8. Capítulo 8: Protocolos de Transporte

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 DNS. 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.

8.1. Funcionamiento de un protocolo de transporte orientado a conexión

Un protocolo de transporte orientado a conexión completo, como TCP, es muy complejo. Por motivos de claridad, se presentan los mecanismos del protocolo de transporte de forma incremental. Comenzaremos 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. Luego se desarrollarán 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.

8.1.1. Servicio de red de entrega confiable ordenada

Se supone 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 que sea altamente fiable
- ✓ 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.
- ✓ 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 (Transport Service, Servicio de Transporte), la información indicada anteriormente. Normalmente, la dirección de usuario se especifica cómo (*estación, puerto*). La variable *"puerto"* representa un usuario TS particular en la estación especificada. En general, 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.

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:

- ✓ 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.

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

Multiplexación

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

Considere, por ejemplo, una entidad de transporte que hace uso de un servicio de red orientado a conexión. ¿Por qué la entidad de transporte debería emplear multiplexación ascendente? Una razón podría porque el proveedor de la red determina lo que cobra a sus clientes en base al número de conexiones de red (circuitos virtuales), porque cada conexión de la capa de red seguramente consume recursos del búfer de los nodos de conmutación. Por lo tanto, si una conexión de la capa de red proporciona suficiente rendimiento para varios usuarios de TS, está indicada la multiplexación ascendente.

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 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. Se analizará esto último más adelante. Primero, vamos a presentar cuatro formas de hacer frente al requisito de control de flujo. La entidad de transporte receptora puede:

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 tiene que ver el control de flujo visto en la capa de enlace de datos, en la primera parte de las presentes notas. Esta técnica estaba basada en:

- ✓ 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, se considera 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.

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 se verá.

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 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 8.1.1.1 ilustra este mecanismo. 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 eso 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.

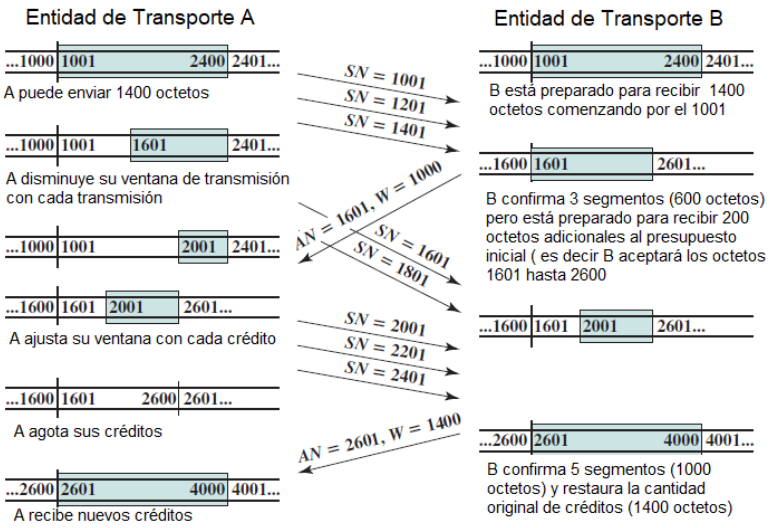

Figura 8.1.1.1: Mecanismo de asignación de créditos

La Figura 8.1.1.2 muestra una visión de este mecanismo según el emisor y según el receptor. 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 8.1.1.1, 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.

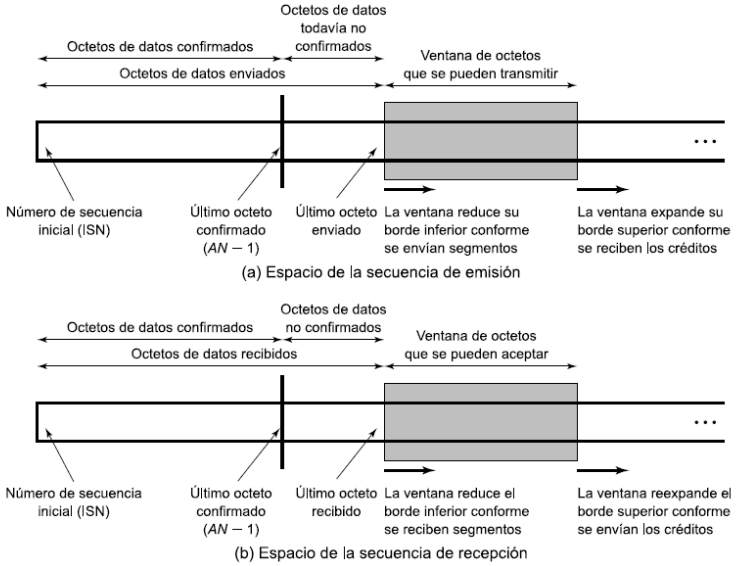

Figura 8.1.1.2: Cómo se ve el flujo de segmentos desde el emisor y el receptor

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 8.1.1.3. 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"*).

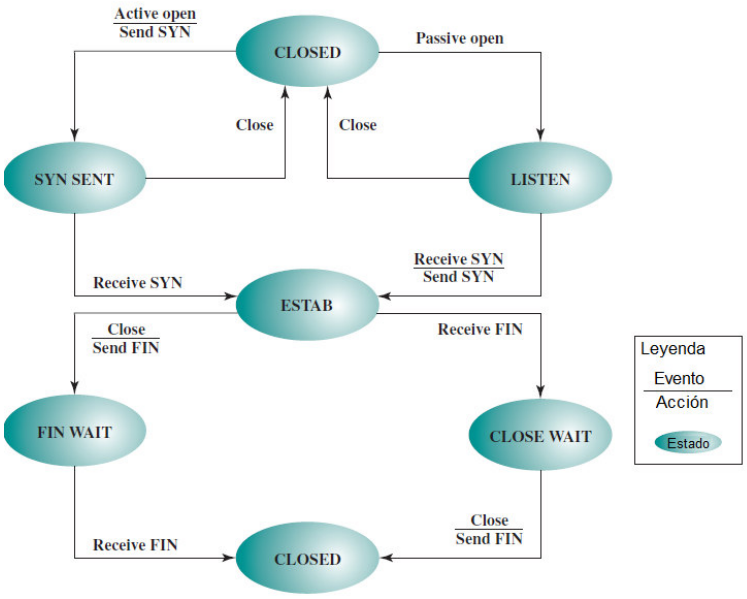

Figura 8.1.1.3: Diagrama de estado de una conexión

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"*.

