Capítulo 13

Control de congestión

Al terminar este capítulo, entenderás:

- Qué es la congestión y cuáles son sus causas.
- Por qué la congestión es un problema grave.
- Qué técnicas se pueden aplicar para prevenir o mitigar la congestión.
- Qué técnicas de control de congestión utiliza TCP y cómo funcionan.

Intuitivamente una congestión es una situación en la que la demanda de recursos supera la capacidad del sistema para proporcionarlos. En el caso de las redes de conmutación de paquetes, la congestión se produce cuando la carga, es decir, la cantidad de paquetes que se introduce en la red, supera su capacidad para manejarlos adecuadamente y llevarlos a su destino.

En este capítulo veremos las causas de la congestión, sus consecuencias y las técnicas que se pueden aplicar para prevenirla o bien controlarla una vez que se ha producido.

13.1. Carga, capacidad y congestión

Podemos cuantificar la **carga** de la subred como la cantidad de paquetes que se encuentran en tránsito en un determinado momento. La subred está formada por lineas de comunicaciones y dispositivos de intercambio, de modo que los bytes que forman los paquetes pueden estar físicamente viajando a través de un enlace o bien en las colas de entrada y salida de los routers. Desde este punto de vista, podemos entender la subred como una especie de almacenamiento temporal, una 'memoria' que contiene paquetes durante el tiempo que transcurre desde emisor los envía envían hasta que son entregados a su destino. La cantidad máxima de paquetes que cabe en esta 'memoria' la podemos entender, por analogía, como la **capacidad** de la subred.

Por simplicidad, hemos hablado de paquetes, pero obviamente esos paquetes son de tamaño muy variado, y lógicamente para una misma cantidad de paquetes, la carga y la capacidad son mayores si hablamos de paquetes de mayor tamaño. Así que, para obtener un cálculo más preciso de carga y capacidad, habría que medirlas en bytes.

Una vez están claros estos conceptos, podemos entender la **congestión** como toda situación en la que la carga supera la capacidad de la subred (ver fórmula 13.1), es decir, se produce una *sobrecarga* de la subred.

$$\text{congestión} = \text{carga} > \text{capacidad} \tag{13.1}$$

Hemos hablado de las colas de entrada y salida de los routers, pero es necesario analizar esta cuestión con detalle. Los paquetes llegan al router a través de las interfaces de red y se almacenan en las colas de entrada, asociadas a cada interfaz. El componente del router que realiza el reenvío (ver § 8.1) va tomando y procesando paquetes desde la cabeza de cada cola. Eso significa que cuantos más paquetes entran al router, mayor es la ocupación de las colas de entrada y más tiempo pasan los paquetes en ellas, lo que aumenta la latencia. Si cuando llega un nuevo paquete, la cola ya está llena, el router lo descarta[1]. Puedes ver una representación de esta situación en la Figura 13.1. La cola de entrada de la interfaz superior está llena y no puede aceptar nuevos paquetes.

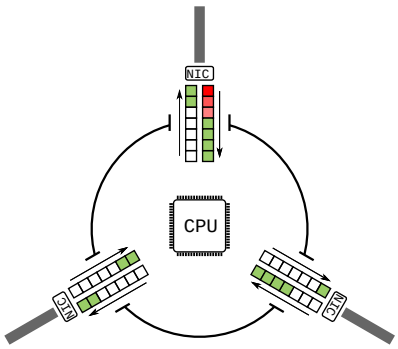

FIGURA 13.1: Esquema de las colas de entrada y salida de un router

Una vez procesado, el paquete se coloca en la cola de salida asociada a la interfaz de red por la que será enviado. La interfaz de salida necesita cierto tiempo para enviar un paquete, por lo que si la velocidad con la que el router reenvía paquetes por esa interfaz (los encola para su envío) es

[1] A menos que el router aplique una política de descarte selectivo

mayor que la velocidad con la que se transmiten al medio, la cola de salida también se puede llegar a llenar y también habrá paquetes descartados por ese motivo.

Como puede verse, cuando la carga se acerca o supera la capacidad de la subred, los routers empezarán a descartar paquetes, que es precisamente la consecuencia más grave de la congestión. Pero hay otros efectos negativos, como el incremento de la latencia, que pueden ocurrir mucho antes. La Figura 13.2 muestra cómo la latencia aumenta con la carga y el impacto que tiene sobre el rendimiento al acercarse la carga a la capacidad de la subred. El rendimiento puede medirse como la cantidad de paquetes que la subred es capaz de llevar a su destino por unidad de tiempo.

Quizá pienses que causa es que las colas del router son demasiado pequeñas. Sin embargo, aumentar el tamaño no soluciona el problema. Si la cola es mayor, los paquetes pasarán más tiempo en ella, aumentando la latencia del flujo y provocando la expiración de los temporizadores de retransmisión, cuando se trate de tráfico confiable. Incluso obviando ese efecto indeseado, mayores colas solo retrasarían el momento en que la congestión se hace patente.

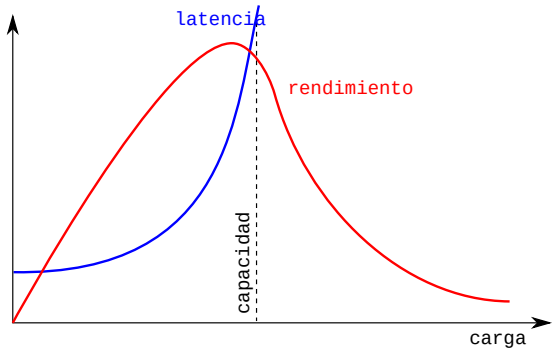

Figura 13.2: Carga, capacidad, latencia y rendimiento

En todo caso, la congestión raramente ocurre a la vez en toda la subred. Suele localizarse en algunos routers o enlaces involucrados en determinadas rutas con más uso en ese momento. Mientras tanto quizá en otras partes todo va bien, aunque de no tomarse medidas, la congestión puede propagarse rápidamente.

