TCP (TRANSMISSION CONTROL PROTOCOL)

TCP (Transmission Control Protocol) se diseñó específicamente para proporcionar un flujo confiable de bytes de extremo a extremo sobre una red no confiable. En una internet, es posible tener diferentes topologías, anchos de banda, retardos, etc. TCP tiene un diseño que se adapta dinámicamente a las propiedades de la internet y que se sobrepone a diversas fallas [8]. TCP es un protocolo orientado a la conexión, que garantiza la entrega de los paquetes [1].

TCP es definido formalmente en el RFC 793 [4]. Cada máquina que soporta TCP tiene una entidad de transporte que maneja flujos e interactúa con la capa IP. Una entidad TCP acepta flujos de datos de usuarios de procesos locales, los divide en fragmentos que no excedan los 64KB, y envía cada fragmento como un datagrama IP independiente. Cuando los datagramas que contienen datos llegan a una máquina, se pasan a la entidad que reconstruye los flujos de bytes originales [8].

La capa IP no ofrece ninguna garantía de que los datagramas se entregarán de manera apropiada, por lo que corresponde a TCP proporcionar la confiabilidad que la mayoría de los usuarios desean y que protocolos de capas inferiores no facilitan [8].

Para cumplir con esas garantías de entrega, TCP hace uso de mecanismos de conexión, control de flujo y errores, y control de congestión, que se discuten a lo largo de este documento; así como aspectos más básicos del protocolo, como lo son el formato del segmento y su máquina de estados.

1 El protocolo TCP

El servicio de flujo confiable provisto por TCP es tan importante que todo el grupo de protocolos se conoce como TCP/IP. A continuación se describen sus principales características.

1.1 Servicios TCP

TCP provee un servicio de flujo de bytes confiable orientado a la conexión [7].El término *orientado a la conexión* implica que las aplicaciones que usan TCP deben establecer una conexión previo al intercambio de datos [7].

TCP brinda *confiabilidad* cumpliendo lo siguiente [7]:

- Divide los datos de aplicación en trozos para su envío. La unidad generada es denominada segmento.
- Cuando TCP envía un segmento mantiene un temporizador en espera de la confirmación de recepción. De no ser recibida la confirmación, se retransmite el segmento.

- Cuando se reciben datos del otro extremo, se envía una confirmación.
- TCP mantiene una suma de comprobación sobre su cabecera y carga útil, para verificar la integridad de los datos recibidos. Si el segmento recibido es inválido, se descarta el segmento y no se envía una confirmación.
- En la recepción, TCP reordena los segmentos que llegan fuera de orden, y descarta los segmentos duplicados.
- TCP también brinda control de flujo.

TCP envía un flujo de bytes continuo, sin importar la forma en que se escriben los datos en la aplicación. Además, no interpreta el contenido de esos bytes; esto es tarea de las aplicaciones de la capa superior en los extremos de la conexión.

1.2 Segmento

TCP usa un único tipo de PDU denominado *segmento*. La cabecera se muestra en la Figura 4.1 [5].

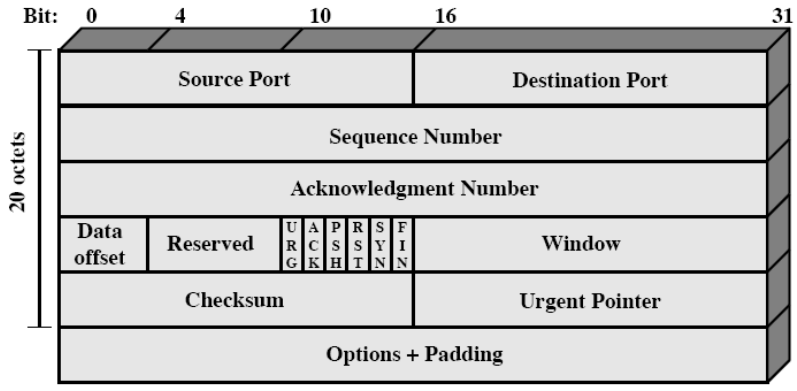

Figura 4.1. Formato de la cabecera del segmento TCP

Los campos son [4][5]:

- Source port. Campo de 16 bits que identifica el puerto fuente.
- Destination port. Campo de 16 bits que identifica el puerto destino.
- Sequence number. Campo de 32 bits que indica el número de secuencia del primer octeto de datos en este segmento, excepto cuando está presente la bandera SYN. Si la bandera SYN está activa, se trata del ISN (Initial Sequence Number) y el primer octeto de datos es ISN + 1.
- Acknowledgement number. Campo de 32 bits que contiene la información del número de secuencia del siguiente octeto que la entidad TCP espera recibir.