8.1.2. Servicio de red de entrega no confiable

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. Todos los mecanismos que se tratan a continuación son utilizados por TCP:

- ✓ Entrega ordenada.
- ✓ Estrategia de retransmisión.
- ✓ Detección de duplicados.
- ✓ Control de flujo.

- ✓ Establecimiento de la conexión.
- ✓ Cierre de la conexión.
- ✓ Recuperación ante fallos.

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 en estas notas. 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 tiene la 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 detallan a continuación junto a una breve explicación:

- ✓ 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. Se tratará el primer caso.

Debe observarse 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 no debe 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 8.1.2.1 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.

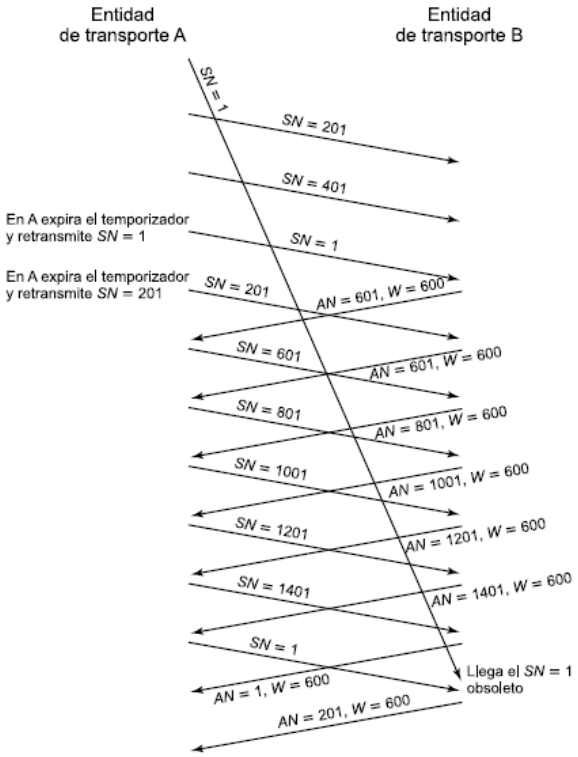

Figura 8.1.2.1: Ejemplo de detección incorrecta de duplicado

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:

- ✓ 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 activa.

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 (descripto en estrategias de retransmisión). A volverá a emitir un SYN cuando el temporizador expire.

La Figura 8.1.2.2 ilustra el funcionamiento típico del diálogo en tres pasos. En la Figura 8.1.2.2(a), 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, Initial sequence numbers) 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 8.1.2.2(b) 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 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 8.1.2.2(c) 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.

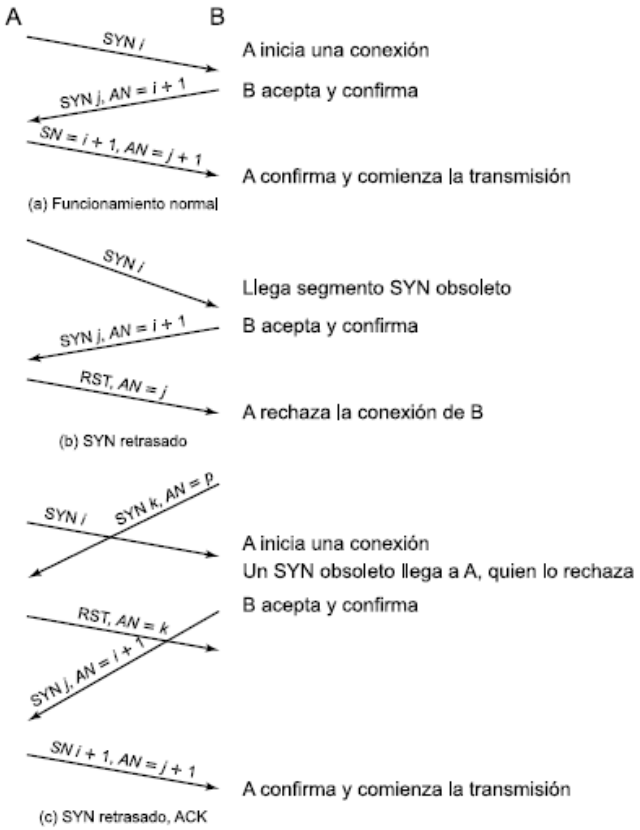

Figura 8.1.2.2: Ejemplos del acuerdo de los tres pasos

Cierre de la conexión

El diagrama de estados de la Figura 8.1.1.3 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 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. Para solucionar este aspecto, 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 "x" y recibir un AN = x + 1.
- ✓ Debe recibir un FIN "y" 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.

