RFC: 793

PROTOCOLO DE CONTROL DE TRANSMISIÓN

DARPA INTERNET PROGRAM

ESPECIFICACIÓN DE PROTOCOLOS

Septiembre de 1981

preparado para

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

por

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

Traducido al español por
Pedro J. Ponce de León (pierre@mail.ono.es),
Domingo Sánchez Ruiz (domingo@toboso.fis.ucm.es) y
José Ignacio Usera Estirado (j_usera@lycos.es)
Marzo de 2002

CONTENIDO

PREFACIO

Este documento describe el protocolo, estándar del DoD, de control de la transmisión (TCP). Ha habido nueve ediciones previas de la especificación de TCP por ARPA sobre las que el presente texto se basa enormemente. Ha habido muchos participantes en este trabajo tanto en términos de conceptos como de texto. Esta edición clarifica varios detalles y elimina los ajustes de tamaño de búfer al "final de carta" y reescribe el mecanismo de carta como una función de entrega inmediata.

Jon Postel

Editor

RFC: 793
Reemplaza a: RFC 761
IENs: 129, 124, 112, 81,
55, 44, 40, 27, 21, 5

PROTOCOLO DE CONTROL DE TRANSMISIÓN

DARPA INTERNET PROGRAM
ESPECIFICACIÓN DE PROTOCOLOS

1. INTRODUCCIÓN

El "protocolo de control de transmisión" ('Transmission Control

Protocol', TCP) está pensado para ser utilizado como un protocolo 'host' a 'host' muy fiable entre miembros de redes de comunicación de computadoras por intercambio de paquetes y en un sistema interconectado de tales redes.

Este documento describe las funciones que debe realizar el protocolo de control de transmisión, el programa que lo implementa, y su interfaz con los programas o usuarios que requieran de sus servicios.

1.1. Motivación

Los sistemas de comunicación entre computadoras están jugando un papel cada vez más importante en entornos militares, gubernamentales y civiles. Este documento centra principalmente su atención en los requisitos militares de comunicación entre computadoras, especialmente la robustez bajo comunicaciones no plenamente fiables y la disponibilidad ante congestiones, aunque muchos de estos problemas pueden encontrarse igualmente en los sectores civil y gubernamental.

A la par que las redes estratégicas y tácticas de comunicación entre computadoras están siendo desarrolladas y desplegadas, es esencial proporcionar medios de interconexión entre ellas y proporcionar protocolos estándares de comunicación entre procesos que puedan soportar un amplio rango de aplicaciones. Anticipando la necesidad de tales estándares, la subsecretaría de defensa del congreso de los diputados para la investigación e ingeniería ('the Deputy Undersecretary of Defense for Research and Engineering') ha declarado el protocolo de transmisión de control (TCP) descrito aquí como la base para la estandarización de los protocolos de comunicación entre procesos, dentro del ámbito de todo el Departamento de Defensa (DoD).

TCP es un protocolo orientado a la conexión, fiable y entre dos extremos, diseñado para encajar en una jerarquía en capas de protocolos que soportan aplicaciones sobre múltiples redes. TCP proporciona mecanismos para la comunicación fiable entre pares de procesos en computadoras 'host' ancladas en redes de comunicación de computadoras distintas, pero interconectadas. Se hacen muy pocas

suposiciones sobre la fiabilidad de los protocolos de comunicación por debajo de la capa de TCP. TCP sólo supone que puede acceder a un servicio de transmisión de datagramas simple, aunque en principio poco fiable, de los protocolos del nivel inferior. En principio, TCP debería ser capaz de operar encima de un amplio espectro de sistemas de comunicaciones que incluye desde conexiones por cables fijos ('hard-wired conections') hasta redes de intercambio de paquetes o redes de circuitos conmutados.

TCP se basa en los conceptos descritos primeramente por Cerf y Kahn en [1]. TCP encaja en una arquitectura de protocolos en capas justo por

encima del protocolo de internet [2], protocolo básico que proporciona un medio para TCP de enviar y recibir segmentos de longitud variable de información envuelta en "sobres" de datagramas de internet. El datagrama de internet proporciona un medio de direccionar TCPs de origen y de destino situados en redes diferentes. El protocolo de internet también trata con la fragmentación y el reensamble de segmentos de TCP que sean necesarios para conseguir el transporte y la entrega sobre múltiples redes y las puertas de enlace que las interconectan. El protocolo de internet también lleva información sobre la prioridad, clasificación de seguridad y compartimentación de los segmentos de TCP, de tal forma que esta información pueda ser comunicada de extremo a extremo entre múltiples redes.

Capas de protocolos

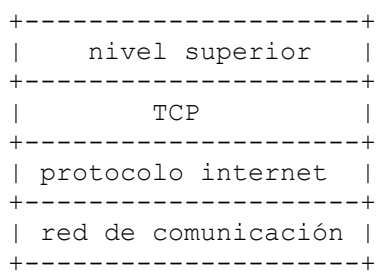


Figura 1

Gran parte de este documento se ha escrito dentro del contexto de las implementaciones de TCP que son corresidentes con protocolos de más alto nivel en la computadora anfitriona. Algunos sistemas de computadoras se conectarán a las redes vía computadoras intermediarias ('front-end computers') que alojan las capas de protocolos TCP e internet, a la vez que el software específico de redes. Esta especificación de TCP describe una interfaz con los protocolos de mayor nivel que resulta ser implementable incluso para el caso de computadoras intermediarias, siempre y cuando se implemente un adecuado protocolo entre el 'host' y la computadora intermediaria.

1.2. Ámbito

TCP está pensado para proporcionar un servicio fiable de comunicación entre procesos en un entorno con múltiples redes. TCP está pensado para ser un protocolo de 'host' a 'host' de uso común en redes múltiples.

1.3. Sobre este documento

Este documento representa una especificación del comportamiento requerido a cualquier implementación de TCP, tanto en sus interacciones con protocolos de más alto nivel como con sus interacciones con otros TCP. El resto de esta sección ofrece una muy breve visión de las interfaces y operaciones del protocolo. La sección 2 resume los presupuestos de corte filosófica para el diseño de TCP. La sección 3 ofrece tanto una descripción detallada de las acciones que se le exigen a TCP cuando varios eventos acontecen (llegada de nuevos segmentos, llamadas de usuario, errores, etc.) así como los detalles de los formatos de los segmentos TCP.

1.4. Interfaces

TCP presenta interfaz por un lado con el usuario o los procesos de aplicación y por el otro con un protocolo de más bajo nivel como es el protocolo de internet (IP).

La interfaz entre un proceso de aplicación y TCP se ilustrará con un detalle razonable. Esta interfaz consiste en un conjunto de llamadas, de forma muy similar a las llamadas que un sistema operativo proporciona a los procesos de aplicación para manipular ficheros. Por ejemplo, hay llamadas para abrir y cerrar conexiones y para enviar y recibir datos por las conexiones establecidas. Se exige también que TCP pueda comunicarse asíncronamente con los programas de aplicación. Aunque se deja considerable libertad a los fabricantes de implementaciones de TCP a la hora de diseñar las interfaces que sean apropiadas para el entorno de un sistema operativo particular, se exige un mínimo de funcionalidad en la interfaz TCP/usuario de cualquier implementación válida.

La interfaz entre TCP y el protocolo de nivel inferior queda esencialmente sin especificar, exceptuando el hecho de que se asume que hay un mecanismo por el cual los dos niveles pueden pasar información asíncronamente el uno al otro. Típicamente, se espera que el protocolo de nivel inferior especifique esta interfaz. Se ha diseñado TCP para trabajar en un entorno muy genérico de redes interconectadas. El nivel inferior que se asumirá a lo largo de este documento es el protocolo de internet [2].

1.5. Operación

Como se ha hecho notar más arriba, el propósito principal de TCP consiste en proporcionar un servicio de conexión o circuito lógico fiable y seguro entre pares de procesos. Para proporcionar este

servicio encima de un entorno de internet menos fiable, el sistema de comunicación requiere de mecanismos relacionados con las siguientes áreas:

- Transferencia básica de datos
- Fiabilidad
- Control de flujo
- Multiplexamiento
- Conexiones
- Prioridad y seguridad

La operación básica de TCP en cada uno de estas áreas se describe en los siguiente párrafos:

Transferencia básica de datos:

TCP es capaz de transferir un flujo continuo de octetos en cada sentido entre sus usuarios empaquetando un cierto número de octetos en segmentos para su transmisión a través del sistema de internet. En general, los módulos de TCP deciden cuándo bloquear y enviar datos según su propia conveniencia.

Algunas veces los usuarios necesitan estar seguros de que todos los datos que habían entregado al módulo de TCP han sido transmitidos. Para este propósito se define una función 'push' ("enviar inmediatamente"). Para asegurar que los datos entregados al módulo de TCP son realmente transmitidos, el usuario emisor debe indicarlo mediante la función 'push'. Un 'push' en un cierto instante causa que los módulos de TCP envíen y entreguen inmediatamente al usuario receptor los datos almacenados hasta ese instante. El instante exacto en que se ejecuta la función 'push' podría no ser visible para el usuario receptor. Tampoco la función 'push' proporciona una marca de límite de registros.

Fiabilidad:

El módulo de TCP debe poder recuperar los datos que se corrompan, pierdan, dupliquen o se entreguen desordenados por el sistema de comunicación del entorno de internet. Esto se consigue asignando un número de secuencia a cada octeto transmitido, y exigiendo un acuse de recibo (ACK, N.T.:del inglés 'acknowledgment') del módulo de TCP receptor. Si no se recibe un ACK dentro de un cierto plazo de expiración prefijado, los datos se retransmiten. En el receptor, se utilizan los números de secuencia para ordenar correctamente los segmentos que puedan haber llegado desordenados y para eliminar los duplicados. La corrupción de datos se trata añadiendo un campo de suma de control ('checksum') a cada segmento transmitido, comprobándose en el receptor y descartando los segmentos dañados.

En tanto en cuanto los módulos de TCP continúen funcionando adecuadamente y el sistema de internet no llegue a quedar particionado de forma completa, los errores de transmisión no afectarán la correcta entrega de datos. TCP se recupera de los errores del sistema de comunicación de internet.

Flujo de control:

TCP proporciona al receptor un medio para controlar la cantidad de datos enviados por el emisor. Esto se consigue devolviendo una "ventana" con cada ACK, indicando el rango de números de secuencia aceptables más allá del último segmento recibido con éxito. La ventana indica el número de octetos que se permite que el emisor transmita antes de que reciba el siguiente permiso.

Multiplexamiento:

Para permitir que muchos procesos dentro de un único 'host' utilicen simultáneamente las posibilidades de comunicación de TCP, el módulo de TCP proporciona una serie de direcciones o puertos dentro de cada 'host'. Concatenadas con las direcciones de red y de 'host' de la capa de comunicación internet conforman lo que se denomina una dirección de conector ('socket'). Un par de direcciones de conector identifica de forma única la conexión. Es decir, un conector puede utilizarse simultáneamente en múltiples conexiones.

La asignación de puertos a los procesos se gestiona de forma independiente en cada 'host'. Sin embargo, resulta de la máxima utilidad asignar a los procesos más utilizados frecuentemente (i.e., un gestor de registros ('logger') o un servicio compartido) conectores fijos que se hacen conocer de forma pública. Estos servicios pueden entonces ser accedidos a través de direcciones conocidas públicamente. El establecimiento y aprendizaje de las direcciones de los puertos de otros procesos puede involucrar otros mecanismos más dinámicos.

Conexiones

La fiabilidad y los mecanismos de control de flujo descritos más arriba exigen que los módulos de TCP inicialicen y mantengan una información de estado para cada flujo de datos. La combinación de esta información, incluyendo las direcciones de los conectores, los números de secuencia y los tamaños de las ventanas, se denomina una conexión. Cada conexión queda especificada de forma única por un par de conectores que corresponden con sus dos extremos.

Cuando dos procesos desean comunicarse, sus módulos de TCP deben establecer primero una conexión (inicializar la información de estado en cada lado). Cuando la comunicación se ha completado, la conexión se termina o cierra con la intención de liberar recursos para otros usos.

Como las conexiones tienen que establecerse entre 'hosts' no fiables y sobre un sistema de comunicación internet no fiable, se utiliza un mecanismo de acuerdo que usa números de secuencia basados en tiempos de reloj para evitar una inicialización errónea de las conexiones.

Prioridad y seguridad:

Los usuarios de TCP pueden indicar el nivel de seguridad y prioridad de su comunicación. Se emplean valores por defecto cuando estas características no se necesiten.

2. FILOSOFIA

2.1. Elementos del sistema de inter-redes.

El entorno de inter-redes consiste en una serie de 'hosts' conectados a varias redes que a su vez están interconectadas vía pasarelas ('gateways'). Aquí se supone que las redes pueden ser tanto redes locales (i.e. de tipo de ETHERNET) o grandes redes (i.e. ARPANET), pero en cualquier caso basadas en tecnología de intercambio de paquetes. Los agentes activos que producen y consumen los mensajes son los procesos. Varios niveles de protocolos en las redes, las pasarelas y 'hosts' dan soporte a un sistema de comunicación entre procesos que proporciona un flujo de datos bidireccional sobre conexiones lógicas entre los puertos de los procesos.

El término paquete o trama se utiliza generalmente aquí para significar el conjunto de datos de una transacción entre un 'host' y su red. El formato de bloques de datos intercambiados dentro de una red, en general, no nos importará aquí.

Los 'hosts' son computadoras unidas a una red, y, desde el punto de vista de la comunicación en la red, constituyen los orígenes y destinos de los paquetes. Los procesos han de verse como los elementos activos en los computadores anfitriones (de acuerdo con la definición bastante común de un proceso como un programa en ejecución). Incluso los terminales y ficheros u otros dispositivos de E/S son vistos como comunicándose unos con otros a través del uso de procesos. Así, toda comunicación puede verse como una comunicación entre procesos.

Ya que un proceso puede necesitar distinguir entre varios flujos de comunicación entre él mismo y otro proceso (o procesos), se supone que un proceso puede tener un cierto número de puertos a través de los cuales se comunica con los puertos de otros procesos.

2.2. Modelo de operación

Los procesos transmiten datos llamando al módulo de TCP y pasando búferes de datos como argumentos de la llamada. El módulo de TCP empaqueta en segmentos los datos provenientes de estos búferes y efectúa una llamada al módulo de internet para que transmita cada segmento al módulo de TCP de destino. El TCP receptor coloca los datos de un segmento en el búfer de recepción del usuario y lo notifica al usuario receptor. Los módulos de TCP incluyen información de control en los segmentos que puede ser utilizada para asegurar una transmisión fiable y ordenada de datos.

El modelo de comunicación del protocolo de internet es de tal forma que hay un módulo del protocolo internet asociado con cada módulo de TCP y que, a su vez, proporciona una interfaz con la red local. Este módulo de internet empaqueta los segmentos de TCP dentro de los datagramas de internet y encamina estos datagramas hacia un módulo de internet de destino por una pasarela intermedia. Para poder

transmitir el datagrama a través de la red local, se encapsula dentro de un paquete de red local.

Los conmutadores de paquetes en la red pueden realizar ulteriores empaquetamientos, fragmentaciones o cualquier otra operación necesaria para conseguir la entrega del paquete local al módulo de internet de destino.

En una pasarela entre redes, el datagrama de internet es "desenvuelto" a partir del paquete de red local y examinado para determinar a través de qué red debería el datagrama viajar a continuación. El datagrama de internet es entonces "envuelto" otra vez por un paquete de red apropiado, envíado a la red siguiente y encaminado hacia la siguiente pasarela, o hacia el destino final.

Se permite que una pasarela divida un datagrama de internet en fragmentos de datagrama más pequeños, si esto es necesario para la transmisión a través de la siguiente red. Para hacer esto, la pasarela produce un conjunto de datagramas de internet; cada uno de ellos transportando un fragmento. Los fragmentos pueden ser subdivididos a su vez en ulteriores pasarelas. El formato de fragmento de datagrama de internet se ha diseñado de tal forma que el módulo de internet de destino puede reensamblar los fragmentos en datagramas de internet.

El módulo de internet de destino desenvuelve el segmento del datagrama (después de haber reensamblado el datagrama, si así era necesario) y lo pasa al módulo de TCP de destino.

Este modelo simple de operación descrito disimula muchos detalles. Una característica importante es el tipo de servicio. Éste proporciona información a la pasarela (o a un módulo de internet) para guiarla en la selección de los parámetros de servicio a utilizar para atravesar la siguiente red. Entre la información del tipo de servicio se encuentra el grado de prioridad del datagrama. Los datagramas pueden también transportar información de seguridad que permitan a los 'hosts' y a las pasarelas que operen en entornos de seguridad multinivel separar adecuadamente datagramas por razones de seguridad.

2.3. El entorno del 'host'

Se supone que el módulo de TCP lo es del sistema operativo. Los usuarios accederán a dicho módulo de una forma muy similar a como acceden al sistema de archivos. El módulo de TCP, a su vez, puede llamar a otras funciones del sistema operativo, por ejemplo, para gestionar las estructuras de datos. Se supone que la interfaz real final con la red se controla mediante un módulo manejador del dispositivo ('device driver') de red. El módulo de TCP no llama directamente al manejador, sino que efectúa llamadas al módulo del protocolo de datagramas de internet que en su lugar llama al manejador

del dispositivo de red.

Los mecanismos de TCP no impiden la implementación de TCP en un procesador intermediario ('front-end'). Sin embargo, en una implementación de este tipo, un protocolo de 'host' a 'front-end' debe proporcionar la funcionalidad necesaria para soportar el tipo de interfaz TCP-usuario descrita en este documento.

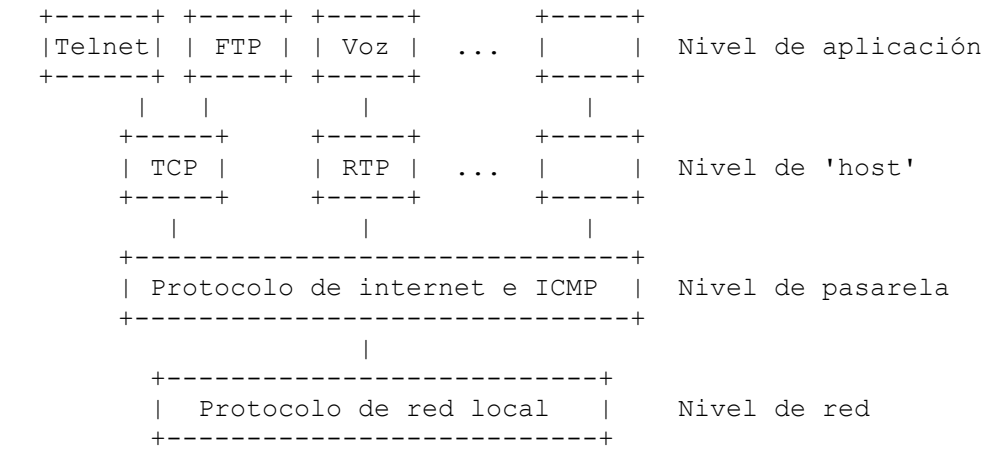
2.4. Interfaces

La interfaz TCP/usuario proporciona al usuario funciones de llamada al módulo de TCP para abrir (OPEN) o cerrar (CLOSE) una conexión, para enviar (SEND) o recibir (RECEIVE) datos, o para obtener información de estado (STATUS) sobre una conexión. Estas llamadas son del mismo tipo que otras llamadas al sistema operativo realizadas desde programas de usuario como, por ejemplo, las llamadas para abrir, leer y cerrar un fichero.