- Data offset. Campo de 4 bits que indica el número de palabras de 32 bits de la cabecera. Algunos autores lo denominan HLEN, refiriéndose a que el campo indica la longitud de la cabecera [2].
- Reserved. Campo de 6 bits reservado para uso futuro.
- Flags. 6 bits bandera que de estar activos (valor 1) indican:
    - URG. El campo urgent pointer es válido.
    - ACK. El campo acknowledgement number es válido.
    - PSH. Función de forzado. Este segmento solicita una operación PUSH.
    - RST. Reiniciar la conexión.
    - SYN. Sincronizar los números de secuencia.
    - FIN. El emisor no enviará más datos.
- Window. Campo de 16 bits que se usa para especificar el tamaño de los buffers, y de esta manera informar cuántos datos está dispuesto a aceptar la entidad.
- Checksum. 16 bits usados para el control de errores, el cálculo incluye todo el segmento más una pseudocabecera.
- Urgent pointer. Campo de 16 bits. Este valor, cuando se suma al número de secuencia del segmento, contiene el número de secuencia del último octeto de la cadena de datos urgentes. Esto permite al receptor conocer la cantidad de datos urgentes recibidos.
- Options. El campo de opciones es de longitud variable, y contiene campos cuya presencia no es obligatoria dentro del segmento.

1.2.1 Opciones TCP

El campo de opciones ocupa el espacio al final de la cabecera del segmento y tienen una longitud múltiplo de 8 bits. Todas las opciones deben incluirse en el cálculo del checksum. Las opciones pueden especificarse en dos formatos [4]:

1 En forma de un único byte por opción.

2 En forma de 3 campos: tipo de opción (un octeto), longitud de la opción (un octeto), y los datos.

La longitud incluye los dos octetos de tipo y longitud de la opción. Existe la posibilidad de que la lista de opciones sea de menor longitud que lo indicado por el campo data offset (no sea múltiplo

de 32 bits). El contenido de la cabecera posterior al final de las opciones debe ser rellenado (padding) para completar el siguiente múltiplo de 32 bits [4].

Algunas opciones definidas actualmente se listan en la Tabla 4.1 [4].

Tipo	Longitud	Significado	Comentario
0	-	End of Option List	Fin de la lista de opciones
1	-	No Operation	Puede usarse para alinear la siguiente opción a frontera de palabra
2	4	MSS	Tamaño máximo de segmento

Tabla 4.1. Opciones TCP

El campo de opción más común es MSS (Maximun Segment Size), de 16 bits. Cada extremo de la conexión especifica esta opción en el primer segmento que se intercambia, para indicar el tamaño máximo del segmento que el emisor desea recibir [7].

1.3 Máquina de estado

La operación de TCP se puede explicar mejor mediante un modelo teórico llamado *máquina de estado finito*. La Figura 4.2 [4] muestra un subconjunto de la máquina de estado de TCP, en la que los rectángulos representan estados y las flechas transiciones. El nombre en cada transición muestra qué recibe TCP para que se genere esa transición y qué envía como respuesta. Por ejemplo, la entidad TCP en cada extremo comienza en un estado CLOSED. El programa de aplicación debe emitir un comando *passive open* en espera de una conexión desde otro extremo, o un comando *active open* para iniciar una conexión. El comando active open obliga la transición al estado SYN SENT. Cuando TCP continúa con la transición, emite un segmento SYN. Cuando el otro extremo devuelve un segmento SYN+ACK, TCP cambia al estado ESTABLISHED y comienza la transferencia de los datos [2].

