(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 996 700**

(51) Int. Cl.:

H04L 47/11 (2012.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **31.03.2015 E 20184668 (0)**

(97) Fecha y número de publicación de la concesión europea: **11.09.2024 EP 3742687**

(54) Título: **Aparato para el control de congestión de red basado en los gradientes de la tasa de transmisión**

(30) Prioridad:

23.04.2014 EP 14382146

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
13.02.2025

(73) Titular/es:

BEQUANT S.L. (100.00%)
Raimundo Fernández Villaverde 61, 5a Planta
28003 Madrid, ES

(72) Inventor/es:

LOPEZ SERRANO, JOSÉ;
PIÑEIRO BLANCA, LUIS y
LOPEZ SERRANO, GUILLERMO

(74) Agente/Representante:

MILTENYI, Peter

DESCRIPCIÓN

Aparato para el control de congestión de red basado en los gradientes de la tasa de transmisión

La presente invención pertenece al campo de las comunicaciones en red, y más exactamente, a un aparato para el control de congestión en una red de comunicaciones.

ANTECEDENTES DE LA INVENCIÓN

Control de congestión

La situación de congestión en una red de comunicación de datos surge siempre que la capacidad de transmisión disponible de un nodo de red o de un enlace es menor que la tasa de datos que necesita retransmitir. Por ejemplo, la Figura 1A muestra un nodo transmisor de datos 110, enviando un flujo de datos 150, a un nodo receptor de datos, 120, a través de una red de comunicaciones 140, que incluye una serie de nodos de red interconectados 141, 142, 143 y posiblemente otros. El flujo de datos 150, es transmitido en este caso a través de los nodos de red 141 y 142. La Figura 1A muestra otro nodo transmisor de datos 111, enviando un flujo de datos 151 a un nodo receptor de datos 121, a través de la misma red 140. Sin embargo, el flujo de datos 151, es transmitido en este caso a través de los nodos de red 141 y 143. Si la tasa de envío agregada de los flujos de datos 150 y 151, es mayor que la capacidad de transmisión del nodo de red 141, donde ambos flujos coinciden, entonces el nodo de red 141 se congestionará y ambos flujos de datos 150 y 151 experimentarán una situación de congestión. Si un nodo de red (tal como el 141) tiene alguna capacidad de almacenamiento temporal ("buffer") de recepción, la situación de congestión ocasionará que esa capacidad de almacenamiento temporal se vaya usando progresivamente hasta llenarse del todo. Una vez que ese almacenamiento temporal esté lleno, si la situación de congestión persiste, parte de los flujos de datos 150 y 151 serán descartados y de ese modo se producirá una pérdida desde el punto de vista de los nodos receptores de datos 120 y 121. Los mecanismos de control de congestión se usan para evitar, mitigar y gestionar situaciones de congestión. Los mecanismos de control de congestión pueden formar parte de la funcionalidad de cualquier capa, típicamente de la capa de enlace, de la capa de red, de la capa de transporte o de la capa de aplicación, y pueden residir en los nodos finales (tales como el 110 o el 111) o en los nodos intermedios (tales como 141, 142 o 143).
Siendo uno de los protocolos de comunicación más ampliamente utilizado en la actualidad, el Protocolo de Control de Transporte (en inglés "Transport Control Protocol" o TCP) emplea mecanismos sofisticados de control de congestión. Muchos de los avances técnicos en el campo del control de congestión han surgido dentro del desarrollo de TCP. Por ello, en adelante, se describirán los mecanismos de control de congestión de TCP. En cualquier caso, muchos de los desarrollos técnicos provenientes de TCP han sido más tarde adoptados por otros protocolos, tales como "Stream Control Transmission Protocol" (SCTP).

Control de Congestión Estándar del Protocolo TCP

El Protocolo de Control de Transporte (TCP) es utilizado ampliamente en redes de comunicación de datos. El TCP, especificado en J. Postel "IETF RFC 793: Transmission control protocol," 1981, que queda incorporado aquí por referencia, proporciona una transmisión de datos fiable entre dos extremos. Los extremos son denominados normalmente "hosts" en la bibliografía sobre tecnología TCP. La referencia a una "transmisión fiable" hace alusión al hecho de que TCP proporciona un mecanismo de petición adaptativa de repeticiones ("Adaptive Repeat Request", ARQ) que permite una transmisión de datos confirmada. En particular, tal y como ilustra Figura 1B, un nodo transmisor de datos 110 transmite un segmento de datos 101 (siendo un segmento el contenido de un datagrama TCP) a través de una red 140, cuya correcta recepción es comprobada por parte del nodo receptor de datos 120. El nodo receptor de datos 120 envía entonces al nodo transmisor 110, a través de una red 140, una confirmación 102 que confirma positivamente la correcta recepción de los datos. De acuerdo con esta respuesta 102, o con la falta de esta respuesta, el nodo transmisor de datos 110 puede retransmitir datos. Las confirmaciones también se transmiten en segmentos TCP y pueden ser acumulativas, es decir, que la confirmación de un segmento implica la confirmación de todos los segmentos TCP consecutivos y previos a éste.

Los paquetes de datos pueden perderse, lo que significa que no llegan dentro de una ventana de tiempo predeterminada (es decir, un periodo de tiempo) al nodo receptor de datos. Más aún, los paquetes de datos pueden sufrir errores de transmisión, que pueden detectarse en el nodo receptor usando técnicas estándar, por ejemplo, códigos de detección y/o corrección de errores tales como códigos de redundancia cíclica ("Cyclic Redundancy Check", CRC) u otros. El retardo o los errores de transmisión pueden ser ocasionados por un aumento de la carga en la red y/o por el empeoramiento de las condiciones de canal. El mecanismo de confirmación proporcionado por TCP permite la recuperación mediante retransmisiones de las pérdidas de paquetes y los datos corruptos.

Sin embargo, si la red experimenta una carga elevada, las retransmisiones reiteradas por parte de múltiples usuarios pueden empeorar la situación y ocasionar la congestión de la red. Con objeto de evitar una situación de esta naturaleza al tiempo que se gestiona la congestión, TCP proporciona varios mecanismos y estrategias de control de congestión, que pueden implantarse en el nodo de transmisión de datos 110 y/o en el nodo receptor de datos 120.

Se puede encontrar un ejemplo de la técnica anterior en el documento US 2005-237929.

A partir de este momento se usará terminología estándar de TCP, tal y como se emplea en la RFC 793, citada anteriormente así como en IETF RFC 5681 “TCP Congestion Control," de Sept. 2009, en particular:

“Host”: nodo de red que es un punto de terminación de una comunicación TCP. El término “host” se empleará también para otros protocolos para los cuales se pueda aplicar la presente invención, en el sentido de un nodo de red que se comunica extremo a extremo desde el punto de vista de esos protocolos.

Conexión: un flujo bidireccional de datos establecido entre dos “hosts”, identificado de forma única, con su propio establecimiento, control de flujo y mecanismos de control de congestión, independientes del de otros flujos de datos.

La Ventana de Congestión ("Congestion Window", cwnd) define la máxima cantidad de datos consecutivos que un “host” TCP puede enviar más allá del último número de secuencia confirmado, tal y como es calculado localmente por el “host” emisor, sin tener en cuenta la ventana anunciada por el “host” receptor.

Ventana de Recepción ("Receive Window", rwnd) es el tamaño de ventana anunciado por el receptor al emisor como parte de los mensajes de confirmación que le envía. El tamaño de ventana especifica la máxima cantidad de datos consecutivos que el receptor está dispuesto a aceptar más allá del último número de secuencia confirmado. Este es el mecanismo utilizado por TCP para implementar el control de flujo, es decir, para impedir que un emisor rápido abrume a un receptor más lento.

La Ventana de Envío es el valor menor de entre la ventana de congestión y la ventana de recepción. El control de congestión estándar de TCP se basa en mecanismos en el emisor, centrándose en fijar un valor adecuado de la ventana de congestión, asumiendo que la ventana de congestión determina la Ventana de Envío.

"Slow Start" (del inglés para Inicio Lento) es un estado de control de congestión en TCP. En el estado "Slow Start", el algoritmo de control de congestión aumenta la ventana de congestión de forma exponencial, de forma que la ventana de congestión se incrementa en un segmento aproximadamente cada vez que se recibe una confirmación. Un “host” emisor TCP en una conexión TCP se dice que está en "Slow Start" cuando está empleando esta forma de incrementar la ventana de congestión.

"Congestion Avoidance" (Prevención de Congestión) es un estado de control de congestión en TCP. En el estado de "Congestion Avoidance" el algoritmo de control de congestión de TCP aumenta la ventana de congestión de forma más lenta que en "Slow Start". En TCP conforme al estándar Reno/New Reno, la ventana de congestión crece en 1/cwnd bytes (el inverso de la ventana de congestión) por cada byte confirmado, es decir la ventana de congestión crecerá en un segmento después de haberse confirmado segmentos que sumen un tamaño total de cwnd bytes. Diferentes variantes de TCP tienen distintos algoritmos de "Congestion Avoidance". Un “host” emisor TCP en una conexión TCP se dice que está en "Congestion Avoidance" cuando está empleando esta forma de incrementar la ventana de congestión.

El umbral de "Slow Start" ("Slow Start Threshold"ssthresh) define la transición entre los estados de "Slow Start" y "Congestion Avoidance": es el valor de la ventana de congestión por debajo del cual se usa "Slow Start" y por encima del cual se incrementa la ventana de congestión conforme a "Congestion Avoidance".

Datos en Vuelo (en inglés "Flight size") es la cantidad de datos, normalmente medida en bytes, transmitidos y aún no confirmados.

"Buffler-bloat" es un término para referirse al uso excesivo de almacenamiento temporal ("buffer") en el camino de la transmisión, por parte de una conexión que emplea una ventana de congestión mayor de la que se requeriría estrictamente para compensar el retardo intrínseco de la conexión y el almacenamiento temporal ("buffer") necesario para la adaptación de ancho de banda en el camino de la transmisión.

La terminología TCP descrita anteriormente se puede usar en cualquier otro protocolo que utilice conceptos similares a los empleados por el control de congestión de TCP.

Las especificaciones de TCP, tales como la IETF RFC 5681 citada anteriormente; la IETF RFC 6582 “The NewReno modification to TCP's fast recovery algorithm," de 2012; la IETF RFC 2018 “TCP selective acknowledgment options" de Oct. 1996; y la IETF RFC 6675: “A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP" de 2012 (todas ellas referenciadas aquí) incluyen algoritmos de control de congestión para establecer la tasa de envío de datos más conveniente para los “hosts” y mecanismos de recuperación de datos que permiten la retransmisión eficiente de datos perdidos a causa de la congestión o por otros motivos. Tal y como se especifica actualmente en los documentos oficiales (RFC) de la "Internet Engineering Task Force" (IETF), el control de congestión de TCP se deriva del mecanismo de control de congestión denominado "Reno" (descrito, por ejemplo, en Jacobson, "Congestion avoidance and control," ACM SIGCOMM Computer Communication Review, 1988), con varios añadidos para mejorar la recuperación de pérdidas de paquetes transmitidos, y se basa en los siguientes principios:

- Los mecanismos de control de congestión se implementan en la funcionalidad TCP de los “hosts” implicados, sin recurrir a nodos intermedios en las capas de red o enlace.

- El receptor envía segmentos de confirmación acumulativos cuando se reciben correctamente segmentos de datos, indicando el número de secuencia del mayor byte consecutivo correctamente recibido.

- La “ventana de envío” definida anteriormente, que, si no es limitada por la ventana de recepción, es igual a la ventana de congestión, controla la cantidad de datos transmitidos por el emisor.

- Al comienzo de la conexión (y tras un vencimiento del temporizador de retransmisiones), el emisor empieza con una ventana de congestión mínima y la incrementa conforme al algoritmo "Slow Start" explicado anteriormente.

- Si no está limitada por la ventana de recepción o por el almacenamiento temporal ("buffer") de salida del emisor, el algoritmo de "Slow Start" provoca que la tasa de envío crezca rápidamente por encima de la capacidad de la red, resultando en pérdidas de paquetes.

- El emisor detecta las pérdidas cuando recibe tres segmentos de confirmación duplicados. Tras un procedimiento de recuperación rápida de las pérdidas, la ventana de congestión se fija a la mitad del valor máximo alcanzado durante la fase "Slow Start" y la conexión cambia a "Congestion Avoidance", lo cual provoca que la ventana de congestión crezca más despacio que en "Slow Start".

- En "Congestion Avoidance" se sigue un esquema de incremento aditivo-decremento multiplicativo ("Additive-Increase-Multiplicative-Decrease", AIMD). La ventana de congestión aumenta en una pequeña cantidad fija por cada segmento de confirmación, y se reduce a la mitad cuando se detecta congestión. La congestión se detecta cuando sucede una pérdida de paquetes (tres confirmaciones duplicadas recibidas consecutivamente). Aunque este mecanismo fuerza la congestión, se ha demostrado matemáticamente que varios flujos TCP que compartan un recurso con cuello de botella y que sigan la estrategia AIMD en su crecimiento de la ventana de congestión, convergerán a un reparto del ancho de banda equitativo.

- Las pérdidas de paquetes muy intensas no son recuperables con el procedimiento de recuperación rápido, lo que ocasionará el vencimiento del temporizador de retransmisiones del emisor TCP. Este vencimiento de temporizador causa la retransmisión del primer segmento enviado y pendiente de confirmación, tras lo cual se inicia la fase de "Slow Start", con la ventana de congestión reducida a un segmento. Este mecanismo hace que todos los emisores reduzcan sus tasas de envío drásticamente en caso de congestión severa, lo que impide un colapso por congestión completo.

Este estándar TCP (a menudo llamado "Reno" o "NewReno") tiene tres limitaciones: (1) bajo rendimiento en redes con velocidad alta y retardos prolongados, ya que la pequeña tasa de crecimiento de la ventana de congestión en "Congestion Avoidance" tarda mucho tiempo en alcanzar los grandes tamaños de ventana de congestión necesarios en dichas redes; (2) uso excesivo de almacenamiento temporal ("buffer-bloat") en la red, incrementando el retardo experimentado por los “hosts” involucrados, ocasionado por el mecanismo de detección de congestión basado en pérdidas, que incrementa la ventana de congestión hasta que el almacenamiento temporal ("buffer") en los nodos de red en el camino de la transmisión se llenan y por tanto los paquetes enviados se pierden, y (3) la competencia contra flujos TCP concurrentes agresivos, es decir, que aumentan su ventana de congestión en forma más agresiva que este TCP estándar cuando comparten un cuello de botella de ancho de banda, y que ocuparán la mayor parte del ancho de banda disponible y privarán de él a los flujos de TCP Reno.

Como se ha mencionado anteriormente, el rendimiento del TCP estándar es limitado en redes de alta velocidad y retardos elevados debido a que el crecimiento lineal de la ventana de congestión en "Congestion Avoidance" es demasiado lento, lo que ocasiona que una parte significativa de la capacidad no se use. Muchas variantes de TCP han propuesto esquemas de crecimiento de la ventana de congestión más agresivos para dichos escenarios, tales como las variantes STCP, HSTCP, BIC-TCP, H-TCP, CUBIC y TCP-Hybla, manteniendo el mismo mecanismo de detección de congestión basado en pérdidas. Tales variantes son en general exitosas a la hora de mejorar el rendimiento de TCP en redes de alta velocidad y retardos elevados, pero no resuelven el problema del uso excesivo del almacenamiento temporal ("buffer-bloat"), ya que usan una detección de la congestión basada en pérdidas. En algunos casos, pueden también tener problemas de reparto del ancho de banda de un cuello de botella con variantes TCP menos agresivas, tales como la Reno estándar, las cuales pueden ser abrumadas por el aumento más agresivo de la ventana de congestión.

Detección de Congestión basada en Retardos para Reducir el uso excesivo del almacenamiento temporal ("buffer-bloat")

Hay variantes de TCP que detectan la congestión de red analizando el retardo extremo a extremo de la conexión mediante medidas del tiempo de ida y vuelta ("Round-Trip Time", RTT), es decir, el tiempo desde que se envía un

segmento hasta que se recibe una confirmación de él. Algunas de tales variantes son Vegas, TCP Vegas-A, TCP New Vegas y FAST-TCP. Las medidas de RTT son traducidas a veces a estimaciones de la tasa de envío, o bien a estimaciones de los segmentos encolados en almacenamiento temporal ("buffers") en el camino de transmisión, pero en realidad la variable independiente que usan para sus decisiones son las medidas de RTT.

En las variantes de TCP basadas en retardos, un incremento de RTT se toma como indicio del inicio de la congestión y, en "Congestion Avoidance", la decisión de aumentar o reducir la ventana de congestión se hace en virtud de estas medidas de RTT. Estos métodos generalmente tienen éxito a la hora de reducir o eliminar las pérdidas por congestión, reduciendo de este modo el uso excesivo del almacenamiento temporal ("buffer-bloat") y el retardo excesivo. Sin embargo, sufren considerablemente cuando compiten contra variantes de TCP basadas en pérdidas. La razón es que los flujos TCP basados en retardos detectan la congestión antes que los flujos TCP basados en pérdidas y reducen sus tasas de envío. Las variantes basadas en pérdidas no tienen ese freno y continúan incrementando su tasa de envío en tanto no llenen el almacenamiento temporal ("buffer") de los cuellos de botella intermedios, reduciendo progresivamente la capacidad destinada a los flujos basados en retardos. Dado que actualmente la inmensa mayoría del TCP en Internet usa control de congestión basado en pérdidas, esto ha sido un obstáculo importante para la adopción de variantes únicamente basadas en retardos.

Variantes Mixtas Basadas en Pérdidas y Retardos

Una propuesta temprana del uso de un modelo mixto, conocida como TCP-DUAL, añade a un control de congestión basado en pérdidas tipo Reno una detección de congestión basada en medidas de RTT, que provoca un decremento multiplicativo en la ventana de congestión. Esta aproximación puede resolver los problemas de uso excesivo del almacenamiento temporal ("buffer-bloat"), pero no puede competir contra variantes basadas en pérdidas debido a su detección basada en RTT, al igual que las variantes basadas sólo en retardos antes mencionadas.