La interfaz TCP/internet proporciona llamadas para enviar y recibir datagramas direccionados a los módulos TCP en cualquier 'host' del sistema de internet. Estas llamadas tienen parámetros para pasar la dirección, el tipo de servicio, la prioridad y otra información de control.

2.5. Relación con otros protocolos

El siguiente diagrama ilustra el lugar de TCP en la jerarquía de protocolos:



Relación entre protocolos

Figura 2.

Se espera que TCP sea capaz de soportar protocolos de nivel superior eficientemente. Debería ser fácil implementar la interfaz de TCP con los protocolos de nivel superior como Telnet de ARPANET o AUTODIN II THP.

2.6. Comunicación fiable

Un flujo de datos envíado sobre una conexión de TCP se entrega de forma fiable y ordenada al destino.

La transmisión es fiable gracias al uso de números de secuencia y de acuses de recibo. Básicamente, se le asigna un número de secuencia a cada octeto de datos. El número de secuencia del primer octeto de datos en un segmento se transmite con ese segmento y se le denomina el número de secuencia del segmento. Los segmentos también llevan un número de acuse de recibo que es el número de secuencia del siguiente octeto de datos esperado en la transmisión en el sentido inverso. Cuando el módulo de TCP transmite un segmento conteniendo datos, pone una copia en una cola de retransmisión e inicia un contador de tiempo; si llega el acuse de recibo para esos de datos, el segmento se borra de la cola. Si no se recibe el acuse de recibo dentro de un plazo de expiración, el segmento se retransmite.

La llegada del acuse de recibo no garantiza que los datos ya hayan sido entregados al usuario final, sino únicamente que el TCP receptor ha asumido la responsabilidad de hacerlo.

Para controlar el flujo de datos entre los módulos de TCP, se utiliza un mecanismo de flujo de control. El TCP receptor devuelve una "ventana" al TCP emisor. Esta ventana especifica el número de octetos, a contar a partir del número del acuse de recibo, que el TCP receptor está en ese momento preparado para recibir.

2.7. Establecimiento y finalización de la conexión

Para identificar los distintos flujos de datos que un módulo TCP puede manejar simultáneamente, TCP proporciona un identificador de puerto. Como los identificadores de puertos son seleccionados de forma independiente por cada TCP, puede que no sean únicos. Para disponer de direcciones únicas dentro de cada TCP, se concatena la dirección de internet que identifica al módulo de TCP con el identificador de puerto para así conformar una dirección de conector ('socket') que será única a largo de todo el conjunto de redes interconectadas.

Una conexión queda completamente especificada por el par de conectores de sus extremos. Un conector local puede participar en muchas conexiones con diferentes conectores foráneos. Una conexión puede ser utilizada para transportar datos en los dos sentidos, es decir, es bidireccional ('full duplex').

Los módulos de TCP pueden asociar libremente los puertos a procesos. Sin embargo, algunos conceptos básicos son necesarios en cualquier implementación. Debe haber un conjunto de conectores públicos que el módulo de TCP asocie sólo con los procesos "apropiados" por algún medio. Se supone la posibilidad de que los procesos puedan "poseer" puertos, y que los procesos puedan iniciar conexiones sólo con los puertos que ellos posean. (Qué medios son necesarios para implementar

estas posesiones es una cuestión local, pero se supone la existencia de un comando de usuario de petición de puerto ('Request Port'), o un método de asignar de forma única un grupo de puertos a un proceso dado, por ejemplo, la asociación de los bits más significativos de un numero de puerto con un proceso dado).

Una conexión se especifica en la llamada de apertura OPEN con los argumentos de un puerto local y una dirección de conector remoto. Como respuesta, el módulo de TCP proporciona un nombre local (corto) a la conexión con la que el usuario puede referirse a la conexión en subsiguientes llamadas. Hay varias cosas que deben recordarse acerca de una conexión. Para guardar esta información podemos imaginar que existe una estructura llamada "Bloque de control de transmisión" (TCB, 'Transmission Control Block'). Una estrategia para su implementación tendría el nombre local de la conexión como un puntero al TCB de esta conexión. La llamada OPEN también especifica si se persigue de forma activa el establecimiento de la conexión o si se espera de forma pasiva.

Una petición pasiva OPEN significa que el proceso desea aceptar peticiones de conexión entrantes en vez de intentar iniciar directamente una conexión. A menudo, el proceso solicitante de un OPEN pasivo aceptará una petición posterior de conexión proveniente de cualquiera. En este caso, la dirección de conector remoto igual a todo ceros se utiliza para denotar un conector sin especificar. Sólo se permiten llamadas OPEN pasivas a conectores remotos de este tipo.

Un proceso de servicio que deseara proporcionar servicios a otros procesos desconocidos lanzaría una petición de OPEN pasivo contra un conector remoto sin especificar. Entonces, se podría establecer una conexión con cualquier proceso que haga una petición de conexión al conector local. Sería de gran ayuda si es público el conector local al que está asociado este servicio.

El uso de conectores públicos ('well-known') constituye un mecanismo conveniente para la asociación a priori de direcciones de conectores con un servicio estándar. Por ejemplo, el proceso "Servidor de telnet" está permanentemente asignado a un conector particular, y del mismo modo se reservan otros conectores para los procesos "Transferencia de ficheros" ('File Transfer'), "Entrada de trabajos remotos" ('Remote Job Entry'), "Generador de texto" ('Text Generator'), "Generador de

eco" ('Echoer') y "Sumidero" ('Sink') (el propósito de los últimos tres servicios es la realización de pruebas). Una dirección de conector podría ser reservada para el acceso a un servicio de "guía" que podría devolver la dirección de conector específica en la que se proporcione un servicio nuevo. El concepto de un conector público es parte de la especificación de TCP, pero la asignación de conectores con servicios queda fuera de esta especificación. (Ver [4]).

Los procesos pueden ejecutar OPEN pasivos y esperar a otros OPEN activos que concuerden y provengan de otros procesos, siendo entonces informados por el módulo de TCP cuando las conexiones se hayan

finalmente establecido. Dos procesos que se dirigen OPEN activos entre sí al mismo tiempo serán conectados correctamente. Esta flexibilidad es crítica para dar soporte a la computación distribuída en la que los componentes pueden actuar asíncronamente unos con respecto a otros.

Hay dos casos principales para la concordancia de conectores entre OPEN pasivos locales y OPEN activos remotos. En el primer caso, un OPEN pasivo local ha especificado completamente el conector remoto. En este caso, la concordancia debe ser exacta. En el segundo caso, los OPEN pasivos locales han dejado el conector remoto sin especificar. En este caso, cualquier conector remoto es aceptable en tanto en cuanto los conectores locales concuerden. Otras posibilidades incluirían concordancias parcialmente restringidas.

Si hay varios OPEN pasivos pendientes (registrados en varios TCB) con el mismo conector local, un OPEN activo remoto concordará con el TCB que contenga el conector remoto específico en el campo OPEN activo remoto, si existe un TCB así, antes de seleccionar un TCB con un conector remoto sin especificar.

Los procedimientos para establecer conexiones utilizan el indicador de control de sincronización (SYN) e involucran un intercambio de tres mensajes. Se ha denominado a este intercambio como el acuerdo en tres pasos ('three-way handshake') [3].

Una conexión se inicia ante la coincidencia de un segmento entrante que contiene un SYN y una entrada de TCB en espera previa, habiendo sido ambos creados por un comando de usuario OPEN. La concordancia de conectores locales y remotos determina cuándo una conexión ha sido iniciada. La conexión queda "establecida" cuando los números de secuencia quedan sincronizados en ambos sentidos.

La finalización de una conexión también involucra el intercambio de segmentos, en este caso transportando un indicador de control FIN.

2.8. Comunicación de datos

Puede pensarse en los datos que fluyen en una conexión como en un flujo de octetos. El usuario emisor indica en cada llamada SEND mediante la puesta a uno del indicador PUSH si los datos de esa llamada (y de cualquier llamada precedente) deben ser entregados inmediatamente al usuario receptor.

Se permite a un TCP emisor recolectar datos del usuario emisor y enviar esos datos en segmentos según propia conveniencia, siempre y cuando no se invoque la función 'push', en ese caso, el TCP emisor debe enviar todos los datos pendientes. Cuando el TCP receptor ve el indicador PUSH, no debe esperar más datos del TCP receptor antes de pasar los datos al proceso receptor.

No hay necesariamente una relación entre el uso de funciones 'push' y

el tamaño de los segmentos. Los datos de cualquier segmento en particular pueden ser el resultado, total o parcial, de una llamada SEND única, o de múltiples llamadas SEND.

El propósito de la función 'push' y del indicador PUSH consiste exclusivamente en transportar inmediatamente los datos desde el usuario emisor al usuario receptor. No proporciona en ningún caso un servicio de registro.

Sí existe una estrecha relación entre el uso de la función 'push' y el uso de los búferes de datos empleados en la interfaz TCP/usuario. Cada vez que un indicador PUSH se asocia con datos colocados en el búfer del usuario receptor, su contenido se devuelve al usuario para ser procesado incluso si el búfer no se ha llenado. Si los datos que llegan llenan el búfer de usuario antes de que se lea un indicador PUSH, los datos se pasan al usuario en las unidades de tamaño del búfer.

TCP también proporciona un mecanismo para comunicar al receptor de los datos que en algún punto más adelante en el flujo de datos que está actualmente leyendo habrá datos urgentes. TCP no intenta definir lo que el usuario debe hacer en concreto cuando se le notifica la existencia de datos urgentes pendientes, sólo da la noción general de que el proceso receptor tendrá que tomar medidas para procesar los datos urgentes de forma rápida.

2.9. Prioridad y seguridad

TCP hace uso del campo de tipo de servicio del protocolo de internet y de la opción de seguridad para proporcionar prioridad y seguridad a los usuarios de TCP, tomando como base cada conexión. No todos los módulos de TCP trabajarán necesariamente en un entorno de seguridad multinivel; algunos puede que estén limitados a usos reservados solamente, y otros puede que operen en un único nivel de seguridad y compartimentación. Consecuentemente, algunas implementaciones y

servicios de TCP para usuarios puede que estén limitados a un subconjunto del caso con seguridad multinivel.

Los módulos de TCP que operen en un entorno con seguridad multinivel deben marcan apropiadamente los segmentos salientes con los niveles de seguridad, compartimentación y prioridad. Estos módulos TCP deben también proporcionar a sus usuarios o a los protocolos de más alto nivel como Telnet o THP una interfaz que les permita especificar el grado de seguridad, compartimentación y prioridad de las conexiones.

2.10. Principio de robustez

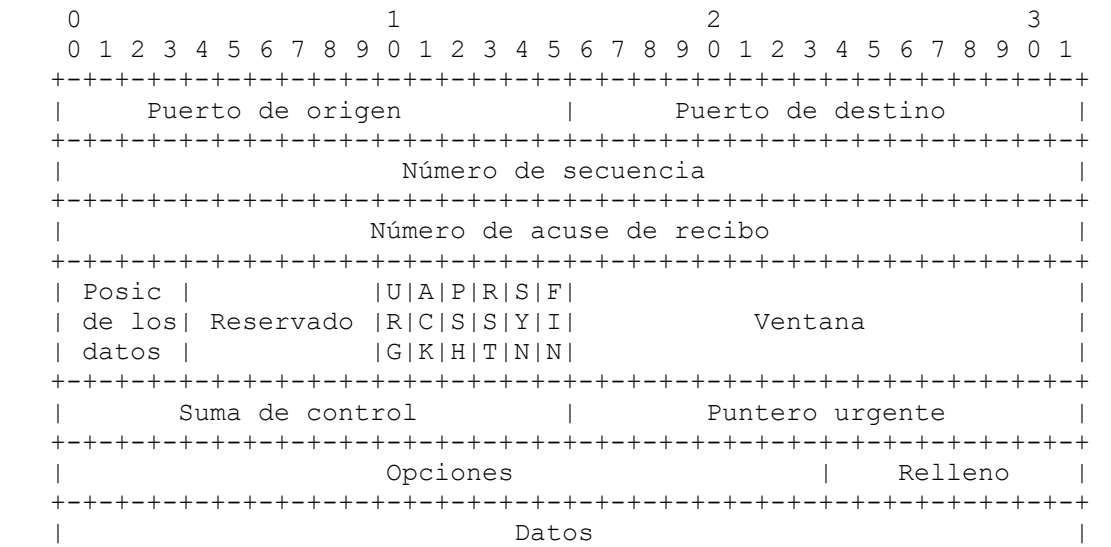
Las implementaciones de TCP seguirán un principio general de robustez: sé conservador en lo que hagas, sé liberal en lo que aceptes de los demás.

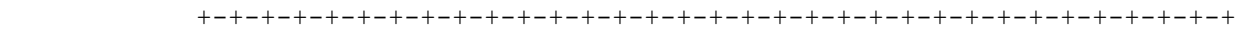
3. ESPECIFICACION FUNCIONAL

3.1. Formato de la cabecera

Los segmentos de TCP se envían como datagramas de internet. La cabecera del protocolo de internet transporta varios campos de información, entre los que se incluyen las direcciones de los 'host' de origen y de destino [2]. Una cabecera de TCP sigue a la cabecera de internet, aportando información específica del protocolo de TCP. Esta división permite la existencia de otros protocolos de la capa de 'host' distintos de TCP.

Formato de la cabecera de TCP





Formato de la cabecera de TCP

Nótese que cada marca horizontal representa un bit.

Figura 3.

Puerto de origen: 16 bits

El número del puerto de origen.

Puerto de destino: 16 bits

El número del puerto de destino.

Número de secuencia: 32 bits

El número de secuencia del primer octeto de datos de este segmento (excepto cuando el indicador SYN esté puesto a uno). Si SYN está puesto a uno es el número de secuencia original (ISN: 'initial sequence number') y, entonces, el primer octeto de datos es ISN+1.

Número de acuse de recibo: 32 bits

Si el bit de control ACK está puesto a uno, este campo contiene el valor del siguiente número de secuencia que el emisor del segmento espera recibir. Una vez que una conexión queda establecida, este número se envía siempre.

Posición de los datos: 4 bits

El número de palabras de 32 bits que ocupa la cabecera de TCP. Este número indica dónde comienzan los datos. La cabecera de TCP (incluso una que lleve opciones) es siempre un número entero de palabras de 32 bits.

Reservado: 6 bits

Reservado para uso futuro. Debe valer 0.

Bits de control: 6 bits (de izquierda a derecha):

URG: Hace significativo el campo "Puntero urgente"
ACK: Hace significativo el campo "Número de acuse de recibo"

PSH: Función de "Entregar datos inmediatamente" ('push')
RST: Reiniciar ('Reset') la conexión
SYN: Sincronizar ('Synchronize') los números de secuencia
FIN: Últimos datos del emisor

Ventana: 16 bits

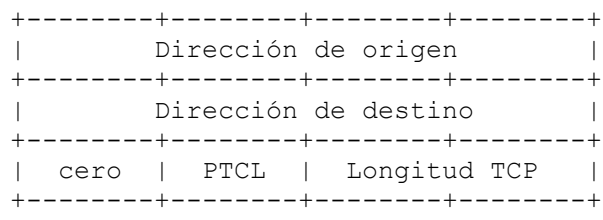
El número de octetos de datos, a contar a partir del número indicado en el campo de "Número de acuse de recibo", que el emisor de este segmento está dispuesto a aceptar.

Suma de control: 16 bits

El campo "Suma de control" es el complemento a uno de 16 bits de la suma de los complementos a uno de todas las palabras de 16 bits de la cabecera y del texto. Si un segmento contiene un número impar de octetos de cabecera y texto, el último octeto se rellena con ceros a la derecha para formar una palabra de 16 bits con el propósito de calcular la suma de control. En el cálculo de la suma de control, el propio campo suma de control se considera formado por ceros.

La suma de control también incluye una pseudocabecera de 96 bits prefijada imaginariamente a la cabecera TCP. Esta pseudocabecera

contiene la dirección de origen, la dirección de destino, el protocolo, y la longitud del segmento de TCP. Esto proporciona una protección ante segmentos mal encaminados. Esta información es transportada por el protocolo de internet y es transferida a través de la interfaz TCP/Red en los argumentos o en los resultados de las llamadas de TCP a IP.



La "longitud TCP" consiste en la suma de la longitud de la cabecera de TCP más la de los datos en octetos (esto no es una cantidad transmitida explícitamente, sino que ha de calcularse), y no incluye los 12 octetos de la pseudo cabecera.

Puntero urgente: 16 bits.

Este campo indica el valor actual del puntero urgente como un desplazamiento positivo desde el número de secuencia de este segmento. El puntero urgente apunta al número de secuencia del octeto al que seguirán los datos urgentes. Este campo es

interpretado únicamente si el bit de control URG está establecido a uno.

Opciones: variable

Los campos de opciones pueden ocupar un cierto espacio al final de la cabecera de TCP, pero siempre de una longitud múltiplo de 8 bits. En el cálculo de la suma de control, se incluyen todas las opciones. Una opción puede empezar en cualquier posición múltiplo de ocho. Existen dos posibilidades para el formato de una opción:

Caso 1: Un octeto único con el tipo de opción.

Caso 2: Un octeto con el tipo de opción, un octeto con la longitud de la opción, y los octetos con los datos propiamente dichos de la opción.

La longitud de la opción tiene en cuenta tanto el octeto con el tipo de opción como el propio octeto de longitud así como los octetos con los datos de la opción.

Nótese que la lista de opciones puede ser más corta que lo que el campo "Posición de los datos" podría implicar. El contenido de la cabecera más allá de la opción "Fin de la lista de opciones" debe ser un relleno de cabecera (es decir, ceros).

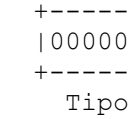
Un módulo de TCP debe implementar todas las opciones.

Las opciones definidas en la actualidad incluyen (donde el tipo se indica en octal):

Tipo	Longitud	Significado
0	-	Fin de la lista de opciones.
1	-	Sin operación.
2	4	Tamaño máximo de segmento.

Definiciones de opciones específicas

Fin de la lista de opciones



Tipo=0

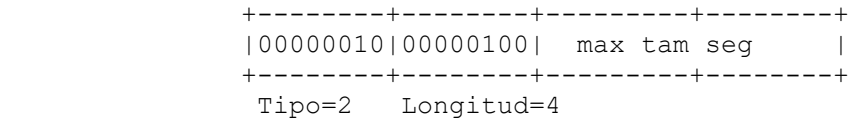
Este código de opción indica el final de la lista de opciones. Éste podría no coincidir con el final de la cabecera de TCP

deducida a partir del campo "Posición de los datos". Esta opción se utiliza al final de todas las opciones, no al final de cada opción, y sólo es necesario utilizarla si el final de las opciones restantes no coincide con el final de la cabecera de TCP.

Sin operación

Este código de opción puede ser utilizado entre opciones, por ejemplo, para alinear el comienzo de una opción subsiguiente con el comienzo de una palabra. No se garantiza que los emisores vayan a utilizar esta opción, por lo que los receptores deben estar preparados para procesar todas las opciones, incluso si no comienzan al principio de una palabra.