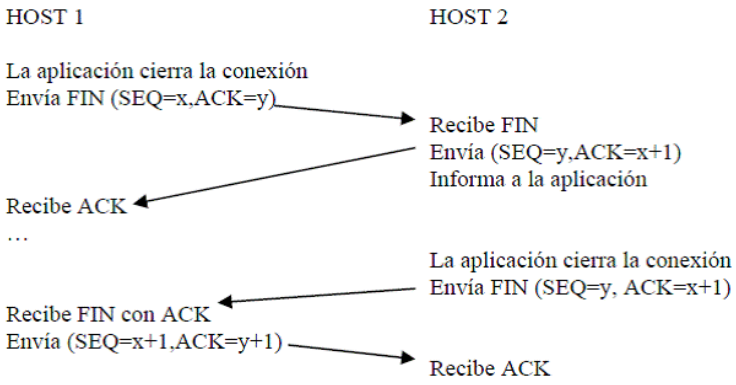

Figura 8.1.2.3: Cierre ordenado en entornos no confiables

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.

8.1.3. Protocolo de Control de Transmisión: TCP (Transmission Control Protocol)

Protocolo de control de transmisión (en inglés Transmission Control Protocol o TCP) es uno de los protocolos fundamentales en Internet. Fue creado entre los años 1973 y 1974 por Vint Cerf[1] y Robert Kahn[2]. El estándar fue publicado como el RFC 793. Entre los servicios que ofrece TCP, se puede destacar:

- ✓ Permite ordenar los segmentos cuando son entregados por el protocolo IP.
- ✓ Implementa mecanismos de control de congestión y así evitar la saturación de la red.
- ✓ Admite que los datos se formen en segmentos de longitud variada para "entregarlos" al protocolo IP.
- ✓ Acepta la multiplexación de los datos, es decir, que la información que viene de diferentes fuentes (por ejemplo, aplicaciones) en la misma línea pueda circular simultáneamente.
- ✓ Por último, permite comenzar y finalizar la comunicación amablemente.

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 8.1.3.1. 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 (Figura 8.1.3.1):

- ✓ Puerto origen (16 bits): usuario TCP origen.
- ✓ Puerto destino (16 bits): usuario TCP destino.

[1] Vinton 'Vint' Gray Cerf (New Haven, Connecticut, Estados Unidos, 23 de junio de 1943) es un científico de la computación estadounidense, considerado uno de los 'padres' de Internet.
[2] Robert Elliot Kahn, (nacido el 23 de diciembre de 1938). Es considerado el vicepresidente de internet. Junto con Vinton Cerf, inventó el protocolo TCP/IP, la tecnología usada para transmitir información en Internet.

- ✓ 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.
- ✓ Número de confirmación (32 bits): contiene el número de secuencia del siguiente octeto que la entidad TCP espera recibir.
- ✓ Longitud de la cabecera (4 bits): número de palabras de 32 bits de la cabecera.
- ✓ Reservado (4 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 del alcance de las presentes notas.
- ✓ Indicadores (8 bits):
    - CWR: ventana de congestión reducida
    - ECE: ECN-Echo; los bits CWR y ECE, definidos en RFC 3168, se utilizan para la función de notificación explícita de congestión
    - URG: el campo de puntero urgente es válido.
    - ACK: el campo de confirmación es válido.
    - PSH: función de forzado.
    - 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.
- ✓ 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.

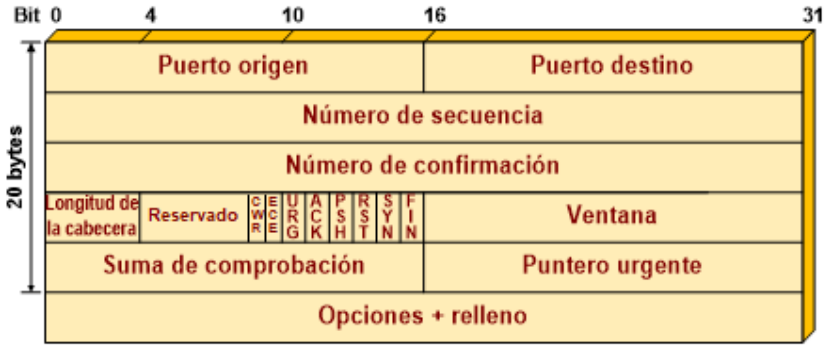

Figura 8.1.3.1: Cabecera de un segmento TCP

El puerto, es un número entero entre 0 y 65535. Por convenio los números 0 a 1023 están reservados para el uso de servicios estándar, por lo que se les denomina puertos bien conocidos o *"well-known ports"*. Cualquier número por encima de 1023 está disponible para ser utilizado libremente por los usuarios. Los valores vigentes de los puertos bien conocidos se pueden consultar por ejemplo en el web de la IANA (Internet Assigned Number Authority) www.iana.org/numbers.html. En la siguiente tabla 8.1.3.2 se muestran algunos de los más más habituales.

Tabla 8.1.3.2: Puertos, protocolos de transporte y aplicaciones

Número de puerto	Protocolo de transporte	Aplicación
20	TCP	FTP Data
21	TCP	FTP Control
22	TCP	SSH
23	TCP	Telnet
25	TCP	SMTP
53	UDP,TCP	DNS
67,68	UDP	DHCP
69	UDP	TFTP
80	TCP	HTTP
110	TCP	POP3
161	UDP	SNMP
443	TCP	SSL
16,384-32,767	UDP	RTP-based Voice and Video

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.

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 de la entrega. Si por cualquier motivo el segmento es capturado y se cambia la dirección ip destino, cuando se calcule el checksum en el destino, el resultado va a diferir del valor que tiene el campo en la cabecera TCP.

La pseudocabecera tiene una longitud de 96 bits (Figura 8.1.3.3), el campo "Dirección origen" y "Dirección destino" es de 32 bits cada uno. Luego hay un campo "Reservado" de 8 bits sin uso que vale todo ceros (0). Luego está el campo que indica número del protocolo de datagrama IP, este valor lo extrae del campo "Protocolo" de la cabecera IP (8 bits) y por último el campo "Longitud del segmento", el cual tiene un tamaño de 16 bits.