Otras variantes de TCP tales como Compound TCP, TCP Libra, TCP Africa, TCP Veno, YeAH-TCP y TCP Illinois también usan modelos mixtos, con control de congestión basado en pérdidas y en retardos. La detección de congestión basada en retardos se usa para modular la agresividad del crecimiento de la ventana de congestión, permitiendo un crecimiento más agresivo cuando no se detecta congestión, lo que normalmente resuelve los problemas de rendimiento asociados a Reno o NewReno en las redes de alta velocidad y retardos elevados. Sin embargo, en todas estas variantes, cuando alguna métrica basada en RTT estima que hay congestión, la ventana de congestión continuará creciendo, aunque más lentamente, hasta que se den pérdidas de paquetes, de modo que el problema de uso excesivo del almacenamiento temporal ("buffer-bloat") persiste, incluso cuando no se compite contra otros flujos.

Otra variante de TCP llamada TCP Vegas+ es un modelo mixto que usa TCP Vegas por defecto y cambia a NewReno cuando se detecta las competencia de otro flujo. Esto debería evitar problemas de uso excesivo del almacenamiento temporal ("buffer-bloat") cuando no hay competencia contra otros flujos, pero persisten los problemas no resueltos de Vegas, por ejemplo su bajo rendimiento en redes de alta velocidad y retardos elevados.

Variantes de TCP con Estimación de Ancho de Banda o de Tasa de Envío

Una variante de TCP llamada Tri-S es una variante temprana de TCP con detección de congestión basada en tasa de transmisión, utilizando para ello su evolución en el tiempo. Sin embargo, sin un filtrado adecuado de las estimaciones de tasa de transmisión o sin una aproximación estadística a la comprobación del crecimiento o estabilidad de la tasa de transmisión medida, los resultados de detección de congestión se estropean por la variabilidad relativamente grande inherente a las medidas de RTT. Más aún, una detección de congestión en tiempo real basada en tasa de transmisión no puede distinguir entre una situación de pura congestión y otra en la cual el flujo TCP compite contra un flujo con una agresividad similar: en ambos casos la ventana crecerá y la tasa de transmisión medida permanecerá constante. Puesto que Tri-S reduce la ventana de congestión en cuanto detecta congestión, no puede competir contra un flujo TCP con control de congestión basado en pérdidas, de igual modo que las variantes de TCP basadas en retardos.

Las variantes de TCP denominadas TCP-.Westwood y TCP-Westwood+ introdujeron estimadores directos del ancho de banda a sus mecanismos de control de congestión, basándose en medidas complejas de los tiempos entre confirmaciones o la tasa de recepción de dichas confirmaciones. Con el filtrado apropiado, aquellas estimaciones obtenidas al tiempo que una pérdida por congestión se toman como el ancho de banda disponible para la conexión TCP. Esta estimación del ancho de banda, junto con el mínimo RTT medido, determinan la ventana de congestión óptima. A partir de ese momento, un modo de "Congestion Avoidance" similar al de Reno llevará la ventana de congestión hasta la congestión y la pérdida de paquetes (causando de esta manera un uso excesivo del almacenamiento temporal o “buffer-bloat”), en cuyo momento una nueva ventana de congestión óptima es calculada basándose en una nueva estimación de ancho de banda.

Hay también variantes más recientes de TCP-Westwood dirigidas a redes con alta velocidad y retardos elevados (tales como LogWestwood+, TCPW-A, TCP-AR y TCP Fusion), con un crecimiento de ventana más agresivo, las cuales se adaptan mejor a cambios en el ancho de banda de la red, pero que sufren todavía de un uso excesivo del almacenamiento temporal ("buffer-bloat") y pueden abrumar a flujos Reno concurrentes menos agresivos.

Transición Mejorada entre "Slow Start" y "Congestion Avoidance"

Las variantes TCP descritas hasta ahora se concentran en su comportamiento durante la fase de "Congestion Avoidance". Sin embargo, la transición de "Slow Start" a "Congestion Avoidance" puede ser muy importante, especialmente en pequeñas descargas, que pasan una parte muy significativa de su existencia en "Slow Start". Detectar congestión en "Slow Start" sólo mediante pérdidas puede ocasionar problemas de uso excesivo del almacenamiento temporal ("buffer-bloat") severos y pérdida de paquetes, ya que la congestión se alcanzará al tiempo que la ventana de envío está creciendo exponencialmente.

Hay distintas formas de detectar congestión en "Slow Start" antes de la pérdida de paquetes y cambiar entonces a un algoritmo menos agresivo de "Congestion Avoidance". Algunas de ellas recurren a medidas de retardos entre confirmaciones, lo cual puede ser impreciso debido al grado de exactitud de las medidas de tiempo y el filtrado sofisticado que se requiere en el emisor. TCP-Vegas propone un "Slow Start" modificado que en realidad produce una transición prematura a "Congestion Avoidance" debido a la naturaleza en ráfagas del tráfico en "Slow Start".

"Limited Slow Start" es una RFC experimental del IETF que se apoya en una constante arbitraria para determinar el punto de la transición. "Adaptative Start" es parte de la variante TCPW-A y usa el ancho de banda estimado para derivar el umbral ssthresh y por tanto depende fuertemente de la calidad de dicha estimación. "Hybrid Start", ahora mismo utilizada por defecto en la mayoría de las versiones del popular sistema operativo Linux, emplea dos algoritmos heurísticos basados en medidas de RTT y de retardos entre confirmaciones. Funciona bien excepto cuando se compite contra flujos TCP en congestión, porque en ese caso el tráfico concurrente aumentará el RTT desde el principio y la transición de "Slow Start" a "Congestion Avoidance" ocurrirá demasiado pronto, causando una reducción de velocidad.

Equidad y Competencia contra Flujos TCP más Agresivos bajo Congestión

En casi toda la bibliografía, el problema de la equidad trata de cómo una variante TCP más agresiva (en relación a su crecimiento de la ventana de congestión) evita abrumar a variantes menos agresivas. Sin embargo, es igualmente importante que un flujo TCP se vuelva más agresivo si determina que otro flujo TCP está compitiendo con él de una forma más agresiva. Esto puede suceder incluso para flujos de la misma variante TCP, cuando el otro flujo está en "Slow Start". Una de las pocas variantes TCP que aborda este problema es TCPW-A, que tiene un mecanismo para incrementar el parámetro ssthresh si estima que resultaría en un mayor ancho de banda, pero lo condiciona a otro mecanismo que detecta si hay otro flujo TCP compitiendo.

Estrategias basadas en red

Las estrategias de control de congestión mencionadas hasta ahora descansan en funcionalidad implementada en los “hosts” finales, principalmente en el lado emisor. Sin embargo, algunas estrategias descansan en funcionalidad en los nodos de red intermedios, tales como enrutadores y “switches”, que pueden alertar a los extremos de una congestión inminente (ej. TCP ECN) o bien tirar paquetes antes de que suceda la congestión (un ejemplo es la gestión de colas mediante el algoritmo "Random Early Detection"). Un nuevo algoritmo de gestión activa de colas, CoDel (K. Nichols, V. Jacobson, “Controlling queue delay”, Communications of the ACM, vol. 55, no. 7, pp. 42-50, 2012) se ha propuesto recientemente para abordar el problema del uso excesivo del almacenamiento temporal ("buffer-bloat"), y también requiere que parte de la funcionalidad se despliegue en enrutadores y “switches” intermedios. El problema con todas estas estrategias es que son muy difíciles de desplegar, ya que hay una base instalada de enrutadores y “switches” inmensa que deberían soportarlas a lo largo de todo el camino, extremo a extremo. Por el contrario, con las soluciones basadas en “hosts”, es suficiente que los dos “hosts” soporten la funcionalidad, y en caso de que la funcionalidad sea solo del lado emisor o lado receptor, solo uno de los “hosts” finales necesita implementarla para poder beneficiarse de ella.

En resumen, el uso excesivo del almacenamiento temporal ("buffer-bloat") continúa siendo un problema importante de las comunicaciones con TCP, causando retardos y uso de recursos innecesarios, debido a que las variantes de TCP más comunes emplean detección de congestión basada en pérdidas, lo cual impide la difusión de variantes basadas en retardos que podrían mitigar el problema. Casi todas las variantes TCP que usan algoritmos basados en ancho de banda y tasa de transmisión para determinar la ventana de congestión usan detección de congestión basado en pérdidas, de manera que el problema del uso excesivo del almacenamiento temporal ("buffer-bloat") permanece. En las pocas estrategias en las que se usa detección de congestión basada en tasas de transmisión para reducir la ventana de congestión, no se emplean filtrados ni métodos estadísticos robustos, de modo que la variabilidad de las medidas no se aborda adecuadamente y la congestión no es bien detectada. De hecho, mientras se usen variantes de TCP basadas en pérdidas (y actualmente son las más utilizadas) el problema del uso excesivo del almacenamiento temporal ("buffer-bloat") será inevitable para cualquier flujo TCP con el que tengan que competir. Sin embargo, hay muchas situaciones en las que una conexión no compite con otras conexiones por un cuello de botella de capacidad, en las cuales sería beneficioso eliminar el uso excesivo del almacenamiento temporal ("buffer-bloat").

Más aún, los intentos más comunes de reducir el uso excesivo del almacenamiento temporal ("buffer-bloat") en la transición entre "Slow Start" y "Congestion Avoidance" se basan en métricas de retardos que pueden causar velocidades menores cuando se enfrentan a la competencia de otros flujos. En resumen, la falta de un buen

mecanismo para la detección de la congestión y de la competencia está causando problemas de uso excesivo del almacenamiento temporal ("buffer-bloat") en algunos casos y problemas de rendimiento en otros, cuando los flujos TCP no reaccionan apropiadamente a la competencia de flujos TCP concurrentes.

RESUMEN DE LA INVENCIÓN

Basados en los inconvenientes del estado anterior de la técnica ya resumidos, sería beneficioso proporcionar un mecanismo de control de congestión que permita la gestión eficiente de las situaciones de congestión y de competencia por el ancho de banda por parte de flujos TCP concurrentes, al tiempo que se evita el uso excesivo del almacenamiento temporal ("buffer-bloat").

Esto se logra mediante las características de la reivindicación independiente.

Las realizaciones ventajosas de la invención son el tema de las reivindicaciones dependientes.

La aproximación particular de la presente invención es detectar la congestión basándose en las tendencias tanto de los datos en vuelo como de la tasa de transmisión con objeto de adaptar la ventana de congestión de acuerdo con los resultados de la detección.

Esta estrategia tiene la ventaja de detectar de manera fiable la congestión y de distinguir entre congestión con competencia injusta y congestión sin competencia injusta, situaciones que pueden beneficiarse de tratamientos diferenciados. La competencia de un flujo de datos concurrente se considera injusta cuando toma de forma continuada una mayor proporción del ancho de banda limitado por la congestión. Más aún, el filtrado aplicado con objeto de determinar las tendencias de la tasa de transmisión y/o de los datos en vuelo consigue que la decisión sobre la presencia de la congestión se realice de manera estable en relación a las variaciones temporales de las medidas.

Ejemplos de realizaciones de los métodos y aparatos de la presente invención se ilustran en las figuras adjuntas, en las cuales números idénticos indican elementos idénticos o similares y en las cuales:

FIGURA 1A es un diagrama de bloques ilustrando la comunicación a través de una red.

FIGURA 1B es un diagrama de bloques ilustrando una comunicación entre dos nodos de red con confirmaciones.

FIGURA 1C es un diagrama de bloques ilustrando una comunicación entre dos nodos de red con confirmaciones interconectados vía un nodo “proxy”.

FIGURA 2 es un diagrama de flujo ilustrando los pasos invocados cuando se procesa un segmento de confirmación entrante, mostrando cómo una realización de la invención puede incorporarse a una implementación de TCP ya existente.

FIGURA 3 es un diagrama de flujo ilustrando los principales pasos de una posible realización de los métodos de esta invención, incluyendo únicamente uno de los cinco pasos de detección de la invención.

FIGURA 4 es un diagrama de flujo ilustrando los principales pasos de otra posible realización de los métodos de esta invención, incluyendo los cinco tipos de detección de congestión de la invención de forma coordinada.

FIGURA 5 es un diagrama de flujo ilustrando los principales pasos de una realización del paso de filtrado de la tasa de transmisión.

FIGURA 6 es un diagrama de flujo ilustrando los principales pasos de una realización del paso de filtrado del gradiente de la tasa de transmisión.

FIGURA 7 es un diagrama de flujo ilustrando los principales pasos de una realización del paso de determinación de la tendencia de los datos en vuelo.

FIGURA 8 es un diagrama de flujo ilustrando los principales pasos de una realización del paso de determinación de la tendencia de la tasa de transmisión.

FIGURA 9 es un diagrama de flujo ilustrando los principales pasos de una realización del paso de detección de la congestión.

FIGURA 10 es un diagrama de flujo ilustrando los principales pasos de una realización del paso de modificación de la ventana de congestión.

FIGURA 11 es un diagrama de flujo ilustrando los principales pasos de una realización alternativa del paso de

determinación de la tendencia de la tasa de transmisión.

FIGURA 12 es un diagrama de bloques ilustrando un aparato que materializa la presente invención.

FIGURA 13 es un diagrama de bloques ilustrando una implementación en el núcleo (en inglés kernel) del sistema operativo de un aparato materializando la presente invención.

FIGURA 14 es un diagrama de bloques ilustrando una implementación fuera el núcleo del sistema operativo de un aparato que materializa la presente invención; y

FIGURA 15 es un diagrama de bloques ilustrando un aparato que materializa la presente invención.

DESCRIPCIÓN DETALLADA

Las comunicaciones eficientes por conmutación de paquetes (“packet-switched communications”) requieren cierto almacenamiento temporal ("buffering") en los nodos de red para permitir la conmutación, para adaptar diferentes tasas de transmisión en los enlaces y para absorber ráfagas de transmisión temporales. Sin embargo, los protocolos que emplean control de congestión basado en pérdidas tienden a usar todo el almacenamiento temporal ("buffer") disponible en el camino de transmisión, más allá de lo necesario para una comunicaciones eficiente, malgastando recursos de memoria y causando retardos innecesarios, provocando un uso excesivo del almacenamiento temporal ("buffer-bloat"). El uso excesivo del almacenamiento temporal ("buffer-bloat") puede ser inevitable en ciertas circunstancias, tales como cuando se compite contra otros flujos de datos bajo congestión, dado que en esos casos los flujos que traten de limitar el uso excesivo del almacenamiento temporal ("buffer-bloat") serán abrumados por aquellos que no lo hagan. Sin embargo, el uso excesivo del almacenamiento temporal ("buffer-bloat") puede evitarse cuando no hay competencia con otros flujos de datos por recursos en congestión (una situación bastante habitual).

La presente invención proporciona métodos y aparatos para el control de congestión de red que abordan el problema del uso excesivo del almacenamiento temporal ("buffer-bloat") cuando puede ser evitado sin penalizar la velocidad. Añade como característica ventajosa adicional la estimación de las tendencias de la tasa de transmisión y de los datos en vuelo, junto con mecanismos de detección de la congestión, que son robustos frente a la variabilidad observada. Ciertas realizaciones particulares de la presente invención especifican estrategias adicionales para detectar la congestión usando medidas de las tendencias de los datos en vuelo y de la tasa de transmisión, lo cual permite distinguir entre distintas situaciones de congestión. De este modo, las situaciones de congestión con flujos de datos compitiendo con una agresividad percibida diferente pueden ser tratadas de manera distinta.

La presente invención puede emplease de forma directa dentro del protocolo TCP y algunas de las realizaciones también prevén la incorporación dentro de algoritmos de gestión de la congestión de TCP. En cualquier caso, la presente invención no está limitada de ningún modo a TCP y es aplicable a cualquier protocolo de comunicación que proporcione transmisiones confirmadas y haga uso del control de congestión, tales como Stream Control Transmission Protocol (SCTP), Datagram Congestion Control Protocol (DCCP) y otros. Además, la presente invención es aplicable en cualquier capa de protocolo y no está limitada a la capa de transporte.

De acuerdo con la presente invención, tanto los datos en vuelo como la tasa de transmisión se miden y usan para determinar las tendencias, con las cuales se establece si hay o no una situación de congestión. En base a la congestión detectada, se selecciona la estrategia de modificación de la ventana de transmisión.

La tendencia de la tasa de transmisión se deriva de su gradiente. El gradiente puede ser cualquier medida que refleje la tendencia. Por ejemplo, puede ser la diferencia entre dos valores medidos en dos instantes de tiempo, o la diferencia entre estimaciones (tales como la media, la mediana, la media ponderada exponencialmente de una colección de medidas) en dos instantes de tiempo, una estimación de la pendiente de una curva ajustada a la secuencia de medidas, la media (ponderada) de varias medidas consecutivas de gradiente o la métrica obtenida de la estimación de tendencia por medio de un test estadístico (como el conocido test de tendencia de Jonckheere). En resumen, el gradiente puede ser cualquier métrica que indique si la secuencia de valores tiene una tendencia ascendente, estable o descendente. En cualquier caso, la medida se tomará repetidas veces, en tiempo real, partiendo de medidas que pueden incluir cantidades de ruido muy significativas, especialmente en el caso de las medidas de tasa de transmisión. Por ello, bien las medidas de tasa de transmisión, bien el gradiente de la tasa de transmisión (la métrica de tendencia), bien ambas, se filtran para reducir los efectos del ruido, que podría de otro modo dificultar a la detección de congestión.

De modo similar, la tendencia de los datos en vuelo puede determinarse como una medida que represente la evolución de los datos en vuelo o de los datos en vuelo filtrados a lo largo del tiempo. En particular, se puede usar un gradiente como el descrito anteriormente.

La presente invención, realizada en métodos y aparatos, proporciona la posibilidad de competir por un cuello de botella con flujos TCP basados en pérdidas, en cuyo caso tratará de igualar la agresividad de los flujos competidores, sin abrumarlos. Cuando no compite por un cuello de botella con flujos de datos agresivos, tratará de limitar su ventana de congestión una vez que se alcance el límite físico de tasa de transmisión, reduciendo de ese modo el uso excesivo

del almacenamiento temporal ("buffer-bloat") sin comprometer la velocidad. La estrategia es medir, desde el punto de vista del emisor, los datos en vuelo (esto es, la ventana en uso) y la tasa de transmisión obtenida con esos datos en vuelo, luego determinar sus tendencias (evolución temporal) con técnicas estadísticas robustas, y finalmente actuar sobre la ventana de congestión cuando detecta que la conexión se encuentra en una situación de congestión predefinida. De acuerdo con una realización ventajosa, hay cinco de tales situaciones de congestión, descritas más abajo. La invención es aplicable a emisores TCP ("hosts"), a los cuales añade la capacidad de detectar las situaciones antes referidas y, cuando se detecta una de dichas situaciones, cambia la ventana de congestión o el algoritmo que controla el crecimiento de la ventana de congestión. Para combatir la gran variabilidad presente en las medidas, especialmente en las medidas de tasa de transmisión, la presente invención emplea técnicas estadísticas robustas y el filtrado de la estimación de las tendencias de las tasas de transmisión.