13.2. Control de congestión

Por supuesto, la congestión es un problema muy grave que puede inutilizar parte de una interred. La solución es lo que llamamos **control de congestión**, que consiste en aplicar técnicas que permitan evitar, o al menos minimizar, los efectos de la congestión.

Veamos una breve descripción de algunas de las técnicas habituales agrupadas en dos categorías: preventivas y reactivas. En ambos casos se basan en la idea de reducir la tasa de emisión de mensajes, o en menor medida, el tamaño de estos. Reduciendo la cantidad de paquetes que se introduce en la red, lógicamente se reduce la carga y con ello la posibilidad de alcanzar la capacidad. Sin embargo, recuerda que el rendimiento óptimo de la subred está cerca de la capacidad, de modo que el objetivo es buscar el equilibrio, llegar al punto de rendimiento máximo sin sobrepasarlo.

13.2.1. Técnicas preventivas

Como su nombre indica, se aplican para evitar que la congestión llegue a producirse. Tanto los emisores como los receptores pueden colaborar. También se llaman *técnicas de bucle abierto* porque no se realizan medidas ni se obtiene información explícita de lo realmente ocurre en la red.

Política de retransmisión
: Cuando un paquete se pierde, el emisor puede aumentar la duración del temporizador de retransmisión, para que en una situación de congestión, los paquetes se retransmitan más tarde de lo habitual, reduciendo así la tasa de emisión.

Política de confirmación
: Si el receptor estima una situación de congestión puede retrasar el envío de mensajes de confirmación, lo que indirectamente reduce la tasa de emisión. Obviamente este retardo adicional debe estar dentro de los límites que eviten retransmisiones innecesarias, que provocarían un aumento de carga, agravando la congestión. También se puede aprovechar la *confirmación acumulativa* que además reduce el número de mensajes en tránsito.

Política de ventana
: Un protocolo de confiabilidad de tipo *repetición selectiva* o *confirmación negativa* evita tráfico innecesario y puede ayudar e evitar la congestión.

Política de descarte
Los routers pueden decidir qué paquetes resulta más conveniente descartar en caso de congestión. Por ejemplo, descartar paquetes que corresponden a protocolos confiables provocará retransmisiones, y eso no ayuda a resolver el problema. Pueden ser técnicas muy efectivas, pero requieren conocimiento detallado de la topología de la red y de los protocolos de aplicación.

Política de admisión
Determina si la red dispone de recursos suficientes para aceptar un nuevo flujo de datos, rechazándolo si no fuera así. Esta técnica es tremendamente efectiva, pero solo es posible en tecnologías de conmutación de circuitos o circuitos virtuales.

13.2.2. Técnicas reactivas

Se aplican una vez que la congestión ha empezado a producirse y algunos de sus síntomas son perceptibles. Se denominan también *técnicas de bucle cerrado* porque se basan en la retroalimentación de información sobre el estado de la red que procede, bien de los receptores, o de otros dispositivos intermedios.

Backpressure
Se trata de una técnica típica de las redes de circuitos virtuales. Cuando un dispositivo de interconexión (conmutador según su terminología[2]) detecta congestión, puede enviar un mensaje al conmutador inmediatamente anterior para que reduzca su tasa. Esto implicará el descarte de mensajes, de modo que este conmutador enviará a su vez un mensaje de notificación de congestión al conmutador previo y así hasta llegar al dispositivo emisor. Este último reducirá su tasa de emisión y la congestión se irá reduciendo progresivamente.

Paquete de estrangulamiento (*choke*)
Es similar a *backpressure*. Se envía un mensaje específico de notificación de congestión directamente al emisor, mientras que los dispositivos intermedios no son notificados.

Señalización implícita
En este caso el emisor interpreta ciertos indicios en el comportamiento del flujo o conexión como señales de que está ocurriendo un episodio de congestión. Uno de estos indicios podría ser la ausencia o retraso en los mensajes de confirmación.

[2]No confundir con conmutadores Ethernet

Señalización explícita
: Es similar al paquete *choke*, pero no se utiliza un mensaje de notificación de congestión específico, sino que se utilizan campos de los propios mensajes de datos. En algunas tecnologías, como en Frame Relay, esta información podía enviarse tanto al emisor (*backward signaling*) como al receptor (*forward signaling*).

13.3. Control de flujo vs. control de congestión

Ambos, control de flujo y control de congestión se basan ajustar la tasa a la que nuevos datos entran en la red. Sin embargo, su objetivo es muy diferente. El control de flujo (de receptor) intenta evitar que un emisor específico sature a un receptor específico, es decir, es un problema que atañe únicamente a un único flujo de datos (normalmente una conexión). Por otro lado, el control de congestión busca evitar que una subred o parte de ella colapse por sobrecarga de tráfico, es decir, la congestión afecta a muchos flujos de datos simultáneamente, y para resolverla probablemente múltiples emisores tendrán que colaborar reduciendo sus tasas de emisión.

Como son dos problemas independientes pueden ocurrir de forma independiente. Pueden darse situaciones en las que un emisor tenga que aplicar control de flujo porque está saturando a su receptor, y sin embargo, la red se encuentre en condiciones de baja carga. También puede ocurrir lo contrario: que un emisor tenga que aplicar control de congestión porque la red está saturada, pero su receptor no tenga ningún problema con la tasa de llegada de los datos. Y obviamente también pueden coincidir ambos problemas al mismo tiempo.