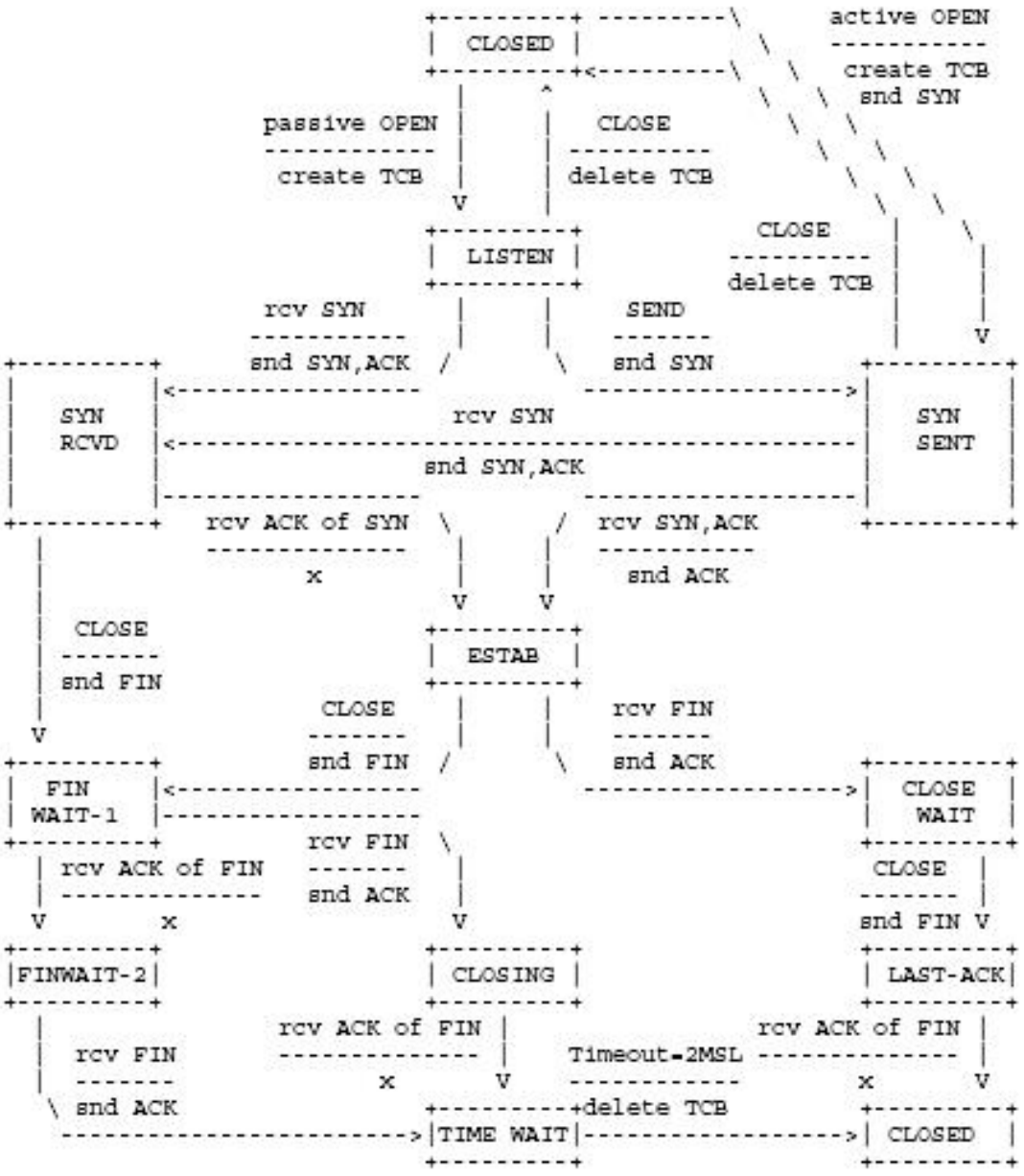

Figura 4.2. Máquina de estados TCP[1]

2. Administración de la conexión

TCP permite que varias aplicaciones en una máquina se comuniquen de forma concurrente y realiza el demultiplexado del tráfico entrante en el destino. Para lograr esto, TCP hace uso de los números de puertos para identificar el extremo final (proceso) dentro de una máquina [2].