Los cinco situaciones específicas que pueden detectarse y frente a las que se puede actuar son: (1) congestión sin competencia injusta en "Slow Start", (2) congestión sin competencia injusta en "Congestion Avoidance", (3) congestión con competencia injusta en "Congestion Avoidance", (4) congestión con competencia injusta tras una reducción de la ventana de congestión, y (5) congestión con competencia injusta tras una transición de "Slow Start" a "Congestion Avoidance". Hay que hacer notar que la presente invención no se limita a evaluar todas las situaciones anteriores y a manejarlas adecuadamente. Más bien, cualquiera de las situaciones anteriores pueden ser distinguidas mediante la estimación de la tendencia de los datos en vuelo y de la tasa de transmisión conforme a la invención. Por tanto, cada una de las cinco situaciones anteriores, su detección y manejo, constituye una realización separada.Además, estas realizaciones particulares pueden combinarse entre sí como verá claramente cualquiera experto en la materia.

Aunque los términos empleados anteriormente para las situaciones se refieren a "Slow Start" y "Congestion Avoidance", la invención no se limita a TCP y se pueden usar en estados correspondientes en otros mecanismos de control de congestión o aparatos (de acuerdo a la definición en la sección de Antecedentes o las especificaciones de TCP que correspondan). Más aún, el estado de congestión puede en general ser detectado sin distinguir entre estados de congestión como "Slow Start" o "Congestion Avoidance". En cualquier caso, la diferenciación de los estados mejora la adaptación de los mecanismos de control de congestión a las condiciones de la red.

El término "competencia injusta" es relativo al "host" emisor que está efectuando la detección de congestión y a su estado actual, es decir, indica que el flujo de datos competidor es más agresivo que el "host" emisor en su estado actual. El término "más agresivo" se refiera a que el flujo de datos competidor incrementa su ventana de congestión (o su tasa de transmisión) a un ritmo mayor que el "host" emisor. De este modo, un "host" emisor puede detectar la situación (1), es decir, congestión sin competencia injusta en "Slow Start", cuando está compitiendo contra otro flujo que también está en estado de "Slow Start". Dicha competencia no se considerará posiblemente injusta ya que el flujo competidor es tan agresivo como el "host" emisor. Sin embargo, si el "host" emisor cambia a "Congestion Avoidance", empezaría a ver el flujo competidor que permanece en "Slow Start" como competencia injusta, ya que dicho flujo le parecería entonces más agresivo al "host" emisor.

El término "congestión" es también relativo al "host" emisor, en el sentido de referirse a los síntomas de congestión detectados por el "host" emisor de acuerdo con ciertos parámetros de transmisión de datos que son medidos y evaluados. La congestión se considera detectada positivamente si cierta condición o condiciones predeterminadas basadas en los parámetros medidos se cumplen. Es ventajoso que los parámetros medidos sean los datos en vuelo y la tasa de transmisión, junto con su evolución en el tiempo (es decir, las tendencias) de estos dos parámetros.
Una realización de la presente invención puede detectar positivamente la situación (2), es decir, congestión sin competencia injusta en estado de "Congestion Avoidance", mediante la detección de un incremento de los datos en vuelo conjuntamente con una estabilización de la tasa de transmisión. Si la situación (2) se detecta positivamente, se reduce la ventana de congestión. En ausencia de límites en la ventana de recepción o en el almacenamiento temporal ("buffer") del "host", la ventana de congestión gobernará la ventana de envío, que gobierna a su vez los datos en vuelo. Reducir la ventana de congestión tenderá por tanto a reducir los datos en vuelo, que a su vez tenderá a mitigar el uso excesivo del almacenamiento temporal ("buffer-bloat").

Alternativa o adicionalmente, una realización de la presente invención puede detectar positivamente la situación (1), es decir, congestión sin competencia injusta en "Slow Start", detectando un aumento en los datos en vuelo conjuntamente con una estabilización de la tasa de transmisión, y cambiar del estado "Slow Start" al estado "Congestion Avoidance". Esto hará que los datos en vuelo tiendan a crecer más lentamente y, de ese modo, mitiga el "buffer-bloat".

Si la distinción entre "Slow Start" y "Congestion Avoidance" no es implementada, la detección positiva de una congestión sin competencia injusta puede efectuarse detectando un incremento de los datos en vuelo conjuntamente con una tasa de transmisión esencialmente invariable. El tratamiento puede consistir en reducir la ventana de congestión o reducir la velocidad de crecimiento de la ventana de congestión.

Alternativa o adicionalmente, una realización de la presente invención puede detectar la condición (3), es decir, una congestión con competencia injusta en "Congestion Avoidance", en la cual hay congestión al tiempo que otro flujos de datos compiten por la capacidad de un cuello de botella de un modo más agresivo (por ejemplo, creciendo su ventana de congestión más rápidamente). Tal congestión con competencia injusta puede detectarse (positivamente), en el

estado de "Congestion Avoidance", detectando incrementos en los datos en vuelo conjuntamente con decrementos en la tasa de transmisión. La detección positiva de competencia injusta en "Congestion Avoidance" puede ser tratada beneficiosamente mediante un incremento más agresivo de la ventana de congestión o cambiando de "Congestion Avoidance" a "Slow Start".

En ausencia de flujos de datos competidores, la detección de congestión basada en tasa de transmisión, al igual que los algoritmos basado en retardos, puede detectar la congestión cuando sucede, normalmente bastante antes de que las pérdidas provocadas por la congestión sucedan, y el uso excesivo del almacenamiento temporal ("buffer-bloat") puede evitarse reduciendo la ventana de congestión o pasando en ese momento del estado "Slow Start" al estado "Congestion Avoidance". Una de las ventajas de detectar la congestión basándose en la evolución temporal de los datos en vuelo y de la tasa de transmisión es que permite distinguir entre congestión con competencia injusta y congestión sin competencia injusta, especialmente en "Congestion Avoidance". En consecuencia, la acción a tomar en cada caso puede ser diferente, lo que mejora los efectos de la gestión de la congestión.

Si hay un flujo competidor que usa detección de congestión basado en pérdidas, no es posible evitar el uso excesivo del almacenamiento temporal ("buffer-bloat"). Si uno de los flujos limita su ventana de congestión cuando detecta congestión (por parte del emisor de ese flujo) entonces se verá abrumado por el flujo de datos competidor con detección basada en pérdidas, el cual seguirá incrementando sus datos en vuelo hasta que el almacenamiento temporal ("buffer") en la red se llene. En "Congestion Avoidance", cuando se alcanza la capacidad del cuello de botella, si el flujo competidor crece más agresivamente, la detección de competencia injusta basada en tasa de transmisión de acuerdo con la realización descrita anteriormente, detectará una tasa de transmisión decreciente conjuntamente con unos datos en vuelo crecientes, y o bien la ventana de congestión se incrementará o bien se pasará a un algoritmo más agresivo como "Slow Start". Esas dos acciones harán al flujo de datos más agresivo y ayudará a competir mejor contra el flujo más agresivo.

Cuando el “host” detecta la congestión (utilizando los métodos o elementos de la presente invención, o mediante otros algoritmos, tales como mecanismos basado en retardos o en la recepción de tres confirmaciones duplicadas), puede suceder que además de una situación de congestión, haya uno o más flujos de datos compitiendo por el ancho de banda disponible, con una tasa de crecimiento de ventana igualmente agresiva. Hasta ese momento, los flujos de agresividad similar podrían haber compartido el cuello de botella congestionado con una tasa de transmisión aproximadamente constante, mientras sus datos en vuelo respectivos crecían paralelamente (utilizando el almacenamiento temporal o "buffer" disponible en la red). Cuando como resultado de la detección de congestión, o bien la ventana de congestión es reducida o bien la conexión en "Slow Start" pasa a "Congestion Avoidance", el flujo ahora menos agresivo perderá parte de su tasa de transmisión, la cual será reclamada por los flujos de datos competidores. Esta situación es denominada "competencia injusta" en esta descripción, ya que, desde el punto de vista del flujo de datos que reduce su agresividad, es injusta. El lado emisor puede detectar esta situación de congestión con competencia injusta *a posteriori*, evaluando los resultados inmediatos de reducir la ventana de congestión o de cambiar a "Congestion Avoidance".

Una realización de la invención puede detectar positivamente la congestión con competencia injusta tras una reducción de la ventana de congestión (es decir, la situación (4) mencionada anteriormente) cuando un poco después de reducir la ventana de congestión (como consecuencia de la detección positiva de congestión), los datos en vuelo no son menores que la cuantía de la reducción en la ventana de congestión, mientras que la tasa de transmisión sí es menor. Al detectar esta situación (4), la reducción efectuada en la ventana de congestión puede deshacerse beneficiosamente, es decir, ser revertida. Por ejemplo, si un flujo de datos competidor crece su ventana de congestión al mismo ritmo, en "Congestion Avoidance", la detección de congestión sin competencia injusta basada en tasa de transmisión descrita anteriormente (2) termina por detectar una congestión y reducir la ventana de congestión, lo que ralentiza al flujo de datos que ha reducido su ventana de congestión, ya que el otro flujo de datos llenará el almacenamiento temporal ("buffer") que se ha liberado. Sin embargo, de acuerdo con la realización del manejo de la situación (4), esta reducción de la tasa de transmisión puede detectarse y la reducción de la ventana de congestión revertirse. añadiendo el valor previamente sustraído a la ventana de congestión.

Alternativa o adicionalmente, una realización de la presente invención puede detectar positivamente congestión con competencia injusta tras una transición de "Slow Start" a "Congestion Avoidance" (correspondiente a la situación (5) mencionada anteriormente) cuando, poco después de haber pasado de "Slow Start" a "Congestion Avoidance" (tras la detección de la congestión), los datos en vuelo no son menores que antes y hay una reducción en la tasa de transmisión tras la transición de "Slow Start" a "Congestion Avoidance". Si se detecta la situación (5), puede ser beneficioso poner la conexión en "Slow Start" de nuevo. De este modo, la detección que valida la transición de "Slow Start" a "Congestion Avoidance" permite determinar que hay una reducción en la tasa de transmisión si hay un flujo de datos competidor en "Slow Start", en cuyo caso el flujo vuelve a "Slow Start". En el último caso, dado que el algoritmo de "Slow Start" crece muy rápidamente, puede ser beneficioso mantener temporalmente una variable con el valor que habría tenido la ventana de congestión de no haberse efectuado el cambio a "Congestion Avoidance". De esta manera, si se requiere la vuelta a "Slow Start", la ventana de congestión puede fijarse al valor de dicha variable.
La incorporación de la invención a una implementación de TCP ya existente, con su capacidad de detectar las cinco situaciones antes descritas y de efectuar las acciones asociadas, puede producir una implementación de TCP que resuelve el problema del uso excesivo del almacenamiento temporal ("buffer-bloat"). No obstante, algunas

implementaciones de TCP se pueden beneficiar de incorporar un subconjunto de dichas detecciones de las cinco situaciones descritas (y sus acciones asociadas). Lo mismo aplica a mecanismos de control de congestión de otros protocolos que pueden estar implementados en otras capas diferentes a la capa de transporte.

Los métodos de esta invención pueden implementase, entre otras posibilidades, como una funcionalidad adicional del TCP de un “host” final de una conexión TCP, o como una funcionalidad adicional del TCP de un “proxy”, tal como muestra la Figura 1C. Un nodo “proxy” TCP 130 es un nodo de red que actúa como intermediario entre dos “hosts” finales 160 y 170 que se comunican vía TCP. El “proxy” TCP 130 termina la conexión TCP hacia los dos “hosts” finales, de una forma transparente con respecto a los “hosts” finales 160, 170. Este tipo de nodo “proxy” 130 puede retransmitir transparentemente la información enviada por un “host” final al otro (150 y 160 en Figura 1C) o puede tratar de añadir valor modificándola de algún modo. De modo similar, los métodos de esta invención podrían aplicarse a otros protocolos además de TCP, donde un nodo “proxy” pueda usarse igualmente. La realización de la presente invención dentro de un nodo “proxy” 130 puede proporcionar beneficios ya que la configuración o implementación de los “hosts” finales 160 y 170 no necesitan cambios para poderse beneficiar de las ventajas de la presente invención.

Debe remarcarse que TCP incluye muchos aspectos aparte del control de la congestión. Dichos aspectos no necesitan ser modificados por la presente invención. Esta invención puede adaptarse fácilmente a un TCP existente con detección de congestión basado en pérdidas añadiendo ciertos pasos en puntos precisos del procesamiento que serán detallados a continuación

La Figura 2 es un diagrama de flujo que ilustra el procesamiento lógico que tiene lugar cuando se recibe 210 un nuevo segmento de confirmación (ACK nuevo) de acuerdo con una realización de la presente invención.

También indica cómo se puede incorporar una realización de la presente invención en una implementación de TCP ya existente. Un ACK nuevo se refiere a un segmento de confirmación que confirma por primera vez y de forma acumulativa datos de usuario de TCP enviados por el “host” destinatario de este ACK nuevo (es decir, los datos no han sido confirmados de forma acumulativa previamente). Hay que destacar que aun cuando es beneficioso efectuar este método tras la recepción de cada ACK nuevo, la presente invención no se limita únicamente a esa implementación. En general, la invención puede efectuar el método con menos frecuencia que con la recepción de cada nuevo ACK. Podría incluso efectuarse periódicamente de acuerdo con algún reloj interno y no con cada nuevo ACK recibido, con lo que podrían aún así estimar las tasa de transmisión.

Además del puerto de origen, el puerto de destino, el número de secuencia, el tamaño de ventana y la suma de comprobación ("checksum" en inglés), la cabecera del protocolo TCP incluye, entre otras, marcas ("flags" en inglés) tales como ACK (indicando la relevancia del campo con el número de secuencia de confirmación), FIN (no más datos del emisor), SYN (número de secuencia de sincronización, enviado únicamente al principio), RST (reinicialización de la conexión) y otros (descritos en detalle en la previamente citada RFC 793). En una implementación TCP, un ACK nuevo es un segmento TCP válido caracterizado por la marca ACK en la cabecera TCP y por no tener ninguna de las otras marcas FIN, SYN y RST. El paso 220 se refiere al procesamiento de un segmento ACK nuevo que puede llevar a cabo el protocolo TCP (u otro protocolo) tal y como se ha descrito previamente, hasta la estimación del RTT. En general, el paso 220 puede incluir la lectura de un segmento de la interfaz de red, la descodificación de los datos de su capa de enlace y red, verificando que el segmento es un segmento TCP correctamente formado, etc. El término SND.UNA en el paso 220 se refiere a un parámetro de estado de la conexión TCP, específicamente al número de secuencia del menor byte aún no confirmado, siendo este parámetro actualizado con el número de secuencia confirmado en cada ACK nuevo recibido. El parámetro cwnd se refiere al tamaño de la Ventana de Congestión en bytes. Como parte de la lógica, un “host” TCP habitualmente calcula el RTT (paso 230). RTT es el tiempo de ida y vuelta ("Round Trip Time" en inglés): el tiempo desde que un cierto segmento TCP es enviado hasta que llega el primer segmento de confirmación que confirma el segmento enviado, y estima el retardo extremo a extremo entre los dos “hosts”.

Una vez calculado el RTT en el paso 230, se ejecutan los pasos 250 de Detección de Congestión y Competencia. Estos pasos sólo se ejecutan si el control de congestión está en "Slow Start" o "Congestion Avoidance" (o modos equivalentes), pero no si está en los modos de "Fast Recovery" o "Fast Retransmit". Esto se ilustra en el paso de decisión 240. En algunas implementaciones de TCP, el RTT no se calcula con cada ACK nuevo. En cualquier caso, incluso en ese caso, la detección de congestión y competencia puede efectuarse en el mismo paso 250 y puede usar el RTT más reciente, valor obtenido por un ACK nuevo previo (por ejemplo el más reciente) para el cual se ha calculado el RTT. En las realizaciones presentadas aquí, se asume que SND.UNA se fija en el paso 220, antes de la Detección de Congestión y Competencia en 250. Sin embargo, SND.UNA podría fijarse tras el paso 250, en cuyo caso el paso 250 usará la secuencia confirmada en el ACK nuevo en vez de SND.UNA.

El paso 260 incluye el resto de lógica requerida por un ACK nuevo en la implementación de TCP. Dicha lógica no es parte de la presente invención y no es requerida por ésta. La forma en que se divide el procesado TCP de un ACK nuevo entre los pasos 220 y 260 no influye en la invención: cualquiera de los pasos 220 o 260 pueden incluir, por ejemplo, actualizar los contadores relativos a TCP o incrementar la Ventana de Congestión de acuerdo con los algoritmos de "Congestion Avoidance" o "Slow Start". El paso 270, Transmisión de datos respetando el tamaño de la Ventana de Congestión, incluye la evaluación de la Ventana de Envío (como función de la Ventana de Congestión, de

la Ventana de Recepción y de los límites del almacenamiento temporal ("buffer") de salida del “host” emisor), y a continuación el envío de los segmentos TCP cuyo número de secuencia final sean menores que la suma de SND.UNA y la Ventana de Envío. Estos segmentos vendrían de la cola de segmentos TCP esperando ser transmitidos en la dirección opuesta a la de recepción del ACK nuevo. Parte del procesamiento del ACK nuevo tras la estimación de RTT 260 podría efectuarse de manera alternativa tras el paso 270.

La implementación del control de congestión puede incluir ya un método para determinar la transición de "Slow Start" a "Congestion Avoidance" antes de que ocurran pérdidas, es decir para fijar el valor del Umbral de Slow Start (ssthresh) igual a la Ventana de Congestión actual en "Slow Start", en función de medidas tomadas durante esa misma fase "Slow Start". En tal caso, la totalidad de dicho método ya existente, que normalmente se invoca al recibir un segmento ACK nuevo, puede reemplazarse por los pasos correspondientes a la detección de congestión en "Slow Start" de acuerdo con una realización de la invención.

Como se explicó anteriormente, la detección de congestión se basa en medidas de la tasa de transmisión y de los datos en vuelo. La tasa de transmisión puede ser tanto la tasa de envío como la tasa de confirmación. La tasa de envío es la tasa a la cual se envían datos de usuario TCP. La tasa de confirmación es la tasa a la cual se confirman datos de usuario TCP.