13.4. Supresión al origen

El protocolo IPv4 ofrecía un mecanismo para control de congestión muy rudimentario mediante el mensaje ICMP *source quench*, que se podría traducir como «enfriamiento en origen».

Cuando un router IP tiene un alto nivel de ocupación en una cola de entrada (señal inequívoca de congestión) y se ve obligado a descartar un paquete, puede enviar un paquete *source quench* al emisor para que reduzca su tasa de emisión. El mensaje incluye como carga útil los primeros 8 bytes (ver § 8.6) del paquete descartado, de modo que el emisor puede identificar a qué flujo corresponde. Conforme a la clasificación anterior, el uso de este tipo de notificación se puede entender como una técnica reactiva similar al *choke packet*.

Un receptor también puede enviar mensajes *source quench* si no puede manejar la tasa de recepción de datos, de modo que en ese caso funciona como un sistema de control de flujo, no de congestión.

Desde hace años el uso de paquetes *source quench* se considera obsoleto por varias razones [15, 30], y de hecho, no se incluyó ningún mecanismo equivalente en IPv6. Las razones más importantes son las siguientes:

- Su eficacia es limitada si el emisor recibe el mensaje tarde, o simplemente el router no lo envía.
- El mecanismo no incluye una medida de la gravedad de la congestión. El emisor debe decidir cuánto y durante cuanto tiempo reducir la tasa después de recibir las notificaciones.
- Un agente malicioso podría enviar notificaciones falsas para reducir la tasa de una víctima consiguiendo así una denegación de servicio (DoS).
- Los mecanismos de control de congestión de TCP son mucho más eficaces.

13.5. Control de congestión en TCP

TCP cuenta con mecanismos de control de congestión muy sofisticados. Se trata de mecanismos de «caja negra», es decir, los elementos intermedios de la red, como los routers, no participan y no tienen que hacer nada específico para que el control de congestión de TCP funcione correctamente.

El concepto clave es la *ventana de congestión* (*congestion window*), abreviado como *cwnd*. La ventana de congestión limita en todo momento la ventana de envío, es decir, controla la tasa de emisión de cada conexión. Claramente esta es una técnica de «política de ventana» según la clasificación que hemos visto en § 13.2. Por supuesto, el valor de la ventana de congestión no es fijo, sino que se recalcula continuamente para adaptarse a las condiciones de la red. Con esto podemos refinar el modo en que se calcula el valor la ventana de envío (ver fórmula 13.2), en contraposición a la versión simplificada que dimos en la fórmula 12.1.

$$\text{swnd} \leq \text{mín}(\text{rwnd}, \text{cwnd}) \tag{13.2}$$

Esta fórmula tan sencilla dice algo interesante: ambos mecanismos, control de flujo y control de congestión, son igualmente importantes y tienen la misma capacidad de limitar la tasa de emisión.

La forma en la que se determina el valor de la ventana de congestión y el comportamiento de las retransmisiones depende de 5 algoritmos:

- Arranque Lento (*Slow Start*)
- Evitación de Congestión (*Congestion Avoidance*)
- Decrecimiento multiplicativo (*Multiplicative Decrease*)
- Retransmisión Rápida (*Fast Retransmit*)
- Recuperación Rápida (*Fast Recovery*)

Veamos cada uno de ellos en las siguientes secciones.

13.5.1. Arranque lento (*slow start*)

El *arranque lento* pretende alcanzar una tasa de emisión significativa partiendo de la mínima, pero con un margen de seguridad que evite provocar congestión. Se aplica en el inicio de la conexión o justo después de un RTO. Aunque el arranque lento es solo una de las fases, es habitual encontrar bibliografía que se refiere a todo el control de congestión de TCP como «arranque lento», probablemente porque en las primeras versiones de TCP era de hecho el único algoritmo que había.

El valor de la ventana de congestión se actualiza en base a rondas. Una «ronda» es el tiempo que transcurre desde que se envían los segmentos asociados a una ventana hasta que comienzan a llegar sus respectivos ACKs, lo que normalmente equivale a la medición e un RTT.

Inmediatamente después de establecerse la conexión, se fija el tamaño de la ventana de congestión (*initial window*) en MSS[3]. Así, el emisor puede empezar enviando un segmento de tamaño MSS. Al llegar su correspondiente confirmación, cwnd crece hasta 2 MSS. Es la segunda ronda puede enviar 2 segmentos de tamaño MSS. Sus respectivas confirmaciones aumentan cwnd hasta 4 MSS, y así sucesivamente. Puedes ver la evolución de cwnd en la Figura 13.3.

Este crecimiento exponencial (2^r, siendo r la ronda) continúa hasta que alcanza *ssthresh* (abreviatura de *slow start threshold*) o bien se detecta congestión. Inicialmente *ssthresh* se fija en un valor alto, habitualmente el tamaño máximo de ventana (2^{16}), pero cuando se detecte congestión, se reducirá sensiblemente.

[3]Esto es una simplificación —de las muchas que aplicaremos en este capítulo— pero sirve para el propósito del libro. Como referencia, las versiones actuales de GNU/Linux suelen fijar un mínimo de cwnd=10 MSS.

Es necesario aclarar que el crecimiento tal como se acaba de explicar solo ocurre si rwnd no es un limitante, es decir, siempre que se cumpla $rwnd > cwnd$. En otro caso, la tasa de emisión estará condicionada por rwnd que a su vez, limitará el crecimiento de cwnd. Si el emisor envía un segmento de, por ejemplo, MSS/2 debido a un valor de rwnd bajo, la confirmación correspondiente aumentará cwnd únicamente en esa cantidad.