[1] TCB: Transmission Control Block, estructura de datos que almacena el estado de la conexión.

TCP utiliza la conexión, y no sólo los números de puerto como su abstracción fundamental; las conexiones se identifican por medio de un par de puntos extremos. Un punto extremo es una dupla (host, puerto), donde el host es identificado por la dirección de red (IP) y el puerto TCP del proceso en cuestión [2].

2.1 Establecimiento de la conexión

TCP transmite datos de modo full dúplex. Esto implica que cada extremo debe iniciar una conexión y obtener aprobación del otro extremo antes de transmitir datos [3].

2.1.1 Three-way Handshaking

El establecimiento de la conexión en TCP es denominado *three-way handshaking*. El proceso inicia cuando uno de los extremos indica al sistema operativo que está listo para aceptar conexiones. El otro extremo hace la solicitud de conexión indicando la dupla que identifica al extremo con el que se quiere conectar. Entonces se comienza el intercambio de segmentos [1][3]. El proceso se describe en la Figura 4.3 [5].

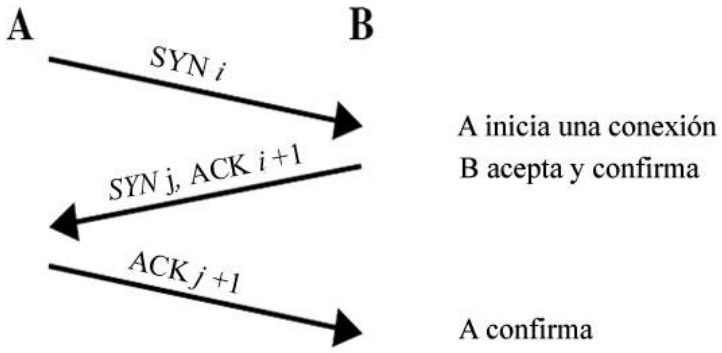

Figura 4.3. Three-way Handshaking

2.1.2 Apertura pasiva y activa

Para establecer la conexión, el programa de aplicación puede realizar una operación de apertura *pasiva* al notificar al sistema operativo que aceptará una conexión entrante; entonces se le asigna un número de puerto para esa conexión. El programa de aplicación del otro extremo debe comunicar al sistema operativo su solicitud de apertura *activa* para establecer una conexión. Una vez que se establece la conexión las aplicaciones pueden comenzar a transmitir los datos [2].

2.1.3 Apertura simultánea

Es posible, aunque poco probable, que dos aplicaciones realicen una apertura activa a la vez. Cada extremo debe enviar un segmento SYN, y los SYNs deben cruzarse en su camino en la red. También es necesario que cada extremo tenga un número de puerto local bien conocido hacia el otro extremo. Esto es llamado una apertura *simultánea*. TCP fue diseñado para manejar aperturas

simultáneas y la regla es que sólo una conexión resulta de este proceso, y no dos [7]. En este caso, ambos extremos cumplen el papel de cliente y de servidor.

2.2 Mantenimiento de la conexión

Aunque los datos se transmiten en segmentos sobre una conexión de transporte, la transmisión se ve desde un punto de vista lógico como un flujo de octetos. Por tanto, cada octeto es numerado módulo 2^{32}. Cada segmento contiene el número de secuencia del primer octeto en el campo de datos. El control de flujo se ejerce usando un mecanismo de créditos (descrito en la sección 3.2) en el cual el crédito es un número de octetos en lugar de un número de segmentos [5].

La entidad de transporte almacena temporalmente los datos, tanto en transmisión como en recepción. TCP aplica su propio criterio para decidir cuándo construir un segmento para transmitirlo y cuándo entregar los datos recibidos al usuario. Si durante el intercambio de datos llega un segmento que aparentemente no va dirigido a la conexión actual, se envía un segmento con el indicador RST activado [5].