Adicionalmente, cuando se envía un segmento TCP (es decir, un paquete TCP) que incluye nuevos datos de usuario, si se emplea la tasa de confirmación como tasa de transmisión, es preferible guardar el número de secuencia de SND.UNA en el momento de envío del segmento, de manera que cuando la confirmación de esos datos es recibida, el valor de SND.UNA así guardado puede indicar el valor de SND.UNA en el momento en que se enviaron los datos ahora confirmados. Este valor puede guardarse para todos los segmentos de datos nuevos enviados o sólo para un subconjunto de ellos, pero en esta último caso, la precisión en el cálculo de la tasa de confirmación será menor.

La Figura 3. muestra una posible realización de los métodos de la invención, implementando uno cualquiera de los cinco posibles pasos de detección de congestión (con sus acciones respectivas). En particular, se proporciona un método de control de congestión en un protocolo de comunicación que emplee comunicación confirmada en la cual el nodo transmisor transmite datos a un nodo receptor y el nodo receptor confirma la recepción de los datos, en donde una ventana de congestión especifica una cantidad máxima de datos sin confirmar que el nodo transmisor puede transmitir antes de recibir la confirmación positiva de todos o parte de esos datos. El método incluye medir 310 unos datos en vuelo que indiquen la cantidad de datos enviados por el nodo transmisor y no confirmados aún por el nodo receptor; medir 320 una tasa de transmisión, la tasa de transmisión correspondiendo al mismo instante de tiempo que los datos en vuelo; determinar 350 una tendencia en los datos en vuelo; determinar 360 una tendencia en la tasa de transmisión; y juzgar 370 si hay congestión o no de acuerdo con la tendencia determinada para la tasa de transmisión y con la tendencia de los datos en vuelo. Si a juicio del paso 370 (paso de detección), la congestión fue positivamente detectada, el paso de modificar la Ventana de Congestión 390 sigue a continuación. Los datos a transmitir son transmitidos en el paso 270 respetando el tamaño de la Ventana de Congestión. En el contexto de la Figura 2, la Figura 3 muestra una posible implementación del paso 250 de detección de congestión y competencia, a ejecutar como parte del procesamiento de un ACK nuevo tras el cálculo 230 del RTT.

Adicionalmente a los pasos incluidos en la Figura 3 hay un paso más que se ejecuta tras los pasos descritos en la Figura 3 (ver Figura 2), específicamente la transmisión de datos respetando el tamaño de la Ventana de Congestión 270, ya descrito anteriormente. Los primeros dos pasos de la realización de la Figura 3 son: Medición de los Datos en Vuelo 310 y Medición de la Tasa de Transmisión 320.

El paso de Medida de los Datos en Vuelo 310 obtiene los datos en vuelo usados posteriormente para detectar la congestión y posiblemente para distinguir entre congestión sin y con competencia injusta. Los datos en vuelo se pueden medir (calcular) como sigue. Si se usa la tasa de envío como tasa de transmisión, una muestra apropiada de los datos en vuelo puede obtenerse substrayendo SND.UNA de SND.NXT en el instante en que se procesa el ACK nuevo. Como resultado del procesamiento de un ACK nuevo, normalmente se envían nuevos segmentos. Así, en vez de SND.NXT sería posible usar una estimación de qué valor tendrá SND.NXT una vez se envíen los nuevos segmentos. SND.NXT es un parámetro de estado de una conexión TCP, el primer número de secuencia del siguiente segmento no enviado aún.

Si se usa la tasa de confirmación como tasa de transmisión, se estiman los datos en vuelo en el instante en que el segmento confirmado se envió. Para permitir dicha estimación, la implementación de la invención puede mantener un registro de (es decir, almacenar) el parámetro SND.UNA en el instante en que los segmentos son enviados, para todos los segmentos aún por confirmar o para un subconjunto de ellos. De este modo, una muestra del número de bytes enviados y aún no confirmados se puede obtener substrayendo el SND.UNA almacenado en el momento de envío del último segmento confirmado del SND.UNA en el instante de procesar el ACK nuevo. Alternativamente, con objeto de obtener la estimación más precisa, en vez de usar en los cálculos el SND.UNA (almacenado) en el instante del envío del último segmento confirmado, se puede usar el SND.UNA en el instante en que se envió el primer segmento que está siendo confirmado (que puede ser diferente del último segmento confirmado, especialmente cuando se usan confirmaciones diferidas). Más aún, puede ser ventajoso cuando se almacenen valores de SND.UNA en el instante en que se envían los segmentos, que cuando se envían consecutivamente varios segmentos en ráfaga, en respuesta a

un único segmento de confirmación ACK recibido, se almacenen también cuantos de dicho segmentos son enviados consecutivamente, de forma que en el cálculo de los datos en vuelo puede asignarse el mismo valor de datos en vuelo a la confirmación de cualquiera de los segmentos de la ráfaga, específicamente los datos en vuelo que corresponderían al último segmento enviado de la ráfaga.

Alternativamente, el mismo parámetro cwnd, ventana de congestión, puede usarse para estimar los datos en vuelo, produciendo resultados similares a usar la tasa de confirmación. Sin embargo, cuando los datos en vuelo están limitados por la rwnd anunciada por el receptor o por limitaciones de almacenamiento temporal ("buffer") en el emisor, la ventana de congestión puede ser una estimación bastante imprecisa de los datos en vuelo.

La tasa de transmisión medida puede calcularse en el paso 320 como los datos en vuelo calculados anteriormente dividido por el RTT medido. Dependiendo de qué datos en vuelo se use, se obtendrá la tasa de envío o la tasa de confirmación. Usar la tasa de confirmación da mejores resultados para la detección de la congestión y de la competencia, porque refleja mejor la tasa de transmisión extremo a extremo y también porque es una medida menos ruidosa, aunque puede ser más intensiva en computación y memoria. Por ello, ambas opciones pueden usarse en la implementación de la invención. En situaciones estables, sin pérdidas significativas, debido a la temporización de confirmaciones inherente a TCP y al hecho de que sólo lo que se ha enviado puede confirmarse, las tasas de envío y confirmación son muy similares. Sin embargo, cuando hay variaciones rápidas en la tasa de transmisión, tal y como ocurre en "Slow Start", la tasa de envío puede ser bastante distinta de la tasa de confirmación. En esta situación, usar la tasa de confirmación produciría mejores resultados para la detección de congestión en "Slow Start" que aplicando la tasa de envío. Dado que en general el uso de tasas de confirmación produce los mejores resultados, las realizaciones descritas utilizan tasas de confirmación y sus datos en vuelo correspondientes, pero sería directo modificarlos para que usasen la tasa de envío sin más que cambiar los datos en vuelo conforme se describió anteriormente.

De acuerdo con la realización descrita en Figura 3, tras medir los datos en vuelo y la tasa de transmisión (pasos 310 y 320), se puede determinar la evolución en el tiempo, es decir la tendencia, tanto de los datos en vuelo como de la tasa de transmisión (pasos 350 y 360).

La Figura 7 ilustra una posible implementación de la determinación de la tendencia en los datos en vuelo (paso 350). En esta figura, flightSizeTrend representa una variable a la que se pueden asignar los siguientes valores: "INDETERMINADO", "UP", "DOWN" y "STABLE", y se fija inicialmente a un valor por defecto de "INDETERMINADO" en el paso 710. La variable flightSizeMeasured almacena el valor del paso 310 y si es menos que un valor configurable flightSizeMin en el paso 715, se abandona la estimación de los datos en vuelo (con flightSizeTrend con valor “INDETERMINADO”). A FlightSizeMin puede asignarse un valor por debajo del cual puede no ser ventajoso determinar la tendencia o detectar la congestión, por ejemplo 20*MSS. MSS es el máximo tamaño de segmento o "Maximum Segment Size" en inglés, y corresponde a la mayor cantidad de datos (usualmente en bytes u octetos) que puede transmitirse o recibirse en un único segmento TCP (es decir, dentro de la carga -"payload" en inglés- de un segmento TCP).

La variable congType, utilizada en la Figura 7 y en algunas otras de las realizaciones ejemplares, representa el tipo de congestión que está siendo detectada, y puede tener los valores: “congSS” (para la detección de congestión sin competencia injusta en "Slow Start"), “congNoUnfair” (para la detección de la congestión sin competencia injusta en "Congestion Avoidance"), “congUnfair” (para la detección de la congestión con competencia injusta en "Congestion Avoidance"), “afterSS” (para la detección de congestión con competencia injusta tras pasar de "Slow Start" a "Congestion Avoidance") y “afterCwnd” (para la detección de congestión con competencia injusta tras reducir la ventana de congestión debido a una detección de congestión en "Congestion Avoidance"). En la realización ilustrada en la Figura 3 se supone que congType ya se ha fijado a uno de sus posibles valores, reflejando el tipo de congestión a detectar. Hay que hacer notar que la presente invención puede implementarse igualmente para sólo uno de los tipos de congestión anteriores, en cuyo caso se detectará sólo la presencia o ausencia de ese tipo particular de congestión En la realización ilustrada por la Figura 4, en la cual se usan de manera coordinada los cinco tipos de detección de congestión, se supone que congType se fija inicialmente a "congSS" cuando empieza la conexión en "Slow Start", y posteriormente se cambia a otros valores, de acuerdo con el estado de la red.

En la Figura 7, que ilustra una realización del paso 350, se supone que congType ya está fijado. Si en el paso 720 se fija a “afterCwnd” o “afterSS”, se comparará el valor de SND.UNA con la variable seqRef del paso 725, que guarda el número de secuencia del siguiente segmento contiguo no enviado en el momento de detección de la congestión y, o bien la ventana de congestión se redujo o el estado de control de congestión se cambió de "Slow Start" a "Congestion Avoidance". Si SND.UNA es menor que seqRef, se abandona el procedimiento de estimación de la tendencia en los datos en vuelo (con flightSizeTrend fijado a “INDETERMINADO”). Si SND.UNA no es menor a seqRef el paso 735 comprueba el valor de validRef, una variable que indica si se ha escogido ya una referencia para la estimación de la tendencia. Si validRef es ""Falso"" y el paso 740 comprueba que congType es "afterCwnd", entonces el paso 750 asignará a flightSizeRef (los datos en vuelo a usar como referencia en la estimación de tendencia) el valor guardado en flightSizePrev, es decir, el valor de flightSizeMeasured justo antes de la detección de congestión indicada por seqRef. Se supone que flightSizePrev tiene el valor medido de datos en vuelo antes del valor de datos en vuelo actual, y que el valor fue almacenado en el paso 310 justo artes de actualizar flightSizeMeasured. Si congType en el paso

740 no es "afterCwnd" (es decir, si es "afterSS") entonces en el paso 745 se fija flightSizeRef al valor de flightSizeMeasured. Tanto tras el paso 745 como el 750, se abandona el procedimiento de estimación de la tendencia en los datos en vuelo (con flightSizeTrend fijado a “INDETERMINADO”). Si en el paso 735 validRef es "VERDADERO" (es decir, hay una referencia válida de datos en vuelo para determinar la tendencia), pero en el paso 760 SND.UNA es menos que seqTest, se abandona el procedimiento de estimación de la tendencia en los datos en vuelo (con flightSizeTrend fijado a “INDETERMINADO”). seqTest es una variable que almacena el número de secuencia tras el cual se efectuará la determinación de la tendencia en caso de que congType sea “afterCwnd” o “afterSS”.
Si en el paso 720 congType no es "afterCwnd" ni "afterSS", el paso 795 evalúa validRef y si no es "Verdadero", el paso 745 fija flightSizeRef al valor actual de flightSizeMeasured y se abandona el procedimiento de estimación de la tendencia en los datos en vuelo (con flightSizeTrend fijado a “INDETERMINADO”).

Si en el paso 760 SND.UNA no es menor que seqTest, o si en el paso 795 validRef es "Verdadero", entonces se ejecutará el paso 765 para calcular la variable flightSizeThresh, que almacena un valor de umbral en el cual se basará la tendencia en los datos en vuelo, medido en bytes en esta implementación particular. Este valor de umbral se puede calcular como un proporción fija de los datos en vuelo de referencia (guardados en flightSizeRef), por ejemplo 1/8, que es un valor apropiado y además puede implementarse como un desplazamiento binario. El valor resultante puede restringirse entre un valor mínimo, flightsizeDiffMin (5*MSS, por ejemplo, sería un valor apropiado) y un valor máximo, flightsizeDiffMax (15*MSS, por ejemplo, sería un valor apropiado).

El paso 770 comprueba si el valor actual de los datos en vuelo es menor que su valor de referencia en más del umbral calculado anteriormente, y de serlo, el paso 775 fija la variable flightSizeTrend a "DOWN" y los valores actuales pasan a ser los nuevos valores de referencia tanto para los datos en vuelo (guardados en flightSizeRef) como para la tasa de transmisión (guardado en rateRef). La variable rateRef es análoga a la variable flightSizeRef, pero se utiliza para el valor de referencia de la tasa de transmisión, el cual se utiliza en el paso 360 para calcular la tendencia de la tasa de transmisión. Si el paso 770 determina que los datos en vuelo no es menor que su valor de referencia en más que el umbral calculado anteriormente, entonces el paso 780 comprueba si el valor actual de los datos en vuelo es mayor que su valor de referencia en más del mismo umbral, y si lo es, el paso 780 fija el valor de flightSizeTrend a "UP". Finalmente, si el paso 780 determina que el valor actual de los datos en vuelo no es mayor que su valor de referencia en más del umbral anterior, entonces el paso 790 fija el valor de flightSizeTrend a "STABLE". Tras cualquiera de los pasos 775, 785 y 790 se abandona el procedimiento de estimación de la estimación de los datos en vuelo (con flightSizeTrend fijado a su valor ya determinado, indicando bien "UP" para una tendencia creciente en los datos en vuelo, o "DOWN" para una tendencia decreciente en los datos en vuelo, o "STABLE" para estable, es decir para una tendencia sin cambios substanciales en los datos de vuelo. La tendencia determinada de este modo puede usarse en el paso 370 para detectar la congestión.

La Figura 8 ilustra una posible implementación de la determinación de la tendencia en la tasa de transmisión (paso 360). El primer paso, paso 330 Filtrado de la Tasa de Transmisión, es opcional solo si se realiza el paso 340 Filtrado Gradiente de Tasa (es decir, o bien el paso 330 o el paso 340 o ambos deben implementarse para compensar la variabilidad de las medidas de Tasa de Transmisión). Qué paso de filtrado se utilice dependerá de qué métrica se utilice para el gradiente de la tasa de transmisión. La Tasa de Transmisión (tanto la tasa de envío como la de confirmación) pueden experimentar gran variabilidad, lo que aconseja que se filtre para eliminar ruido de las medidas, a menos que el filtrado se efectúe sobre el gradiente calculado. Un filtro paso bajo muy utilizado y eficiente es un estimador básico de la media de la clase denominada algoritmos de predicción de error recursivos, también llamado Media Móvil con Ponderación Exponencial ("Exponentially Weighted Moving Average", EWMA). La tasa de transmisión filtrada se actualiza a un nuevo valor (RFn) que es función del valor previo RFn-1, el nuevo valor medido Rmeasured y un parámetro de suavizado g:

$$RFn = (1-g)\bullet RFn-1 + g\bullet Rmeasured = RFn-1 + g\bullet(Rmeasured - RFn-1)$$

El signo "•” significa multiplicación, que en la Figuras aparece también como “*”. Por ejemplo, 1/16 es un valor apropiado para g cuando las muestras se obtienen por cada recepción de un ACK nuevo, pero un valor mayor puede ser beneficioso para muestras menos frecuentes. En cualquier caso, la presente invención no está limitada a dichos valores. Más bien, se puede seleccionar un valor mediante pruebas que resulte en un mecanismo de control de congestión más robusto. Se podrían usar otros algoritmos de filtrado, tales como filtros paso bajo de segundo grado.
Una implementación ilustrativa de los principales pasos del filtrado de la Tasa de Transmisión 330 se muestra en la Figura 5, en la que el primer paso 505 es el de guardar una copia del valor actual de rateFiltered en ratePrev, antes de que rateFiltered se actualice. En la Figura 5 rateFilterRestart representa una variable que puede tener los valores "Verdadero" o "Falso". La variable rateMeasured es una variable que guarda la tasa de transmisión medida previamente en el paso 320. La variable rateFiltered guarda el resultado del filtrado de las medidas de tasa de transmisión. Si el valor de restartRateFilter es "Verdadero" en el paso 510, el filtrado se reinicializa fijando rateFiltered al valor de rateMeasured y fijando en el paso 520 restartRateFilter a "Falso". De otro modo (si restartRateFilter es "Falso") el paso 530 actualiza la variable rateFiltered como sigue:

$$rateFiltered = rateFiltered + g\bullet(rateMeasured – rateFiltered)$$

Es de destacar que la implementación del filtrado descrito anteriormente es sólo un ejemplo ventajoso. Sin embargo,

la presente invención de ningún modo se limita a esta implementación particular. Cualquier tipo de filtrado puede aplicarse a la presente invención, incluyendo por ejemplo cualquier filtrado de medias (ponderado) o cualquier filtro paso bajo.

Volviendo a la Figura 8, rateTrend representa una variable a la que se pueden asignar los siguientes valores: "INDETERMINADO", "UP", "DOWN" y "STABLE", y se fija inicialmente a un valor por defecto de "INDETERMINADO" en el paso 810. La variable rateFiltered guarda el valor filtrado de tasa de transmisión del paso 330 (o el valor medido en el paso 320 si no se usa filtrado). Tras el paso 810, si el paso 815 fija congType a "afterCwnd" o "afterSS", el paso 820 compara SND.UNA con la variable seqRef. Si SND.UNA es menor que seqRef, se abandona el procedimiento de estimación de la tendencia en la tasa de transmisión (con rateTrend fijado a “INDETERMINADO”). Si SND.UNA no es menor que seqRef, el paso 825 comprueba el valor de validRef. Si validRef es "FALSO" entonces, el paso 830 asignará a la variable rateRef (la tasa de transmisión a usar como referencia en la estimación de tendencia) el valor guardado en ratePrev, es decir, el valor de rateFiltered justo antes de la detección de congestión indicada por seqRef. Se supone que ratePrev guarda el valor previo de la tasa filtrada, si se usa el paso 330 para filtrar las medidas, o el valor previo de la medida de tasa si no se está usando el paso 330 para filtrar las medidas (en cuyo caso la variable ratePrev debería actualizarse justo antes de que el paso 320 recoja un nuevo valor de rateMeasured). Adicionalmente, el paso 830 fijará las variables restartRateFilter y restartGradFilter a "Verdadero", como una forma de reiniciar el filtrado una vez se haya elegido la referencia de tasa de transmisión para la estimación de la tendencia; la variable validRef se fija a "Verdadero" para indicar que ambos valores de referencia para la tasa de transmisión y para los datos en vuelo se han tomado, y el parámetro SND.NXT se guardará en la variable seqTest, ya que ese es el valor de secuencia en el cual se calculará la tendencia cuando congType sea "afterCwnd" o "afterSS". Tras el paso 830, se abandona el procedimiento de estimación de la tendencia en la tasa de transmisión (con rateTrend fijado a “INDETERMINADO”). Si en el paso 825 validRef es "VERDADERO" (es decir, hay una tasa de transmisión válida como referencia para determinar la tendencia), pero SND.UNA es menor que seqTest en el paso 835, también se abandona el procedimiento de estimación de la tendencia en la tasa de transmisión (con rateTrend fijado a “INDETERMINADO”).