En resumen, durante el arranque lento, cwnd aumenta la cantidad de nuevos bytes (N) enviados por cada mensaje de confirmación, siempre que esa cantidad no supere MSS (ver fórmula 13.3). Si efectivamente, los segmentos son de tamaño MSS, cwnd crece a un ritmo MSS por cada confirmación efectiva, o lo que es lo mismo, cwnd se duplica al final de cada ronda.

$$\text{cwnd} \mathrel{+}= \text{mín}(\text{N}, \text{MSS}) \tag{13.3}$$

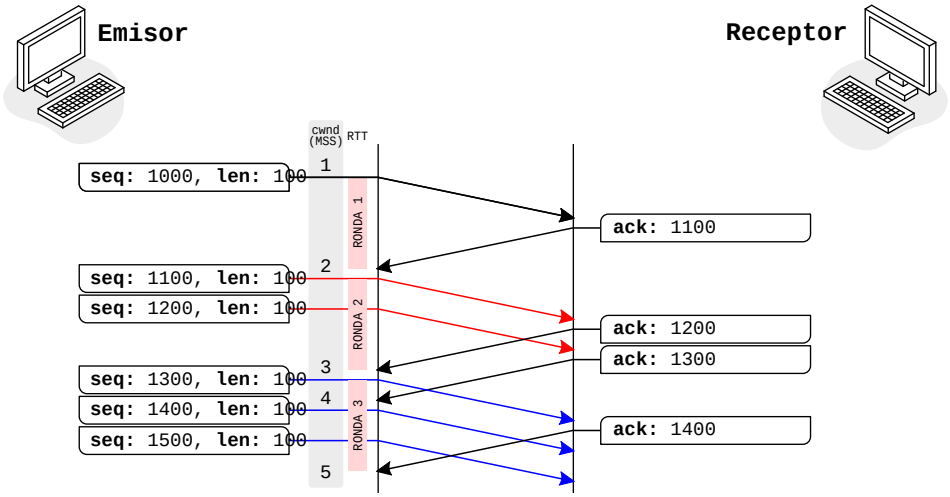

FIGURA 13.3: Evolución de la ventana de congestión durante una fase de arranque lento

Cuando $cwnd > ssthresh$, la fase de arranque lento termina y comienza una fase de *evitación de congestión*.

13.5.2. Evitación de congestión (*congestion avoidance*)

En esta fase cwnd crece de forma lineal según la fórmula 13.4 con cada confirmación neta recibida, que corresponde aproximadamente a una MSS/k siendo k el número de segmentos enviados en esa ronda. Dicho de otro modo, al final de cada ronda, cwnd habrá crece aproximadamente MSS, siempre

que como se indicó antes, rwnd no sea un limitante. La Figura 13.4 muestra un ejemplo de esta fase.

$$\text{cwnd} \mathrel{+}= \text{MSS}^2/\text{cwnd} \tag{13.4}$$

En este caso no se aplica ningún umbral, la ventana continúa creciendo indefinidamente hasta que se detecte algún problema. Sin embargo, el valor de cwnd respecto a *ssthresh* se puede utilizar para determinar en qué fase se encuentra la conexión:

- Si $cwnd < ssthresh$, la conexión está en una fase de arranque lento.
- Si $cwnd > ssthresh$, la conexión está en una fase de evitación de congestión.
- Si $cwnd = ssthresh$, el emisor puede decidir qué algoritmo es más conveniente en cada caso, es decir, en ocasiones puede continuar con arranque lento, mientras que en otras puede cambiar a evitación de congestión.

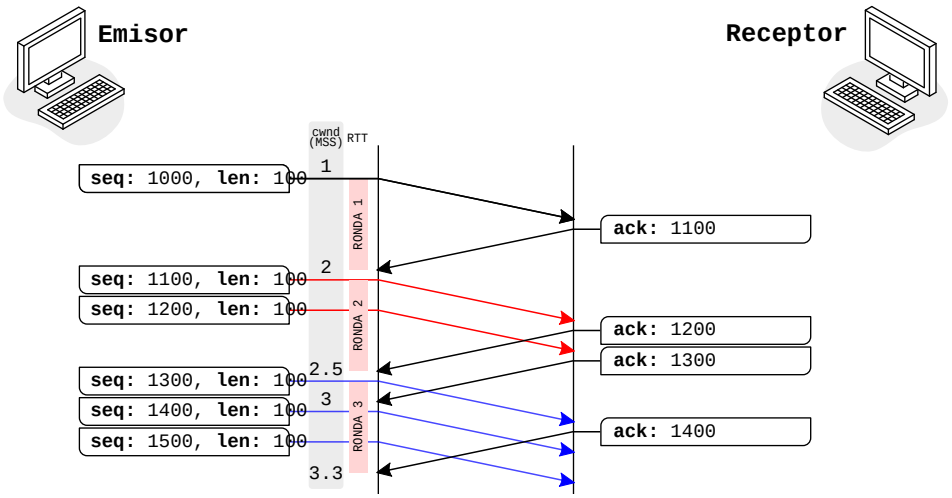

FIGURA 13.4: Evolución de la ventana de congestión durante una fase de evitación de congestión

13.6. Decrecimiento multiplicativo

Esta fase, como indica su nombre, es una reducción drástica de la ventana de congestión que se aplica cuando se detecta congestión. Aunque estaba

diseñada para limitar la tasa durante cierto tiempo, en la actualidad solo se aplica durante una ronda, por lo que en realidad actúa más como transición que como fase estable.