2.3 Finalización de la conexión

Cualquiera de los dos extremos de la conexión puede cerrarla. Mientras que para iniciar una conexión son necesarios 3 segmentos, para el cierre es necesario el intercambio de 4 segmentos. Esto se debe al *half close* de TCP. Debido a que la conexión es full dúplex, cada dirección debe ser cerrada de forma independiente. Cada extremo puede enviar un FIN cuando haya culminado su transmisión de datos, indicando que no habrá más flujo de datos en esa dirección, más no en la contraria. El extremo que envía el primer FIN realiza un cierre activo, y el otro extremo realiza un cierre pasivo [7].

Cuando un extremo recibe un FIN responde con un ACK. Posteriormente este extremo (el que envía el ACK) envía un FIN para cerrar el otro sentido de la conexión, el cual debe ser confirmado mediante un ACK [7].

2.3.1 Cierre pasivo y activo

Se dice que el extremo que solicita en primera instancia el cierre de la conexión realiza el cierre *activo*, y el otro extremo realiza el cierre *pasivo* [7].

2.3.2 Cierre simultáneo

Es posible que ambos extremos realicen un cierre activo, ejecutándose un cierre *simultáneo*. En este caso se intercambia el mismo número de mensajes que en un cierre normal [7].

3. Control de Flujo

TCP usa el mecanismo de ventanas deslizantes para controlar el flujo. Su implementación difiere de otras en que desacopla la confirmación de las unidades de datos recibidas de la concesión de

permiso para el envío de unidades adicionales. Esto implica que en TCP uno de los extremos puede confirmar datos entrantes sin conceder permiso para enviar nuevos datos [6].

3.1 Ventanas deslizantes

La técnica de ventanas deslizantes es una forma más compleja de confirmaciones positivas que el simple mecanismo de Stop and Wait[2]. Las ventanas deslizantes hacen un uso más eficiente del ancho de banda, ya que permiten la transmisión de varios segmentos antes de la recepción de una confirmación [2].

3.2 Asignación de créditos

En este mecanismo se considera que cada byte individual que se transmite tiene un número de secuencia. Además de los datos, cada segmento incluye en la cabecera 3 campos relacionados con el control de flujo: secuence number (SN), acknowledgement number (AN) y window (W). Cuando una entidad TCP envía un segmento, incluye el número de secuencia del primer octeto en el campo de datos del segmento. Una entidad TCP confirma un segmento devolviendo otro que incluya (AN = *i*, W = *j*) con la siguiente interpretación [6]:

- Se confirman todos los octetos hasta el número de secuencia SN = *i* -1; se espera el octeto con el número de secuencia *i*.
- Se concede permiso para enviar una ventana de W = *j* octetos de datos.

3.3 Efecto del tamaño de la ventana en el rendimiento

El rendimiento de la transmisión depende del tamaño de la ventana, del retardo de propagación y de la velocidad de transmisión de los datos. Se plantea la siguiente formulación [6]:

W = tamaño de la ventana TCP (en octetos)

R = velocidad de transmisión de datos (bps) en la fuente TCP

D = retardo de propagación (seg) entre el origen y el destino TCP en una conexión dada

En aras de mantener la simplicidad, se ignoran los bits de sobrecarga del segmento. Suponga que una entidad TCP comienza a transmitir un flujo de bytes por una conexión. Al primer octeto le tomará *D* segundos llegar al destino, y se necesitan otros *D* segundos adicionales para que llegue su confirmación. Durante ese tiempo la fuente (si no está limitada) puede transmitir un total de 2*RD* bits, o *RD*/4 octetos. De hecho, la fuente está limitada al tamaño de la ventana de W octetos hasta que se recibe una confirmación. Por tanto, si W > *RD*/4, se alcanzará el máximo rendimiento

[2] Stop and Wait es un mecanismo que impide la transmisión de un segmento hasta tanto no se reciba la confirmación de recepción del segmento anterior.

posible. Si W < *RD*/4, el máximo rendimiento alcanzable es el cociente de W entre *RD*/4. Así pues, el rendimiento normalizado S se puede expresar como [6]:

$$S = \begin{cases} 1 & , W \geq RD/4 \\ 4W/RD & , W < RD/4 \end{cases}$$

Ecuación 4.1.