Si en el paso 815 congType no es "afterCwnd" ni "afterSS", el paso 840 evalúa validRef y si no es "Verdadero", el paso 845 fija rateRef al valor actual de rateFiltered, al tiempo que la variable validRef se fija a "Verdadero" para indicar que los valores de referencia de la tasa de transmisión y de los datos en vuelo están tomados, y se abandona el procedimiento de estimación de la tendencia en los datos en vuelo (con flightSizeTrend fijado a “INDETERMINADO”). Si en el paso 835 SND.UNA no es menor que seqTest, o si en el paso 840 validRef es "Verdadero", entonces el paso 850 calculará el gradiente de la tasa de transmisión y la almacenará en la variable rateGradient. Tras el paso 850, el paso 340 Filter Rate Gradient, puede ejecutarse (los detalles internos del paso 340 se describen más delante). Como se expuso previamente, bien el paso 330, bien el paso 340, o bien ambos deben implementarse para combatir la variabilidad en las medidas de tasas de transmisión. Tras el paso 340 (o tras el paso 850 si no se efectúa Transmission Rate Filtering), el paso 855 calculará las variables gradientThreshUp y gradientThreshDown, que almacenan los valores de umbral en los que se basa la determinación de la tendencia en la tasa de transmisión. El gradiente de tasa de transmisión en el paso 850 puede calcularse de distintas maneras y esto influye en cómo se filtra en el paso 340, y en el cálculo del umbral para determinar la tendencia en el paso 855.

Una alternativa de cálculo del gradiente de la tasa de transmisión es como la diferencia entre la tasa de transmisión actual (rateFiltered si se usa el paso 330 para filtrar la tasa de transmisión, rateMeasured en caso contrario) y el valor de referencia (rateRef), es decir:

$$rateGradient = rateFiltered - rateRef$$

Esta alternativa es especialmente ventajosa en caso de un cálculo único del gradiente, como en el caso de congType con valores “afterCwnd” o “afterSS”, donde el objetivo es ver si la tasa de transmisión ha descendido tras un evento específico (el evento puede ser un cambio de "Slow Start" a "Congestion Avoidance" o una reducción de la ventana de congestión debida a una detección de congestión previa).

Cuando se calcula el gradiente como la diferencia entre la tasa de transmisión actual y la tasa de referencia, el filtrado se puede hacer directamente sobre la tasa de transmisión (mediante el paso 330) o en la variable rateGradient en el paso 340, pero se pueden obtener mejores resultados filtrando en el paso 330, ya que el filtrado tiene lugar incluso cuando SND.UNA es menor que seqTest. En ese caso, el umbral para determinar la tendencia ascendente, gradientThreshUp, puede calcularse como una fracción (ejemplo 1/2) de la proporción entre el umbral de datos en vuelo y los datos en vuelo de referencia (es decir, flightSizeThresh/flightSizeRef), aplicada a la tasa de transmisión de referencia (almacenada en rateRef). El valor resultante puede forzarse a que sea menor que cierta fracción de la tasa de transmisión de referencia (por ejemplo 1/16) ya que el valor del umbral se usará para comprobar si la tendencia en la tasa de transmisión es estable, y un valor de umbral demasiado pequeño podría producir más falsas detecciones de estabilidad. El umbral para determinar si hay una tendencia decreciente, gradientThreshDown, puede calcularse ventajosamente como dos veces gradientThreshUp, pero con signo negativo. Se toman dos veces, o algún factor mayor que 1, con objeto de tener menor probabilidad de detectar incorrectamente una tendencia descendente. Resumiendo, los cálculos de gradientThreshUp y gradientThreshDown pueden realizarse como (el signo “•” significa multiplicación y “/” significa división):

gradientThreshUp = ((flightSizeThresh / flightSizeRef) / 2)•rateRef

gradientThreshUp = min(gradientThreshUp, rateRef/16)

gradientThreshDown = - 2 • gradientThreshUp

Alternativamente, los umbrales de gradiente pueden ligarse a la variabilidad en la tasa de transmisión medida. En tal caso, se necesita un estimador de la variabilidad, tal y como la desviación absoluta media. La desviación media en la tasa de transmisión puede calcularse junto con el filtrado en el paso 330, y guardado en una variable rateMeanDev, utilizando una media ponderada exponencial (por ejemplo con un factor a=1/16). Un múltiplo de esta variable (por ejemplo 2 para un umbral de tendencia ascendente y 4 para un umbral de tendencia descendente) puede usarse para calcular los umbrales de tasas de transmisión. Igual que en el cálculo previo, puede ser beneficioso forzar un valor máximo para el umbral. Resumiendo ("||" significa valor absoluto).

rateMeanDev = rateMeanDev + a•(|rateFiltered – rateMeasured| - rateMeanDev)

gradientThreshUp = (2•rateMeanDev, rateRef/16)

gradientThreshDown = - 2 • gradientThreshUp

Otra alternativa para calcular el gradiente puede ser determinar el cambio en la tasa de transmisión asociado a cada cambio en los datos en vuelo. Dado que puede haber varios ACK nuevos contiguos con la misma medida de datos en vuelo y diferentes medidas de la tasa de transmisión (en "Congestion Avoidance", por ejemplo, la ventana de congestión es incrementada una vez por cada cwnd bytes, lo que debe reflejarse normalmente del mismo modo en los datos en vuelo), un filtrado inicial incluiría determinar la tasa de transmisión media para cada valor de datos en vuelo (estos valores medios podrían guardarse en una variable rateFiltered, por ejemplo). Entonces, un gradiente de tasa de transmisión puede ser la división entre el cambio en la tasa de transmisión media y el cambio en datos en vuelo con respecto a un valor previo de datos en vuelo, es decir:

rateGradient = (rateFiltered – ratePrev) / (flightSizeMeasured – flightSizePrev)

En ese caso, el valor de ratePrev correspondería a una tasa media de transmisión calculada cuando los datos en vuelo tenían el valor flightSizePrev, mientras rateFiltered correspondería a la tasa media de transmisión calculada cuando los datos en vuelo eran flightSizeMeasured. Este valor se calcularía una vez por cada cambio en los datos en vuelo. Este gradiente se normalizaría dividiéndolo por (ratePrev/ flightSizePrev). Así normalizado, un valor de 1 correspondería a una situación en la cual la tasa de transmisión crece, como porcentaje, al mismo ritmo que los datos en vuelo, mientras un valor de 0 indicaría una tendencia estable, y un valor negativo una tendencia decreciente. Sin embargo, un gradiente calculado de este modo mostrará una variabilidad relativamente grande, de forma que el paso 340 Filter Rate Gradient sería muy ventajoso para reducir la variabilidad observada. El paso 340 podría implementarse como muestra la Figura 6, que replica directamente la realización del filtrado de ratio de transmisión mostrado en la Figura 5. En ese caso, valores apropiados de umbral serian 0.5 para gradientThreshUp y -1 para gradientThreshDown.

El paso 860 comprueba si el valor actual del gradiente de la tasa de transmisión (posiblemente filtrado) es menor que su umbral gradientThreshDown calculado anteriormente, y si es menor, el paso 865 fija la variable rateTrend a "DOWN". Si el paso 860 determina que la tasa de transmisión filtrada no es menor que su umbral gradientThreshDown calculado anteriormente, entonces el paso 870 comprueba si el valor actual de la tasa de transmisión filtrada es mayor que su umbral gradientThreshUp, y si lo es, el paso 875 fija el valor de variable rateTrend "UP". Finalmente, si el paso 870 determina que el valor actual de la tasa de transmisión filtrada no es mayor que su umbral gradientThreshUp, entonces el paso 880 fija el valor de rateTrend a "STABLE". Tras cualquiera de los pasos 865, 875 y 880, se abandona el procedimiento de estimación de la tendencia de la tasa de transmisión (con rateTrend fijado al valor que se haya determinado, UP, DOWN o STABLE).

Las implementaciones de la determinación de tendencia descritas anteriormente (Figura 7 y 8) son sólo ejemplos. Hay que hacer notar que la presente invención puede adoptar otras implementaciones, entre ellas aquellas especificadas más adelante en este documento para la determinación de tendencias en la tasa de transmisión (usando métodos estadísticos no paramétricos y modelos de residuos). En general, la tendencia en los datos en vuelo o en la tasa de transmisión puede estimarse comparando el gradiente de los datos en vuelo o de la tasa de transmisión con un umbral. El gradiente puede calcularse, por ejemplo, como la diferencia entre el valor actual y previo de los datos en vuelo o de la tasa de transmisión respectivamente. La diferencia puede calcularse entre el valor previo filtrado y el valor actual filtrado, lo cual da lugar a resultados más robustos. Sin embargo, las diferencias pueden calcularse entre los valores previos filtrados y los valores actuales medidos, lo cual puede dar lugar a reacciones más rápidas a cambios temporales. Es beneficioso elegir como valor previo el valor inmediatamente precedente al actual (filtrado o medido). Sin embargo, esto no limita la presente invención y el valor previo puede ser cualquier valor precedente. Más aún, el cálculo de la tendencia puede también efectuarse tomando más de un valor precedente medido o filtrado, los cuales pueden proporcionar la ventaja de una estimación de tendencia más robusta.

La Figura 9. ilustra una posible implementación de la detección de congestión (correspondiente al paso 370)., En el paso 985, si congType es "afterSS" (en el paso 905), añadirá MSS bytes a la variable shadowCwnd. La variable shadowCwnd almacena el valor que habría alcanzado la ventana de congestión si hubiese permanecido en "Slow Start" (congType fijado a "afterSS" significa que se ha abandonado recientemente el estado de "Slow Start"), con objeto de ser capaces de revertir a ese valor en caso necesario. Si congType es fijado a "afterCwnd" en el paso 910 o si es "afterSS" tras el paso 995 (es decir, cuando se trata de detectar congestión con competencia injusta tras una reducción de ventana de congestión o un cambio de "Slow Start" a "Congestion Avoidance"), el paso 915 comprobará si la tendencia de datos en vuelo (flightSizeTrend) previamente determinada (en el paso 350) es "STABLE" o "UP", y si no lo es (es decir, si es "DOWN" o "INDETERMINADO") entonces el paso 930 fijará la variable congestionDetected a "Falso" y el procedimiento de detección de congestión resultará en una detección de congestión negativa (ninguna congestión detectada). Si el flightSizeTrend es "STABLE" o "UP" en el paso 915, el paso 920 fijará la variable validRef a "Falso", de forma que los pasos subsiguientes de determinación de tendencia (pasos 350 y 360) se vean forzados a fijar nuevos valores de referencia. Tras el paso 920, si la tendencia de la tasa de transmisión (rateTrend) previamente determinada (en el paso 360) es "DOWN" en el paso 925, entonces el paso 935 fijará la variable congestionDetected a "Verdadero" y terminará el procedimiento de detección de congestión con un resultado de detección de congestión positivo (la congestión en la red es detectada). De otro modo, si en el paso 925 el rateTrend no es "DOWN" (es decir, si es UP, STABLE o UNDETERMINATE) entonces el paso 930 fijará la variable congestionDetected a "Falso" y terminará el procedimiento de detección de congestión (detección negativa).

Si en el paso 910 congType no es ni "afterCwnd" ni "afterSS", pero en el paso 940 congType es "congUnfair" (es decir, cuando se está tratando de detectar congestión con competencia injusta en modo "Congestion Avoidance"), el paso 945 comprobará si la tendencia de los datos en vuelo (flightSizeTrend) determinada previamente (en el paso 350) es "UP", y si no lo es (es decir, si es "STABLE", "DOWN" o "UNDETERMINATE") entonces el paso 960 fijará la variable congestionDetected a "Falso" y terminará el procedimiento de detección de congestión. Si el flightSizeTrend es "UP" en el paso 945, el paso 950 fijará la variable validRef a "Falso", de forma que los pasos subsiguientes de determinación de tendencia (pasos 350 y 360) se vean forzados a fijar nuevos valores de referencia. Tras el paso 950, si la tendencia de la tasa de transmisión (rateTrend) previamente determinada (en el paso 360) es "DOWN" en el paso 955, entonces el paso 965 fijará la variable congestionDetected a "Verdadero" y terminará el procedimiento de detección de congestión. De otro modo, si en el paso 955 el rateTrend no es "DOWN" (es decir, si es UP, STABLE o UNDETERMINATE) entonces el paso 960 fijará la variable congestionDetected a "Falso" y terminará el procedimiento de detección de congestión (detección negativa).

Si en el paso 940 congType no es "congUnfair" (es decir, cuando se está tratando de detectar congestión sin competencia injusta en modo "Slow Start" o "Congestion Avoidance"), el paso 970 comprobará si la tendencia de los datos en vuelo (flightSizeTrend) determinada previamente (en el paso 350) es "UP", y si no lo es (es decir, si es "STABLE", "DOWN" o "UNDETERMINATE") entonces el paso 990 fijará la variable congestionDetected a "Falso" y terminará el procedimiento de detección de congestión. Si el flightSizeTrend es "UP" en el paso 970, el paso 975 fijará la variable validRef a "Falso", de forma que los pasos subsiguientes de determinación de tendencia (pasos 350 y 360) se vean forzados a fijar nuevos valores de referencia. Tras el paso 975, si la tendencia de la tasa de transmisión (rateTrend) previamente determinada (en el paso 360) es "STABLE" en el paso 980, entonces el paso 985 fijará la variable congestionDetected a "Verdadero" y terminará el procedimiento de detección de congestión. De otro modo, si en el paso 980 el rateTrend no es "STABLE" (es decir, si es UP, DOWN o UNDETERMINATE) entonces el paso 990 fijará la variable congestionDetected a "Falso" y terminará el procedimiento de detección de congestión.

La Figura 9 ilustra la detección de una situación de congestión y tiene en cuenta todas las variantes de congestión mencionadas anteriormente (congestiones en "Slow Start" o "Congestion Avoidance", congestión con o sin competencia). Sin embargo, la presente invención puede también implementar únicamente la detección y manejo de un tipo de congestión o de un subconjunto de estos tipos de congestión. En ese caso, sólo se implementarán subconjuntos de los pasos mostrados en la Figura 9. Por ejemplo, para detectar congestión sin competencia injusta, sólo los pasos 970-990 deben implementarse. Por ejemplo, para detectar congestión con competencia injusta, sólo los pasos 945-965 deben implementarse. En particular, en el método de acuerdo con una realización de esta invención, el paso 370 que detecta si hay o no congestión, incluye la detección de si hay o no congestión con competencia injusta. La congestión con competencia injusta es detectada positivamente (paso 965) cuando la tendencia de los datos en vuelo es creciente (paso 945) y la tendencia de la tasa de transmisión es decreciente (paso 955), y detectada negativamente en caso contrario. Alternativa o adicionalmente, en el método 250 el paso 370 para detectar si hay o no congestión puede incluir la detección de si hay o no congestión sin competencia injusta; y la congestión sin competencia injusta puede detectarse positivamente cuando la tendencia de los datos en vuelo es creciente (paso 970) y la tendencia de la tasa de transmisión es substancialmente igual (paso 980), y detectada negativamente en caso contrario.

La Figura 10 ilustra una posible implementación de la modificación de la ventana de congestión (correspondiente al paso 390). En ambas realizaciones descritas en las Figura 3 y Figura 4, este paso se ejecuta solo si se ha detectado previamente algún tipo de congestión en el paso 370 de detección de congestión. El paso 390 contiene las acciones a realizar en cada uno de los cinco tipos de congestión detectados por el paso 370 y especificado en la variable congType. Si congType es "congSS" en el paso 1010 (es decir, si se ha detectado congestión sin competencia injusta en "Slow Start", por ejemplo en el paso 935), entonces el paso 1015 fijará la variable de umbral de "Slow Start" ssthresh

al valor actual de la ventana de congestión (cwnd), de este modo cambiando el estado de control de congestión a "Congestion Avoidance". Tras el paso 1015, el paso 1020 fijará el shadowCwnd a cwnd, de forma que esta variable se puede usar para seguir el valor que hubiese tenido la ventana de congestión de haber permanecido en "Slow Start" (en las realizaciones descritas, shadowCwnd es actualizado posteriormente en el paso 370). Sin embargo, el paso 1015 solo es necesario si tras la detección de la congestión sin competencia injusta en "Slow Start", congType se fija a "afterSS" para detectar si hay congestión con competencia injusta justo después del cambio de "Slow Start" a "Congestion Avoidance". De este modo, el paso 1015 no sería requerido en la realización general ilustrada en la Figura 3 pero sí lo sería en la realización más específica de la Figura 4 (descrita a continuación en detalle).

Si congType es "congNoUnfair" en el paso 1025 (es decir, si una congestión sin competencia injusta ha sido detectada en "Congestion Avoidance"), entonces el paso 1030 reducirá la ventana de congestión en una cantidad deltaCwnd. La cantidad en bytes (deltaCwnd) pude ser una proporción fija de la ventana de congestión (por ejemplo, un 20%) o una proporción del umbral de datos en vuelo utilizado para determinar la tendencia de los datos en vuelo (flightSizeThresh). En este último caso, es aconsejable que sea algo mayor (por ejemplo 1.2xflightSizeThresh) de forma que si se ha detectado congestión, la ventana de congestión se reduzca por debajo del punto real de congestión, y de esta manera permita oscilar alrededor de ese punto de congestión en rondas sucesivas de crecimiento o reducción de la ventana de congestión por detección de congestión.

El paso 1035 sigue bien el paso 1020 o bien el paso 1030, pero es solo necesario si tras la detección de la congestión sin competencia injusta, congType se fija a "afterSS" o "afterCwnd", para detectar si hay congestión con competencia injusta justo tras la transición de "Slow Start" a "Congestion Avoidance" o tras la reducción de ventana de congestión, respectivamente. De esta manera, el paso 1035 no sería requerido en la realización ilustrada en la Figura 3 pero sería implementado para la de la Figura 4. El paso 1035 incluye el fijar la variable seqRef a SND.NXT, que es el número de secuencia en el cual se elegirán los valores de referencia de los datos en vuelo y de la tasa de transmisión (en realidad, se toman los valores justo anteriores a seqRef, excepto en el caso de datos en vuelo cuando congType es afterCwnd, en cuyo caso se toma el valor justo después de seqRef, tal y como muestran las Figura 7 y Figura 8).