13.6.1. Indicios de congestión

Aunque hemos hablado varias veces de 'detección de congestión', en realidad TCP asume que la conexión está sufriendo los efectos de un episodio de congestión debido a dos *indicios* muy concretos:

- Ha expirado el RTO. TCP asume que la explicación más probable de la expiración del RTO y la correspondiente pérdida de un segmento es que un router ha descartado el paquete debido a la alta ocupación de sus colas, señal inequívoca de congestión.
- El emisor ha recibido 3 confirmaciones duplicadas (*3dupack*[4]), es decir, han llegado un total 4 mensajes de confirmación idénticos desde un receptor al que se le están enviado datos nuevos. Las confirmaciones duplicadas se producen porque el receptor está recibiendo segmentos correctos, pero fuera de secuencia (ver § 12.12.5). La Figura 13.5 muestra un ejemplo. Por supuesto, un emisor podría recibir más de 3 en una ronda, dependiendo de cuántos segmentos la formen.

La detección de la congestión en base a estos indicios se puede interpretar como una técnica de 'señalización implícita' según la clasificación de § 13.2. Estos indicios tienen consecuencias diferentes. La expiración del RTO es un evento más grave y es más probable que sea síntoma de congestión. La aparición de *3dupack* en cambio indica que los mensajes siguientes, correspondientes a la ronda, siguen llegando al destino. Por tanto, la congestión, de haberla, no parece tan grave.

Cuando ocurre una expiración del RTO, la fase en curso termina inmediatamente. Se inicia entonces una nueva fase de arranque lento, y se reajustan los valores de cwnd (fórmula 13.5) y *ssthresh* (fórmula 13.6) del siguiente modo.

$$\text{cwnd} = \text{MSS} \tag{13.5}$$

$$\text{ssthresh} = \text{máx}(2\text{MSS}, \frac{\text{datos en vuelo}}{2}) \tag{13.6}$$

[4]En muchos documentos técnicos, o en código fuente, es habitual encontrar «dupack» para indicar las 3 confirmaciones duplicadas. Aquí utilizaremos el más explícito «3dupack» para evitar cualquier confusión.

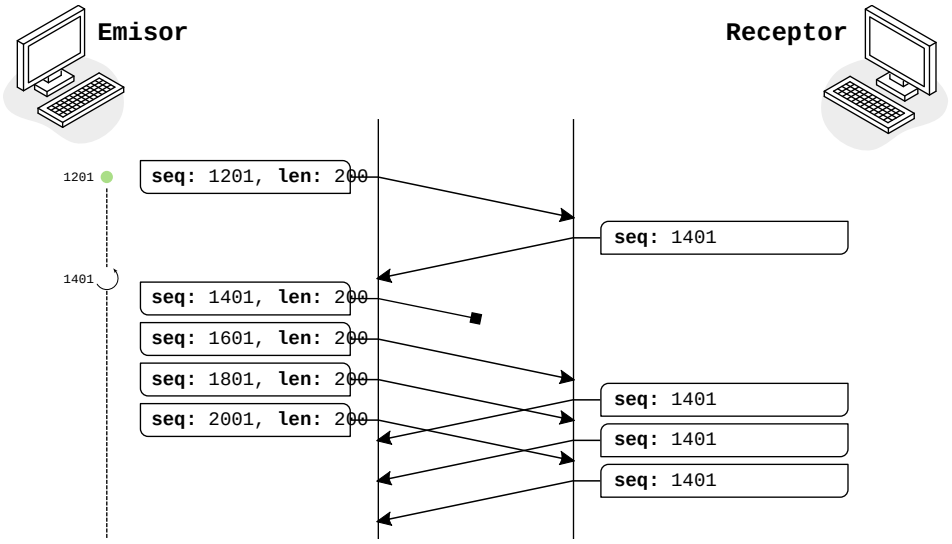

FIGURA 13.5: 3 ACK duplicados, como consecuencia de la pérdida (o retraso) de un segmento

El nuevo valor de *ssthresh* provoca que, en la siguiente fase de arranque lento, el crecimiento de cwnd sea más conservador que en el inicio de la conexión. Por eso, la tasa de emisión se adapta con más suavidad a las condiciones de la red. El parámetro *datos en vuelo* (*FlightSize*) se refiere a la cantidad de datos que el emisor ha enviado y que todavía no han sido confirmados.

Cuando se produce *3dupack*, la fase en curso termina. En ese momento se aplica el algoritmo de 'retransmisión rápida' (*fast retransmit*) que inmediatamente realiza una retransmisión del segmento al que corresponde la confirmación duplicada. Se llama así porque no espera la expiración del RTO.

Fíjate que la aparición de los ACK duplicados no implica necesariamente que el mensaje se haya perdido. Podría acabar llegando y el receptor enviaría una confirmación que, de llegar a tiempo, evitaría la retransmisión.

Después de la retransmisión rápida se aplica el algoritmo de 'recuperación rápida' (*fast recovery*). Primero fija cwnd y *ssthresh* según la fórmula 13.6. Después lleva a cabo un período de «inflación» que incrementa cwnd en 3 MSS para compensar las confirmaciones duplicadas. A partir de ahí, incrementa cwnd en MSS por cada confirmación duplicada adicional. Durante este período, el emisor puede enviar segmentos con datos nuevos. Cuando por fin se reciba la confirmación para el segmento retransmitido, se produce la «deflación», es decir, cwnd vuelve al valor *ssthresh* calculado previamente. La conexión continua en una fase de evitación de congestión.

Los mecanismos de retransmisión y recuperación rápida fueron optimizaciones introducidas en las versiones Tahoe y Reno, respectivamente. Antes de eso (versión Vegas), al recibir *3dupack*, el emisor cambiaba directamente a una fase de evitación de congestión fijando cwnd a $swnd/2$, algo que más tarde se consideró una caída demasiado brusca de la tasa de envío.