Máximo tamaño de segmento



Datos de la opción "Máximo tamaño de segmento": 16 bits

Si esta opción está presente, entonces indica el tamaño máximo de segmento que puede recibir el módulo de TCP que envía este segmento. Este campo debe enviarse únicamente en la petición inicial de conexión (i.e., en los segmentos con el bit de control SYN puesto a uno). Si no se utiliza esta opción, se permite cualquier tamaño de segmento.

Relleno: variable

El relleno de la cabecera de TCP se utiliza para asegurar que la cabecera de TCP finaliza, y que los datos comienzan, en una posición múltiplo de 32 bits. El relleno está compuesto de ceros.

3.2. Terminología

Antes de empezar a discutir cualquier cosa sobre el modo de operación de TCP, es necesario introducir con cierto detalle algo de terminología. El mantenimiento de una conexión de TCP requiere el

almacenamiento y seguimiento de varias variables. Estas variables han de imaginarse almacenadas en un registro de conexión denominado "Bloque de control de la transmisión" o TCB ('Transmission Control Block'). Entre las variables almacenadas en el TCB se hallan las direcciones de los conectores local y remoto, los valores de seguridad y prioridad de la conexión, los punteros a los búferes de envío y recepción del usuario, los punteros a la cola de retransmisión y al segmento actual. También se almacenan en el TCB algunas variables relacionadas con los números de secuencia de envío y recepción.

Variables de la secuencia de envío

SND.UNA - envío sin acuse de recibo recibido ('unacknowledged')
SND.NXT - envío siguiente ('next')
SND.WND - ventana ('window') de envío
SND.UP - puntero urgente ('urgent pointer') de envío
SND.WL1 - número de secuencia del segmento utilizado en la última ('last') actualización de la ventana
SND.WL2 - número de acuse de recibo del segmento utilizado en la última actualización de la ventana
ISS - número de secuencia de envío inicial ('initial send sequence')

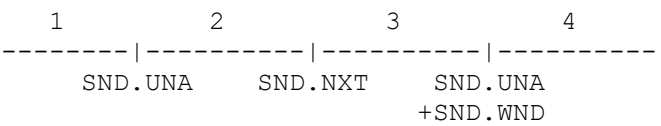
Variables de la secuencia de recepción

RCV.NXT - siguiente recepción
RCV.WND - ventana de recepción
RCV.UP - puntero urgente de recepción
IRS - número de secuencia de recepción inicial ('initial receive sequence')

Los siguientes diagramas pueden ayudar a relacionar alguna de estas

variables con el espacio de secuencias.

Espacio de secuencias de envío



1 - anteriores números de secuencia de los que ya se ha recibido acuse de recibo
2 - número de secuencia de datos sin acuse de recibo recibido
3 - número de secuencia permitido en la siguiente transmisión de datos
4 - futuros números de secuencia no permitidos todavía en

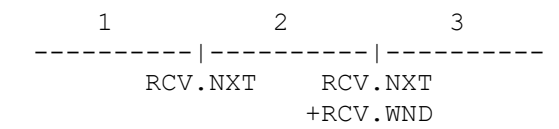
la siguiente transmisión

Espacio de secuencias de envío

Figura 4.

La ventana de envío es la porción del espacio de números de secuencia etiquetada como 3 en la figura 4.

Espacio de secuencias de recepción



1 - anteriores números de secuencia de los que ya se han envíado el acuse de recibo
2 - números de secuencia permitidos para una nueva recepción
3 - futuros números de secuencia que todavía no están permitidos

Espacio de secuencias de Recepción

Figura 5.

La ventana de recepción es la porción del espacio de secuencias etiquetada como 2 en la figura 5.

Además hay algunas variables que se utilizarán con frecuencia en la exposición que sigue más adelante y que toman sus valores de los campos del segmento actualmente en proceso.

Variables del segmento actual

SEG.SEQ - número de secuencia del segmento
SEG.ACK - número de acuse de recibo del segmento
SEG.LEN - longitud ('length') del segmento
SEG.WND - ventana del segmento
SEG.UP - puntero urgente del segmento
SEG.PRC - valor de prioridad del segmento

Una conexión progresa de acuerdo con una serie de estados durante su tiempo de vida. Los estados son: 'LISTEN' (en escucha), 'SYN-SENT' (SYN enviado), 'SYN-RECEIVED' (SYN recibido), 'ESTABLISHED' (establecida), 'FIN-WAIT-1' ("en espera de fin-1"), 'FIN-WAIT-2' ("en espera de fin-2"), 'CLOSE-WAIT' (en espera de cierre), 'CLOSING'

(cerrándose), 'LAST-ACK' (último acuse de recibo), 'TIME-WAIT' (en espera), y el estado ficticio 'CLOSED' (cerrada). 'CLOSED' es un estado ficticio porque representa el estado en el que no existe TCB, y por lo tanto, no hay conexión. De forma breve, los significados de los estados son (N.T.:de aquí en adelante no se entrecomillarán los estados y llaamdas salvo cuando aparezcan en frases en mayúsculas):

LISTEN - representa la espera de una solicitud de conexión proveniente de cualquier TCP y puerto remotos.

SYN-SENT - representa la espera de una solicitud de conexión concordante tras haber enviado previamente una solicitud de conexión.

SYN-RECEIVED - representa la espera del acuse de recibo confirmando la solicitud de conexión tras haber recibido tanto como envíado una solicitud de conexión.

ESTABLISHED - representa una conexión abierta, los datos recibidos pueden ser entregados al usuario. El estado normal para la fase de transferencia de una conexión.

FIN-WAIT-1 - representa la espera de una solicitud de finalización de la conexión proveniente del TCP remoto, o del acuse de recibo de la solicitud de finalización previamente envíada.

FIN-WAIT-2 - representa la espera de una solicitud de finalización del TCP remoto.

CLOSE-WAIT - representa la espera de una solicitud de finalización de la conexión proveniente del usuario local.

CLOSING - representa la espera del paquete, proveniente del TCP remoto, con el acuse de recibo de la solicitud de finalización.

LAST-ACK - representa la espera del acuse de recibo de la solicitud de finalización de la conexión previamente envíada al TCP remoto (lo que incluye el haber envíado el acuse de recibo de la solicitud

remota de finalización de la conexión).

TIME-WAIT - representa la espera durante suficiente tiempo para asegurar que el TCP remoto recibió el acuse de recibo de su solicitud de finalización de la conexión.

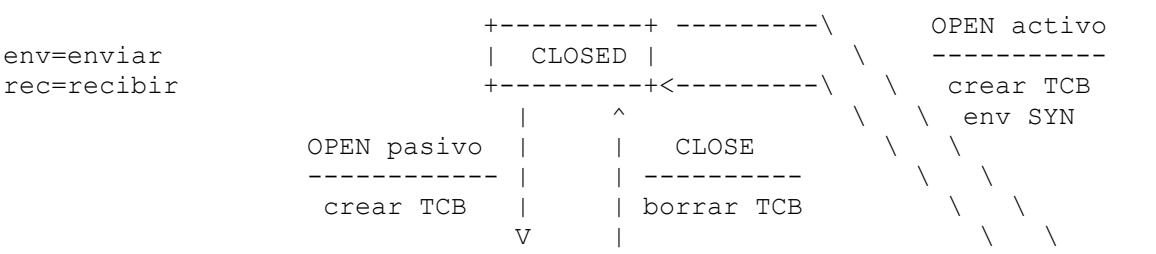
CLOSED - representa un estado sin conexión en absoluto

Una conexión de TCP progresa de un estado a otro en respuesta a eventos. Los eventos son: las llamadas de usuario, OPEN ("abrir"),

SEND ("enviar"), RECEIVE ("recibir"), CLOSE ("cerrar"), ABORT ("interrumpir"), and STATUS ("mostrar el estado"); la llegada de segmentos, particularmente aquellos que contengan alguno de los indicadores SYN, ACK, RST y FIN, y la expiración de plazos de tiempos.

El diagrama de estados de la figura 6 ilustra sólo los cambios de estados, junto con los eventos causantes y las acciones resultantes, pero no aborda ni las condiciones de error ni las acciones que no estén relacionadas con cambios de estado. En una sección posterior, se ofrecerá más detalle sobre todo lo relacionado con la reacción de TCP ante eventos.

NOTA BENE: este diagrama únicamente es un resumen y no debe ser tomado como la especificación total



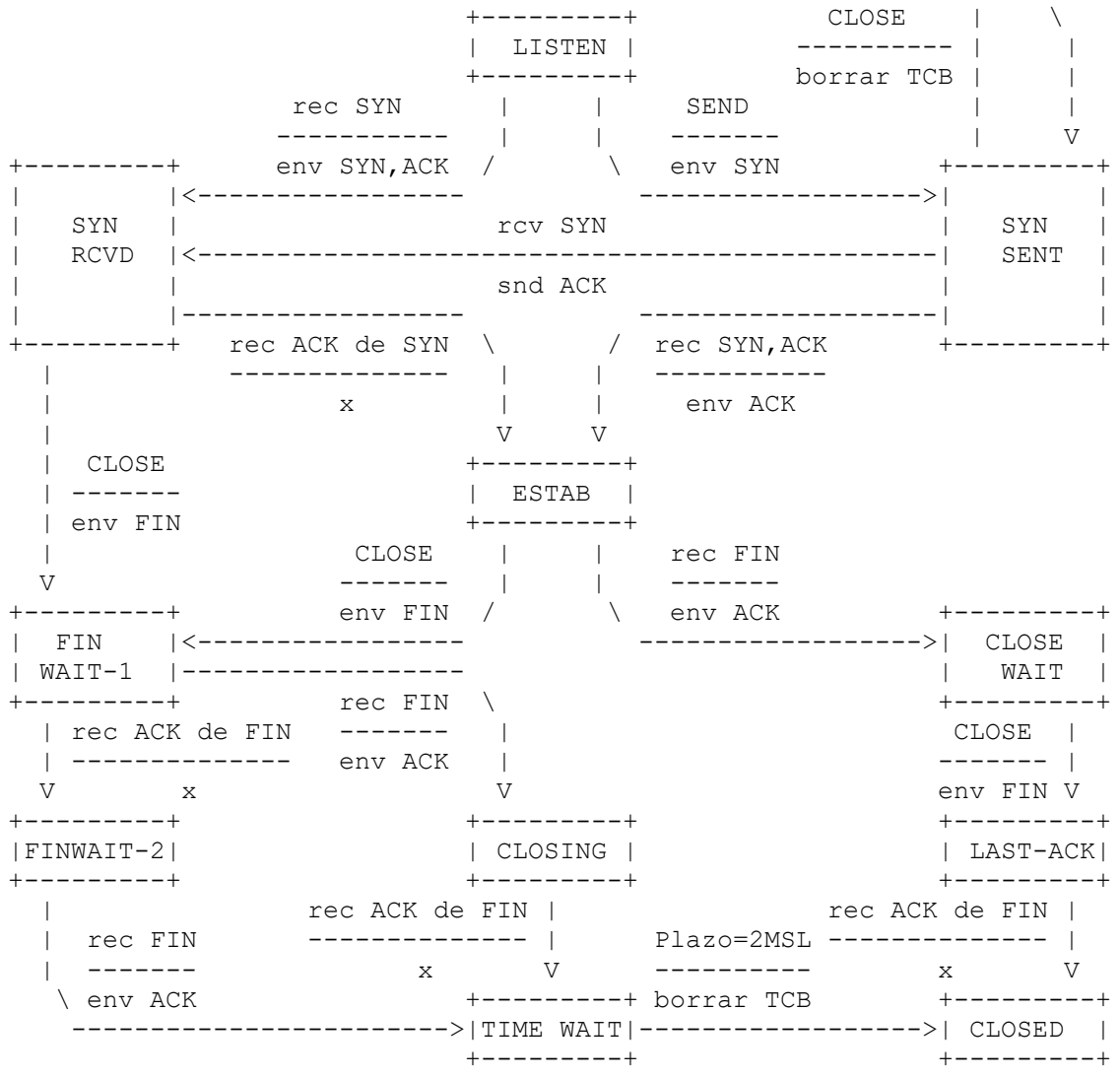


Diagrama de estados de una conexión de TCP
Figura 6.

3.3. Números de secuencia

El hecho de que todo octeto de datos envíado por una conexión de TCP tenga asociado un número de secuencia constituye una noción fundamental en el diseño de TCP. Ya que cada octeto se secuencia,

puede realizarse un acuse de recibo de cada uno de ellos. El mecanismo de acuse de recibo empleado es acumulativo, de tal forma que el acuse de recibo de un número de secuencia X indica que todos los octetos hasta, pero no incluyendo, X, han sido recibidos. Este mecanismo permite la detección fácil y directa de duplicados generados por retransmisiones. La numeración de octetos dentro de un segmento es de la siguiente forma: el primer octeto de datos inmediatamente tras la cabecera es el de menor número y los octetos siguientes son numerados consecutivamente.

Es esencial tener presente que el espacio real de números de secuencia es finito, aunque muy grande. Este espacio abarca desde el 0 hasta 2**32 -1. Como el espacio es finito, toda la aritmética con números de secuencia debe ser desarrollada módulo 2**32. Esta aritmética sin signo preserva la relación entre números de secuencia incluso cuando se rota desde 2**32 -1 a 0 de nuevo. Hay algunas sutilezas en los cálculos de la aritmética de módulos, por tanto se ha de tener un especial cuidado a la hora de programar las funciones de comparación de valores. El símbolo "=<" significa "menor o igual" (modulo 2**32).

Las típicas comparaciones de números de secuencia que TCP debe realizar incluyen:

(a) Determinar si un acuse de recibo se refiere a algún número de secuencia envíado pero del que no se ha recibido todavía su acuse de recibo

(b) Determinar si se ha recibido el acuse de recibo de todos los números de secuencia ocupados por un segmento (por ejemplo, para eliminar un segmento de la cola de retransmisión).

(c) Determinar si un segmento entrante contiene números de secuencia esperados (es decir, si el segmento cabrá en la ventana de recepción).

Como respuesta a sus envíos de datos, el módulo de TCP recibirá acuses de recibo. Es necesario realizar las siguientes comparaciones para procesar los acuses de recibo.

SND.UNA = el menor número de secuencia sin acuse de recibo recibido

SND.NXT = número de secuencia del próximo envío

SEG.ACK = acuse de recibo procedente del TCP receptor (próximo número de secuencia esperado por el TCP receptor)

SEG.SEQ = primer número de secuencia de un segmento

SEG.LEN = el número de octetos ocupados por los datos en el segmento (incluyendo SYN y FIN)

SEG.SEQ+SEG.LEN-1 = último número de secuencia de un segmento

Un nuevo acuse de recibo (denominado un "acuse de recibo aceptable"), es aquél para el cual la siguiente desigualdad es cierta:

SND.UNA < SEG.ACK =< SND.NXT

Un segmento ubicado en la cola de retransmisión queda completamente confirmado por un acuse de recibo si la suma de su número de secuencia inicial y su longitud es menor o igual que el valor del acuse de recibo indicado en el segmento entrante.

Es necesario realizar las siguientes comparaciones cuando se reciben datos.

RCV.NXT = siguiente número de secuencia esperado de los segmentos entrantes, lo que define el borde izquierdo o inferior de la ventana de recepción

RCV.NXT+RCV.WND-1 = último número de secuencia esperado en un segmento entrante, lo que define el borde derecho o superior de la ventana de recepción

SEG.SEQ = primer número de secuencia ocupado por el segmento entrante

SEG.SEQ+SEG.LEN-1 = último número de secuencia ocupado por el segmento entrante

Se considera que un segmento ocupa una porción válida del espacio de secuencias de recepción si

RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND

o si

RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

La primera parte de esta comprobación mira a ver si el comienzo del

segmento cae dentro de la ventana, la segunda parte comprueba si el final del segmento cae dentro de la ventana; si el segmento pasa cualquiera de las dos partes de la comprobación, entonces es que contiene datos dentro de la ventana.

La realidad es algo más compleja que todo esto. Debido a la existencia de ventanas y segmentos de tamaño cero, tenemos cuatro posibilidades a considerar a la hora de aceptar un segmento entrante:

Longitud segmento	Ventana recep.	Comprobación
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND
>0	0	no aceptable
>0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND o RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

Nótese que cuando el tamaño de la ventana de recepción es cero, no se debería aceptar ningún segmento excepto los segmentos ACK. Así, es perfectamente posible para un TCP mantener una ventana de recepción de tamaño nulo mientras esté transmitiendo datos y recibiendo segmentos ACK. Sin embargo, incluso cuando la ventana de recepción es cero, un módulo de TCP debe procesar los campos RST y URG de todos los segmentos entrantes.

Se ha aprovechado el esquema de numeración para proteger también cierta información de control. Esto se consigue incluyendo implícitamente algunos indicadores de control en el espacio de secuencias de tal forma que puedan ser retransmitidos y confirmados sin posibilidad de confusión (i.e., se considerará una y sólo una copia de la información de control). La información de control no se transporta físicamente en el espacio de datos del segmento. Consecuentemente, se deben adoptar reglas implícitas de asignación de números de secuencia para los indicadores de control. SYN y FIN son los únicos indicadores de control que requieren esta protección, y estos indicadores se utilizan únicamente para la apertura y cierre de la conexión. Para propósitos de numeración de secuencias, se considera que el SYN ocurre antes que el primer octeto de datos reales del segmento en el que es transportado, mientras que el FIN se considera que ocurre después del último octeto de datos del segmento que lleva el FIN. La longitud del segmento (SEG.LEN) incluyen tanto los datos como el espacio de secuencias ocupado por los indicadores de control. Cuando un SYN está presente, entonces SEG.SEQ es el número de secuencia del SYN.

Selección del número de secuencia inicial

Este protocolo no pone ninguna restricción sobre el hecho de reutilizar una y otra vez la misma conexión. Una conexión se define por el par de conectores. Cualquier nueva instancia de una conexión será referida como una encarnación de la conexión. El problema que aparece aquí es: "¿ cómo identifica TCP los segmentos duplicados de encarnaciones previas de la conexión ?" Este problema resulta evidente si la conexión se abre y se cierra en una sucesión muy rápida, o si la conexión se corta acompañada de pérdidas de memoria y después se reestablece.

Para evitar una confunsión, se debe evitar que los segmentos de una encarnación de una conexión utilizen los números de secuencia que todavía posiblemente estén siendo utilizados en la red por otra encarnación anterior. Se desea asegurar esto, incluso si un TCP se cuelga y pierde todo conocimiento de los números de secuencia que estaba empleando. Cuando se crean nuevas conexiones se utiliza un generador de números iniciales de secuencia (ISN, 'initial sequence number') que selecciona un nuevo ISN de 32 bits. El generador estará asociado a un reloj (posiblemente ficticio) de 32 bits, cuyo bit menos significativo se incrementa aproximadamente cada 4 microsegundos. Así, el ISN rota aproximadamente cada 4.55 horas. Como queda asumido que los segmentos no permanecerán más tiempo en la red que la "vida máxima del segmento" (MSL, 'Maximum Segment Lifetime') y que el MSL es menor que 4.55 horas, podemos por tanto razonablemente presuponer que los ISN serán únicos.

Para cada conexión existe un número de secuencia de envío y un número de secuencia de recepción. El número de secuencia de envío inicial (ISS, 'initial send sequence') lo elige el TCP emisor, y el número de secuencia de recepción inicial (IRS, 'initial receive sequence') se asume durante el procedimiento de establecimiento de la conexión.