4. Control de Errores

TCP hace uso de BEC (Backward Error Correction) para el control de errores. El primer paso para el control de errores es la detección. Para esto, TCP hace uso de la técnica de checksum a través del campo destinado para ello en la cabecera.

4.1 Checksum de TCP

El checksum de TCP cubre la cabecera, el payload y una pseudo-cabecera de 12 bytes. Esta pseudo-cabecera incluye algunos campos de la cabecera IP. Si el receptor detecta un error, el segmento es descartado y no se genera un mensaje de error [7].

4.2 Estrategia de Retransmisión

En TCP no hay confirmaciones negativas explicitas. Por el contrario, se apoya exclusivamente en confirmaciones positivas y en la retransmisión, cuando no llega una confirmación dentro de un plazo de tiempo [6].

Hay dos sucesos que hacen necesaria la retransmisión de un segmento. En primer lugar éste puede resultar dañado en tránsito, y pese a ello llegar a su destino. En este caso, el checksum de la cabecera permite detectar el error y descartar el segmento. La contingencia más común es que el segmento no llegue. En cualquier caso, la entidad emisora no sabe que la transmisión del segmento no ha tenido éxito.

4.2.1 Tiempo límite y retransmisión

Para gestionar estas situaciones, debe existir un temporizador asociado a cada segmento que se envía. Si este expira antes de recibir una confirmación para ese segmento, el emisor debe retransmitirlo [6].

Un aspecto clave de diseño en TCP es el valor del temporizador de retransmisión. Si el valor es muy pequeño habrá muchas retransmisiones innecesarias; si el valor es demasiado grande el protocolo será muy perezoso en la respuesta a la pérdida de un segmento. El temporizador se

debería activar con un tiempo algo superior al RTT (Round Trip Time). Por supuesto, ese retardo es variable, incluso bajo condiciones de carga constante [6].

Dos estrategias son evidentes. La primera consiste en usar un temporizador fijo, basado en el conocimiento típico de la red. La segunda consiste en un mecanismo adaptativo, que se explica a continuación.

4.2.2 Temporizador de retransmisiones adaptativo

Virtualmente todas las implementaciones de TCP tratan de estimar el RTT actual observando el patrón del retardo de los segmentos recientes, y estableciendo en el temporizador un valor algo mayor que ese estimado [6].

Un enfoque consistiría en tomar simplemente la media de los RTTs observados para un número de segmentos. Si la media predice con precisión los futuros RTTs, el plazo de temporización de retransmisiones ofrecerá un buen rendimiento.

En una media común se otorga el mismo peso a cada término. En condiciones normales, sería deseable otorgar mayor peso a las instancias más recientes, puesto que es más probable que reflejen el comportamiento futuro. Una técnica común para este cálculo es el uso de promedios pesados, donde las mediciones recientes poseen un peso mayor a las anteriores [6].

5. Opciones de políticas en la implementación de TCP

TCP proporciona una especificación precisa del protocolo a usar entre entidades. Sin embargo, ciertos aspectos del protocolo admiten varias opciones de implementación. Aunque dos implementaciones que escojan opciones alternativas serán interoperables, habrá consideraciones que atañen al rendimiento. Las tareas de diseño para las cuales se han especificado opciones son [6]:

- Política de envíos. En ausencia de envío forzado (PUSH) y de una ventana de transmisión cerrada, una entidad TCP emisora es libre de transmitir datos según su propia conveniencia. TCP puede construir un segmento para cada grupo de datos proporcionados por el usuario, o puede esperar hasta que haya una cierta cantidad de datos acumulados antes de construir y enviar un segmento.

- Política de entregas. En ausencia de envío forzado (PUSH), una entidad TCP receptora es libre de entregar los datos al usuario a su propia conveniencia. Puede entregar los datos conforme se reciben segmentos en secuencia, o puede almacenar los datos de varios segmentos en el buffer de recepción antes de su entrega.

- Política de aceptación. Cuando en una conexión TCP llegan en orden todos los segmentos de datos, estos se colocan en un buffer de recepción para su entrega al usuario. Sin

embargo, es posible que lleguen segmentos desordenados. En este caso, la entidad TCP receptora tiene dos opciones:

  - En-orden. Acepta sólo los segmentos que llegan en secuencia.
  - En-ventana. Se aceptan todos los segmentos que estén dentro de la ventana de recepción.