Si congType es "congUnfair" en el paso 1040 (es decir, si se ha detectado congestión con competencia injusta en "Congestion Avoidance"), entonces el paso 1045 fijará la variable de umbral de "Slow Start" (ssthresh) a un valor mayor, por ejemplo 3 veces el valor actual de la ventana de congestión (cwnd), de este modo cambiando el estado de control de congestión a "Slow Start".

Si congType es "afterSS" en el paso 1050 (es decir, si se ha detectado una congestión con competencia injusta tras una transición de "Slow Start" a "Congestion Avoidance"), entonces el paso 1055 fijará la ventana de congestión al valor de la variable shadowCwnd (que refleja el valor que habría alcanzado cwnd si la conexión hubiese permanecido en "Slow Start"). El paso 1055 también fijará la variable ssthresh a un valor mayor, por ejemplo tres veces el valor de la ventana de congestión (cwnd), y de este modo cambia el estado de control de congestión a "Slow Start".

Si congType es "afterCwnd" en el paso 1060 (es decir, si se ha detectado congestión con competencia injusta tras una reducción de la ventana de congestión debida a una detección de congestión), entonces el paso 1065 añadirá a la ventana de congestión el valor almacenado previamente en deltaCwnd reflejando la reducción en cwnd cuando la congestión se detectó previamente. De esta manera, se deshace la reducción en la cwnd.

Para la realización ilustrada en Figura 3, que usa los pasos 330, 340, 350, 360, 370 y 390 tal y como se ilustra en las Figura 5 a 10, es beneficioso inicializar algunas de las variables usadas, en particular las variables restartGradFilter, restartRateFilter y validRef deben inicializarse a "FALSO" cada vez que la conexión empieza o hay un vencimiento del temporizador RTO o un vencimiento del temporizador de inactividad. Adicionalmente, congType debería fijarse al inicio de la conexión al tipo de detección de congestión requerido.

La Figura 10 muestra el manejo de todas las situaciones de congestión descritas anteriormente, y en particular la modificación 390 de la cwnd. Sin embargo, la presente invención puede implementarse solo para condiciones individuales o para subconjuntos de ellas. En general, el paso 390 de modificar la cwnd incluye el incremento (paso 1045) del ssthresh cuando se detecta positivamente la congestión con competencia injusta (paso 1040), lo que cambiará la conexión a "Slow Start". La Figura 10 ejemplifica el incremento de sshthresh como la multiplicación de la cwnd actual por tres. Sin embargo, este es sólo un ejemplo y el incremento puede hacerse con un multiplicador diferente o de otra manera. En el caso de TCP, el incremento de la cwnd al detectar positivamente congestión con competencia injusta se ejecuta de forma beneficiosa si el nodo transmisor está en el estado de control de congestión "Congestion Avoidance", y no se ejecuta en caso contrario.

Alternativa o adicionalmente, el paso 390 de modificar la cwnd puede incluir reducir la cwnd (paso 1030) al detectar positivamente (paso 1025) la congestión sin competencia injusta. Cuando se usa TCP, la reducción de la cwnd al detectar positivamente la congestión sin competencia injusta se ejecuta si el nodo transmisor está en el estado de control de congestión "Congestion Avoidance" ("No" en el paso 1010 al no estar en "Slow Start"), y no se ejecuta en caso contrario.

Alternativa o adicionalmente a las estrategias de gestión de la cwnd descritas anteriormente, el paso 390 de

modificación de la cwnd puede incluir cambiar el actual estado de control de congestión a un nuevo estado de control de congestión al detectar positivamente la congestión con competencia injusta, donde los estados de control de congestión actual y nuevo especifican respectivamente una regla actual y otra nueva para incrementar la cwnd al recibir una confirmación de datos, cuando no se ha detectado ni congestión ni pérdida de datos, y la nueva regla especifica un ritmo de incremento de la cwnd mayor que la regla actual. En el caso de TCP, el cambio del estado de control de congestión actual al nuevo estado al detectar positivamente la congestión con competencia injusta puede ejecutarse si el estado de control de congestión actual es el estado de TCP de "Congestion Avoidance" y no se ejecuta en caso contrario, y el nuevo estado de control de congestión puede corresponder al estado de TCP "Slow Start". En el caso de TCP, el cambio del estado de control de congestión actual al nuevo estado al detectar positivamente la congestión con competencia injusta puede ejecutarse beneficiosamente si el estado de control de congestión actual es el estado de TCP "Congestion Avoidance" y no ejecutarse en caso contrario, y el nuevo estado de control de congestión puede corresponder al estado de TCP "Slow Start".

Alternativa o adicionalmente a las estrategias de gestión de la cwnd descritos anteriormente, el paso 390 de modificación de la cwnd puede incluir cambiar el actual estado de control de congestión a un nuevo estado de control de congestión al detectar positivamente la congestión sin competencia injusta, donde los estados de control de congestión actual y nuevo especifican respectivamente una regla actual y otra nueva para incrementar la cwnd al recibir una confirmación de datos, cuando no se ha detectado ni congestión ni pérdidas de datos, y la nueva regla especifica un ritmo de incremento de la cwnd más lento que la regla actual. En el caso de TCP, el cambio del estado de control de congestión actual al nuevo estado al detectar positivamente la congestión sin competencia injusta puede ejecutarse ventajosamente si el estado de control de congestión actual es el estado de TCP "Slow Start" y no ejecutarse en caso contrario, y el nuevo estado de control de congestión puede corresponder al estado de TCP "Congestion Avoidance".

La Figura 4 muestra una posible realización de los métodos de la invención, incluyendo los cinco tipos de pasos de detección de congestión (con sus acciones asociadas respectivas) de una manera coordinada. La Figura 4 muestra una posible implementación del paso 250 de detección de congestión y competencia, que se supone que se ejecutará como parte del procesamiento de un ACK nuevo tras el cálculo 230 del RTT. Adicionalmente a los pasos incluidos en la Figura 4 hay un paso más que es parte de la invención y que se ejecuta tras los pasos descritos en la Figura 4 (ver Figura 2): Transmitir Datos Respetando la Ventana de Congestión (270), ya descrito anteriormente.

La realización de la invención descrita en la Figura 4 utiliza todos los pasos de la realización descrita en la Figura 3 (310, 320, 350, 360, 370, 380 y 390) y añade más pasos para gestionar la variable congType, que especifica el tipo de detección de congestión utilizado. De este modo, esta realización detectará congestión sin competencia injusta cuando se está en "Slow Start", y cuando se detecta dicha congestión, la conexión se pasará a "Congestion Avoidance". Justo cuando se pasa a "Congestion Avoidance", determinará si hay congestión con competencia injusta en relación al instante en que se hizo la transición, y si así fuera, revertiría la conexión a "Slow Start". Una vez en "Congestion Avoidance", comprobará si hay congestión sin competencia injusta, que será tratada reduciendo la cwnd. Esta reducción de la cwnd debido a la congestión disparará una comprobación especial para verificar si hay congestión con competencia injusta en relación al instante en que se redujo la cwnd, y si se encuentra congestión con competencia injusta, se deshará la reducción en la ventana de congestión. Si la congestión sin competencia injusta no es detectada en "Congestion Avoidance", comprobará también si hay congestión con competencia injusta, y si la hay, cambiará la conexión a "Slow Start".

Sigue una descripción más detallada de los pasos en la realización ilustrada en la Figura 4. Si es parte de una realización como la descrita en la Figura 2, los pasos en la Figura 4 se invocan para todo mensaje ACK nuevo, cuando se está en "Slow Start" o en "Congestion Avoidance". En el paso 405, si la conexión está en "Slow Start", la variable congType se fija a "congSS" en el paso 410, para determinar la detección de congestión sin competencia injusta en "Slow Start". Si no, congType permanecerá en su valor previo. En ambos casos, los siguientes dos pasos se ejecutarán para medir los datos en vuelo, 310, y para medir la tasa de transmisión, ambos pueden implementarse como se ha descrito anteriormente (cuando se describieron los mismos pasos para la realización ilustrada en la Figura 3). Los siguientes pasos serán determinar los datos en vuelo 350 (con una posible realización ilustrada en la Figura 7), determinar la tendencia de la tasa de transmisión 360 (con una posible realización ilustrada en la Figura 8) y detectar la congestión 370 (con una posible realización ilustrada en la Figura 9), todos ellos ya descritos en detalle como parte de la descripción de la realización en la Figura 3.

Entonces, si la variable congestionDetected es "Verdadero" en el paso 380 (es decir, que se detectó algún tipo de congestión en el paso 370), el paso 390 (con una posible realización en la Figura 10) se ejecutará para modificar la cwnd como corresponda. Entonces, si en el paso 450 congType es "afterCwnd" (congestión con competencia injusta tras una reducción de cwnd), el paso 455 fijará congType a "congNoUnfair" (detección de congestión sin competencia injusta en "Congestion Avoidance"). Sin embargo, si congType es "congSS" (detección de congestión sin competencia injusta en "Slow Start"), el paso 460 llevará al paso 465, en el cual se fijará congType a "afterSS" (detección de congestión con competencia injusta tras una transición a "Slow Start"). Finalmente, si congType es "congNoUnfair" (detección de congestión sin competencia injusta en "Congestion Avoidance"), el paso 470 llevará al paso 475, en el cual se fijará congType a "afterCwnd" (detección de congestión con competencia injusta tras una reducción de la ventana de congestión).

Por otro lado, en la Figura 4 si congestionDetected es "Falso" en el paso 380, es decir, no se detectó congestión, entonces el paso 415 verificará si SND.UNA es mayor que seqTest en los casos en que congType sea "afterSS" o "afterCwnd", y si es mayor, el paso 420 fijará congType a "congNoUnfair". Esto se asegura que la detección de congestión con competencia injusta en relación a eventos específicos (la transición a "Congestion Avoidance" y la reducción de la cwnd debido a congestión) se hace únicamente hasta el número de secuencia especificado en seqTest, y tras esto, el modo de detección de congestión cambiará a detección de congestión sin competencia injusta en "Congestion Avoidance". De hecho, si la verificación en el paso 415 no es positiva y si el paso 425 determina que congType es "congNoUnfair" (es decir, se ha comprobado que no se ha encontrado congestión sin competencia injusta en "Congestion Avoidance"), entonces el paso 430 fijará congType a "congUnfair" (congestión con competencia injusta en "Congestion Avoidance") y se ejecutará de nuevo el paso 370 de detección de congestión (con una posible realización ilustrada en la Figura 9), con la misma información de tendencia, pero buscando un tipo de congestión diferente. Si el paso 435 verifica que esta nueva detección de congestión ha sido negativa, entonces el paso 440 fija de nuevo la variable congType a"congNoUnfair", de forma que cuando se reciba el siguiente ACK nuevo se comprobará de nuevo la congestión sin competencia injusta y, si no se encuentra, se comprobará la congestión con competencia injusta. Si el paso 435 detecta la congestión con competencia injusta, entonces el paso 390 (la Figura10 ilustra una posible realización) modificará la cwnd adecuadamente, y finalmente el paso 445 fijará congType a "congNoUnfair". Si la conexión pasa a "Slow Start", tal y como hace el paso 390 ilustrado en la Figura 10, congType no se cambiará a "congSS" hasta que el siguiente ACK nuevo, cuando el paso 405 detecta la congestión en "Slow Start".

Para la realización ilustrada en Figura 4, que usa los pasos 350, 360, 370 y 390 tal y como se ilustra en las Figura 7 a 10, es beneficioso inicializar algunas de las variables usadas, específicamente las variables congType a "congSS", mientras que restartGradFilter, restartRateFilter y validRef deben inicializarse a "FALSO" cada vez que la conexión empieza o hay un vencimiento de del temporizador RTO o un vencimiento del temporizador de inactividad.

Determinación de la tendencia de la tasa de transmisión usando métodos estadísticos no paramétricos

Dado que la tasa de transmisión filtrada puede presentar gran variabilidad y que esa variabilidad puede no seguir una distribución estadística bien definida, puede ser beneficioso usar un método estadístico no paramétrico para determinar la tendencia en la tasa de transmisión. En ese caso, una posible implementación del paso 360 podría todavía basarse en la Figura 8, pero con ciertas diferencias en relación a la implementación descrita anteriormente. La principal diferencia sería que en vez de la variable rateFiltered, se usaría una nueva variable rateSample, siendo rateSample un vector o “array” de N (por ejemplo 5) muestras de rateFiltered. Un paso inicial posible, tras el paso 330 y antes del paso 810, por ejemplo, asignaría valores a los N componentes de rateSample. De este modo, rateSample tendría los últimos N valores de rateFiltered, o un subconjunto de estos últimos valores. En cualquier caso, suponiendo que toda instancia de rateFiltered en la Figura 8 fuera ahora una instancia de rateSample, otro cambió necesario sería que las variables ratePrev y rateRef serían también “arrays” o vectores de N componentes. Cuando en el paso 830 se asigna el valor de rateFiltered a rateRef, y cuando en el paso 845 se asigna el valor de ratePrev, estas asignaciones serían asignaciones vectoriales, es decir, se asignarían todos los componentes del vector o “array”. Otro cambio necesario sería no usar el paso 850 (Calcular el Gradiente de la Tasa de Transmisión) y el paso 340 (Filtrar el Gradiente de la Tasa de Transmisión), e ir directamente al cálculo de gradientThreshUp y GradientThreshDown, los cuales continuarían siendo variables escalares (es decir, no serían vectores o “arrays”), y que se podrían calcular como una fracción del cociente del umbral de los datos en vuelo y de los datos en vuelo de referencia (es decir, flightSizeThresh/flightSizeRef), aplicada a la referencia media de tasa de transmisión (derivada de las muestras guardadas en rateRef). Esto es similar a lo ya descrito anteriormente en una de las posibles realizaciones del paso 855, pero usando la media de las muestras de rateRef, es decir:

$$gradientThreshUp = ((flightSizeThresh / flightSizeRef) / 2) \bullet mean(rateRef)$$

$$gradientThreshUp = min(gradientThreshUp, mean(rateRef)/16)$$

$$gradientThreshDown = -2 \bullet gradientThreshUp$$

Los cambios más importantes estarían en los pasos 860 y 870. El paso 860 comprobaría si el gradiente de la tasa de transmisión es positivo verificando si el vector rateSample es mayor que el vector resultante de la suma de gradientThreshUp a cada uno de los N componentes de rateRef. El paso 870 comprobaría si el gradiente de la tasa de transmisión es negativo verificando si el vector rateSample es menor que el vector resultante de la resta de gradientThreshUp a cada uno de los N componentes de rateRef. Esto significa dos tests de desigualdad (">" y "<"), donde se pueden usar los métodos estadísticos no paramétricos, en particular, el conocido test U de Mann-Whitney (también llamado Mann–Whitney–Wilcoxon, test Wilcoxon de suma de rangos o test de Wilcoxon–Mann–Whitney) que puede usarse para determinar si dos muestras de N valores (cada una representado por un vector o ”array” de N componentes) son mayores (vienen de una población de mediana mayor), menores (vienen de una población con menor mediana) o iguales (vienen de la misma población).

El test se realiza fácilmente asignando un rango a cada uno de los N valores de los dos vectores a comparar, es decir,

ordenando los 2*N valores en una secuencia y asignándoles su número de orden (a los empates se les asigna el mismo número). Entonces, tomando el vector o "array" con el valor de mediana menor, para cada uno de sus N componentes, se cuenta el número de observaciones en el otro vector o "array" que tienen un rango menor (se cuenta 1/2 por cada valor que sea igual). La suma de esas cuentas es el estadístico U. Para N=5, que es un tamaño de muestra apropiado, de acuerdo con las tablas estándar de un test unilateral con un 90% de intervalo de confianza, un valor de U de 5 o menos significa que la muestra (vector o "array") en este caso es menor que la otra, mientras que un valor de 20 significa que la muestra (vector o "array") es mayor que la otra.

Determinación de la tendencia de la tasa de transmisión usando los residuos de un modelo

Una alternativa al modo de determinar la tendencia de la tasa de transmisión es usar un test estadístico para detectar la tendencia en las medidas de tasa de transmisión y usar una métrica producida por el test como el gradiente, filtrando la métrica (por ejemplo, acumulándola o promediándola a lo largo del tiempo). Un posible test para detectar una tendencia ascendente sería ajustar dos modelos a las medidas, uno con el comportamiento esperado sin congestión y el otro con el comportamiento esperado con congestión. Entonces, una métrica que compara cómo de bien se ajustan los dos modelos a las medidas se puede usar como una forma de gradiente del cual derivar la tendencia de la tasa de transmisión. Por ejemplo, cada medida de la tasa de transmisión se puede comparar con un modelo que predice que sea igual que el valor anterior (esto es, un modelo de recorrido aleatorio, que podría esperarse en una situación de congestión), y comparar también con un modelo que predice un incremento en la tasa de transmisión respecto al valor anterior en la misma proporción que el incremento medido en los datos en vuelo (lo que podría esperarse en una situación de no congestión). Esto produciría un error residual en cada caso, los cuales, elevados al cuadrado y divididos entre sí, producirían una métrica que indicaría cuál de los modelos se ajusta mejor a los datos.

Una posible implementación de este método es ilustrada en la Figura 11, que es idéntica a la Figura 8 excepto en unos cuantos aspectos que se pasa a describir seguidamente. El gradiente descrito en el párrafo anterior puede usarse sólo para detectar una tendencia ascendente de la tasa de transmisión, de forma que otro gradiente es necesario para probar si la tendencia es decreciente. Así, en la realización ejemplificada en el paso 360 mostrada en la Figura 11, el gradiente producido por el paso 851 será una variable de dos valores: un gradiente para el test de tendencia decreciente (almacenado en la variable rateGradientUp) y un gradiente para el test de tendencia ascendente (almacenado en la variable rateGradientDown). Cada uno de estos gradientes se filtrarán independientemente en el paso 895 y luego el paso 856 producirá un umbral para el test de tendencia ascendente (guardado en la variable gradientThreshUp) y otro umbral para el test de tendencia decreciente (guardado en la variable gradientThreshDown).