La Figura 13.6 muestra el diagrama de transición entre fases según lo explicado.

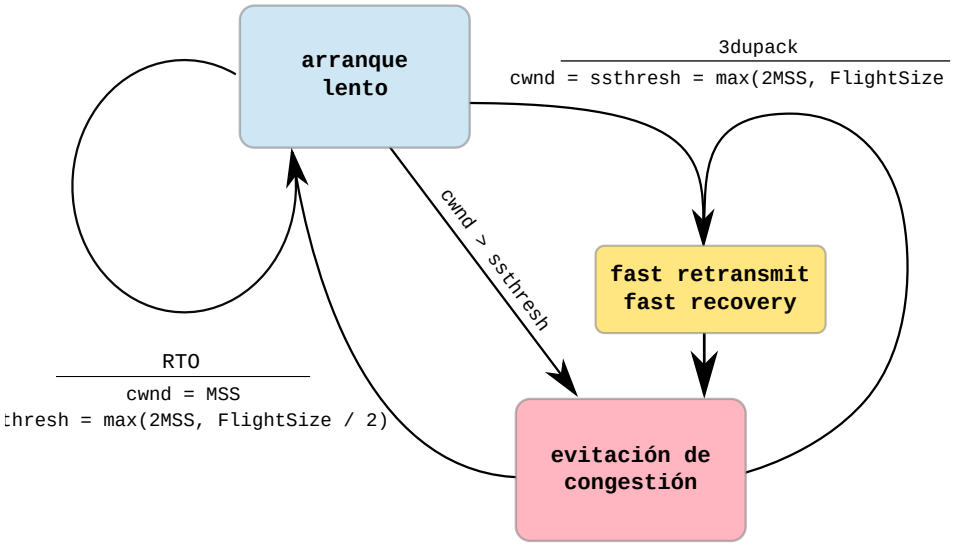

FIGURA 13.6: Transición entre los algoritmos de control de congestión de TCP

13.6.2. Modelo simplificado

Por facilitar el estudio y comprensión de estos mecanismos podemos aplicar una serie de simplificaciones que no afectan significativamente a su finalidad, pero que nos permiten realizar cálculos más sencillos.

En la fase de evitación de congestión, en lugar de aplicar la fórmula 13.4 para determinar el crecimiento de cwnd, utilizaremos la fórmula 13.7, que divide el crecimiento de MSS de la ronda en los k segmentos que se envían en ella, con lo que al final de la ronda cwnd también habrá crecido MSS.

$$\text{cwnd} += \text{MSS}/k \tag{13.7}$$

Cuando expira el RTO, calcularemos el nuevo valor de *ssthresh* según la fórmula 13.6 que utiliza swnd como aproximación a *datos en vuelo*. También

obviamos el mínimo de 2 MSS. El cálculo simplificado es el de la fórmula 13.8.

$$\text{ssthresh} = \text{swnd}/2 \tag{13.8}$$

Como parte de la simplificación, aplicaremos el enfoque de TCP Vegas, es decir, después de *3dupack* se aplica la fórmula 13.9 y se pasa a una fase de evitación de congestión, obviando la retransmisión rápida y la recuperación rápida.

$$\text{cwnd} = \text{swnd}/2 \tag{13.9}$$

La Figura 13.7 muestra el diagrama de estados entre las fases arranque lento y evitación de congestión, aplicando la simplificaciones indicadas.

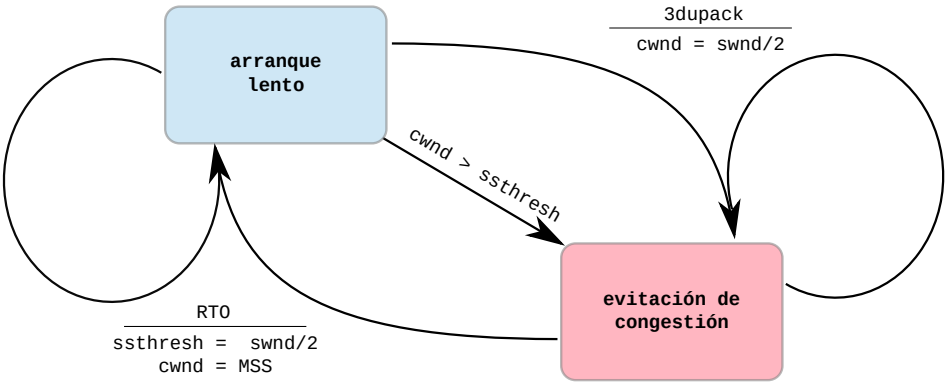

FIGURA 13.7: Transición entre los algoritmos de control de congestión de TCP

En la Figura 13.8 aparece un ejemplo más completo para entender cómo evoluciona la ventana de congestión conforme ocurren diferentes eventos a lo largo de la conexión. El valor de cwnd aparece en el eje vertical expresado en MSS mientras que el eje horizontal muestra las rondas, o períodos RTT equivalente. Por simplicidad este ejemplo supone que $rwnd > cwnd$ durante toda la conexión, de forma que $swnd = cwnd$.

Veamos cómo progresa la ventana de congestión respondiendo a los indicios que ocurren durante la conexión:

Ronda 0 – Al inicio de la conexión hay siempre una fase de arranque lento (AL) que fija $cwnd = MSS$ y lo duplica en cada ronda.