Para que una conexión quede establecida o inicializada, los dos TCP deben sincronizarse entre sí mediante sus números de secuencia iniciales. Esto se consigue durante el establecimiento de la conexión mediante el intercambio de segmentos que transportan un bit de control denominado "SYN" (acrónimo de 'syncronize' o sincronizar en inglés) y los números de secuencia iniciales. A modo de abreviatura, los segmentos que transportan el bit SYN se denominan también "SYN". Por tanto, la solución requiere de un apropiado mecanismo para elegir un número de secuencia inicial y una forma de acuerdo ligeramente complicada para intercambiar los ISN.

La sincronización requiere que cada parte envíe su propio número de secuencia inicial y que reciba una confirmación de su llegada en la forma de un acuse de recibo de la otra parte. Cada parte debe también recibir el número de secuencia inicial de la otra parte y enviar un acuse de recibo como confirmación.

1) A --> B SYN mi número de secuencia es X
2) A <-- B ACK tu número de secuencia es X
3) A <-- B SYN mi número de secuencia es Y
4) A --> B ACK tu número de secuencia es Y

Como los pasos 2 y 3 pueden combinarse en un único mensaje, este procedimiento se denomina acuerdo en tres pasos (o en tres mensajes).

Es necesario un acuerdo en tres pasos porque los números de secuencia no están sujetos a un reloj global de la red, y los TCP puede que tengan diferentes mecanismos para elegir los ISN. El receptor del primer SYN no tiene forma de saber si el segmento era uno anterior retrasado o no, a menos que recuerde el último número de secuencia utilizado en la conexión (lo que no siempre es posible), y por tanto debe pedir al emisor que verifique este SYN. El acuerdo en tres pasos y las ventajas de un esquema basado en relojes se discuten en [3].

Saber cuándo permanecer en silencio

Para estar seguro de que un TCP no cree un segmento que transporte un número de secuencia que podría estar duplicado por la existencia de otro segmento anterior que todavía permanezca en la red, TCP debe permanecer en silencio durante el tiempo de vida máximo de un segmento (MSL, 'maximum segment lifetime') antes de que asigne cualquier número de secuencia tras arrancar o recuperarse de una caída en la que se perdieron los números de secuencia en uso. En esta especificación, el MSL se toma como de 2 minutos. Esta es la elección de un ingeniero, y puede ser cambiada si la experiencia indica que es conveniente hacerlo. Nótese que si un TCP se reinicializa de alguna forma en la que todavía retenga en memoria los números de secuencia en uso, entonces no es necesaria ninguna espera en absoluto; en ese caso, basta con utilizar números de secuencia mayores que los utilizados con anterioridad.

El concepto de "tiempo en silencio" ('Quiet Time') de TCP

Esta especificación establece que los 'hosts' que se "cuelguen" sin haber retenido ningún conocimiento de los últimos números de secuencia transmitidos en cada conexión activa (i.e., sin cerrar) deberían retrasar la emisión de cualquier segmento de TCP durante al menos el "tiempo de vida máximo de segmento" (MSL) acordado. Más abajo, se da una explicación para esta especificación. Los fabricantes de TCP pueden violar esta restricción del "tiempo en silencio", pero sólo asumiendo el riesgo de causar que algunos datos viejos sean aceptados como nuevos o que los nuevos datos sean rechazados como duplicados de otros anteriores por algunos receptores en el entorno de internet.

Los TCP van consumiendo el espacio de números de secuencia según van

formando segmentos e introduciéndolos en la cola de salida de red en un 'host' de origen. La detección de duplicados y el algoritmo de

secuenciación del protocolo de TCP se apoya en la asociación única de datos del segmento a un espacio de secuencias hasta el punto de que los números de secuencia no pasarán por todos los 2**32 valores antes de que los datos del segmento asociados a aquellos números de secuencia hayan sido entregados y confirmados en su entrada por el receptor y que todas las copias duplicadas hayan sido "drenadas" del entorno internet. Si no se asumiera esto, dos segmentos TCP distintos muy bien podrían tener asignados los mismos, o coincidentes en parte, números de secuencia, causando confusión al receptor sobre cuáles son los datos viejos y cuáles los nuevos. Recuérdese que cada segmento se asocia con tantos números de secuencia como octetos tenga el segmento.

Bajo condiciones normales, los TCP llevan la cuenta del siguiente número de secuencia para emitir y del último acuse de recibo que se espera para evitar utilizar otra vez de forma equivocada un número de secuencia antes de que la entrega de su primer uso haya sido confirmada. Esto por sí solo no garantiza que los datos duplicados antiguos desaparezcan de internet, por ello el espacio de secuencia se ha definido muy grande para reducir la probabilidad de que un duplicado errante pueda causar problemas en su llegada. A 2 megabits/seg. lleva 4.5 horas el utilizar todos los 2**32 octetos del espacio de secuencias. Como la vida máxima de un segmento en la red probablemente no supere unas pocas decenas de segundos, se piensa que esto es suficiente protección para futuras redes, incluso si las transferencias de datos escalan hasta varias decenas de megabits/seg. A 100 megabits/seg, el tiempo de una vuelta es de 5,4 minutos, lo que puede ser un poco corto, pero todavía dentro de lo razonable.

El algoritmo básico de detección de duplicados y de secuenciación de TCP puede confundirse, sin embargo, si un TCP de origen no mantiene ninguna memoria de los números de secuencia que utilizó en su última conexión. Por ejemplo, si el módulo TCP comenzara todas las conexiones con el número de secuencia 0, entonces después de una caída y un reinicio, un TCP podría volver a formar una conexión anterior (posiblemente después de una resolución de la conexión medio abierta) y emitir paquetes con números de secuencia idénticos o coincidentes en parte con los paquetes que todavía circulan por la red y que fueron emitidos en la anterior encarnación de la misma conexión. En ausencia de cualquier conocimiento de los números de secuencia empleados en una conexión concreta, la especificación de TCP recomienda que el origen retrase la emisión de cualquier segmento en la conexión durante MSL segundos, para dejar tiempo suficiente a los segmentos de la anterior encarnación de la conexión a que desaparezcan del sistema.

Incluso los 'hosts' que puedan recordar la hora del día y utilizarla para seleccionar los valores de los números de secuencia no son inmunes a este problema (es decir, incluso si la hora del día se utilizara para seleccionar un número de secuencia inicial en cada nueva encarnación de la conexión).

Supóngase, por ejemplo, que se abre una conexión con un número de secuencia S. Supóngase que no se utiliza mucho esta conexión y que eventualmente la función que da el número de secuencia inicial a partir de la hora (ISN(t)) toma un valor igual al número de secuencia, por ejemplo S1, del último segmento enviado por este TCP en una conexión concreta. Ahora supóngase que, en este instante, el 'host' se cuelga, se recupera y establece una nueva encarnación de la conexión. El número de secuencia inicial elegido es S1 = ISN(t) -- ¡ el último número de secuencia utilizado en la anterior encarnación de la conexión ! Si la recuperación sucede de forma suficientemente rápida, cualquier anterior duplicado en la red que transportan números de secuencia próximos a S1 pueden llegar y ser tratados como paquetes nuevos por el receptor de esta nueva encarnación de la conexión.

El problema consiste en que el 'host' que se recupera puede que no sepa por cuánto tiempo estuvo caído y si todavía quedan viejos duplicados en el sistema pertenecientes a anteriores encarnaciones de la conexión.

Una forma de tratar este problema consite en retrasar de forma deliberada la emisión de segmentos durante un MSL tras recuperarse de una caída - ésta es la especificación del "tiempo en silencio". Los 'hosts' que prefieran evitar esperar están exponiéndose al riesgo de una posible confusión entre viejos y nuevos paquetes. Los fabricantes pueden proporcionar a los usuarios de TCP la posibilidad de seleccionar conexión por conexión si se debe esperar tras una caída o no, o puede que implementen informalmente el "tiempo en silencio" para todas las conexiones. Obviamente, incluso si el usuario elige esperar, esto no es necesario si el 'host' ha estado funcionando durante al menos MSL segundos.

Resumiendo: cada segmento emitido ocupa uno o más números de secuencia del espacio de secuencias, los números utilizados por un segmento están "ocupados" o "en uso" hasta que hayan pasado MSL segundos, tras una caída un cierto bloque de espacio-tiempo es ocupado por los octetos del último segmento emitido, si una nueva conexión empieza demasiado pronto y utiliza cualquiera de los números de secuencia de la huella dejada en el espacio-tiempo por el último segmento de la encarnación previa de la conexión, hay un peligro potencial de que haya una coincidencia en una cierta área de número, lo que podría causar confusión al receptor.

3.4. Establecemiento de una conexión

El procedimiento de acuerdo en tres pasos se utiliza para establecer una conexión. Normalmente, este procedimiento se inicia por un TCP y responde otro TCP distinto. El procedimiento también funciona si dos TCP simultáneamente inician el procedimiento. En el caso de intentos simultáneos, cada TCP recibe un segmento 'SYN', que no lleva acuse de recibo, tras haber envíado su 'SYN'. Por supuesto, la llegada de un segmento 'SYN' anterior duplicado puede, potencialmente, hacer creer al receptor que está en progreso una iniciación simultánea de conexión. Un uso adecuado de los segmentos de tipo 'reset' puede eliminar la ambigüedad de estos casos.

A continuación, se dan algunos ejemplos de iniciación de conexión. Aunque estos ejemplos no muestren la sincronización de la conexión con segmentos que transporten datos, esto último es algo perfectamente legítimo, siempre y cuando el TCP receptor no entregue los datos al usuario hasta que no esté claro que los datos son válidos (es decir, los datos deben permanecer en los búferes del receptor hasta que la conexión alcance el estado ESTABLISHED o de conexión establecida). El acuerdo en tres pasos reduce la posibilidad de una conexión falsa. Es la implementación de la negociación entre la memoria y los mensajes quien debe proporcionar la información para esta comprobación.

El acuerdo en tres pasos más simple posible se muestra en la figura 7 de más abajo. Las figuras deben ser interpretadas de la siguiente manera. Cada línea se numera para referencias futuras. Las flechas hacia la derecha (-->) indican la salida de un segmento TCP desde el TCP A dirigido hacia el TCP B, o la llegada de un segmento en B proveniente de A. Las flechas hacia la izquierda (<--), indican lo contrario. Los puntos suspensivos (...) indican un segmento que todavía está en la red (retrasado). "XXX" indica un segmento que se perdió o fue rechazado. Los comentarios aparecen en paréntesis. Los estados de TCP representan el estado DESPUÉS de la salida o llegada del segmento (del que se muestra su contenido en el centro de cada línea). El contenido del segmento se muestra de forma abreviada, con el número de secuencia, los indicadores de control y el valor del campo de número de acuse de recibo (ACK). Otros campos tales como la ventana, direcciones, longitudes y el texto han sido omitidos en aras de la claridad.

```
     TCP A                                                TCP B

1.   CLOSED                                               LISTEN

2.   SYN-SENT    --> <SEQ=100><CTL=SYN>               --> SYN-RECEIVED

3.   ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK>  <-- SYN-RECEIVED

4.   ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK>       --> ESTABLISHED

5.   ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK><DATOS>--> ESTABLISHED
```

Acuerdo en 3 pasos básico de
sincronización de la conexión

Figura 7.

En la línea 2 de la figura 7, el TCP A comienza enviando un segmento SYN que además indica que va a utilizar números de secuencia comenzando por el número 100. En la línea 3, el TCP B envía un SYN confirmando la recepción del SYN que le envió el TCP A. Nótese que el campo de acuse de recibo indica que el TCP B está esperando recibir el 101 de la secuencia, confirmado la recepción del SYN que ocupó el lugar 100 de la secuencia.

En la línea 4, el TCP A responde con un segmento vacío conteniendo un ACK para el SYN del TCP B; y en la línea 5, el TCP A envía algunos datos. Nótese que el número de secuencia del segmento en la línea 5 es el mismo que el de la línea 4 porque el ACK no consume un número del espacio de secuencias (si se hiciera, ¡se estaría abocado a confirmar segmentos ACK!

La iniciación simultánea es sólo un poco más compleja, como se muestra en la figura 8. Cada TCP pasa de CLOSED a SYN-SENT, luego a SYN-RECEIVED y por último a ESTABLISHED.

```
     TCP A                                                TCP B

  1. CLOSED                                               CLOSED

  2. SYN-SENT     --> <SEQ=100><CTL=SYN>                  ...

  3. SYN-RECEIVED <-- <SEQ=300><CTL=SYN>                  <-- SYN-SENT

  4.               ... <SEQ=100><CTL=SYN>                 --> SYN-RECEIVED

  5. SYN-RECEIVED --> <SEQ=100><ACK=301><CTL=SYN,ACK>     ...

  6. ESTABLISHED  <-- <SEQ=300><ACK=101><CTL=SYN,ACK>     <-- SYN-RECEIVED

  7.               ... <SEQ=101><ACK=301><CTL=ACK>        --> ESTABLISHED
```

Sincronización simultánea de la conexión

Figura 8.

El principal motivo del uso del acuerdo de tres pasos es evitar que las iniciaciones de conexiones duplicadas anteriores causen confusión. Para tratar esto, se ha previsto un mensaje de control especial, el de reinicio o 'reset'. Si el TCP receptor está en un estado no sincronizado (i.e., SYN-SENT, SYN-RECEIVED), vuelve al estado LISTEN tras la recepción de un "reset" aceptable. Si el TCP está en uno de los estados sincronizados (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING. LAST-ACK, TIME-WAIT), corta abruptamente la conexión e informa de ello a su usuario. Se discutirá este último caso en la sección dedicada a las conexiones "medio abiertas" ('half-open') de más adelante.

```
     TCP A                                                TCP B

  1. CLOSED                                               LISTEN

  2. SYN-SENT    --> <SEQ=100><CTL=SYN>               ...

  3. (duplicado) ... <SEQ=90><CTL=SYN>                --> SYN-RECEIVED

  4. SYN-SENT    <-- <SEQ=300><ACK=91><CTL=SYN,ACK>   <-- SYN-RECEIVED

  5. SYN-SENT    --> <SEQ=91><CTL=RST>                --> LISTEN

  6.             ... <SEQ=100><CTL=SYN>               --> SYN-RECEIVED

  7. SYN-SENT    <-- <SEQ=400><ACK=101><CTL=SYN,ACK>  <-- SYN-RECEIVED

  8. ESTABLISHED --> <SEQ=101><ACK=401><CTL=ACK>      --> ESTABLISHED
```

Recuperación ante un SYN duplicado anterior

Figura 9.

Como ejemplo simple de una recuperación ante la presencia de duplicados anteriores, considérese la figura 9. En la línea 3, un SYN duplicado anterior llega al TCP B. El TCP B no puede discernir si se trata de un duplicado anterior, y por tanto responde de forma normal (línea 4). El TCP A detecta que el campo ACK es incorrecto y devuelve un RST ("reset") con su campo SEQ elegido de tal forma que el segmento sea creíble. El TCP B, al recibir el RST, vuelve al estado LISTEN. Cuando el SYN original (N.T: en el original inglés aquí aparece entre

paréntesis la expresión "juego de palabras intencionado", ya que 'original SYN' se pronuncia igual en inglés que "pecado original") llega finalmente al otro extremo en la línea 6, la sincronización procede de forma normal. Si el SYN de la línea 5 hubiera llegado antes que el RST, podría haberse dado un intercambio más complejo con RST enviados en ambas direcciones.

Conexiones "medio abiertas" y otras anomalías

Una conexión establecida se dice que está "medio abierta" ('half-open') si uno de los TCP ha cerrado o interrumpido la conexión en su extremo sin avisar a la otra parte, o si los dos extremos de la conexión han quedado desincronizados por culpa de una caída que causó pérdidas de memoria. Tales conexiones se reiniciarán automáticamente (mediante un "reset") si hay un intento de enviar datos en cualquier sentido. Sin embargo, es de esperar que las conexiones medio abiertas se den con poca frecuencia, y además el procedimiento de recuperación no es excesivamente complicado.

Si en el extremo A la conexión deja de existir, entonces un intento

por parte del usuario en el extremo B de envíar datos sobre dicha conexión causará que el TCP receptor del extremo B reciba un mensaje de control de reinicio (un 'reset'). Este tipo de mensaje indica al TCP de B que algo va mal, y que se espera de él que interrumpa la conexión.

Asúmase que dos procesos de usuario A y B están comunicándose entre sí cuando una caída ocurre causando pérdidas de memoria en el TCP de A. Dependiendo del sistema operativo que dé soporte al TCP de A, es probable que exista algún mecanismo de recuperación ante errores. Cuando el TCP esté funcionando otra vez, es probable que A empiece de nuevo desde el comienzo o desde un punto de recuperación. Como resultado, A intentará probablemente abrir la conexión de nuevo mediante una llamada OPEN o intentará llamadas SEND sobre la conexión que A considera abierta. En el último caso, recibirá el mensaje de error "conexión no abierta" proveniente del TCP local (de A). En un intento de establecer la conexión, el TCP de A enviára un segmento conteniendo un SYN. Este escenario conduce al ejemplo mostrado en la figura 10. Después de que el TCP de A se cuelga, el usuario intenta reabrir la conexión. El TCP B, mientras tanto, piensa que la conexión sigue abierta.

	TCP de A		TCP de B
1.	(CAÍDA)	(enviar el 300, recibir el 100)	
2.	CLOSED		ESTABLISHED
3.	SYN-SENT --> <SEQ=400><CTL=SYN>	-->	(??)

```
4.  (!!)     <-- <SEQ=300><ACK=100><CTL=ACK>     <-- ESTABLISHED

5.  SYN-SENT --> <SEQ=100><CTL=RST>              --> (¡¡Interrumpir!!)

6.  SYN-SENT                                        CLOSED

7.  SYN-SENT --> <SEQ=400><CTL=SYN>              -->
```

Descubrimiento de una conexión "medio abierta"

Figura 10.

Cuando, en la línea 3, el SYN llega, el TCP de B, estando en un estado sincronizado y el segmento entrante siendo mayor que la ventana, responderá con un acuse de recibo indicando qué número de secuencia espera recibir a continuación (ACK 100). El TCP de A aprecia que este segmento no confirma nada envíado y, estando desincronizado, envía un 'reset' (RST) porque ha detectado que la conexión está medio-abierta. En la línea 5, el TCP de B interrumpe la conexión. EL TCP de A continuará intentando reestablecer la conexión; el problema queda reducido ahora al acuerdo en tres pasos básico descrito en la figura 7.

Un caso alternativo interesante se da si el TCP de A se cuelga y el TCP de B intenta envíar datos sobre lo que él piensa es una conexión sincronizada. Esto se ilustra en la figura 11. En este caso, los datos llegando al TCP de A provenientes del TCP de B (línea 2) no son aceptables porque no existe tal conexión, por tanto el TCP de A envía un RST. El RST es aceptable, por tanto el TCP B lo procesa e interrumpe la conexión.

```
      TCP de A                                          TCP de B

1.  (CAIDA)                              (envíar el 300,recibir el 100)

2.  (??)    <-- <SEQ=300><ACK=100><DATA=10><CTL=ACK> <-- ESTABLISHED

3.          --> <SEQ=100><CTL=RST>                --> (¡¡INTERRUMPIR!!)
```

Un extremo activo causa el descubrimiento de una conexión "medio abierta"

Figura 11.