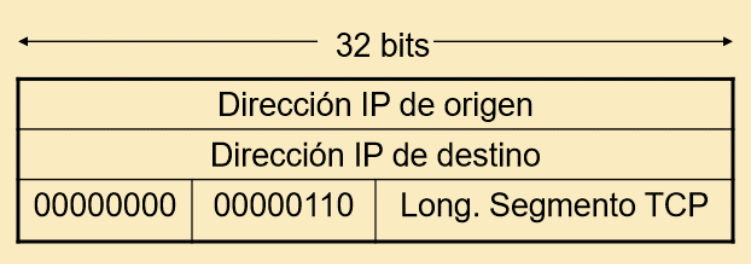

Figura 8.1.3.3: Pseudocabecera de un segmento TCP

Cuestiones de Perfomance: el campo "Opciones"

La mayoría de las opciones de TCP que se analizan son utilizadas solo durante la fase inicial SYN y SYN / ACK del protocolo de enlace de 3 vías que realiza TCP para establecer un enlace virtual antes de transferir cualquier dato. Sin embargo, se pueden utilizar otras opciones a voluntad durante la sesión de TCP.

También es importante tener en cuenta que las *"Opciones"* de TCP pueden ocupar espacio al final del encabezado de TCP y son múltiplos de 8 bits de longitud. Esto significa que, si usamos una opción TCP que tiene 4 bits de longitud, debe haber otros 4 bits de relleno para cumplir con el RFC de TCP. Por lo tanto, la longitud de las opciones de TCP DEBE estar en múltiplos de 8 bits, es decir, 8, 16, 24, 32 etc.

Las opciones disponibles que se fueron incorporando a lo largo de distintos RFC son las siguientes: Maximum Segment Size (MSS), Window Scaling, Selective Acknowledgements (SACK), Timestamps, Nop. A continuación, se describen las dos primeras.

Maximum Segment Size (MSS)

El tamaño máximo de segmento se utiliza para definir el segmento máximo que se utilizará durante una conexión entre dos hosts. Como tal, solo debería ver esta opción utilizada durante la fase SYN y SYN / ACK del protocolo de enlace de 3 vías. La opción MSS TCP ocupa 4 bytes (32 bits) de longitud.

Si la opción MSS se omite en uno o ambos extremos de la conexión, se utilizará el valor de 536 bytes. El valor MSS de 536 bytes está definido por RFC 1122 y se calcula tomando el valor predeterminado de un datagrama IP, 576 bytes, menos la longitud estándar del encabezado IP y TCP (40 bytes), lo que nos da 536 bytes.

En general, es muy importante utilizar el mejor valor posible de MSS para su red porque el rendimiento de su red podría ser extremadamente bajo si este valor es demasiado grande o demasiado pequeño.

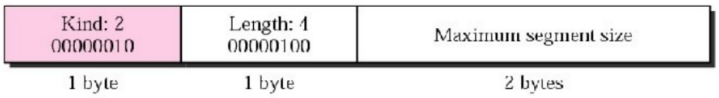

Figura 8.1.3.4: Campo opciones configurado con MSS

Windows Scaling (Escalado de Ventana)

El máximo tamaño de ventana estándar de TCP es de 64 KBytes. El tamaño de ventana establece la máxima cantidad de datos que pueden estar pendientes de confirmación en una comunicación. Cuando la comunicación utiliza un canal de elevada capacidad o gran latencia (es decir elevado valor de RTT) es posible que el emisor tenga que esperar a recibir confirmación antes de seguir enviando.

En una comunicación vía satélite es normal tener valores de RTT de 500 mseg; si un host transmite datos a otro mediante TCP por un enlace vía satélite de 2 Mb/s no podrá enviar más de 64 Kbytes cada 500 mseg, ya que una vez ha llenado la ventana tiene que esperar a recibir el ACK; pero 64 Kbytes cada 500 mseg equivale a 64 * 1024 * 8 / 0,5 = 524.288 / 0,5 = 1,049 Mb/s, por lo que el enlace solo podrá aprovecharse al 50% aproximadamente. En general el rendimiento de una conexión siempre vendrá limitado por la fórmula: *Capacidad máxima = Tamaño de ventana / RTT*

Es decir, siempre que el producto de la capacidad * RTT de una conexión sea superior al tamaño de ventana el rendimiento vendrá limitado por ésta ya que la conexión no es capaz de 'llenar la tubería' de datos y se producirán tiempos de espera en la comunicación. Cuando se diseñó TCP era impensable tener este tipo de problemas, que aparecieron inicialmente con la disponibilidad de enlaces vía satélite de alta velocidad (2 Mb/s). Hoy en día la posibilidad de tener enlaces de alta velocidad en redes de área extensa plantea otras situaciones en las que también se da este problema, por ejemplo, una comunicación TCP sobre una línea ATM OC3 de 155,52 Mb/s con un RTT de 8 ms puede obtener con la ventana estándar un ancho de banda máximo de 524.288 / 0,008 = 65,5 Mb/s; para aprovechar completamente la línea sería precisa una ventana mínima de 155.520.000 * 0,008 = 1.244.160 bits = 152 Kbytes. El RFC 1323, incorporado en muchas

implementaciones de TCP, resuelve este problema ya que permite utilizar un factor de escala para ampliar el tamaño de ventana hasta 2^{30} (1 GByte).

La opción de escala de ventana se usa solo durante el protocolo de enlace de tres vías de TCP (Figura 8.1.3.5). El valor de la escala de la ventana representa el número de bits para desplazar hacia la izquierda el campo de tamaño de la ventana de 16 bits. El valor de la escala de la ventana se puede configurar de 0 (sin cambio) a 14.

Para calcular el tamaño real de la ventana se multiplica el tamaño de la ventana por 2^s, donde "s" es el valor de escala.