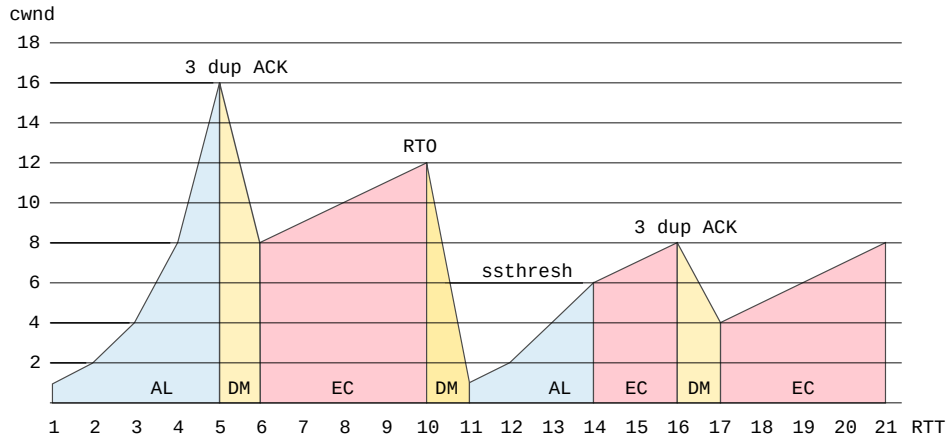

FIGURA 13.8: Ejemplo de evolución de la ventana de congestión en TCP

Ronda 4 – Se detecta una situación 3dupack. TCP responde según el diagrama de la Figura13.7 fijando $cwnd = swnd/2 = 16/2 = 8MSS$ con una transición de disminución multiplicativa (DM) y cambia a evitación de congestión (EC). La fase EC incrementa el valor de cwnd en MSS en cada ronda.

Ronda 9 – Aparece otro problema: la expiración del RTO. Esta vez la respuesta es más agresiva, cwnd se reduce a MSS y comienza una nueva fase de arranque lento, pero también se ajusta $ssthresh = swnd/2 = 12/2 = 6MSS$.

Ronda 13 – El valor de cwnd alcanza *ssthresh*, por lo que entra en evitación de congestión.

Ronda 15 – De nuevo otra situación 3dupack, y de nuevo se aplica $cwnd = swnd/2 = 8/2 = 4MSS$. A partir de aquí cwnd sigue creciendo de forma lineal hasta la ronda 20.

13.7. Gestión activa de colas

Que los routers descarten paquetes cuando sus colas de entrada desbordan tiene algunas consecuencias negativas adicionales a la propia pérdida de los paquetes. Es muy probable que paquetes que corresponden a conexiones diferentes lleguen al router mientras la cola está llena, y todos ellos sufrirán la misma suerte. Esto puede provocar que esas conexiones *sincronicen* sus retransmisiones, un efecto que no es deseable en absoluto.

La Gestión Activa de Colas (en inglés AQM) es un conjunto de técnicas para detectar la congestión antes de que las colas de entrada del router se llenen por completo y, en ese caso, informar de la situación a los nodos finales.

En Internet, este tipo de notificaciones se realizan mediante Notificación Explícita de Congestión (en inglés ECN) que, como su nombre indica, es una técnica de bucle cerrado. Los routers AQM no crean mensajes específicos sino que etiquetan con la notificación Congestion Experienced (CE) la cabecera de los paquetes IP en tránsito. Estos paquetes «marcados» siguen su camino del modo habitual hacia su destino.

Con AQM, los routers tienen dos opciones al detectar una congestión inminente. Pueden *notificar* la congestión de forma implícita: descartando un paquete aunque no haya desbordamiento de la cola de entrada, o de forma explícita: haciendo llegar información específica sobre la congestión al emisor responsable de ese tráfico, es decir, aplicando el marcado ECN.

Con AQM la congestión no se notifica cuando la cola desborda, sino que se utiliza un algoritmo como RED. Este algoritmo calcula la ocupación media de la cola en cierto período de tiempo. Si ese valor sobrepasa determinado umbral, considera que existe un riesgo significativo de congestión y optará por descartar un paquete de la cola o bien aplicar ECN. Es interesante destacar que incluso en el caso de descarte de paquetes, los efectos negativos no son tan graves como cuando ocurre porque la cola desborda, ya que se reduce mucho la probabilidad no deseada de retransmisiones sincronizadas entre varias conexiones.

AQM utiliza ECN únicamente si la carga útil del paquete es un ECT (ECN Capable Transport), es decir, un protocolo de transporte que soporta ECN. En caso contrario, descarta el paquete. ECN utiliza un campo del mismo nombre que hay en la cabecera del paquete IP[5]. Este campo de 2 bits sirve para que el router notifique CE, pero también para indicar a los routers que el paquete encapsula un ECT. Por concretar el significado de estos bits:

00: El protocolo de transporte no soporta ECN.
01: Es un ECT de tipo 1 (ECT 1), el modelo avanzado.
10: Es un ECT de tipo 0 (ECT 0), el modelo estándar, que usa TCP.
11: Un router ha detectado congestión (CE).

13.7.1. ECN en TCP

Un emisor TCP con soporte ECN reacciona a una notificación CE reduciendo su ventana de congestión a la mitad ($cwnd = cwnd/2$) [31], de forma similar a lo que ocurre con la detección de *3dupack*.

[5]Tanto en IPv4 como en la de IPv6

Sin embargo, para que el emisor TCP sea informado se necesita la colaboración del receptor. Para ello se utilizan 2 flags en la cabecera TCP:

ECE (ECN Echo) Lo envía el receptor (probablemente en un segmento de confirmación) para indicar al emisor la llegada de un paquete marcado con CE.

CWR (Congestion Window Reduced) Lo envía el emisor (probablemente en el siguiente segmento de datos) para indicar al receptor que ha recibido la notificación de congestión y ya está reduciendo su tasa de emisión. El receptor puede entonces dejar de marcar sus ACK con el bit ECE.