En la figura 12, encontramos los TCP de A y de B ambos con conexiones pasivas esperando un SYN. Un duplicado anterior llega al TCP de B (línea 2) y provoca que B entre en acción. Le devuelve un SYN-ACK (línea 3) lo que causa que el TCP de A genere un RST (pues el ACK de la línea 3 no es aceptable). El TCP de B acepta el "reset" y vuelve a

su estado pasivo LISTEN.

```
    TCP A                                                TCP B

1.  LISTEN                                               LISTEN

2.       ... <SEQ=Z><CTL=SYN>                  -->  SYN-RECEIVED

3.  (??) <-- <SEQ=X><ACK=Z+1><CTL=SYN,ACK>     <--  SYN-RECEIVED

4.       --> <SEQ=Z+1><CTL=RST>                -->  (vuelve a LISTEN!)

5.  LISTEN                                               LISTEN
```

Un SYN anterior duplicado inicia un 'reset' en dos conectores pasivos

Figura 12.

Otros muchos casos son posibles, estando todos ellos contemplados por la siguientes reglas de generación y proceso de segmentos RST.

Generación de reinicios ('resets')

Como regla general, se debe envíar un 'reset' (RST) siempre que llegue un segmento que aparentemente no esté destinado para la conexión actual. No debe enviarse un 'reset' si no está claro que éste sea el caso.

Hay tres grupos de estados:

1. Si la conexión no existe (estado CLOSED) entonces se envía un 'reset' como respuesta a cualquier segmento entrante excepto si es otro 'reset'. En particular, los SYN enviados a una conexión no existente serán rechazados de esta forma.

   Si en el segmento entrante tiene significado el campo ACK, el 'reset' escoge como su número de secuencia el valor del campo ACK del segmento, en otro caso el número de secuencia del 'reset' tiene valor cero y el campo ACK se establece a la suma del número de secuencia y la longitud del segmento entrante. La conexión permanece en el estado CLOSED.

2. Si la conexión está en cualquier estado no sincronizado (LISTEN, SYN-SENT, SYN-RECEIVED), y el segmento entrante confirma un número

todavía no enviado (el segmento lleva un ACK inaceptable), o si un segmento entrante tiene un nivel de seguridad o compartimentación que no concuerda exactamente con el solicitado para la conexión, entonces, se envía un 'reset'.

Si nuestro SYN no ha sido confirmado y el nivel de prioridad del segmento entrante es más alto que el nivel de prioridad solicitado entonces o se aumenta el nivel de prioridad local (si esto está permitido por el usuario y el sistema) o se envía un 'reset'; o si el nivel de prioridad del segmento entrante es menor que el nivel de prioridad solicitado entonces se continúa como si los niveles de prioridad hubieran concordado exactamente (si el TCP remoto no puede elevar el nivel de prioridad para concordar el nuestro sería detectado en el próximo segmento que enviara, y la conexión se finalizaría entonces). Si nuestro SYN ha sido confirmado (quizás en este mismo segmento entrante), el nivel de prioridad del segmento entrante debe concordar exactamente con el nivel de prioridad local, si no es así, se envía un 'reset'.

Si el segmento entrante tiene un campo ACK, el 'reset' escoge su número de secuencia del campo ACK del segmento, en otro caso el "reset" tiene como número de secuencia cero y el campo ACK se establece a la suma del número de secuencia y la longitud del segmento entrante. La conexión permanece en el mismo estado.

3. Si la conexión está en un estado sincronizado (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), cualquier segmento inaceptable (con un número de secuencia fuera de la ventana o con un número de acuse de recibo inaceptable) debe

desencadenar únicamente el envío de un segmento de acuse de recibo vacío de datos, que llevaría el actual número de secuencia y un valor de acuse de recibo indicando el siguiente número de secuencia que se espera recibir; entonces la conexión permanece en el mismo estado.

Si el segmento entrante tiene un nivel de seguridad, o compartimentación, o prioridad que no concuerda exactamente con los solicitados para la conexión, entonces se envía un 'reset' y la conexión pasa al estado CLOSED. El 'reset' escoge como su número de secuencia el valor del campo ACK del segmento entrante.

Procesamiento de 'resets'.

En todos los estados exceptuando SYN-SENT, todos los segmentos de tipo 'reset' (RST) son validados comprobando sus campos SEQ. Un 'reset' es válido si su número de secuencia está dentro de la ventana. En el estado SYN-SENT (con un RST recibido en respuesta a un SYN inicial), el RST es aceptable si el campo ACK confirma el SYN.

El receptor de un RST primero lo comprueba, entonces cambia de estado. Si el receptor estaba en el estado LISTEN, lo ignora. Si el receptor estaba en el estado SYN-RECEIVED y estaba previamente en el estado LISTEN, entonces el receptor vuelve al estado LISTEN, en cualquier otro caso el receptor interrumpe la conexión y pasa al estado CLOSED. Si el receptor, estaba en cualquier otro estado, interrumple la conexión, avisa de ello al usuario y pasa al estado CLOSED.

3.5. Cierre de una conexión

CLOSE (N.T.: "cerrar" en inglés) es una operación que significa "no tengo más datos que enviar". La noción de cerrar una conexión bidireccional tiene una interpretación ambigua, desde luego, dado que puede no resultar obvio como tratar a la parte receptora de la conexión. Se ha elegido tratar a CLOSE de una forma simple. El usuario que hace la llamada CLOSE puede continuar recibiendo mediante llamadas RECEIVE hasta que se le indique que el otro lado ha cerrado también la conexión mediante otro CLOSE. Por tanto, un programa puede iniciar varias llamadas SEND seguidas por una llamada CLOSE y continuar recibiendo (con RECEIVE) hasta que se le indique que falló una llamada RECEIVE debido a que su interlocutor cerró la conexión con un CLOSE. Se asume que TCP idicará al usuario que el otro lado de la conexión ha cerrado ésta incluso si no hay llamadas RECEIVE pendientes, de forma que pueda cerrar su conexión limpiamente. Una conexión TCP entregará de forma fiable todos los búferes enviados en llamadas SENT antes de que la conexión sea cerrada, de forma que un usuario que no espera datos de respuesta sólo necesite esperar a que la conexión sea cerrada con éxito para saber que todos sus datos fueron recibidos en el TCP de destino. Los usuarios deben mantenerse leyendo en las conexiones que cierren para envío hasta que TCP notifique que no hay más datos.

Existen fundamentalmente tres casos:

1) El usuario inicia el cierre de la conexión enviando la llamada CLOSE a TCP.

2) El TCP remoto inicia el cierre enviando una señal de control FIN.

3) Ambos usuarios realizan la llamada CLOSE simultáneamente.

Caso 1: el usuario local inicia el cierre

En este caso se puede construir un segmento FIN y ponerlo en la cola de salida de segmentos. TCP no aceptará más llamadas SEND y entrará en el estado FIN-WAIT-1. En este estado se permiten las llamadas RECEIVE. Todos los segmentos que preceden al FIN, incluído éste, serán retransmitidos hasta que se reciban los correspondientes acuses de recibo. Cuando el TCP remoto haya realizado el acuse de

recibo del FIN y enviado su propio FIN, el TCP local puede realizar el acuse de recibo de este FIN. Nótese que un TCP que recibe un FIN enviará su ACK pero no enviará su propio FIN hasta que su usuario haya cerrado también la conexión con un CLOSE.

Caso 2: TCP recibe un FIN desde la red

Si llega un FIN no solicitado desde la red, el TCP receptor puede responder con un ACK y advertir al usuario de que la conexión se está cerrando. El usuario responderá con una llamada CLOSE, ante la cual TCP puede enviar un FIN al otro TCP tras enviar los datos restantes. Entonces TCP espera hasta el acuse de recibo de su propio FIN y elimina la conexión. Si el ACK no llega en el tiempo de espera del usuario, la conexión se aborta y así se notifica al usuario.

Caso 3: ambos usuarios cierran simultáneamente

El cierre simultáneo a ambos lados de la conexión causa el intercambio de segmentos FIN. Cuando todos los segmentos que preceden a los FIN se han procesado y confirmado con acuses de recibo, cada TCP puede responder con un ACK el FIN que ha recibido. Ambos, ante la recepción de los ACK, eliminarán la conexión.


```
     TCP A                                                TCP B

  1. ESTABLISHED                                          ESTABLISHED

  2. (Close)
     FIN-WAIT-1  --> <SEQ=100><ACK=300><CTL=FIN,ACK>  --> CLOSE-WAIT

  3. FIN-WAIT-2  <-- <SEQ=300><ACK=101><CTL=ACK>      <-- CLOSE-WAIT

  4.                                                      (Close)
     TIME-WAIT   <-- <SEQ=300><ACK=101><CTL=FIN,ACK>  <-- LAST-ACK

  5. TIME-WAIT   --> <SEQ=101><ACK=301><CTL=ACK>      --> CLOSED
```

6. (2 MSL)
   CLOSED

Secuencia de cierre normal

Figura 13.

```
    TCP A                                                TCP B

1.  ESTABLISHED                                          ESTABLISHED

2.  (Close)                                              (Close)
    FIN-WAIT-1  --> <SEQ=100><ACK=300><CTL=FIN,ACK>  ... FIN-WAIT-1
                <-- <SEQ=300><ACK=100><CTL=FIN,ACK>  <--
                ... <SEQ=100><ACK=300><CTL=FIN,ACK>  -->

3.  CLOSING     --> <SEQ=101><ACK=301><CTL=ACK>      ... CLOSING
                <-- <SEQ=301><ACK=101><CTL=ACK>      <--
                ... <SEQ=101><ACK=301><CTL=ACK>      -->

4.  TIME-WAIT                                            TIME-WAIT
    (2 MSL)                                              (2 MSL)
    CLOSED                                               CLOSED
```

Secuencia de cierre simultáneo

Figura 14.

3.6. Prioridad y seguridad

La intención es que la conexión sólo se permita entre puertos que operen exactamente con los mismos valores de seguridad y compartimentación y con un nivel de prioridad igual al mayor de los solicitados por ambos puertos.

Los parámetros de prioridad y seguridad usados en TCP son exactamente los definidos en el protocolo de internet (IP) [2]. A lo largo de esta especificación de TCP el término "seguridad/compartimentación" indica los parámetros de seguridad utilizados por IP que incluye los de

seguridad, compartimentación, grupo de usuario y manejo de restricciones.

Un intento de conexión con valores de seguridad/compartimentación erróneos o con un nivel de prioridad menor debe ser rechazado enviando un 'reset'. El rechazo de una conexión debido a un nivel de prioridad demasiado bajo ocurre solamente tras la recepción del acuse de recibo del SYN.

Nótese que los módulos de TCP que operan únicamente al nivel de prioridad por defecto deberán comprobar áun así la prioridad de los

segmentos entrantes y posiblemente elevar el nivel de prioridad que usen en la conexión.

Los parámetros de seguridad pueden ser utilizados incluso en un entorno inseguro (los valores indicarían datos reservados secretos), por lo que las máquinas en entornos inseguros deben estar preparadas para recibir los parámetros de seguridad, aunque no es obligatorio que los envíen.

3.7. Comunicación de datos

Una vez establecida la conexión los datos se transmiten mediante el intercambio de segmentos. Dado que los segmentos pueden perderse debido a errores (fallo en la suma de comprobación) o congestión de la red, TCP utiliza la retransmisión (tras un tiempo de espera) para asegurar la entrega de cada segmento. Pueden llegar segmentos duplicados debido a la red o a la retransmisión de TCP. Tal y como se explica en la sección sobre números de secuencia, TCP realiza ciertas comprobaciones sobre los números de secuencia y de acuse de recibo en los segmentos para verificar su admisibilidad.

El emisor de los datos mantiene un registro del próximo número de secuencia a usar en la variable SND.NXT. El receptor de los datos mantiene un registro del siguiente número de secuencia esperado en la variable RCV.NXT. El emisor mantiene también un registro del número de secuencia sin acuse de recibo menor en la variable SND.UNA. Si el flujo de datos queda momentáneamente inactivo y de todos los datos enviados se tiene acuse de recibo entonces las tres variables serán idénticas.

Cuando el emisor crea un segmento y lo transmite, incrementa SND.NXT. Cuando el receptor acepta un segmento incrementa RCV.NXT y envía un acuse de recibo. Cuando el emisor de los datos recibe un acuse de recibo incrementa SND.UNA. La diferencia entre los valores de estas variables es una medida del retardo en la comunicación. El valor en el cual se incrementan estas variables es el de la longitud de los datos del segmento. Nótese que, una vez en el estado ESTABLISHED todos los segmentos deben llevar la información del acuse de recibo actualizada.

La llamada de usuario CLOSE implica la función de entrega inmediata 'push', tal y como sucede con el indicador de control FIN en un

segmento entrante.

Tiempo de espera de retransmisión

Debido a la variabilidad de las redes que componen un sistema de redes de internet y la gran cantidad de casos de conexiones TCP el tiempo de espera de retransmisión se debe determinar dinámicamente. Se ilustra a

continuación un procedimiento para determinar un tiempo de espera de retransmisión.

Un ejemplo de procedimiento de tiempo de espera de retransmisión

Mídase el tiempo transcurrido entre el envio de un octeto de datos con un número de secuencia determinado y la recepción de un acuse de recibo que incluya ese número de secuencia (los segmentos enviados no tienen por qué concordar con los segmentos recibidos). Este tiempo medido es el "tiempo de ida y vuelta ('Round Trip Time' o RTT). Calcúlese después el "tiempo de ida y vuelta suavizado" ('Smoothed Round Trip Time' o SRTT) como (N.T.: esta fórmula aparece exactamente así en el original):

```
SRTT = ( ALPHA * SRTT ) + ((1-ALPHA) * RTT)
```

y basándose en este, calcúlese el tiempo de espera de retransmisión (RTO) como:

```
RTO = min[COTASUP,max[COTAINF,(BETA*SRTT)]]
```

donde COTASUP es una cota superior del tiempo de espera (i.e., 1 minuto), COTAINF es una cota inferior del tiempo de espera (i.e., 1 segundo), ALPHA es un factor de suavizado (i.e., entre 0,8 y 0,9) y BETA es un factor de varianza del retardo (i.e., entre 1,3 y 2,0).

La comunicación de información urgente

El objetivo del mecanismo de urgencia de TCP consiste en permitir al usuario emisor la posibilidad de estimular al usuario receptor para que acepte ciertos datos urgentes y en permitir al TCP receptor que indique al usuario receptor cuándo se le han entregado todos los datos urgentes conocidos hasta el momento.

Este mecanismo permite elegir un punto en el flujo de datos como el final de la información urgente. Siempre que este punto esté más adelante del número de secuencia de recepción (RCV.NXT) en el TCP receptor, éste debe avisar al usuario para que cambie al "modo urgente"; cuando el número de secuencia de recepción alcance el puntero urgente, TCP debe avisar al usuario que cambie al "modo normal". Si el puntero urgente se actualiza mientras el usuario está en el "modo urgente", dicha actualización no será visible para el usuario.

El método emplea un campo urgente em todos los segmentos transmitidos. El indicador de control URG indica que el campo urgente tiene significado y debe ser añadido al número de secuencia de segmento para obtener el puntero urgente. La ausencia de este indicador denota la ausencia de datos urgentes.

Para enviar una indicación de urgencia el usuario debe enviar también al menos un octeto de datos. Si el usuario emisor indica también una llamada 'push', entonces se acelera la entrega de la información urgente al proceso de destino.

Manejo de la ventana

La ventana, que se envia en cada segmento, indica el rango de números de secuencia que el emisor de la ventana (el receptor de los datos) está preparado para aceptar en ese momento. Se supone que normalmente este rango está relacionado con el espacio de almacenamiento de datos disponible en ese momento para la conexión.

Indicar una ventana grande favorece las transmisiones. Si llegan más datos de los que pueden ser aceptados, serán descartados. Esto resultará en un exceso de retransmisiones, añadiendo una carga innecesaria a la red y a los módulos de TCP. Indicar una ventana pequeña puede restringir la transmisión de datos hasta el punto de introducir un retardo de ida y vuelta entre cada nuevo segmento transmitido.

Los mecanismos proporcionados permiten a un TCP anunciar una ventana grande y seguidamente anunciar una mucho menor sin tener que aceptar todos los datos. Esta práctica, conocida como "reducción la ventana", no se recomienda bajo ningún concepto. El principio de robustez dicta que cada módulo de TCP no reducirá la ventana por sí mismo, pero sí que deben estar preparados para esperar este comportamiento por parte de otros TCP.

El TCP emisor debe estar preparado para aceptar datos del usuario y enviar al menos un octeto de nuevos datos, incluso si el tamaño de la ventana de envío es cero. El TCP emisor debe retransmitir regularmente al receptor TCP incluso cuando el tamaño de la ventana es cero. Cuando el tamaño de la ventana es cero, se recomienda un intervalo de retransmisión de dos minutos. Esta retransmisión es esencial para garantizar que, siempre que uno de los TCP tenga una ventana de tamaño cero, la reapertura de la ventana sea notificada de forma fiable al otro.

Cuando el TCP receptor tiene una ventana de tamaño cero y llega un segmento debe todavía enviar un acuse de recibo con su próximo número de secuencia esperado y su tamaño de ventana actual (cero).

El TCP emisor empaqueta los datos para transmitir en segmentos que caben en la ventana actual y puede que reempaquete segmentos de la cola de retransmisión. Este reempaquetamiento no es obligatorio, pero

puede resultar útil.

En una conexión con un flujo de datos de una sola dirección, la información de la ventana se transmite en segmentos de acuse de recibo, todos con el mismo número de secuencia, de manera que no hay forma de reordenarlos si llegan desordenados. Este no es un problema serio, pero permite que en ocasiones la información de ventana esté basada en información anticuada del receptor. Un refinamiento para evitar este problema consiste en escoger la información de ventana de los segmentos con el número de acuse de recibo más alto (es decir, los segmentos con número de acuse de recibo igual o mayor que el más alto recibido hasta el momento).

El procedimiento de gestión de ventana tiene una influencia significativa sobre el rendimiento de la comunicación. Los comentarios siguientes son sugerencias para los implementadores.

Sugerencias sobre la gestión de la ventana

Disponer de una ventana muy pequeña provoca que los datos sean transmitidos en muchos segmentos pequeños, cuando se puede conseguir un mejor rendimiento usando menos segmentos de mayor tamaño.

Una sugerencia para evitar las ventanas pequeñas es que el receptor retrase la actualización de una ventana hasta que el tamaño disponible sea al menos un X por ciento del tamaño máximo posible para la conexión (donde X podría estar entre 20 y 40).

Otra sugerencia consiste en que el emisor evite el envío de pequeños segmentos esperando hasta que la ventana sea lo bastante grande para enviar los datos. Si el usuario indica una función de entrega inmediata 'push' entonces los datos deberán ser enviados incluso si se trata de un segmento pequeño.

Nótese que los acuses de recibo no deben retrasarse o se producirán retransmisiones innecesarias. Una estrategia podría consistir en enviar un acuse de recibo cuando llega un segmento pequeño (sin actualizar la información de ventana), y luego enviar otro acuse de recibo con la nueva información de ventana cuando ésta sea mayor.