Por ejemplo: Si el tamaño de la ventana es 65.535 bytes con un factor de escala de ventana de 3. Tamaño de ventana real = 65535 * 2^3, por lo tanto, el Tamaño de ventana real = 524280 bytes. Para el caso de un factor de escala de 14, el tamaño de la ventana es de 1.073.725.440 Bytes, casi 10 Gb.

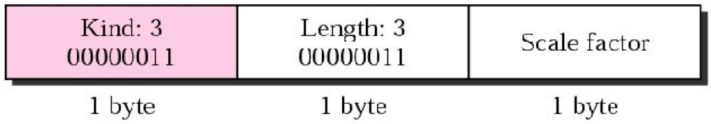

Figura 8.1.3.5: Campo "Opciones" configurado con escalado de ventana

A continuación, se muestra una tabla con el resumen de las alternativas y el significado de cada una de ellas, todas configurables a través del campo "Opciones". Tabla 8.1.3.6.

Tabla 8.1.3.6: Valores que puede tomar el campo "Opciones"

Tipo	*Longitud*	*Valor*	*Descripción*
0	-	-	*Fin de lista de Opciones*: Un byte único al final de las opciones. Su único propósito es alinear las opciones en palabras de 32 bits, cuando se da el caso de que el fin de una opción no coincida con el fin del encabezado.
1	-	-	*No Operación*: Un byte que se usa como separador entre opciones, para contribuir a la alineación en palabras de 32 *bits*.
2	4	MSS	*Maximum Segment Size*: Opción sólo usada en un segmento con el bit *SYN* levantado, para anunciar al otro extremo el tamaño máximo del segmento *TCP*. El valor depende de la *MTU*.
3	3	Factor de Escalamiento de Ventana	*Escalamiento de Ventana*: Se implementa esta opción cuando el tamaño de la ventana anunciada supere el valor máximo que se puede anunciar en el campo *Ventana*, de (65.535 *bytes*). El valor es una potencia de dos por la que hay que multiplicar el valor anunciado en el campo *Ventana*, para obtener el verdadero valor que se está utilizando.
4	2	-	*SACK permitido*: Opción para indicar que el extremo soporta ACK Selectivo.
5	variable	Block ACK	*SACK*: Si se permite SACK, esta opción despliega los bloques de datos no contiguos recibidos, para que no sean retransmitidos
8	10	Sellos de Tiempo	*Sellos de Tiempo*: Opción utilizada para medir tiempos de ida y vuelta en una comunicación. También se puede usar como protección cuando el campo *Número de Secuencia* se consume y vuelve a empezar dentro de una misma conexión.
14	3	Algoritmo de Checksum Alternativo	*Requermiento de Checksum Alternativo:* Opción que permite solicitar un algoritmo de generación de checksum diferente. Debe haber un acuerdo entre ambas partes.
15	variable	Checksum Alternativo	*Checksum Alternativo*: Si el valor del checksum alternativo no cabe en el campo de 16 *bits* del encabezado, se coloca en esta opción.

Funcionamiento 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 como se explicó en la Sección 8.1. 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 + 1mediante 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.

Una conexión está unívocamente 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

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 8.1.

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"* (cerrar). 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.

Administración de temporizadores

TCP usa varios temporizadores (al menos de manera conceptual) para hacer su trabajo. El más importante de éstos es el RTO (Retransmission TimeOut, Temporizador de Retransmisión). Cuando se envía un segmento, se inicia un temporizador de retransmisiones. Si la confirmación de recepción del segmento llega

antes de que expire el temporizador, éste se detiene. Por otro lado, si el temporizador termina antes de que llegue la confirmación de recepción, se retransmite el segmento (y se inicia de nuevo el temporizador). Surge entonces la pregunta: ¿qué tan grande debe ser el intervalo de expiración del temporizador?

La solución es usar un algoritmo dinámico que ajuste de manera constante el intervalo de expiración del temporizador, con base en mediciones continuas del desempeño de la red. El algoritmo utilizado por lo general por el TCP se lo debemos a Jacobson (1988) y funciona de la siguiente manera. Por cada conexión, TCP mantiene una variable llamada SRTT (Smoothed Round Trip Time, Tiempo de Ida y Vuelta Suavizado), que es la mejor estimación actual del tiempo de ida y vuelta al destino en cuestión. Al enviarse un segmento, se inicia un temporizador, tanto para ver el tiempo que tarda la confirmación de recepción como para activar una retransmisión si se tarda demasiado. Si llega la confirmación de recepción antes de expirar el temporizador, TCP mide el tiempo que tardó la confirmación de recepción, por decir R. Luego actualiza el SRTT de acuerdo con la fórmula:

$$SRTT = \alpha\ SRTT + (1 - \alpha)\ R$$

en donde α es un factor de suavizado que determina la rapidez con que se olvidan los valores anteriores. Por lo común, $\alpha = 7/8$. Este tipo de fórmula es un EWMA (Exponentially Weighted Moving Average, Promedio Móvil Ponderado Exponencialmente) o un filtro pasa bajas, que descarta el ruido en las muestras.

En el RFC 2988 se proporcionan más detalles acerca de cómo calcular este tiempo de expiración, incluyendo los valores iniciales de las variables. El temporizador de retransmisión también se mantiene en un mínimo de 1 segundo, sin importar las estimaciones. Éste es un valor conservador que se seleccionó para evitar retransmisiones espurias con base en las mediciones.

El temporizador de retransmisiones no es el único temporizador que TCP utiliza. El temporizador de persistencia es el segundo de ellos. Está diseñado para evitar el siguiente interbloqueo. El receptor envía una confirmación de recepción con un tamaño de ventana de 0 para indicar al emisor que espere. Después el receptor actualiza la ventana, pero se pierde el paquete con la actualización. Ahora, tanto el emisor como el receptor están esperando a que el otro haga algo. Cuando expira el temporizador de persistencia, el emisor transmite un sondeo al receptor. La respuesta al sondeo proporciona el tamaño de la ventana. Si aún es cero, se inicia el temporizador de persistencia una vez más y se repite el ciclo. Si es diferente de cero, ahora se pueden enviar datos.