Para contar con la colaboración del receptor, es necesario que ambos extremos verifiquen que el otro también soporta ECN. Para ello, un cliente que soporta ECN debería activar los flags CWR y ECE en el segmento inicial junto al flag SYN. Si el servidor también soporta ECN, debe responder activando el flag ECE junto con los flags SYN y ACK, propios del proceso de conexión habitual. El uso de estos flags durante el establecimiento de conexión se llama «negociación ECN» y durante su ejecución no tienen el significado de notificación previamente descrito. Solo en el caso en que se haya dado esta negociación ECN con éxito, los participantes marcarán los paquetes IP con ECN, y reconocerán y activarán los flags ECE y CWR durante la conexión.

Veamos una secuencia de mensajes que aparecerían en una conexión que utiliza ECN, que está también representada en la Figura 13.9.

1. El cliente realiza la conexión enviando un segmento marcando los flags SYN, ECE y CWR.
2. El servidor responde con un segmento marcando los flags SYN, ACK y ECE. A partir de este momento, cliente y servidor marcarán los paquetes IP que envíen con el valor ECT(0)[6].
3. En algún momento, un router detecta congestión y elige un paquete que corresponde a esta conexión. Cambia el valor de su campo ECN a 11, es decir, lo marca como CE.
4. El paquete marcado con CE llega al receptor TCP.
5. En las siguientes confirmaciones, el receptor activa el flag ECE en la cabecera TCP para notificar al emisor.

[6]En realidad, solo marcan los paquetes IP que contengan segmentos TCP con datos.

6. El emisor TCP recibe un segmento con ECE, reduce su tasa de emisión y, en el siguiente segmento de datos, activa el flag CWR.

7. El receptor recibe el segmento marcado con CWR y deja de enviar segmentos marcados con ECE para que el emisor deje de aplicar la reducción de tasa.

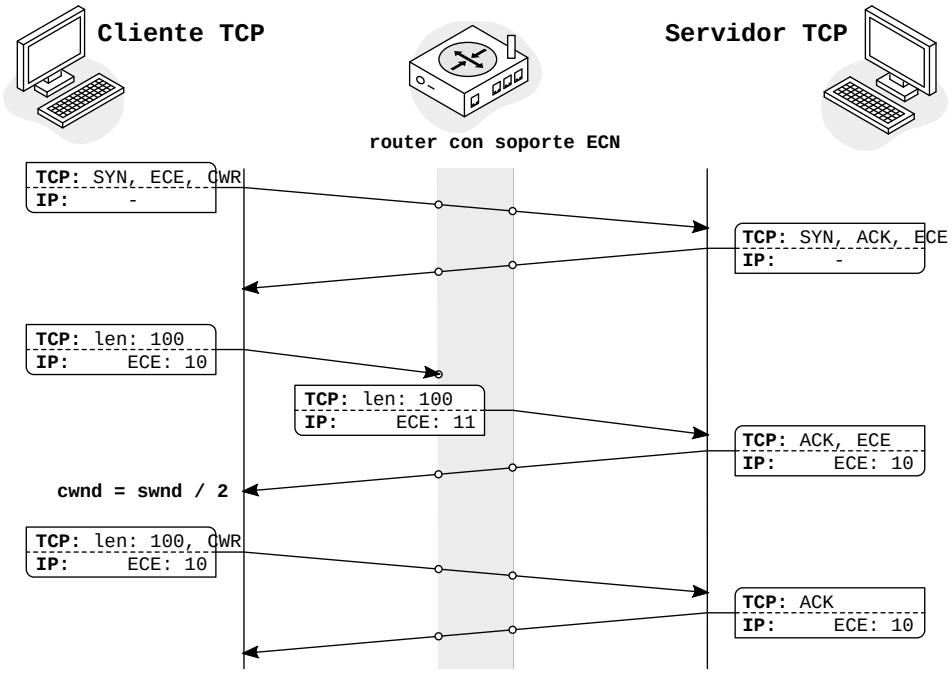

FIGURA 13.9: Conexión TCP con soporte ECN bajo un episodio de congestión

13.7.2. ECN con otros protocolos

ECN no es para uso específico de TCP, si bien requiere que los protocolos de transporte proporcionen la funcionalidad necesaria para realizar la necesaria gestión de tasa de emisión por sus propios medios. Los protocolos de aplicación también pueden beneficiarse de ECN si están diseñados para ello. Por ejemplo, protocolos modernos como QUIC, que de hecho se encapsula en UDP, lo incorporan [32].

Y ¿qué más?

TCP es un protocolo complejo, lleno de detalles y algoritmos que han sido refinados y optimizados a lo largo de sus muchos años de vida. La expe-

riencia empírica ha permitido a varias generaciones de ingenieros ir adaptando su funcionamiento para trabajar con aplicaciones y servicios que ni se concebían cuando se diseñó a finales de los 70. A pesar de ello, su funcionamiento básico sigue siendo esencialmente el mismo, y su desempeño es francamente robusto para la mayoría de los casos, algo que resulta especialmente meritorio si se piensa por un momento la tremenda evolución que ha experimentado hardware y software desde entonces.

Eso no significa que sea perfecto, ni mucho menos. Algunos de sus problemas más graves son: ausencia total de mecanismos de seguridad, bajo desempeño en comunicaciones móviles, rendimiento pobre en enlaces de alta latencia, el problema del «bloqueo de línea de cabecera» (*head-of-line blocking*), etc.

Estos problemas y sus soluciones son temas avanzados que superan el alcance actual de este libro, pero pueden dar al lector una idea de cómo el estudio y desarrollo de nuevos protocolos de transporte sigue siendo un tema de investigación abierto aún hoy día.