Para el test de tendencia ascendente, se podría usar una métrica que compara el ajuste de los dos modelos descritos, por ejemplo, el cociente de residuos de error al cuadrado.

$$rateGradientUp = (rateMeasured - ratePrev)^2 / [rateMeasured - ratePrev \cdot (1 + (flightSizemeasured - flightSizePrev)/flightSizePrev)]^2$$

Para el test de tendencia decreciente, sería posible usar un gradiente como ya se ha descrito anteriormente en la realización ejemplo del paso 360.

$$rateGradientDown = rateFiltered - rate\ Ref$$

Entonces, el paso 895 Filtrar rateGradienUp y rateGradienDown, podría usar una media ponderada exponencial de ambos gradientes (como se ha descrito anteriormente). Otros métodos de filtrado de rateGradienUp serían posibles, como acumular independientemente la suma de los cuadrados de los residuos para cada uno de los dos modelos, y entonces producir un gradiente filtrado dividiendo las sumas acumuladas de residuos al cuadrado.

Tras el paso 895, el paso 856 podría calcular los dos umbrales:

$$gradientThreshUp = ((flightSizeThresh / flightSizeRef) / 2) \cdot rateRef$$

$$gradientThreshDown = -2 \cdot \min(gradientThreshDown, rateRef/16)$$

$$gradientThreshUp = 2$$

El umbral usado para el cálculo de la tendencia decreciente es el mismo descrito anteriormente en las realización ejemplo del paso 360, mientras el umbral de 2 usado para el gradientThreshUp se alcanzaría cuando los residuos al cuadrado (los errores) en el modelo que predice congestión son mayores (dos veces mayores) que los residuos a cuadrado (errores) en el modelo sin congestión.

Tras el paso 856, los pasos 861 y 871 compararán el umbral y tomarán la determinación de la tendencia casi del mismo modo que en las otras realizaciones descritas previamente del paso 360, excepto que habría un gradiente específico para la determinación de la tendencia ascendente y otro para la determinación de la tendencia descendente.

Una realización de la presente invención incluye un aparato que se configura para ejecutar cualquiera de los métodos descritos anteriormente. Un ejemplo de dicho aparato es ilustrado en la Figura 12. Por ejemplo, se proporciona un aparato para controlar, en un nodo transmisor (tal y como el “host” transmisor 110 o 111, o el “proxy” en 130), la congestión en un protocolo de comunicación que emplee confirmaciones de comunicación, en el cual el nodo emisor envía datos a un nodo receptor y el nodo receptor confirma la recepción de los datos, donde una ventana de congestión especifica la cantidad máxima de datos sin confirmar que el nodo transmisor puede transmitir antes de recibir la confirmación positiva de todos o de parte de esos datos. Dicho aparato comprende una unidad de medida de datos en vuelo 1210 para medir unos datos en vuelo que indiquen una cantidad de datos enviados por el nodo transmisor y no confirmados aún por el nodo receptor; una unidad de medida de tasa de transmisión 1220 para medir una tasa de transmisión, la tasa de transmisión correspondiendo al mismo instante de tiempo que los datos en vuelo; una unidad de cálculo de tendencia de datos en vuelo 1250 para determinar una tendencia de los datos en vuelo; una unidad cálculo de tendencia de tasa de transmisión 1260 para determinar una tendencia en la tasa de transmisión; una unidad de detección de congestión 1270 para detectar si hay o no una congestión de acuerdo con la tendencia determinada para la tasa de transmisión y con la tendencia de los datos en vuelo; una unidad de gestión de congestión 1280 para, al detectarse positivamente la congestión por la unidad de detección de congestión, modificar la ventana de congestión; y una unidad de transmisión 1290 para transmitir datos respetando el tamaño de la ventana de congestión.

En particular, cuando el aparato implementa TCP, dicho aparato incluye ventajosamente una "Unidad de Proceso de Lógica TCP" 1200 y una "Unidad de Detección de Congestión y Competencia" 1295, que al menos lógicamente (y posiblemente también físicamente, es decir, implementados por un dispositivo de procesamiento único) comprende las unidades mencionadas previamente.

La Unidad de Proceso de Lógica TCP 1200 incluye la lógica requerida para enviar y recibir datos usando el protocolo TCP. La Unidad de Recepción 1205 en la Unidad de Proceso de Lógica TCP comprende la funcionalidad de recibir segmentos TCP mientras la Unidad de Transmisión 1290 comprende la funcionalidad de enviar segmentos TCP, en ambos casos de acuerdo con los estándares TCP, tales como la IETF RFC 793 y otros citados anteriormente. En la Unidad de Transmisión en particular también se incluye el control de la ventana de congestión y de la ventana de envío de TCP, de modo que sólo se envíen los paquetes permitidos por dichos parámetros.

La Unidad de Detección de Congestión y Competencia 1295 comprende los otros elementos de la invención, aparte de los mencionados previamente en la Unidad de Transmisión: una Unidad de Medida de Datos en Vuelo 1210, una Unidad de Medida de Tasa de Transmisión 1220, una Unidad de Cálculo de Tendencia de Datos en Vuelo 1250, una Unidad de Cálculo de Tendencia de Tasa de Transmisión 1260, una Unidad de Detección de Congestión 1270 y una Unidad de Gestión de Congestión 1280. La Unidad de Proceso de Lógica TCP 1200 invoca la funcionalidad de la Unidad de Detección de Congestión y Competencia para cada ACK nuevo (tal y como de definió anteriormente) recibido por la Unidad de Recepción para conexiones en los estados de "Slow Start" o "Congestion Avoidance", posiblemente incluyendo el valor de la variable SND.UNA en el momento que el segmento de datos confirmado por el ACK nuevo. La Unidad de Detección de Congestión y Competencia tiene acceso (por ejemplo a través de la memoria) a los valores actuales de las variables SND.NXT, SND.UNA, último RTT medido, cwnd y ssthresh de las conexiones TCP en "TCP Logic Processing Unit", incluyendo la posibilidad de cambiar los valores almacenados en cwnd y ssthresh. Adicionalmente, las unidades internas de la Unidad de Detección de Congestión y Competencia pueden comunicarse entre ellas a través de variables almacenadas en memoria.

La Unidad de Proceso de Lógica TCP 1200 y la Unidad de Detección de Congestión y Competencia 1295, pueden implementarse como parte del núcleo ("kernel") del Sistema Operativo 1310, tal y como ilustra la Figura 13. El núcleo del sistema operativo incluye los controladores ("drivers") de las interfaces de red y las librerías de software que permiten el envío y recepción de paquetes de datos a y desde otros sistemas a través de redes de datos. En tal implementación, la Unidad de Proceso de Lógica TCP 1200 puede ser la funcionalidad TCP del núcleo de la mayoría de sistemas operativos (por ejemplo, Linux, Windows, BSD, OSX, ANdroid, iOS) adaptado para invocar a la Unidad de Detección de Congestión y Competencia 1295 para cada ACK nuevo como se describió anteriormente, permitiendo el acceso a SND.UNA, SND.NXT, último RTT medido, cwnd y ssThresh, también como se detalló antes. La Unidad de Transmisión 1290 y la Unidad de Proceso de Lógica TCP 1200 no requerirían adaptaciones en ese caso, excepto para almacenar el valor de la variable SND.UNA cuando se envían segmentos.

Alternativamente, la Unidad de Proceso de Lógica TCP 1200 y la Unidad de Detección de Congestión y Competencia 1295, pueden implementarse en espacio de usuario ("user-space") tal y como ilustra la Figura 14, y no como parte del "kernel" del sistema operativo 1410. En tal implementación, el aparato tendría un “kernel” del sistema operativo, incluyendo los controladores ("drivers") de las interfaces de red y las librerías de software que permiten el envío y recepción de paquetes de datos a y desde otros sistemas a través de redes de datos. La Unidad de Proceso de Lógica TCP 1200 en esta caso implementa el protocolo TCP en espacio de usuario (fuera del núcleo o “kernel”), incluyendo la Unidad de Transmisión 1290 para enviar los segmentos de datos de acuerdo con el protocolo TCP, es decir, respetando la cwnd y ventana de envío para la conexión, como se describió anteriormente, a través de las Librerías del Sistema 1420, que son la interfaz para comunicarse con el “kernel” del sistema operativo. La Unidad de Proceso de Lógica TCP 1200 almacena es este caso el valor de SND.UNA cuando se envían los segmentos.

La implementación de las unidades comprendidas en la Unidad de Detección de Congestión y Competencia 1295

puede realizarse configurando la ejecución de los pasos descritos anteriormente, como parte de la realización ejemplar de los métodos de esta invención. En particular, la Unidad de Medida de Datos en Vuelo 1210 se configura para ejecutar el paso 310, Medir Datos en Vuelo; la Unidad de Medida de Tasa de Transmisión 1220 se configura para ejecutar el paso 320, Medir la Tasa de Transmisión; la Unidad de Cálculo de Tendencia de Datos en Vuelo 1250 se configura para ejecutar el paso 350, Determinar Tendencia de Datos en Vuelo; la Unidad de Cálculo de Tendencia de Tasa de Transmisión 1260 se configura para ejecutar el paso 360, Determinar Tendencia de Tasa de Transmisión; la Unidad de Detección de Congestión 1270 se configura para ejecutar el paso 370, Detección de Congestión; y la Unidad de Gestión de Congestión 1280 se configura para ejecutar el paso 390, Modificar la Ventana de Congestión, pero sólo en casos cuando la Unidad de Detección de Congestión ha detectado congestión.

Además, la presente invención puede realizarse en un aparato como el mostrado en la Figura 15, correspondiente a un dispositivo de procesamiento que comprende uno o más procesadores 1510, una memoria o conjunto de memorias 1520 y una o más interfaces de red 1530, todo ello interconectado, con la memoria almacenando las instrucciones que, al ser ejecutadas por el procesador, hacen que el procesador realice operaciones que comprenden los pasos de cualquiera de los métodos antes descritos. En particular, el o los procesadores pueden configurarse para recibir mensajes de ACK nuevos (paso 220 descrito como parte de una realización de los métodos de la invención e ilustrada en la Figura 2); procesar los mensajes de ACK nuevos, posiblemente incluyendo el actualizado del parámetro SND.UNA (paso 220 descrito como parte de la realización de los métodos de la invención e ilustrado en la Figura 2); actualización de medidas de RTT (el paso 230, descrito como parte de una realización de los métodos de la invención e ilustrado en la Figura 2); si la conexión del ACK recibido está en "Slow Start" o "Congestion Avoidance", entonces se efectúa la actualización de la detección de congestión y competencia (el paso 250 descrito como parte de una realización de los métodos de la invención e ilustrada en la Figura 2); el procesamiento adicional de los ACK nuevos tal y como requiere TCP (el paso 260 descrito como parte de una realización de los métodos de la invención e ilustrada en la Figura 2); y transmitir datos respetando la actualización de la cwnd (paso 270 descrito como parte de una realización de los métodos de la invención e ilustrada en la Figura 2). El aparato de la Figura 15 puede guardar instrucciones para ejecutar cualquiera de los métodos descritos más arriba en la memoria 1520 o en una pluralidad de tales memorias. Las instrucciones pueden ejecutarse por el procesador 1510 o por una pluralidad de tales procesadores y el control de congestión puede efectuarse basado en la comunicación, tal y como confirmaciones o datos, transmitidos o recibidos a través de la interfaz de red 1530 y evaluados tal y como se describe anteriormente (medido, filtrado, procesado para obtener tendencias, condiciones del test).

La Detección de Congestión y Competencia puede ser implementada por ejemplo como se especificó previamente en la descripción detallada de la realización descrita en la Figura 3. Puede implementarse también por ejemplo como se especificó previamente en la descripción detallada de la realización descrita en la Figura 4.

Los procesadores utilizados para la implementación del aparato pueden ser por ejemplo procesadores de propósito general, microcontroladores, procesadores optimizados para teléfonos móviles o tabletas, ASIC a medida, FPGA u otros dispositivos similares. La memoria puede ser RAM, ROM, EPROM, EEPROM, discos duros, discos de estado sólido, o dispositivos similares, o cualquier combinación de ellos. Las interfaces de red pueden permitir el envío y recepción de datos a través de redes vía puertos alámbricos (ejemplos: Ethernet, cable, fibra, ISDN, ADSL) o puertos inalámbricos (ejemplos: Wifi, WiMax, GPRS, UMTS, HSPA, LTE, enlaces de microondas). Las redes a través de las cuales pueden comunicarse el aparado que realiza la presente invención son, por ejemplo, LAN, WAN, satélite, inalámbrico 3G, inalámbrico 4G, cable, red de transporte interna ("backhaul"), etc.

Los aparatos ejemplificados anteriormente implementan comunicaciones TCP. Sin embargo, la presente invención no está limitada a ello y pude usarse también para cualquier protocolo de comunicación que use confirmaciones.

Además, los aparatos descritos pueden aplicarse a los extremos tales como “hosts” (ilustrado en la Figura 1A) y también a los nodos “proxy” (ilustrados en la Figura 1B). En el último caso, por ejemplo, en un “proxy” TCP, un aparato realizando la invención aplicaría la detección y control de congestión y competencia a la funcionalidad de envío.

En resumen, la presente invención pertenece al campo del control de congestión en comunicaciones con confirmación a través de redes. La congestión es detectada basándose en las tendencias tanto de los datos en vuelo como de la tasa de transmisión con objeto de adaptar la Ventana de Congestión de acuerdo con los resultados de detección. Tal detección de la congestión permite, por ejemplo, distinguir entre la congestión con o sin competencia injusta. Además, la tasa de transmisión medida puede filtrarse para compensar las variaciones temporales. La invención puede estar incorporada en un nodo final o en un proxy.

REIVINDICACIONES

1. Un aparato para controlar para un nodo de transmisión, la congestión en un protocolo de comunicación de datos que emplea comunicación confirmada en la cual el nodo transmisor transmite datos a un nodo receptor y el nodo receptor confirma la recepción de los datos, comprendiendo el aparato:
uno o más procesadores (1510); y
una o más memorias (1520) que almacenan instrucciones que, cuando son ejecutadas por el uno o más procesadores, hacen que el uno o más procesadores realice los pasos de:
medir (310) datos en vuelo que indiquen una cantidad de datos enviados por el nodo transmisor y no confirmados aún por el nodo receptor;
determinar (320) una tasa de transmisión, la tasa de transmisión correspondiendo al mismo instante de tiempo que los datos en vuelo;
determinar (350) una tendencia en los datos en vuelo,
determinar (360) una tendencia en la tasa de transmisión, donde la tendencia se deriva de un cálculo del gradiente de la tasa de transmisión, en el cual se filtra la tasa de transmisión determinada o los cálculos del gradiente de tasa de transmisión, o ambos, con objeto de reducir su variabilidad en el tiempo,
detectar (370) si hay o no una congestión de acuerdo con la tendencia determinada para la tasa de transmisión y con la tendencia de los datos en vuelo,
al detectar positivamente la congestión, modificar los datos en vuelo (390); y
transmitir datos respetando una cantidad máxima de datos sin confirmar que el nodo transmisor puede transmitir (270).

2. El aparato según la reivindicación 1, donde
el paso de detectar si hay o no congestión, la congestión es detectada positivamente cuando la tendencia de los datos en vuelo es creciente o substancialmente estable y la tendencia de la tasa de transmisión es decreciente, y detectada negativamente en caso contrario.

3. El aparato según la reivindicación 2, donde el paso de modificar los datos en vuelo incluye incrementar los datos en vuelo al detectar positivamente la congestión.

4. El aparato según la reivindicación 3, donde
en el paso de determinar una tendencia de los datos en vuelo, la tendencia de los datos en vuelo se determina a lo largo de un periodo de tiempo que comienza en el instante posterior y más próximo a la reducción más reciente de los datos en vuelo debida a una detección de congestión; y
en el paso de determinar una tendencia de la tasa de transmisión, la tendencia de la tasa de transmisión se determina a lo largo de un periodo de tiempo que comienza en un instante previo y más próximo a dicha reducción más reciente de los datos en vuelo debido a una detección de congestión.

5. El aparato según la reivindicación 2, donde
el paso de modificar los datos en vuelo incluye cambiar de un estado actual de control de congestión a un nuevo estado de control de congestión al detectar positivamente la congestión, donde los estados de control de congestión actual y nuevo especifican respectivamente una regla actual y una nueva regla para incrementar los datos en vuelo al recibir confirmaciones de datos, y
la nueva regla produce un ritmo de incremento de los datos en vuelo más rápido que la regla actual.

6. El aparato según la reivindicación 5, donde
en los pasos de determinar una tendencia de los datos en vuelo y de determinar una tendencia de la tasa de transmisión, la tendencia de los datos en vuelo y la tendencia de la tasa de transmisión se determinan a lo largo de un período de tiempo que empieza en un instante anterior y más próximo al cambio más reciente de un estado de control de congestión previo al estado de control de congestión actual;
el estado de control de congestión previo y el estado de control de congestión actual especifican respectivamente una regla previa para modificar los datos en vuelo y una regla actual para modificar los datos en vuelo al recibir confirmaciones de datos; y
el estado de control de congestión previo produce un ritmo de incremento de los datos en vuelo más rápido que el estado de control de congestión actual.

7. El aparato según la reivindicación 1,
donde en el paso de detectar si hay o no congestión, la congestión se detecta positivamente cuando la tendencia de los datos en vuelo es creciente y la tendencia de la tasa de transmisión es sustancialmente estable, y se detecta negativamente en caso contrario.

8. El aparato según la reivindicación 7, donde el paso de modificar los datos en vuelo incluye reducir los datos en vuelo al detectar positivamente la congestión.

9. El aparato según la reivindicación 7, donde
el paso de modificar los datos en vuelo incluye cambiar de un estado de control de congestión actual a un nuevo estado de control de congestión al detectar positivamente la congestión,

el estado de control de congestión actual y el nuevo estado de control de congestión especifican respectivamente una regla actual para modificar los datos en vuelo y una nueva regla para modificar los datos en vuelo al recibir confirmaciones de datos, y
la nueva regla produce valores de datos en vuelo menores que la regla actual.

10. El aparato según cualquiera de las reivindicaciones 1 a 9, donde
la tendencia de los datos en vuelo se calcula como una comparación entre los datos en vuelo actuales y una estimación de los datos en vuelo de referencia.