El segmento que se envía para sondear una ventana de tamaño cero puede también dar comienzo a un particionado de los datos transmitidos en segmentos más y más pequeños. Si un segmento que contiene un único octeto de datos enviado para sondear una ventana de tamaño cero se acepta, entonces se consume un octeto de la ventana ahora disponible. Si el TCP emisor simplemente envía todo lo que puede como cuando el tamaño de la ventana es distinto de cero, los datos transmitidos serán partidos en segmentos alternativamente grandes y pequeños. Conforme avance el tiempo, las pausas ocasionales en el receptor para permitir el ajuste de

la ventana producirán una partición de cada segmento grande en uno pequeño y otro no tan grande como el original. Después de un rato la transmisión de datos se producirá en su mayor parte en segmentos pequeños.

La sugerencia aquí es que las implementaciones de TCP tienen que intentar, de forma activa, combinar las ventanas pequeñas en ventanas de mayor tamaño, dado que en las implementaciones simples, los mecanismos de gestión de ventana tienden a producir muchas ventanas pequeñas.

3.8. Interfaces

Existen, por supuesto, dos interfaces de interés: la interfaz usuario/TCP y la interfaz TCP/nivel-inferior. Tenemos un modelo completamente elaborado de la interfaz usuario/TCP, pero dejamos sin especificar aquí la interfaz con el protocolo de nivel inferior, ya que estará especificado en detalle en la especificación del protocolo de nivel inferior. Para el caso en que el protocolo de nivel inferior sea IP, apuntamos algunos de los valores de los parámetros que los TCP pueden utilizar.

Interfaz usuario/TCP

La siguiente descripción funcional de las órdenes de usuario al módulo de TCP es, en el mejor de los casos, ficticia, dado que cada sistema operativo dispondrá de distintos recursos. En consecuencia, debemos advertir al lector que diferentes implementaciones de TCP pueden tener diferentes interfaces de usuario. Sin embargo, todos los TCP deben proporcionar un cierto conjunto mínimo de servicios para garantizar que todas las implementaciones TCP pueden soportar la misma jerarquía de protocolo. Esta sección especifica las interfaces funcionales obligatorias para todas las implementaciones de TCP.

Ordenes de usuario de TCP

Las siguientes secciones caracterizan de forma funcional una interfaz USUARIO/TCP. La notación utilizada es similar a la mayoría de las llamadas a procedimiento o función de los lenguajes de alto nivel, lo cual no significa que se excluyan las llamadas de servicio tipo 'trap' (i.e., SVC, UUO, EMT).

Las órdenes de usuario descritas más abajo especifican las funciones básicas que debe ejecutar TCP para soportar la comunicación entre procesos. Las implementaciones individuales deben definir su propio formato exacto, y pueden proporcionar combinaciones o subconjuntos de las funciones básicas en llamadas simples. En particular, algunas implementaciones pueden querer realizar la llamada de apertura OPEN de la conexión

automáticamente en el primer SEND o RECEIVE enviado por el usuario para una conexión dada.

Al proporcionar recursos para la comunicación entre procesos, el TCP no debe limitarse a aceptar órdenes, sino que también debe devolver información al proceso que sirve. Esta información consistirá en:

(a) información general acerca de una conexión (i.e., interrupciones, cierre remoto, enlace a un conector remoto sin especificar).

(b) respuestas a órdenes de usuario específicas indicando éxito o varios tipos de error.

Open (N.T: "abrir" en inglés)

Formato: OPEN (puerto local, conector remoto, 'active'/'passive' [, tiempo de espera] [, prioridad] [, seguridad/compartimentación] [, opciones]) -> nombre de la conexión local

(N.T.: 'active' es "activa" en inglés, 'passive' es "pasiva")

Se asume que el TCP local conoce la identidad de los procesos a los que sirve y que comprobará que el proceso tiene autoridad para utilizar la conexión especificada. Dependiendo de la implementación del TCP, los identificadores del TCP y de la red local serán proporcionados bien por el módulo de TCP o bien por el protocolo de nivel inferior (i.e. IP). Estas consideraciones derivan de cuestiones de seguridad, de forma que ningún TCP pueda hacerse pasar por otro, y así sucesivamente. De forma similar, ningún proceso puede hacerse pasar por otro sin la connivencia del módulo de TCP.

Si el indicador 'active'/'passive' está puesto a 'passive', entonces ésta será una llamada para escuchar (LISTEN) una conexión entrante. Un OPEN pasivo puede tener o bien un conector remoto completamente especificado para esperar una conexión particular o un conector remoto sin especificar para esperar cualquier llamada. Una llamada pasiva completamente especificada puede activarse mediante la ejecución subsecuente de un SEND.

Se crea un bloque de control de transmisión ('Transmision control block' o TCB) y se rellena parcialmente con los datos de los parámetros de la orden OPEN.

En una orden OPEN activa, el TCP iniciará el procedimiento para sincronizar (es decir, establecer) la conexión en una sola vez.

El tiempo de espera, si está presente, permite al llamador aplicar un tiempo de espera para todos los datos enviados por el

TCP. Si los datos no se entregan con éxito en el destino dentro del tiempo de espera, TCP interrumpirá la conexión. El valor por defecto estándar en la actualidad es de 5 minutos.

TCP o algún componente del sistema operativo verificará la autorización de los usuarios para abrir una conexión con los valores de prioridad o seguridad/compartimentación especificados. La ausencia de valores de prioridad o seguridad/compartimentación en la llamada OPEN indica que se deben utilizar los valores por defecto.

TCP aceptará peticiones entrantes como válidas sólamente si la información sobre seguridad/compartimentación es exactamente la misma y sólo si la prioridad es igual o mayor que la prioridad solicitada en la llamada OPEN.

La prioridad en la conexión es el mayor de los valores solicitados en la llamada OPEN y recibidos de la solicitud entrante, y fijado a este valor durante el resto de la conexión. Los implementadores pueden querer dar el control de la negociación de la prioridad al usuario. Por ejemplo, se podría permitir al usuario especificar que la prioridad sea exactamente la misma, o que cualquier intento de elevar la prioridad sea confirmado por el usuario.

TCP devolverá al usuario el nombre de la conexión local. Este nombre puede ser usado después como un atajo para denotar la conexión definida por el par <conector local, conector remoto>.

Send (N.T: "enviar" en inglés)

Formato: SEND (nombre de la conexión local, dirección del búfer, contador de bytes, indicador PUSH, indicador URGENT [, tiempo de espera])

Esta llamada hace que se envíen mediante la conexión indicada los datos contenidos en el búfer de usuario indicado. Si la conexión no se ha abierto, el SEND se considera un error. Puede que algunas implementaciones permitan a los usuarios enviar un SEND primero; en este caso, se efectuará un OPEN automático. Si el proceso llamador no está autorizado a usar esta conexión, se devuelve un error.

Si el indicador PUSH está activado, los datos deben ser transmitidos cuanto antes al receptor, y se activará el indicador PUSH en el último segmento TCP creado a partir del búfer. Si el indicador PUSH no está activado, los datos pueden ser combinados con datos de llamadas SEND subsiguientes en favor de la eficiencia de la transmisión.

Si el indicador URGENT está activado, los segmentos enviados al TCP de destino tendrán un valor en el campo de puntero urgente.

El TCP receptor señalará la condición de urgencia al proceso receptor si el puntero urgente indica que los datos que le preceden no han sido consumidos por el proceso receptor. El propósito de este indicador es estimular al receptor para que procese los datos urgentes e indicar al receptor cúando se han recibido todos los datos urgentes actualmente conocidos. El número de veces que el TCP del usuario emisor señala una condición de urgencia no será necesariamente igual al número de veces que el usuario receptor será notificado de la presencia de datos urgentes.

Si no se ha especificado un conector remoto en la llamada OPEN, pero se establece la conexión (i.e., debido a que una conexión en estado de escucha LISTEN se ha hecho específica debido a la llegada de un segmento remoto al conector local), se envía el búfer indicado al conector remoto implícito. Los usuarios que hacen uso de la llamada OPEN sin especificar un conector remoto pueden usar la llamada SEND sin conocer explícitamente la dirección del conector remoto.

Sin embargo, si se intenta una llamada SEND antes de que el conector remoto se haya vuelto específico, se devolverá un error. Los usuarios pueden utilizar la llamada STATUS para determinar el estado de la conexión. En algunas implementaciones TCP puede notificar al usuario cuándo un conector sin especificar queda fijado.

Si se especifica un tiempo de espera, el tiempo de espera actual para esta conexión se cambia al nuevo valor.

En la implementación más simple, la llamada SEND no devolverá el control al usuario hasta que se complete la transmisión o se supere el tiempo de espera. Sin embargo, este método simple está sujeto a puntos muertos ('deadlocks') (por ejemplo, ambos lados de la conexión podrían intentar realizar operaciones SEND antes de hacer ninguna operación RECEIVE) y ofrece un rendimiento pobre, por lo cual no se recomienda. Una implementación más sofisticada devolverá el control inmediatamente para permitir al proceso ejecutarse concurrentemente con la E/S de red y, además, permitir que múltiples operaciones SEND tengan lugar a la vez. Las operaciones SEND múltiples son atendidas según van llegando, de forma que TCP pondrá en cola aquellas a las que no puede atender inmediatamente.

Hemos asumido implícitamente una interfaz de usuario asíncrona en la cual una llamada SEND provoca más tarde algún tipo de

señal o pseudo-interrupción en el TCP servidor. Una posible alternativa es devolver una respuesta inmediatamente. Por ejemplo, las llamadas SEND podrían devolver un acuse de recibo local inmediato, incluso si el acuse de recibo del segmento enviado no ha sido aún recibido. Se puede asumir, de forma

optimista, que la operación tendrá eventualmente éxito. Si no es así, la conexión se cerrará de todas formas debido a la superación del tiempo de espera. En implementaciones de este tipo (síncronas), existirán todavía algunas señales asíncronas, pero éstas estarán relacionadas con la propia conexión, no con segmentos o búferes específicos.

Con objeto de que el proceso distinga entre las distintas indicaciones de error o éxito correspondientes a las distintas llamadas SEND, puede resultar apropiado devolver la dirección del búfer junto con la respuesta codificada a la solicitud SEND. Las señales de TCP al usuario se discuten más abajo, indicando qué información debe ser devuelta al proceso llamador.

Receive (N.T: "recibir" en inglés)

Formato: RECEIVE (nombre de la conexión local, dirección del búfer, número de bytes) -> número de bytes, indicador 'urgent', indicador 'push'.

Esta orden inicializa un búfer de recepción asociado con la conexión especificada. Si esta orden no viene precedida de una llamada OPEN o el proceso solicitante no está autorizado a usar esta conexión, se devuelve un error.

En la implementación más simple, el control no volverá al programa solicitante hasta que el búfer se llene u ocurra algún error, pero este esquema tiene un alto riesgo de puntos muertos. Una implementación más sofisticada permitiría que varios RECEIVE se ejecutaran a la vez. Estos se irían completando según van llegando los segmentos. Esta estrategia permite incrementar el rendimiento a costa de un esquema más elaborado (posiblemente asíncrono) para notificar al programa solicitante que se ha detectado una llamada de entrega inmediata PUSH o se ha llenado un búfer.

Si llegan datos suficientes como para llenar el búfer antes de que se detecte el PUSH, el indicador PUSH no se activará en la respuesta al RECEIVE. El búfer será llenado con tantos datos como le quepan. Si se detecta un PUSH antes de que el búfer esté lleno, éste será devuelto parcialmente lleno y con el indicador PUSH activado.

Si hay datos urgentes el usuario habrá sido informado tan pronto

como llegaron por medio de una señal de TCP al usuario. El usuario receptor debe entonces estar en 'modo urgente'. Si el indicador URGENT está activado, es que quedan todavía datos urgentes. Si está desactivado, esta llamada a RECEIVE ha devuelto todos los datos urgentes y el usuario puede ahora abandonar el "modo urgente". Nótese que los datos que siguen al puntero urgente (datos no urgentes) no pueden ser entregados al usuario en el mismo búfer con los datos urgentes precedentes a

menos que el límite entre ellos esté claramente marcado para el usuario.

Para distinguir entre varias llamadas RECEIVE pendientes y para tener en cuenta el caso de un búfer no saturado, el código de retorno viene acompañado de un puntero al búfer y un número de bytes que indica la longitud de los datos recibidos.

En algunas implementaciones alternativas, el TCP se encargaría de reservar el espacio de almacenamiento para el búfer, o bien podría compartir un búfer circular con el usuario.

CLOSE

Formato: CLOSE (nombre de la conexión local)

Esta orden cierra la conexión especificada. Si la conexión no está abierta o el proceso solicitante no está autorizado a usar dicha conexión, se devuelve un error. El cierre de conexiones está pensado para que sea una operación limpia en el sentido de que las llamadas SEND pendientes serán transmitidas (y retransmitidas), en la medida en que el control de flujo lo permita, hasta que todas hayan sido servidas. Por tanto, es aceptable enviar varias órdenes SEND seguidas de una llamada CLOSE, y suponer que todos los datos serán enviados a su destino. Debe quedar claro que los usuarios pueden continuar recibiendo por conexiones que se están cerrando, ya que el otro lado de la conexión puede estar intentando transmitir el resto de sus datos. Por tanto, cerrar una conexión significa "no tengo nada más que enviar" pero no "no recibiré nada más". Puede suceder (si el protocolo del nivel de usuario no está bien ideado) que el lado de la conexión que cierra no sea capaz de deshacerse de todos sus datos antes de que se cumpla el tiempo de espera. En este caso, la llamada CLOSE se convierte en una llamada ABORT, y el TCP que cierra la conexión deja de seguir.

El usuario puede cerrar la conexión en cualquier momento bajo su propia iniciativa, o en respuesta a varios avisos del TCP (i.e., ejecutado un cierre remoto, excedido el tiempo de espera de transmisión, destino inaccesible).

Dado que cerrar una conexión requiere la comunicación con el TCP remoto, las conexiones pueden permanecer en el estado de cierre durante un corto espacio de tiempo. Los intentos de reabrir la conexión antes de que el TCP responda a la orden CLOSE darán como resultado respuestas de error.

La orden CLOSE implica la función de entrega inmediata 'push'.

Status (N.T. "estado" en inglés)

Formato: STATUS (nombre de la conexión local) -> datos de estado

Esta orden de usuario depende de la implementación y puede excluírse sin efectos adversos. La información devuelta provendrá típicamente del TCB asociado con la conexión.

Esta orden devuelve un bloque de datos que contiene la siguiente información:
- conector local,
- conector remoto,
- nombre de la conexión local,
- ventana de recepción,
- ventana de envío,
- estado de la conexión,
- número de búferes en espera del acuse de recibo,
- número de búferes pendientes de recepción,
- estado urgente,
- prioridad,
- seguridad/compartimentación,
- y tiempo de espera de transmisión

Dependiendo del estado de la conexión, o de la implementación misma, parte de esta información puede no estar disponible o no tener ningún significado útil. Si el proceso solicitante no está autorizado a utilizar esta conexión se devuelve un error. Esto evita que los procesos no autorizados puedan obtener información sobre una conexión.

ABORT

Formato: ABORT (nombre de la conexión local)

Esta orden aborta la ejecución de todas las órdenes SEND y RECEIVE pendientes, elimina el TCB y envía un mensaje especial RESET al otro lado de la conexión. Dependiendo de la implementación, los usuarios puede recibir avisos de abandono

por cada orden SEND o RECEIVE pendiente, o simplemente recibir una confirmación de la ejecución de la orden ABORT.

Mensajes de TCP a usuario

Se da por supuesto que el sistema operativo proporciona algún medio para que TCP pueda enviar una señal asíncrona al programa de usuario. Cuando TCP envía una señal a un programa de usuario pasa cierta información al usuario. A menudo en la especificación la información será un mensaje de error. En otros casos habrá información relativa a la finalización del procesamiento de una orden SEND o RECEIVE o cualquier otra llamada del usuario.

Se proporciona la siguiente información:

Nombre de la conexión local	Siempre
Texto de respuesta	Siempre
Dirección del búfer	SEND y RECEIVE
Número de bytes (núm. de bytes recibidos)	RECEIVE
Indicador de entrega inmediata	RECEIVE
Indicador urgente	RECEIVE

Interfaz TCP/nivel inferior

Realmente, TCP realiza llamadas al módulo del protocolo de nivel inferior para enviar y recibir información a través de una red. Uno de estos casos es el del sistema de interconexión de redes ARPA, donde el módulo del nivel inferior es el protocolo de internet (IP) [2].

Si el protocolo de nivel inferior es IP, éste proporciona argumentos para un tipo de servicio y un tiempo de vida. TCP utiliza los siguientes valores para estos parámetros:

Tipo de servicio = prioridad: habitual, retardo: normal, rendimiento: normal, fiabilidad: normal; ó 00000000.

Tiempo de vida = un minuto, ó 00111100.

Nótese que el tiempo de vida máximo asumido para un segmento es de dos minutos. Aquí se requiere explícitamente que un segmento sea destruido si no puede ser entregado por el sistema de internet en un minuto.

Si el nivel inferior es IP (u otro protocolo que proporcione esta característica) y se utiliza encaminamiento de origen ['source routing'], la interfaz debe permitir que se pueda comunicar la información sobre la ruta. Esto es especialmente importante de forma que las direcciones de origen y destino usadas en la suma de comprobación de TCP sean las direcciones de

origen y destino definitivas. También es importante conservar la ruta de retorno para contestar preguntas sobre la conexión.

Todo protocolo de nivel inferior debe suministrar la dirección de origen, la dirección de destino, y los campos del protocolo, además de alguna forma de determinar la "longitud de TCP", para suministrar un servicio equivalente al de IP y poder ser utilizados en la suma de comprobación de TCP.

3.9. Procesamiento de eventos

El procesamiento explicado en esta sección es un ejemplo de una posible implementación. Otras implementaciones pueden tener secuencias de procesamiento ligeramente diferentes, pero sólo en los detalles, no en lo esencial.

Se puede caracterizar la actividad del TCP como una serie de respuestas a eventos. Los eventos posibles se pueden clasificar en tres categorías: llamadas del usuario, segmentos entrantes y fin de tiempos de espera. Esta sección describe el procesamiento que realiza TCP en respuesta a cada uno de estos eventos. En muchos casos el procesamiento necesario depende del estado de la conexión

Eventos posibles

Llamadas del usuario

OPEN
SEND
RECEIVE
CLOSE
ABORT
STATUS

Segmentos entrantes

LLEGADA DEL SEGMENTO

Fin del tiempo de espera

FIN DEL TIEMPO DE ESPERA DE USUARIO

FIN DEL TIEMPO DE ESPERA DE RETRANSMISION
FIN DEL TIEMPO DE ESPERA EN EL ESTADO 'TIME-WAIT'

El modelo de la interfaz TCP/usuario se basa en que las órdenes de usuario reciben una respuesta inmediata y posiblemente otra retardada por medio de un evento o una pseudo-interrupción. En las descripciones siguientes, el término "señal" significa "causa una respuesta retardada".

Las respuestas de error se dan en forma de cadenas de caracteres. Por ejemplo, las órdenes de usuario que hacen referencia a conexiones que no existen reciben lo siguiente: 'error: connection not open' ("error: conexión sin abrir").

Téngase en cuenta en lo que sigue que toda la aritmética sobre números de secuencia, números de acuse de recibo, números asociados a ventanas, etcétera, es módulo 2**32, el tamaño del espacio reservado para los números de secuencia. Nótese también que "=<" significa "menor o igual que (módulo 2**32)".