Un tercer temporizador que utilizan algunas implementaciones es el temporizador de seguir con vida (keepalive). Cuando una conexión ha estado inactiva durante demasiado tiempo, el temporizador de seguir con vida puede expirar para ocasionar que un lado compruebe que el otro aún está ahí. Si no se recibe respuesta, se termina la conexión. Esta característica es controversial puesto que agrega sobrecarga y puede terminar una conexión saludable debido a una partición temporal de la red.

También existen 3 (tres) temporizadores que tienen que ver con un cierre ordenado de una conexión (Figura 8.1.3.7). El primer temporizador es el FIN-WAIT-1, y lo inicializa el host que inicia el cierre activo de la conexión (por ejemplo, A). Este temporizador se desactiva cuando el mencionado host recibe el ACK del host que hace el cierre pasivo (por ejemplo, B). En ese mismo momento A inicializa un temporizador FIN-WAIT-2, el cual contabilizará el tiempo esperando que el host B envío un segmento de FIN. En caso de que A reciba el segmento de FIN del host B o que hubiera expirado el tiempo de FIN-WAIT-2, el host A inicia un temporizador conocido como TIME WAIT, el cual mantiene la conexión "viva" durante 2 MSL (Maximum Segment Lifetime), es decir dos veces el máximo tiempo de vida de un segmento. Esto lo hace por las dudas existieran segmentos de la conexión que aún no hubieran llegado a destino. Luego de ese tiempo, la conexión muere definitivamente.

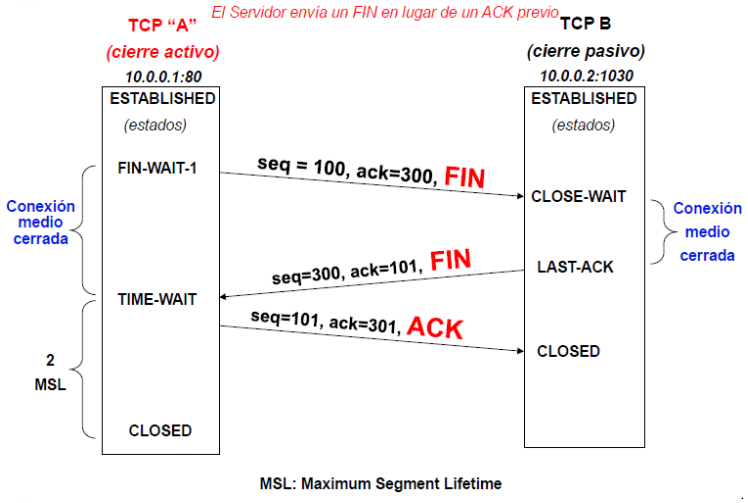

Figura 8.1.3.7: Temporizadores de cierre de la conexión

8.1.4. Control de congestión TCP: gestión de temporizadores y de ventana

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, 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.

Las etiquetas Tahoe, Reno y NewReno de la Tabla 8.1.4.1 se refieren a paquetes de implementación de control de congestión disponibles en muchos sistemas operativos que admiten al protocolo TCP. Las técnicas que utilizan los mencionados paquetes se pueden agrupar, en un sentido amplio, en dos categorías: gestión de temporizadores de retransmisión y gestión de la ventana. A continuación, por razones de alcance del contenido de la asignatura se describirán las técnicas relacionadas con la gestión de ventana.

Tabla 8.1.4.1: Soluciones de Control de Congestión TCP

Medida contra la congestión	RFC 1122	TCP Tahoe	TCP Reno	NewReno
RTT Variance Estimation	✓	✓	✓	✓
Exponential RTO Backoff	✓	✓	✓	✓
Karn's Algorithm	✓	✓	✓	✓
Slow Start	✓	✓	✓	✓
Dynamic Window Sizing on Congestion	✓	✓	✓	✓
Fast Retransmit		✓	✓	✓
Fast Recovery			✓	✓
Modified Fast Recovery				✓

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. Se describirán 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.

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 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:

AW = MIN [RW, CW], dónde

- ✓ AW: ventana permitida, en segmentos. Éste es el número de segmentos que TCP tiene actualmente permitido enviar sin recibir confirmaciones adicionales.
- ✓ Ventana de congestión, CW: en segmentos. Ventana utilizada por TCP durante el inicio y para reducir el flujo durante los periodos de congestión.
- ✓ Ventana de recepción, RW: 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 CW 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 (V., 1988) 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. De esta forma, el crecimiento exponencial 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 (STT) = CW/2.
2. Establecer cwnd = 1 y ejecutar el procedimiento de arranque lento hasta que CW = STT

   En esta fase, cwnd se incrementa en 1 por cada ACK recibido.

3. Para CW >= STT, incrementar cwnd en uno por cada tiempo de ida y vuelta.

La Figura 8.1.4.2 muestra este comportamiento. Se observa 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.