- Política de retransmisión. TCP gestiona una cola de segmentos que se han enviado, pero que aún no se han confirmado. La especificación de TCP indica que se retransmitirá un segmento si no se recibe confirmación para el mismo en un plazo de tiempo dado. Una implementación de TCP puede utilizar una de estas estrategias de retransmisión:
  - Sólo el primero. Gestiona un temporizador de retransmisión para toda la cola. Si se recibe una confirmación, se eliminan los segmentos apropiados de la cola y se reinicia el temporizador. Si vence el plazo de tiempo, se retransmite el segmento situado al principio de la cola y se reinicia el temporizador.
  - Batch (por lotes). Se gestiona un temporizador de retransmisión para toda la cola. Si se recibe una confirmación, se eliminan los segmentos apropiados de la cola y se reinicia el temporizador. Si vence el plazo de tiempo, se retransmiten todos los segmentos de la cola y se reinicia el temporizador.
  - Individual. Se gestiona un temporizador de retransmisión para cada segmento de la cola. Si se recibe una confirmación, se eliminan los segmentos apropiados de la cola y se destruyen los temporizadores. Si vence algún plazo de tiempo, se retransmite de forma individual el segmento correspondiente y se reinicia su temporizador.
- Política de confirmaciones. Cuando llega un segmento de datos en secuencia, la entidad TCP receptora tiene dos opciones en relación a la temporización de la confirmación:
  - Inmediata. Cuando se aceptan los datos, se transmite de inmediato un segmento vacío que contiene la confirmación apropiada.
  - Acumulativa. Cuando se aceptan los datos, se registra la necesidad de confirmación, pero se espera un segmento saliente con datos en el que superponer una confirmación.

En cada caso, la política real dependerá de consideraciones de rendimiento.

6. Control de congestión

Dentro de una red conmutada, se puede usar el enrutamiento dinámico para ayudar a aliviar la congestión, los algoritmos de enrutamiento pueden distribuir la carga entre routers y redes para aliviar la congestión. Sin embargo, estas medidas sólo son efectivas para cargas no equilibradas y breves repuntes de tráfico. En última instancia, la congestión sólo se puede controlar limitando la cantidad de tráfico total que entra en la red [6].

El mecanismo de control de flujo basado en créditos de TCP se diseñó para permitir que el destino restrinja el flujo de segmentos de una fuente y así evitar la saturación de la memoria temporal del destino. Este mismo mecanismo se puede usar para identificar el comienzo de la congestión y reaccionar mediante la reducción del flujo de datos. Si varias estaciones operan de esta forma, la congestión de la red se puede aliviar [5].

Varias técnicas han sido implementadas para mejorar las características del control de congestión en TCP. Las técnicas se pueden agrupar en dos categorías: gestión de temporizadores de retransmisión y gestión de la ventana.

6.1 Gestión del temporizador de retransmisión

Las primeras técnicas a evaluar se ocupan del cálculo del valor del temporizador de retransmisiones (RTO). El valor de este temporizador puede tener un efecto crítico en la capacidad de reacción de TCP frente a la congestión. Las técnicas se describen a continuación [6].

6.1.1 Estimación de la varianza del RTT (algoritmo de Jacobson)

La técnica especificada en el estándar de TCP, descrita en la sección 4.2.2, permite a una entidad adaptarse a los cambios del RTT. Sin embargo, no es adecuada en situaciones en las que el RTT exhibe una varianza relativamente alta [6].

Un enfoque más efectivo consiste en estimar la variabilidad de los valores del RTT y usarlos como entrada en el cálculo del RTO. Una posibilidad sería calcular la desviación estándar de la muestra. Sin embargo, esto implica el cálculo de un cuadrado y de una raíz cuadrada. Una medida de variación que es más sencilla de estimar es la desviación media, que se calcula en base a la diferencia en valor absoluto de los valores a la media.

Jacobson, que propuso el uso de una estimación dinámica de la variabilidad para estimar el RTT, sugiere el uso de la misma técnica de suavizado que se usa para el cálculo del RTT [6].