Una forma natural de pensar en el procesamiento de segmentos entrantes es imaginar que primero se comprueba que su número de secuencia sea el adecuado (es decir, que su contenido caiga dentro del rango de la "ventana de recepción" esperada en el espacio de números de secuencia) y que después generalmente son puestos en cola y procesados según el orden por números de secuencia.

Cuando un segmento se solapa con otros segmentos ya recibidos se reconstruye el segmento de forma que contenga únicamente los nuevos datos, y se ajustan los campos de cabecera para que sean consistentes.

Nótese que si no se menciona un cambio de estado TCP permanece en el mismo estado.

Llamada OPEN

ESTADO 'CLOSED' (i.e., el TCB no está definido)

Se crea un nuevo bloque de control de transmisión (TCB) que soporta información sobre el estado de la conexión. Se completa el identificador de conector local e información relativa al conector remoto, prioridad, seguridad/compartimentación y tiempo de espera de usuario. Nótese que algunos detalles del conector remoto pueden quedar sin especificar durante un OPEN pasivo y habrán de ser cumplimentados en los parámetros del segmento SYN entrante. Están permitidas para este usuario, tanto verificaciones de seguridad, como solicitudes de prioridad. De no ser así, se devuelve 'error: precedence not allowed' ("error: prioridad no admitida") o 'error: security/compartment not allowed' ("error: seguridad/compartimentación no admitidas"). En el caso pasivo, se pasa al estado LISTEN y se retorna. En el caso activo, y si el conector remoto queda sin especificar, se devuelve 'error: foreign socket unspecified' ("error: conector remoto sin especificar"). En el caso activo, y si se especifica el conector remoto, se envía un segmento SYN. Se selecciona un número de secuencia inicial (ISS). Se envía un segmento SYN de la forma <SEQ=ISS><CTL=SYN>. Se fija SND.UNA a ISS, SND.NXT a ISS+1, se pasa al estado SYN-SENT y se retorna.

Si el llamador no tiene acceso al conector local especificado, se

devuelve 'error: connection illegal for this process' ("error: conexión ilegal para este proceso"). Si no quedase espacio para crear una nueva conexión, se devuelve 'error: insufficient resources' ("error: recursos insuficientes").

ESTADO 'LISTEN'

Si es el caso activo y para un conector remoto especificado, se cambia la conexión de pasiva a activa, se selecciona un ISS. Se envía un segmento SYN , se fija SND.UNA a ISS, SND.NXT a ISS+1. Se pasa al estado SYN-SENT. Los datos asociados a SEND pueden enviarse mediante un segmento SYN o pasarse a la cola de transmisión después de entrar en el estado ESTABLISHED. El bit urgente, en caso de ser requerido en el comando, deberá enviarse junto con los segmentos de datos enviados como consecuencia de la ejecución de este comando. Si no existiera espacio para almacenar la petición en cola, se responde con 'error: insufficient resources'. Si el conector remoto no hubiera sido especificado, se devolverá 'error: foreign socket unspecified'.

ESTADO 'SYN-SENT'
ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSE-WAIT'

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se devuelve 'error: connection already exists' ("error: la conexión ya existe").

Llamada SEND

ESTADO 'CLOSED' (i.e., TCB no existe)

Si el usuario no tuviera acceso a tal conexión, entonces se devuelve 'error: connection illegal for this process' ("error: conexión ilegal para este proceso).

En caso contrario, se devuelve 'error: connection does not exist' ("error: la conexión no existe").

ESTADO 'LISTEN'

Si el conector remoto hubiera sido especificado, la conexión pasa de pasiva a activa, se selecciona un ISS. Se envía un segmento SYN, se fija SND.UNA a ISS, SND.NXT a ISS+1. Se pasa al estado SYN-SENT. Los datos asociados con SEND podrán ser enviados mediante un segmento SYN o almacenados en la cola de transmisión tras activarse el estado ESTABLISHED. El bit urgente, en caso de ser requerido en el comando, deberá enviarse junto con los

segmentos de datos enviados como consecuencia de la ejecución de este comando. Si no existiera espacio para almacenar la petición en cola, se responde con 'error: insufficient resources'. Si el conector remoto no hubiera sido especificado, devolverá 'error: foreign socket unspecified'.

ESTADO 'SYN-SENT'
ESTADO 'SYN-RECEIVED'

Almacenamiento en cola de los datos para la transmisión tras haber pasado al estado ESTABLISHED. Si no hubiera espacio disponible en la cola, responde con 'error: insufficient resources'.

ESTADO 'ESTABLISHED'
ESTADO 'CLOSE-WAIT'

Segmentar el búfer y enviarlo cargado con un acuse de recibo (valor de acuse de recibo = RCV.NXT). Si no hubiera espacio suficiente para almacenar el contenido de este búfer, se devuelve simplemente "error: insufficient resources".

Si el indicador de urgente vale uno, se envía SND.UP <- SND.NXT-1 y se pone el puntero urgente en los segmentos salientes.

ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se devuelve 'error: connection closing' y no presta el servicio solicitado.

Llamada RECEIVE

ESTADO 'CLOSED' (i.e., no existe TCB)

Si el usuario no tuviera acceso a tal conexión, se devuelve "error: connection illegal for this process".

En caso contrario se devuelve "error: connection does not exist".

ESTADO 'LISTEN'
ESTADO 'SYN-SENT'
ESTADO 'SYN-RECEIVED'

Almacenamiento en la cola de procesado tras haber entrado en el estado ESTABLISHED. Si no hubiera espacio para almacenar en cola esta solicitud, se responde con 'error: insufficient resources'.

ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'

Si hubiera una cantidad insuficiente de segmentos entrantes en cola para satisfacer la solicitud, se almacena en cola la solicitud. Si no hubiera espacio en cola para recordar el RECEIVE, se responde con "error: insufficient resources".

Reensamblado de los segmentos entrantes en cola en el búfer de recepción y devolución al usuario. Se marca 'push seen' ("'push' visto") (PUSH) si este fuera el caso.

Si se adelanta un RCV.UP a los datos que actualmente se están pasando al usuario se notifica al usuario sobre la presencia de datos urgentes.

Cuando el TCP se responsabiliza de la entrega de datos al usuario, este hecho debe ser comunicado al emisor por medio de un acuse de recibo. La construcción de tal acuse de recibo se describe más abajo en la discusión del procesado de un segmento entrante.

ESTADO 'CLOSE-WAIT'

Dado que la parte remota ya ha enviado un FIN, los RECEIVE deberán ser rellenados con texto ya disponible, pero aún no entregado al usuario. Si no hubiera texto en espera para ser entregado, el RECEIVE obtendrá una respuesta 'error: connection closing' ("error: cerrando la conexión"). En caso contrario, cualquier texto presente podrá usarse para alimentar el RECEIVE.

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se devuelve 'error: connection closing'.

Llamada CLOSE

ESTADO 'CLOSED' (i.e., no existe TCB)

Se devuelve 'error: connection illegal for this process'.

En caso contrario se devuelve 'error: connection does not exist'.

ESTADO 'LISTEN'

A cualesquiera órdenes RECEIVE relevantes, se devuelven respuestas

"error: closing". Se borra el TCB, se pasa al estado CLOSED, y se retorna.

ESTADO 'SYN-SENT'

Se borra el TCB y se devuelven respuestas "error: closing" a cualesquiera órdenes SEND o RECEIVE.

ESTADO 'SYN-RECEIVED'

Si no se hubieran emitido órdenes SEND y no hubiera datos pendientes de envío, se construye un segmento FIN y se envía, tras lo que se pasa al estado FIN-WAIT-1; en caso contrario, se almacena en la cola de procesamiento tras haber pasado al estado ESTABLISHED.

ESTADO 'ESTABLISHED'

Se deja en cola hasta que todas las órdenes SEND precedentes hayan sido segmentadas, entonces se construye un segmento FIN y se envía. En cualquier caso, se pasa al estado FIN-WAIT-1.

ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'

Estrictamente hablando, se trata éste de un error, y debería recibir una respuesta 'error: connection closing'. Una respuesta 'ok' ("correcto") sería aceptable también, ya que no se ha emitido aún un segundo FIN (no obstante, el primer FIN puede ser retransmitido).

ESTADO 'CLOSE-WAIT'

Esta solicitud se almacena en cola hasta que todas las órdenes SEND precedentes hayan sido segmentadas; entonces se envía un segmento FIN, y se pasa al estado CLOSING.

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se contesta con 'error: connection closing'.

Llamada ABORT

ESTADO 'CLOSED' (i.e., no existe TCB)

Si el usuario no tuviera acceso a tal conexión, se devuelve 'error: connection illegal for this process'.

En caso contrario se devuelve 'error: connection does not exist'.

ESTADO 'LISTEN'

A cualesquiera órdenes RECEIVE relevantes deben devolverse respuestas 'error: connection reset' ("error: conexión reiniciada"). Se borra el TCB, se pasa al estado CLOSED, y se retorna.

ESTADO 'SYN-SENT'

Todas las órdenes SEND y RECEIVE en cola deberían recibir la notificación 'connection reset'. Borrado de TCB, paso al estado CLOSED, y retorno.

ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSE-WAIT'

Envío de un segmento de reinicio ('reset'):

<SEQ=SND.NXT><CTL=RST>

Todas las órdenes SEND y RECEIVE en cola deberían recibir una notificación 'connection reset'; todos los segmentos en cola de transmisión (excepto el RST construído más arriba) o retransmisión deberían ser abandonados; se borra el TCB , se pasa al estado CLOSED, y se retorna.

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se responde con 'ok' y se borra el TCB, se pasa al estado CLOSED, y se retorna.

Llamada STATUS

ESTADO 'CLOSED' (i.e., TCB no está definido)

En caso de que el usuario no consiguiera acceso a tal conexión, se

devuelve 'error: connection illegal for this process'.

En cualquier otro caso, se devuelve 'error: connection does not exist'.

ESTADO 'LISTEN'

Se devuelve 'state = LISTEN', y el puntero de TCB.

ESTADO 'SYN-SENT'

Se devuelve 'state = SYN-SENT', y el puntero de TCB.

ESTADO 'SYN-RECEIVED'

Se devuelve 'state = SYN-RECEIVED', y el puntero de TCB.

ESTADO 'ESTABLISHED'

Se devuelve 'state = ESTABLISHED', y el puntero de TCB.

ESTADO 'FIN-WAIT-1'

Se devuelve 'state = FIN-WAIT-1', y el puntero de TCB.

ESTADO 'FIN-WAIT-2'

Se devuelve 'state = FIN-WAIT-2', y el puntero de TCB.

ESTADO 'CLOSE-WAIT'

Se devuelve 'state = CLOSE-WAIT', y el puntero de TCB.

ESTADO 'CLOSING'

Se devuelve 'state = CLOSING', y el puntero de TCB.

ESTADO 'LAST-ACK'

Se devuelve 'state = LAST-ACK', y el puntero de TCB.

ESTADO 'TIME-WAIT'

Se devuelve 'state = TIME-WAIT', y el puntero de TCB.

LLEGADA DE SEGMENTO ('SEGMENT ARRIVES')

Si el estado es CLOSED (i.e., no existe TCB) entonces

se descartan todos los datos del segmento entrante. Cualquier segmento que contenga un RST es descartado. Cualquier segmento entrante que no contenga un RST origina el envío de un RST como respuesta. Se seleccionan los valores de los campos de acuse de recibo y de secuencia para que el número de secuencia del segmento de reinicio sea aceptable para el TCP que envió el segmento.

Si el bit de ACK vale cero, se utiliza el número de secuencia cero,

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

Si el bit ACK vale uno,

<SEQ=SEG.ACK><CTL=RST>

Se retorna.

Si el estado es LISTEN, entonces

Primero se comprueba si es un RST

Se deberá ignorar cualquier RST entrante. Se retorna.

Segundo se comprueba si es un ACK

Cualquier acuse de recibo será defectuoso si llega cuando una conexión está todavía en estado LISTEN. Debería construirse un segmento de reinicio aceptable para cualquier segmento entrante que transporte un ACK. El RST deberá tener el siguiente formato:

<SEQ=SEG.ACK><CTL=RST>

Se retorna.

Tercero se comprueba si es un SYN

Si el bit SYN vale 1, se comprueba la seguridad. Si la seguridad/compartimentación del segmento entrante no coincidiera con la seguridad/compartimentación del TCB, se envía un segmento de reinicio y se retorna.

<SEQ=SEG.ACK><CTL=RST>

Si SEG.PRC es mayor que TCB.PRC, y en caso de que el usuario y el sistema lo permita, se fija TCB.PRC<-SEG.PRC, si no estuviera permitido, se envía un reinicio y se retorna.

<SEQ=SEG.ACK><CTL=RST>

Si SEG.PRC es menor que TCB.PRC, se prosigue.

Se pone RCV.NXT a SEG.SEQ+1, IRS a SEG.SEQ y cualquier otro control o texto deberá quedar en la cola para su procesado posterior. Deberá seleccionarse ISS y enviarse un segmento SYN de la forma:

<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>

SND.NXT se fija a ISS+1 y SND.UNA a ISS. El estado de la conexión deberá cambiar a SYN-RECEIVED. Nótese que será procesado cualquier otro control entrante o datos (combinados con SYN) en el estado SYN-RECEIVED, pero el procesado de SYN y ACK no deberían repetirse. Si la escucha no hubiera sido completamente especificado (i.e., el conector remoto no hubiera sido completamente especificado), deberán cumplimentarse en este momento los campos que quedaron sin especificar.

Cuarto otro texto o control

Cualquier otro segmento portador de control o de texto (que no contenga un SYN) deberá llevar un ACK, y así será descartado por el procesado del ACK. Un segmento entrante RST podría no ser válido, ya que podría no haber sido enviado como respuesta a nada que haya sido enviado por esta encarnación de la conexión. Así que, es improbable encontrarse en esta situación, pero si fuera así, ha de descartarse el segmento y retornar.

Si el estado es SYN-SENT, entonces

Primero se comprueba el bit de ACK

Si el bit ACK vale uno

Si SEG.ACK =< ISS, o bien SEG.ACK > SND.NXT, se envía una orden de reinicio (a no ser que el bit de RST esté puesto a uno. Si fuera así, se descarta el segmento y se retorna)

<SEQ=SEG.ACK><CTL=RST>

y se descarta el segmento. Se retorna.

Si SND.UNA =< SEG.ACK =< SND.NXT entonces el ACK es aceptable.

Segundo, se comprueba el bit RST

Si el bit RST vale uno

Si el ACK era aceptable, entonces se indica al usuario "error:

connection reset", se abandona el segmento, se pasa al estado CLOSED, se borra el TCB, y se retorna. En cualquier otro caso (sin ACK) se abandona el segmento y se retorna.

Tercero, se comprueba la seguridad y la prioridad

Si la seguridad/compartimentación del segmento no coincidieran exactamente con la seguridad/compartimentación del TCB, se envía una orden de reinicio.

Si hubiera un ACK

<SEQ=SEG.ACK><CTL=RST>

En otro caso

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

Si hubiera un ACK

La prioridad del segmento deberá coincidir con la prioridad del TCB. De no ser así, se enviará una orden de reinicio.

<SEQ=SEG.ACK><CTL=RST>

Si no hubiera ACK

Si la prioridad del segmento fuera mayor que la prioridad del TCB entonces, si lo permiten el usuario y el sistema, elévese la prioridad del TCB hasta el valor correspondiente al segmento. Si no se concediera tal permiso para elevar la prioridad, deberá enviarse una orden de reinicio.

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

Si la prioridad del segmento fuera menor que la prioridad en el TCB, continúese.

Si se hubiera enviado una orden de reinicio, descártese el segmento y retórnese.

Cuarto, comprobar el bit SYN

Este paso sólo debería tener lugar si el ACK fuera correcto, o en caso de no haber ACK, y siempre que el segmento no contuviera una orden RST.

Si el bit SYN vale uno y la seguridad/compartimentación y prioridad tienen valores aceptables RCV.NXT se pone a SEG.SEQ+1, IRS se pone a SEG.SEQ. SND.UNA debería incrementarse hasta igualar SEG.ACK (si hubiera un ACK), y cualesquiera segmentos en cola de retransmisión que queden así confirmados deberán ser

eliminados.

Si SND.UNA > ISS (nuestro SYN ha sido confirmado mediante un segmento ACK), se cámbia al estado de la conexión a ESTABLISHED (establecida), se constrúye un segmento ACK

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

y se envia. Podrán incluirse cualesquiera datos o controles que estuvieran almacenados para su transmisión. Si existieran otros controles o texto en el segmento, entonces se continúa procesando en el sexto paso indicado más abajo donde el bit URG queda comprobado, en cualquier otro caso, se retorna.

En caso contrario, se pasa a SYN-RECEIVED, se constrúye un segmento SYN,ACK

<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>

y se envia. Si existieran otros controles o texto en el segmento, se ponen en cola para procesar despues de que el paso al estado ESTABLISHED se haya verificado. Se retorna.

Quinto, si ninguno de los bits SYN o RST valiera uno, entonces se abandona el segmento y se retorna.

En caso contrario,

primero se comprueba el número de secuencia

ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSE-WAIT'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Los segmentos se procesan secuencialmente. Aunque se practican comprobaciones iniciales de llegada con objeto de descartar los duplicados caducados, se efectúa un procesado ulterior en el orden según SEG.SEQ. Si los contenidos de un segmento produjeran un solapamiento de fronteras entre segmentos viejos y nuevos, sólo deberán procesarse las partes nuevas.

Existen cuatro casos posibles para la comprobación de aceptabilidad de un segmento entrante:

Segment Longit	Ventan Recepc	Comprobación
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND
>0	0	no aceptable
>0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND o bien RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

Si el RCV.WND valiera cero, no se aceptará ningún segmento, pero deberá dejarse margen a la aceptación de órdenes ACK válidas, así como de segmentos URG y RST.

Si un segmento entrante fuera no aceptable, deberá enviarse un acuse de recibo en respuesta (siempre y cuando el bit RST valga uno. Si es así, se deshecha el segmento y se retorna):

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

Tras enviar el acuse de recibo, desháchese el segmento no aceptable y se retorna.

En lo que sigue se supondrá siempre que el segmento es un segmento ideal que comienza con RCV.NXT y no sobrepasa el tamaño de la ventana. Uno podría siempre ajustar los segmentos dados de forma que satisfagan esta suposición recortando cualquier fragmento que caiga fuera de la ventana (incluyendo SYN y FIN), y continuando con el procesamiento únicamente si el segmento comienza con RCV.NXT tras la operación. Los segmentos con números de secuencia mayores se podrán mantener para su posterior procesamiento.

Segundo, se comprueba el bit RST,

ESTADO 'SYN-RECEIVED'

Si el bit RST bit vale uno

Si esta conexión hubiera sido iniciada con un OPEN pasivo (i.e., provino de un estado LISTEN), entonces póngase de nuevo la misma al estado LISTEN y se retorna. El usuario no tiene por qué ser informado de esto. Si esta conexión se hubiera iniciado con un OPEN activo (i.e., provino de un estado SYN-

SENT), entonces es que la conexión fue rechazada, se indíca al usuario "connection refused". En cualquiera de ambos casos, todos los segmentos en la cola de retransmisión deberán ser eliminados. Y en el caso de un OPEN activo, se pasa al estado CLOSED, se borra el TCB, y se retorna.

ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSE-WAIT'

Si el bit RST vale uno, cualesquiera órdenes relevantes RECEIVEs y SEND deberían recibir respuestas "reset". Todas las colas de segmento deberán ser inicializadas. Los usuarios deberán también recibir un aviso unilateral (no solicitado) general de "connection reset". Se pasa al estado CLOSED, se borra el TCB, y se retorna.

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Si el bit RST vale a uno, se pasa al estado CLOSED, se borra el TCB, y se retorna.

Tercero, comprobación de la seguridad y prioridad

ESTADO 'SYN-RECEIVED'

Si la seguridad/compartimentación y prioridad del segmento no coincidieran exactamente con la seguridad/compartimentación y prioridad del TCB, deberá enviarse una orden de reinicio ('reset'), y retornar.

ESTADO 'ESTABLISHED'

Si la seguridad/compartimentación y prioridad del segmento no coincidieran exactamente con la seguridad/compartimentación y prioridad del TCB, deberá enviarse una orden de reinicio, y cualesquiera RECEIVEs y SEND deberán recibir respuestas "reset". Todas las colas de segmento deberán ser inicializadas. Los usuarios deberán también recibir un aviso unilateral general de "connection reset". Se pasa al estado CLOSED, se borra el TCB, y se retorna.

Nótese que esta comprobación se ubica a continuación de la comprobación de secuencia para evitar que un segmento procedente de una conexión anterior entre los mismos puertos y con distintos valores de seguridad o prioridad origine una interrupción de la conexión actual.

Cuarto, comprobación del bit SYN,

ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'

ESTADO 'CLOSE-WAIT'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Si el SYN estuviera en la ventana, denotará un error. Se envía un 'reset', y cualesquiera órdenes RECEIVE y SEND relevantes, deberían recibir respuestas "reset", todas las colas de segmentos deberán ser inicializadas, el usuario deberá también recibir una señal no solicitada y general de "connection reset". Se pasa al estado CLOSED, se borra el TCB, y se devuelve el resultado.

Si el SYN no estuviera en la ventana, no se alcanzaría este paso y se habría recibido un mensaje de acuse de recibo en el primer paso (comprobación del número de secuencia).

Quinto, comprobación del campo de ACK,

si el bit de ACK vale cero, se descarta el segmento y se retorna.

si el bit de ACK vale uno

ESTADO 'SYN-RECEIVED'

Si SND.UNA =< SEG.ACK =< SND.NXT entonces se pasa al estado ESTABLISHED y se continúa procesando.

Si el acuse de recibo del segmento no fuera aceptable, se construye un segmento de reinicio,

<SEQ=SEG.ACK><CTL=RST>

y se envia.

ESTADO 'ESTABLISHED'

Si SND.UNA < SEG.ACK =< SND.NXT entonces, se envia SND.UNA <- SEG.ACK. Cualesquiera segmentos en la cola de retransmisión que se confirmen en esta llegada de segmentos serán

eliminados. Los usuarios deberían recibir confirmaciones positivas de los búferes que hayan sido completamente enviados y y confirmados (i.e., el buffer SEND debería ser respondido con un mensaje de 'ok'). Si el ACK está duplicado (SEG.ACK < SND.UNA), puede ignorarse. Si el ACK confirma algún tramo aún sin enviar (SEG.ACK > SND.NXT), entonces se envia un ACK, se descarta el segmento, y se retorna.

Si SND.UNA < SEG.ACK =< SND.NXT, la ventana de envío deberá actualizarse. Si (SND.WL1 < SEG.SEQ o bien (SND.WL1 = SEG.SEQ y SND.WL2 =< SEG.ACK)), establézcase SND.WND <- SEG.WND, así

como SND.WL1 <- SEG.SEQ, y envíese SND.WL2 <- SEG.ACK.

Nótese que SND.WND es un desplazamiento desde SND.UNA, que SND.WL1 registra el número de secuencia del último segmento usado para actualizar SND.WND, y que SND.WL2 registra el número de acuse de recibo correspondiente al último segmento usado para actualizar SND.WND. La comprobación aquí impide el uso de segmentos antiguos para actualizar la ventana.

ESTADO 'FIN-WAIT-1'

Además del procesamiento para el estado ESTABLISHED, si nuestro FIN queda ahora confirmado con un acuse de recibo, se pasa a FIN-WAIT-2 y se continúa procesando en ese estado.

ESTADO 'FIN-WAIT-2'

Además del procesamiento para el estado ESTABLISHED, si la cola de retransmisión estuviera vacía, podrá reconocerse un CLOSE por parte del usuario ('ok') pero no deberá borrarse el TCB.

ESTADO 'CLOSE-WAIT'

Repítase el mismo procesamiento válido para el estado ESTABLISHED.

ESTADO 'CLOSING'

Además del procesado para el estado ESTABLISHED, si el ACK acusa recibo de nuestro FIN, entonces se pasa al estado TIME-WAIT, en cualquier otro caso, se ignora el segmento.

ESTADO 'LAST-ACK'

Lo único que puede llegar en este estado es un acuse de recibo de nuestro FIN. Si nuestro FIN no fuera reconocido, se borra

el TCB, se pasa al estado CLOSED, y se retorna.

ESTADO 'TIME-WAIT'

Lo único que puede llegar en este estado es la retransmisión de un FIN remoto. Se realiza el acuse de recibo correspondiente y se reinicia la cuenta del tiempo de vida 2 MSL.

Sexto, se comprueba el bit URG,

ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'

Si el bit URG vale uno, RCV.UP <- max(RCV.UP,SEG.UP), se infórme al usuario de que la parte remota tiene datos urgentes si el puntero de urgencia (RCV.UP) está adelantado respecto a los datos pasados al usuario. Si el usuario ya hubiera sido informado (o estuviera aún en el "modo urgente") para esta secuencia continua de datos urgentes, no se informará al usuario de nuevo.

ESTADO 'CLOSE-WAIT'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Esto no debería suceder, ya que se ha recibido un FIN desde la parte remota. Ignórese el URG.

Séptimo, se procesa el texto del segmento,

ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'

Una vez en el estado ESTABLISHED, es posible entregar texto del segmento a los búferes RECEIVE del usuario. El texto de los segmentos podrá depositarse en los búferes hasta que, o bien el búfer esté lleno, o el segmento quede vacío. Si el segmento se vacía y lleva un indicador PUSH, entonces se notifica al usuario, cuando el búfer le es devuelto, de que se ha recibido un PUSH.

Cuando el módulo de TCP asume la responsabilidad de la entrega de datos al usuario, deberá también confirmar la recepción de los datos.

Toda vez que el módulo de TCP asuma la responsabilidad de los

datos, avanzará RCV.NXT respecto a los datos aceptados, y ajustará RCV.WND en lo apropiado respecto a la disponibilidad del buffer actual. El total de RCV.NXT y RCV.WND así asignados no deberá ser reducido.

Se ruega tener en cuenta las sugerencias sobre gestión de ventanas en la sección 3.7.

Envíese un acuse de recibo de la forma:

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

Este acuse de recibo deberá ser transportado (N.T.: 'piggybacked') en un segmento y transmitido si fuera posible sin incurrir en retraso indebido.

ESTADO 'CLOSE-WAIT'

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Esto no debería suceder, ya que se ha recibido un FIN desde la parte remota. Ignórese el texto del segmento.

Octavo, se comprueba el bit FIN,

No deberá procesarse el FIN si el estado fuera CLOSED, LISTEN o SYN-SENT, ya que el SEG.SEQ no podrá ser validado; se descarta el segmento y se retorna.

Si el bit FIN vale uno, se informa al usuario mediante "connection closing" y se devuelve cualesquiera mensajes RECEIVE pendientes mediante el mismo mensaje, se avanza RCV.NXT respecto al FIN, y se envia un acuse de recibo para el FIN. Nótese que FIN implica PUSH para cualquier texto del segmento que aún no ha sido entregado al usuario.

ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'

Se pasa al estado CLOSE-WAIT.

ESTADO 'FIN-WAIT-1'

Si nuestro FIN ha sido confirmado por un acuse de recibo (quizá en este mismo segmento), entonces se pasa a TIME-WAIT, se inicia el contador de tiempo 'time-wait', se interrumpen todos los otros contadores; en cualquier otro caso, se pasa al estado CLOSING.

ESTADO 'FIN-WAIT-2'

Se pasa al estado TIME-WAIT. Se inicia el contador de tiempo 'time-wait', se interrumpen todos los demás contadores de tiempo.

ESTADO 'CLOSE-WAIT'

Permanecer en el estado CLOSE-WAIT.

ESTADO 'CLOSING'

Permanecer en el estado CLOSING.

ESTADO 'LAST-ACK'

Permanecer en el estado LAST-ACK.

ESTADO 'TIME-WAIT'

Permanecer en el estado TIME-WAIT. Reiniciar el tiempo de vida 2 MSL y retornar.

TIEMPO DE ESPERA DE USUARIO

Si expirase el tiempo de espera del usuario para cualquier estado, inicializar todas las colas, informar al usuario mediante "error: connection aborted due to user timeout". En general, y para cualesquiera llamadas relevantes, se borra el TCB, se pasa al xestado CLOSED y se retorna.

TIEMPO DE ESPERA DE RETRANSMISIÓN

Para cualquier estado, si expirase el tiempo de vida de retransmisión de un segmento en la cola de retransmisión, se envia de nuevo el segmento que está al comienzo de la cola de retransmisión, se reinicia el contador de tiempo de retransmisión, y se retorna.

TIEMPO DE ESPERA EN EL ESTADO 'TIME-WAIT'

Si expirase el tiempo de espera en el estado TIME-WAIT de una conexión, se borra el TCB, se pasa al estado CLOSED y se retorna.

GLOSARIO

1822
: Informe BBN 1822 "The Specification of the Interconnection of a Host and an IMP". La especificación de la interfaz entre un 'host' u ARPANET.

ACK
: Un bit de control (del inglés 'acknowledge') que no ocupa posición del espacio de números de secuencias y que indica que el campo de acuse de recibo de este segmento especifica el próximo número de secuencia que el emisor de este segmento espera recibir, por lo que también realiza un acuse de recibo de todos los números de secuencia anteriores.

ARPANET, mensaje de
: La unidad de transmisión entre un 'host' y un IMP en ARPANET. Su tamaño máximo es de 1012 octetos (8096 bits).

ARPANET, paquete de
: Una unidad de transmisión que ARPANET utiliza internamente entre los IMP. Su tamaño máximo es de 126 octetos (1008 bits).

cabecera
: Información de control al comienzo de un mensaje, segmento, fragmento, paquete o bloque de datos.

conector
: En inglés: 'socket'. Dirección que específicamente incluye un identificador de puerto, es decir, la concatenación de una dirección de internet con un puerto de TCP.

conexión
: Un camino lógico de comunicación identificado por un par de conectores.

datagrama
: Un mensaje que se envía en una red de comunicaciones de ordenadores por intercambio de paquetes.

dirección de destino
: La dirección de destino, habitualmente los identificadores de red y de 'host'.

dirección de origen
: la dirección de origen, generalmente los identificadores de red y de 'host'.

envío, secuencia de
: Siguiente número de secuencia que el TCP local (emisor) utilizará en la conexión. Se selecciona inicialmente de una serie de números de secuencias iniciales (ISN, 'initial sequence number') y se incrementa por cada octeto de datos o de control que se transmita.

envío, ventana de
: Representa los números de sencuencia que el TCP remoto (receptor) espera recibir. Es el valor del campo de ventana que se especifica en los segmentos provenientes del TCP remoto (receptor de los datos). El rango de los nuevos números de secuencia que un TCP puede emitir va desde SND.TXT hasta SND.UNA + SND.WND - 1. (Las retransmisiones de los números de secuencia entre SND.UNA y SND.NXT son de esperar, por supuesto).

FIN
: Un bit de control (de 'finis') que ocupan una posición del espacio de números de secuencia y que indica que el emisor no

enviará más datos o informacion de control que ocupe posiciones del espacio de números de secuencia.

fragmento
: Una porción de una unidad lógica de datos, en particular, un fragmento de internet es una porción de un datagrama de internet.

FTP
: Un protocolo de transferencia de ficheros.

host
: Un computador. En particular, un origen o destino de los mensajes desde el punto de vista de la red de comunicaciones.

identificación
: Un campo del protocolo de internet. Identifica el valor asignado por el emisor que sirve de ayuda para el ensamblaje de los fragmentos de un datagrama.

IMP
: La interfaz de procesamiento de mensajes o 'Interface Message Processor', el commutador de paquetes de ARPANET.

internet, dirección de
: Una dirección de origen o de destino específica del nivel de 'host'.

internet, datagrama de
: La unidad de datos que se intercambia entre un módulo de internet y el protocolo de nivel superior más la cabecera de internet.

internet, fragmento de
: Una porción de los datos de un datagrama de internet con una cabecera de internet.

IP
: El protocolo de internet o 'Internet Protocol'.

IRS
: El número de secuencia de recepción inicial ('Initial Receive Sequence number'). El primer número utilizado por el emisor en una conexión.

ISN
: El número de secuencia inicial ('Initial Sequence Number'). El primer número de secuencia utilizado en una conexión (el ISS o el IRS). Se selecciona por un procedimiento basado en el tiempo de reloj.

ISS
: El número de secuencia de envío inicial ('Initial Send Sequence'). El primer número de secuencia utilizado por el emisor en la conexión.

líder
: Información de control del comienzo de un mensaje o bloque de datos. En particular, en ARPANET, la información en un mensaje de ARPANET en la interfaz 'host'-IMP.

módulo
: Una implementación, habitualmente de 'software', de un protocolo u otro procedimiento.

MSL
: Vida máxima del segmento ('Maximum Segment Lifetime'), el tiempo que un segmento de TCP puede existir en el sistema de inter-redes. Se define por convención en 2 minutos.

octeto
: Un byte de ocho bits.

opciones
: Un campo de opción puede contener varias opciones, y cada opción puede tener varios octetos de longitud. Las opciones se utilizan sobre todo en situaciones de pruebas; por ejemplo para transportar marcas de tiempo.

paquete
: Un conjunto de datos con una cabecera que puede estar o no lógicamente completa. Más a menudo, se refiere a un empaquetamiento físico de datos que lógico.

paquete local
: La unidad de transmisión dentro de una red local.

puerto
: La porción de un conector que especifica qué entrada lógica o canal de salida se asocian con los datos.

puntero urgente
: Un campo de control significativo sólamente cuando el bit URG

está establecido a uno. Este campo comunica el valor del puntero urgente que indica el octeto de datos asociado con la llamada urgente del usuario emisor.

proceso
: Un programa en ejecución. Un origen o destino de datos desde el punto de vista de TCP o cualquier otro protocolo de 'host' a 'host'.

PUSH
: Un bit de control que no ocupa posición en el espacio de número de secuencias, y que indica que un segmento que contiene datos debe entregarse inmediatamente ('pushed') al usuario receptor.

RCV.NXT
: Siguiente número de secuencia de recepción.

RCV.UP
: Puntero urgente de recepción.

RCV.WND
: Ventana de recepción.

recepción, número de secuencia de recepción
: Número de secuencia que el TCP local espera recibir.

recepción, ventana de
: Representa los números de secuencias que el TCP local (receptor) espera recibir. De este modo, el TCP local considera que los segmentos que intersecten con el rango RCV.NXT a RCV.NXT + RCV.WND - 1 transportan datos o bits de control aceptables. Los segmentos que contienen números de secuencia completamente fuera de este rango se consideran duplicados y se descartan.

RST
: Un bit de control ('reset' o reinicio), que no ocupa posición en el espacio de secuencias, y que indica que el receptor debe eliminar la conexión sin más interacción. El receptor puede determinar, basándose en los campos de número de secuencia y de número de acuse de recibo, si debe obedecer o ignorar la orden de reinicio. En ningún caso, la recepción de un segmento que contiene un RST dará lugar a una respuesta con otro RST.

RTP
: Protocolo de tiempo real ('Real Time Protocol'): un protocolo

de nivel 'host' a 'host' para la comunicación de información crítica de tiempo.

secuencia a la izquierda
: Número de secuencia siguiente en espera de que el TCP reciba su acuse de recibo (o el número de secuencia más bajo en la actualidad sin confirmación) y que a veces se denomina el borde izquierdo de la ventana de envío.

SEG.ACK
: Acuse de recibo del segmento.

SEG.LEN
: Longitud del segmento.

SEG.PRC
: Valor de prioridad del segmento.

SEG.SEQ
: Secuencia del segmento.

SEG.UP
: Campo de puntero urgente del segmento.

SEG.WND
: Campo de ventana del segmento.

segmento
: Una unidad lógica de datos, en particular un segmento de TCP es la unidad de datos transferida entre dos módulos de TCP.

segmento, acuse de recibo del
: El número de secuencia en el campo de acuse de recibo del segmento entrante.

segmento, longitud del
: La cantidad del espacio de número de secuencias utilizada por un segmento, incluyendo cualquier bit de control que ocupe posiciones del espacio de número de secuencias.

segmento, secuencia del
: El número en el campo de número de secuencia del segmento entrante.

SND.NXT
: Secuencia de envío.

SND.UNA

Secuencia a la izquierda.

SND.UP
: Puntero urgente de envío.

SND.WL1
: Número de secuencia del segmento utilizado en la última actualización de la ventana.

SND.WL2
: Número de acuse de recibo del segmento utilizado en la última actualización de la ventana.

SND.WND
: Ventana de envío.

SYN
: Un bit de control del segmento entrante, que ocupa un número de secuencia, y que se utiliza en el inicio de una conexión para indicar dónde empezará la numeración secuencial.

TCP
: Bloque de control de la transmisión ('Transmission control block'), la estructura de datos que registra el estado de una conexión.

TCB.PRC
: La prioridad de una conexión

TCP
: Procotocolo de control de transmisión ('Transmission Control Protocol': un protocolo de 'host' a 'host' para las comunicaciones fiables en entornos de inter-redes.

TOS
: Tipo de servicio ('Type of Service'), un campo del protocolo de internet.

Tipo de servicio
: Un campo del protocolo de internet que indica el tipo de servicio para este fragmento de internet.

URG
: Un bit de control (urgente), que no ocupa posición en el espacio de números de secuencia y que se utiliza para indicar que se le debería notificar al usuario receptor que realice un procesamiento urgente tan pronto como los datos le vayan

siendo entregados mientras que sus números de secuencia sean menores que los indicados en el puntero urgente.

REFERENCIAS

[1] Cerf, V., and R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol. COM-22, N°. 5, págs 637-648, Mayo de 1974.

[2] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, USC/Information Sciences Institute, Septiembre de 1981. (N.T. Versión en castellano por P.J. Ponce de León: "Protocolo Internet", Mayo de 1999)

[3] Dalal, Y. and C. Sunshine, "Connection Management in Transport Protocols", Computer Networks, Vol. 2, No. 6, pp. 454-473, Diciembre de 1978.

[4] Postel, J., "Assigned Numbers", RFC 790, USC/Information Sciences Institute, September de 1981.

Nota del traductor

Este documento y las traducciones al español mencionadas en las referencias pueden encontrarse en:
http://lucas.hispalinux.es/htmls/estandares.html

El proyecto de traducción de RFC al español tiene su web de desarrollo en:
http://www.rfc-es.org