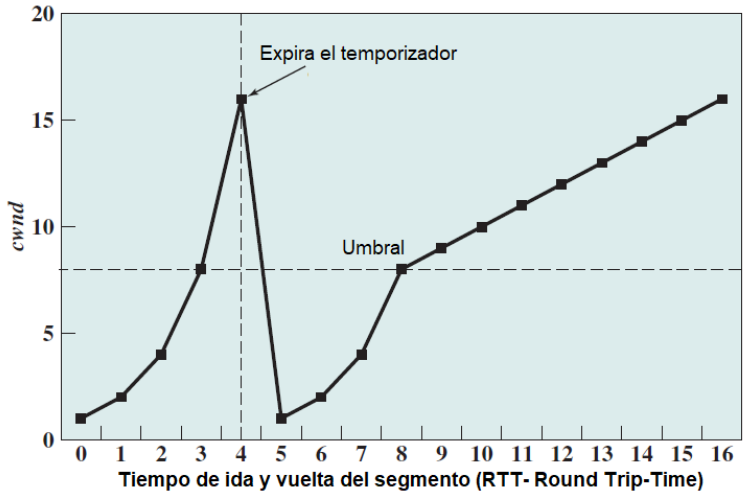

Figura 8.1.4.2: Arranque lento y prevenir la congestión

Retransmisión rápida (Fast Retransmit)

El temporizador de retransmisión (RTO) que utiliza una entidad TCP para determinar cuándo retransmitir un segmento será, por lo general, notablemente más largo que el tiempo real de ida y vuelta (RTT) del ACK para ese segmento que el receptor envía al remitente. Tanto el algoritmo original del RFC 793 como el de Jacobson establecen el valor de RTO en algo mayor que el tiempo de ida y vuelta estimado SRTT (un RTT suavizado). Varios factores hacen que este margen sea deseable:

1. El RTO se calcula sobre la base de una predicción del próximo RTT, estimado a partir de valores pasados de RTT. Si los retrasos en la red fluctúan, entonces el estimado RTT puede ser más pequeño que el RTT real.
2. Del mismo modo, si las demoras en el destino fluctúan, el RTT estimado se convierte en no fidedigno.
3. El sistema de destino puede que por momentos no confirme segmento por segmento, y comience a confirmar con un solo ACK múltiples segmentos, mientras que al mismo tiempo envía ACK cuando tiene algún dato para mandar. mandar. Este comportamiento contribuye a las fluctuaciones en RTT.

Una consecuencia de aplicar un RTO con valores altos, justificado por los tres factores antes descriptos, es que, si se pierde un segmento, TCP puede ser lento para retransmitir. Si el TCP de destino está usando una política de aceptación en orden, entonces se pueden perder muchos segmentos. Incluso en el caso más probable de que el destino TCP esté usando una política de aceptación en ventana, una retransmisión lenta puede causar problemas.

Para ver esto, se supone que A transmite una secuencia de segmentos, el primero de los cuales se pierde. Mientras su ventana de envío no esté vacía y el RTO no expire, A puede continuar transmitir sin recibir un acuse de recibo. B recibe todos estos segmentos excepto el primero. Pero B debe almacenar en el búfer todos estos segmentos entrantes hasta que se retransmita el que falta; no puede despejar su búfer enviando los datos a una aplicación hasta que llegue el segmento que falta. Si la retransmisión del segmento perdido se retrasa demasiado, B tendrá que empezar a descartar los segmentos entrantes.

Jacobson propuso dos procedimientos, llamados Fast Retransmit y Fast Recovery, que en algunas circunstancias mejoran el rendimiento proporcionado por RTO. La retransmisión rápida se aprovecha de la siguiente regla en TCP. Si una entidad TCP recibe un segmento fuera de orden, debe emitir inmediatamente un ACK para el último segmento en orden que recibió. TCP continuará repitiendo este ACK con cada segmento entrante hasta que llegue el segmento que falta para "tapar el agujero" en su buffer.

Cuando el agujero está tapado, TCP envía un ACK acumulativo para todos los segmentos en orden recibidos hasta el momento. Cuando una fuente TCP recibe un ACK duplicado, significa que: el segmento que sigue al segmento confirmado se retrasó de manera que finalmente llegó fuera de orden o ese segmento se perdió. En el primer caso, el segmento llega finalmente y por lo que TCP no debe retransmitir. Pero en el segundo caso, la llegada de un ACK duplicado puede funcionar como un sistema de alerta temprana para indicar al TCP de origen que un segmento se ha se ha perdido y debe ser retransmitido. Para asegurarse de que se está en presencia del segundo caso, Jacobson recomienda que un emisor TCP espere hasta que reciba tres ACK duplicados ACK’s del mismo segmento (es decir, un total de cuatro ACK’s del mismo segmento). En estas circunstancias, es muy probable que el siguiente segmento al segmento confirmado, se haya perdido, y debe ser retransmitido inmediatamente, en lugar de esperar un tiempo de espera.

Fast Recovery (Recuperación Rápida)

Cuando una entidad TCP retransmite un segmento utilizando la retransmisión rápida sabe (o más bien asume) que un segmento se ha perdido, aunque todavía no se haya agotado el tiempo de espera en ese segmento. En consecuencia, la entidad TCP debe tomar medidas para evitar la congestión. Una estrategia obvia es el procedimiento de *"inicio lento/ evitar la congestión"* que se utiliza cuando se produce un tiempo de espera. Es decir, la entidad podría establecer nuevo umbral STT = CW/2, luego asignar la nueva ventana de congestión en uno, es decir poner CW = 1 y comenzar el proceso de inicio lento exponencial hasta que CW = STT, y entonces incrementar CW linealmente. Jacobson argumenta que este enfoque es innecesariamente conservador. Como se acaba de señalar, el mismo hecho de que se hayan devuelto múltiples ACK's indica que los segmentos de datos están llegando con bastante regularidad al otro lado. Así que Jacobson propone una técnica de recuperación rápida (esta técnica evita el proceso inicial exponencial de arranque lento):

1. Retransmitir el segmento perdido
2. Reducir cwnd a la mitad
3. Proceder con el aumento lineal de cwnd.

RFC 3782 (The NewReno Modification to TCP's Fast Recovery Mechanism) modifica el algoritmo de recuperación rápida para mejorar la respuesta cuando dos segmentos se pierden dentro de una misma ventana. Utilizando la retransmisión rápida, un emisor retransmite un segmento antes de que se agote el tiempo porque infiere que el segmento se ha perdido. Esta modificación se puede describir de la siguiente forma: si el emisor recibe posteriormente un acuse de recibo que no cubre todos los segmentos transmitidos antes de que se iniciara la retransmisión rápida, el emisor puede inferir que se perdieron dos segmentos de la ventana actual y retransmitir un segmento adicional. Los detalles de la recuperación rápida y de la recuperación rápida modificada son complejos, pero en caso de conocer a fondo los detalles se deben leer los RFC's 5681 y 3782, los cuales están fuera del alcance del presente apunte.