6.1.2 Retraso exponencial del RTO

Cuando en un emisor vence el plazo de tiempo de un segmento este se debe retransmitir. El RFC 793 [4] asume que se usará el mismo valor de RTO para este segmento retransmitido. Sin embargo, puesto que el vencimiento del plazo se debe probablemente a la congestión de la red, no es nada aconsejable mantener el mismo valor de RTO [6].

Una opción consiste en instar a la fuente a incrementar su RTO cada vez que se retransmite el mismo segmento; esto a veces se conoce como *retroceso*. Una sencilla técnica para implementar el retroceso del RTO consiste en multiplicar el RTO de un segmento por un valor constante en cada retransmisión.

6.1.3 Algoritmo de Karn

Si no se retransmite ningún segmento el proceso de muestreo para el algoritmo de Jacobson es sencillo, se puede incluir el RTT de cada segmento en el cálculo. Sin embargo, suponiendo que vence el plazo de tiempo de un segmento y se debe retransmitir, si se recibe una confirmación hay dos posibilidades:

1. Se trata del ACK de la primera transmisión del segmento.
2. Se trata del ACK de la segunda transmisión.

La fuente no puede distinguir entre estos dos casos. Si el segundo caso es cierto, y la entidad mide el RTT desde la primera transmisión hasta la llegada del ACK, el tiempo medido será demasiado grande. El RTT medido será del orden del RTT actual más el RTO. Si se introduce este RTT falso en el algoritmo de Jacobson se producirá un valor de SRTT innecesariamente elevado, e igual con el RTO.

Peor aún sería medir el RTT desde la segunda transmisión hasta la recepción del ACK. Si de hecho se trata del ACK de la primera transmisión, el RTT medido será demasiado pequeño, lo que causaría la obtención de un valor demasiado bajo para el SRTT y el RTO.

El algoritmo de Karn resuelve este problema con las siguientes reglas [6]:

1. No usar el RTT medido para un segmento retransmitido para actualizar el RTT suavizado.
2. Calcular el retroceso del RTO cuando se produzca una retransmisión.
3. Usar el valor del retroceso del RTO para los segmentos sucesivos hasta que llegue una confirmación de un segmento que no se haya retransmitido.

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

6.2 Gestión de la ventana

El tamaño de la ventana de emisión puede tener un efecto decisivo para que TCP pueda ser utilizado eficientemente sin causar congestión [5].

6.2.1 Arranque lento

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

Una estrategia que se podría seguir consiste en que el emisor empezara a enviar con una ventana relativamente grande pero no con su máximo tamaño, esperando aproximarse al tamaño máximo proporcionado por la conexión. Este esquema es arriesgado, ya que el emisor podría inundar la red con muchos segmentos antes de darse cuenta por los temporizadores de que el flujo era excesivo. En lugar de eso, se necesita algún medio para expandir gradualmente la ventana hasta que se reciban las confirmaciones. Este es el propósito del mecanismo de arranque lento [5].

Este mecanismo sondea la red para asegurarse de que la entidad TCP no esté enviando demasiados segmentos en un entorno ya congestionado. Conforme van llegando las confirmaciones, se le permite abrir la ventana hasta que el flujo se controle mediante las confirmaciones que se reciben [5].

7. Referencias

1. Blank, A. TCP/IP Foundations. Sybex. 2004.
2. Comer, D. TCP/IP: Principios básicos, protocolos y arquitectura. Prentice Hall. 3ª ed. 1996.
3. Forouzan, B. Data Communications and Networking. Mc Graw Hill. 4th ed. 2007.
4. DARPA. Transmission Control Protocol. RFC 793. 1981.
5. Stallings, W. Comunicaciones y Redes de Computadores. Prentice Hall. 7ª ed. 2004.
6. Stallings, W. Redes e Internet de Alta Velocidad: Rendimiento y Calidad de Servicio. Prentice Hall. 2ª ed. 2005.
7. Stevens, R. TCP/IP Illustrated, Volume 1, The Protocols. Addison Wesley. 1994.
8. Tanenbaum, A. Redes de Computadores. Prentice Hall. 4ª ed. 2003.