TCP Reno

TCP RENO incluye algunas mejoras con respecto a implementaciones anteriores de TCP, entre las cuales están algoritmos para evitar la congestión, retransmisión y recuperación rápida. La última mejora evita que el canal se quede desocupado luego de que se haga una retransmisión. La idea tras la rápida recuperación es que el transmisor reenvíe el paquete que lo hizo entrar en ese modo. Luego que ha entrado en el modo de recuperación rápida no hay necesidad de entrar en el modo de inicio lento. Para evitar entrar en el modo de inicio lento RENO usa reconocimientos adicionales para temporizar posteriores paquetes.

El correcto funcionamiento del algoritmo de control de congestión de RENO depende de la pérdida de paquetes, estas pérdidas permiten determinar el ancho de banda disponible y así evolucionar el tamaño de la ventana. Luego de que se pasa el

umbral preestablecido para el modo de inicio lento, se pasa al modo de control de flujo (Congestión Avoidance), en este modo el tamaño de la ventana de RENO va a seguir aumentando en uno cada vez que hay un RTT disponible hasta que haya una pérdida de un paquete, momento en el cual RENO retransmite, y reduce el tamaño de su ventana a la mitad. Este comportamiento se llama incremento aditivo y decremento multiplicativo el cual es aplicado en el modo de control de flujo.

La siguiente Figura 8.1.4.3 representa el comportamiento de TCP RENO a lo largo del tiempo, pudiéndose observar que el objetivo es encontrar un tamaño de ventana que pueda evitar la congestión.

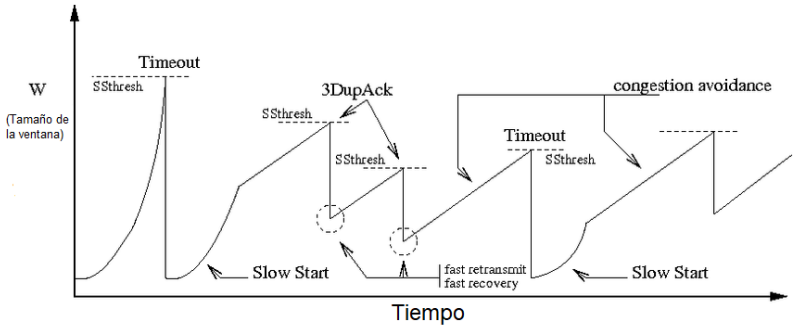

Figura 8.1.4.3: Implementación de TCP Reno

8.1.5. Control de congestión TCP: notificación de congestión explícito (ECN, Explicit Congestion Notification)

Un ejemplo de señalización explícita de congestión es la capacidad de notificación explícita de congestión que se ofrece en IP y TCP y que se define en el RFC 3168. La cabecera IP incluye el campo de notificación explícita de congestión de 2 bits, lo que le permite a los routers notificar a los nodos finales cuales son los paquetes que están experimentando congestión, sin la necesidad de descartar inmediatamente dichos paquetes. Un valor de 00 indica un paquete que no está usando ECN. Un valor de 01 o 10 es establecido por el emisor de datos para indicar que los puntos finales del protocolo de transporte son aptos para ECN. Un valor de 11 es establecido por un router para indicar que se ha encontrado una congestión. La cabecera TCP incluye dos campos de un bit: el campo ECN-Echo y el campo CWR (ventana de congestión reducida). Una secuencia típica de eventos que involucra de estos bits es la siguiente:

- ✓ El campo ECN se pone a 10 o 01 en los paquetes transmitidos por el emisor para indicar que ECN es soportado por las entidades de transporte para estos paquetes.
- ✓ Un router con capacidad ECN detecta una congestión inminente y detecta que un ECN está establecido en 10 o 01 en el paquete que está a punto de abandonar. En lugar de descartar el paquete, el router elige establecer el campo a 11 en la cabecera IP y reenvía el paquete.

- ✓ El receptor recibe el paquete con el valor 11, y establece en 1 el campo ECN-Echo en su siguiente ACK TCP enviado al emisor.
- ✓ El remitente recibe el ACK TCP con ECN-Echo activado y reacciona a la congestión como si se hubiera perdido un paquete.
- ✓ El emisor establece el valor 1 al campo CWR en la cabecera TCP del siguiente paquete enviado al receptor para notificarlo de haber recibido el aviso y haber reaccionado ante el valor del ECN-Echo. Normalmente, la reacción es reducir la ventana de congestión.

8.2. Funcionamiento de un protocolo de transporte no orientado a conexión

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, cuya cabecera se muestra en la Figura 8.2.1. 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 por ejemplo en el protocolo de resolución de nombres (DNS, Domain Name System) o el protocolo de gestión de red (SNMP, Simple Network Management Protocol).

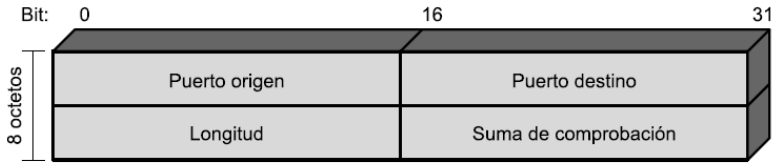

Figura 8.2.1: Cabecera UDP

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. Además, 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 video, 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 8.2.1. 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.