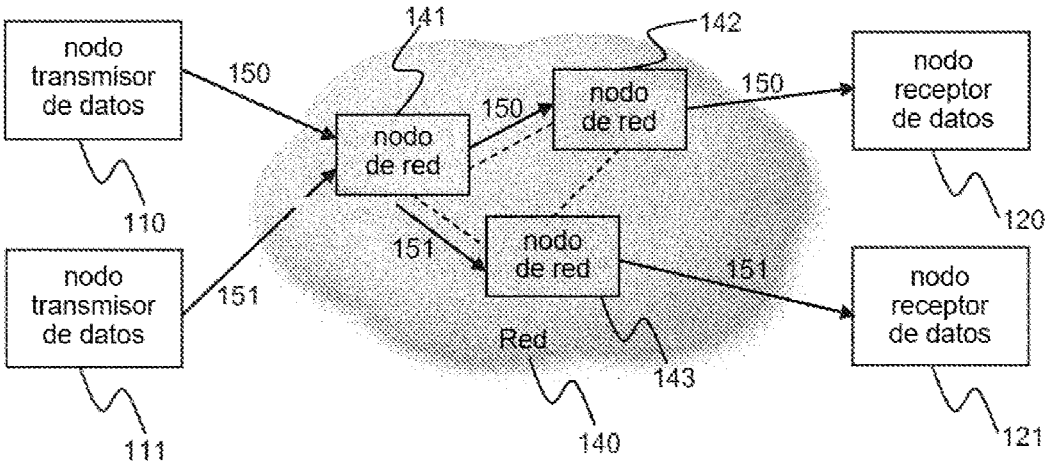

Fig. 1A

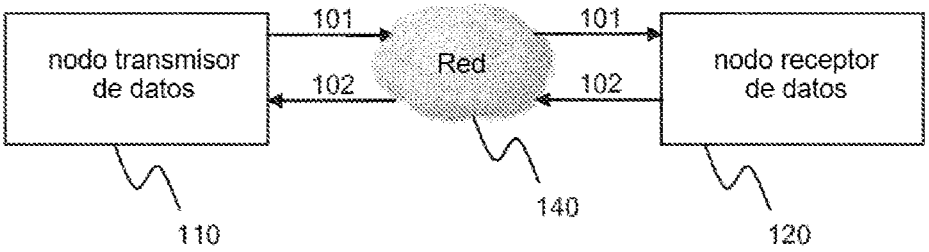

Fig. 1B

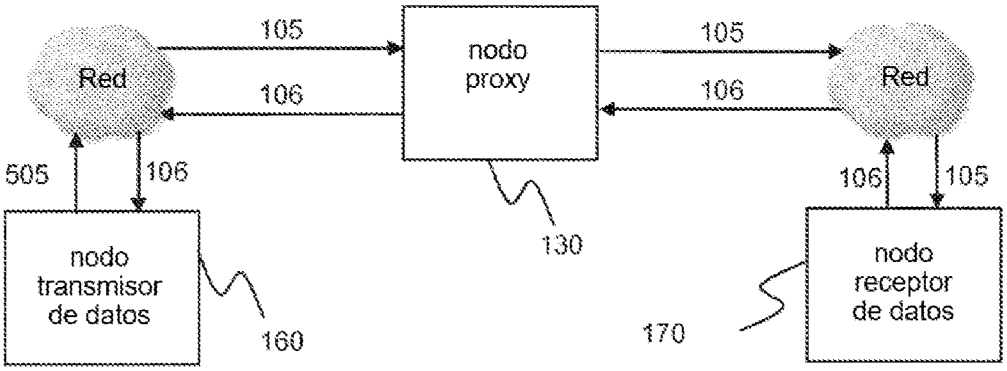

Fig. 1C

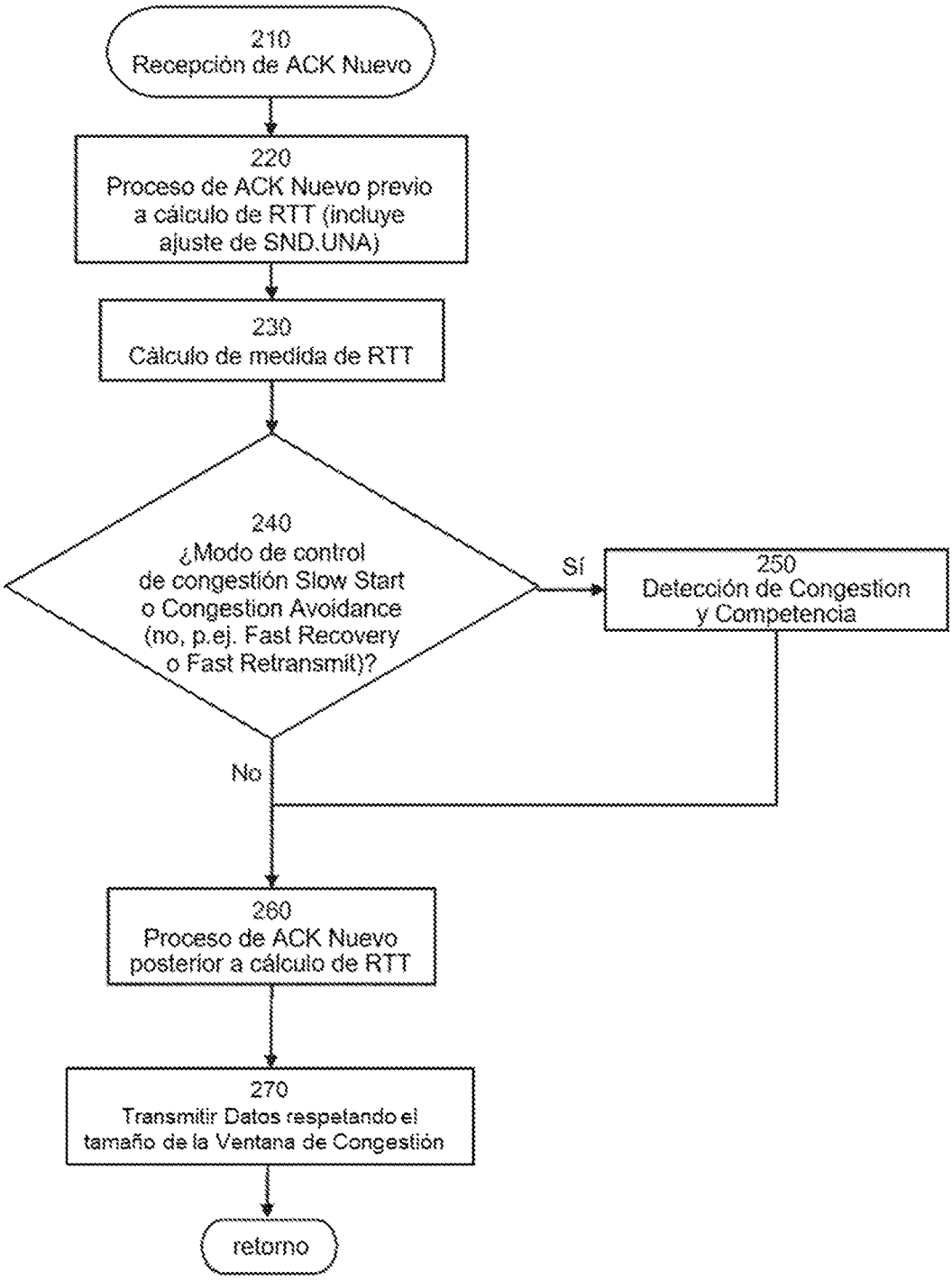
210
Recepción de ACK Nuevo
220
Proceso de ACK Nuevo previo a cálculo de RTT (incluye ajuste de SND.UNA)
230
Cálculo de medida de RTT
240
¿Modo de control de congestión Slow Start o Congestion Avoidance (no, p.ej. Fast Recovery o Fast Retransmit)?
Sí
250
Detección de Congestion y Competencia
No
260
Proceso de ACK Nuevo posterior a cálculo de RTT
270
Transmitir Datos respetando el tamaño de la Ventana de Congestión
retorno

Fig. 2

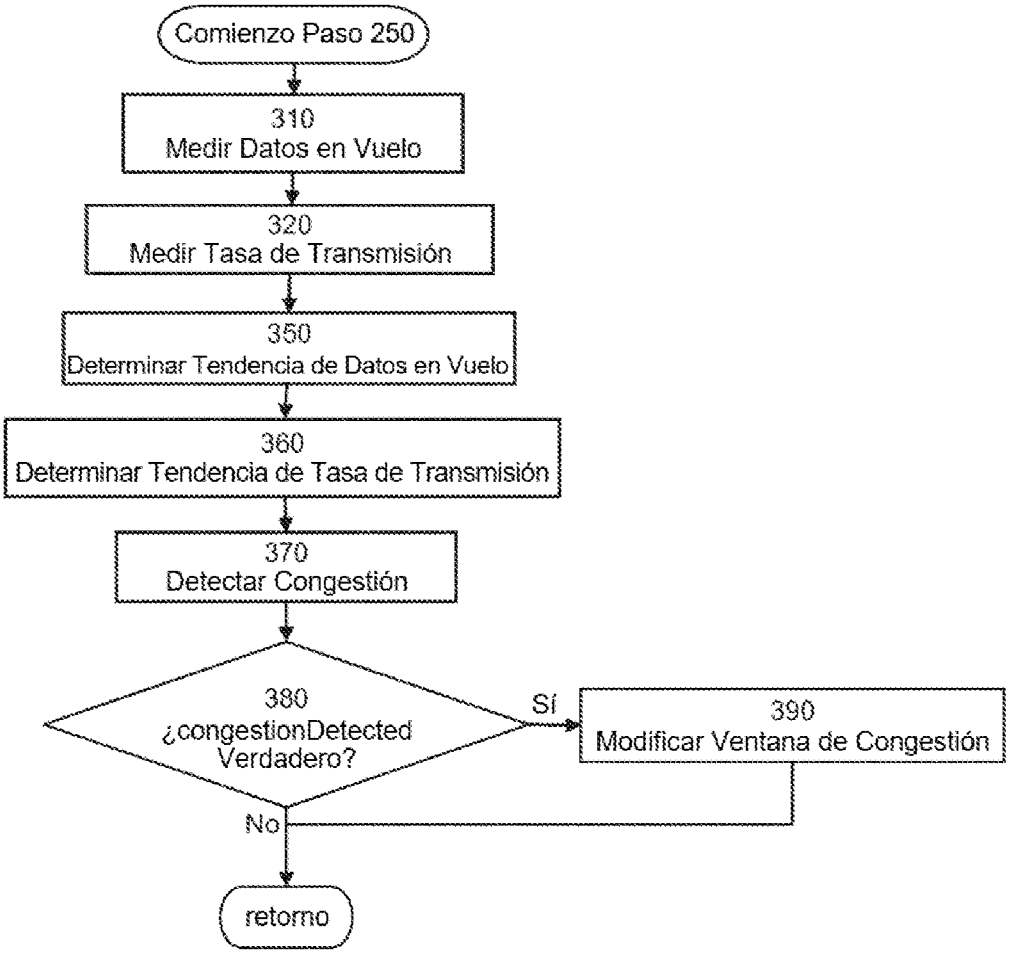
Comienzo Paso 250
310
Medir Datos en Vuelo
320
Medir Tasa de Transmisión
350
Determinar Tendencia de Datos en Vuelo
360
Determinar Tendencia de Tasa de Transmisión
370
Detectar Congestión
380
¿congestionDetected
Verdadero?
Sí
390
Modificar Ventana de Congestión
No
retorno

Fig. 3

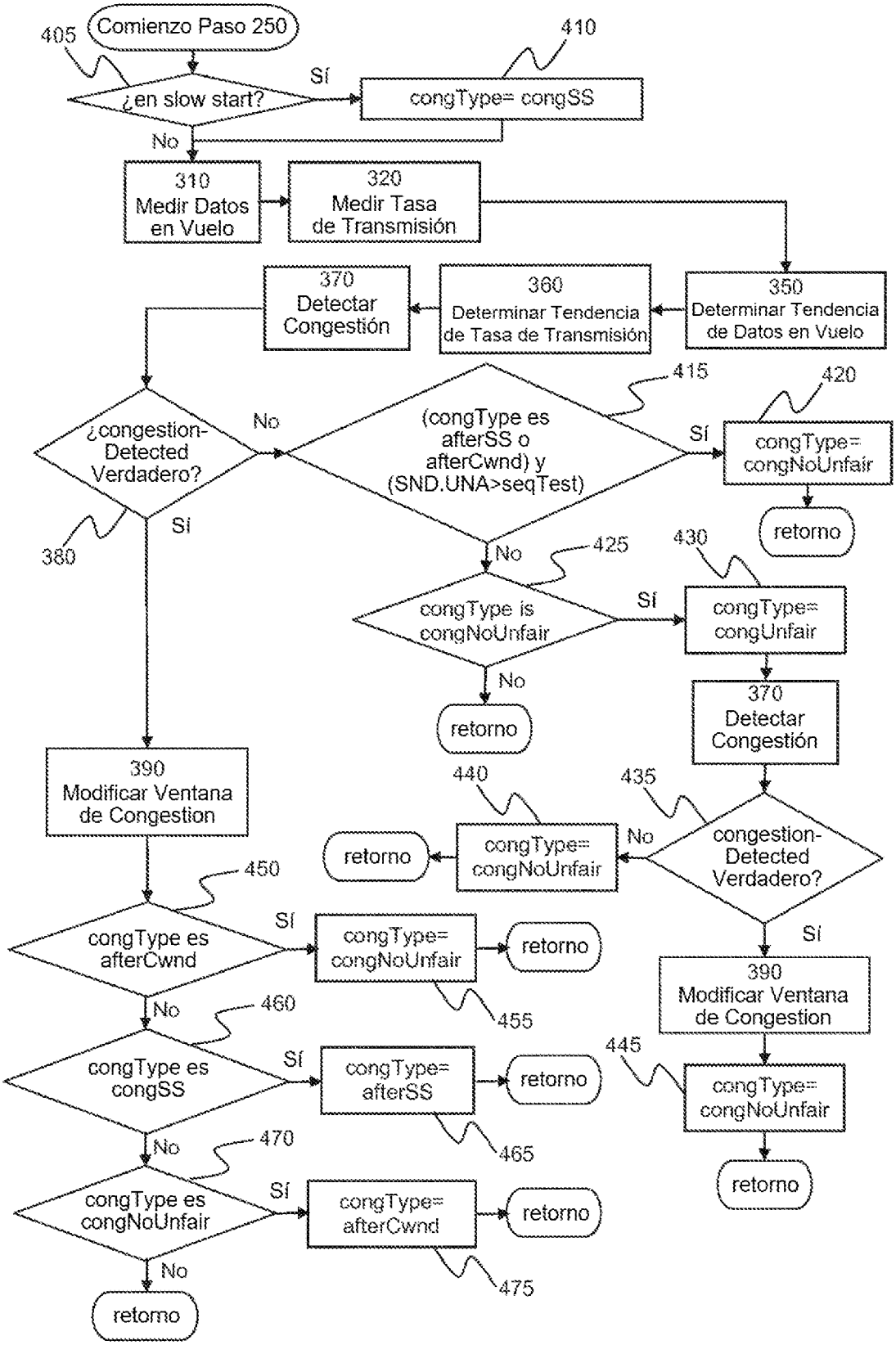
405
Comienzo Paso 250
¿en slow start?
Sí
410
congType= congSS
No
310
Medir Datos en Vuelo
320
Medir Tasa de Transmisión
350
Determinar Tendencia de Datos en Vuelo
360
Determinar Tendencia de Tasa de Transmisión
370
Detectar Congestión
¿congestion-Detected Verdadero?
No
Sí
380
415
(congType es afterSS o afterCwnd) y (SND.UNA>seqTest)
Sí
420
congType= congNoUnfair
retorno
No
425
congType is congNoUnfair
Sí
430
congType= congUnfair
No
retorno
370
Detectar Congestión
390
Modificar Ventana de Congestion
440
congType= congNoUnfair
retorno
435
congestion-Detected Verdadero?
No
Sí
450
congType es afterCwnd
Sí
congType= congNoUnfair
retorno
455
No
460
congType es congSS
Sí
congType= afterSS
retorno
465
No
470
congType es congNoUnfair
Sí
congType= afterCwnd
retorno
475
No
retorno
390
Modificar Ventana de Congestion
445
congType= congNoUnfair
retorno

Fig. 4

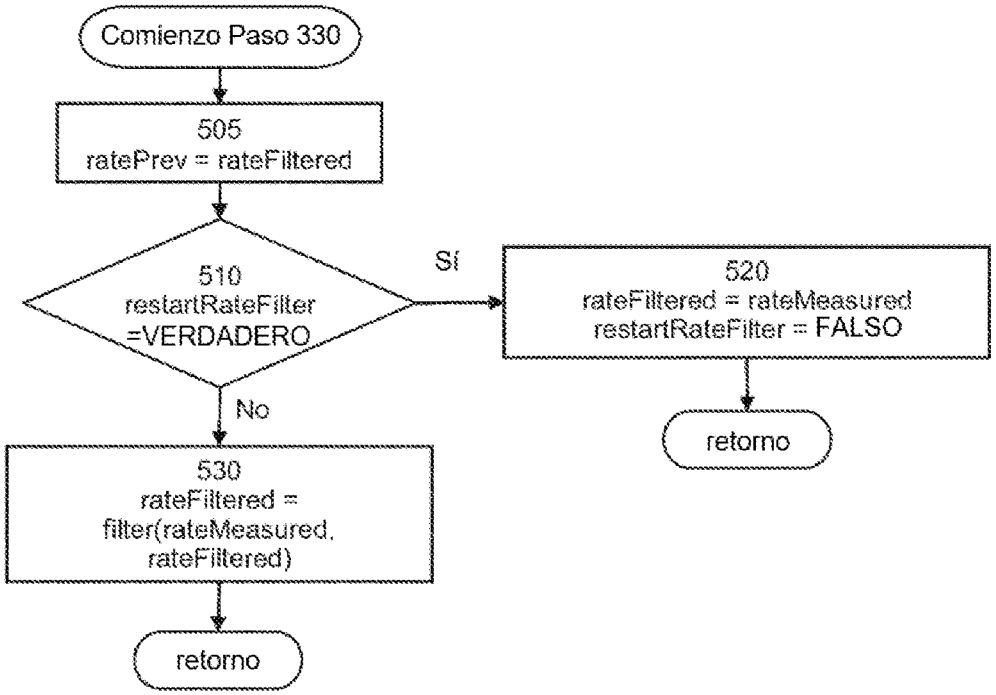
Comienzo Paso 330
505
ratePrev = rateFiltered
510
restartRateFilter
=VERDADERO
Sí
520
rateFiltered = rateMeasured
restartRateFilter = FALSO
retorno
No
530
rateFiltered =
filter(rateMeasured,
rateFiltered)
retorno

Fig. 5

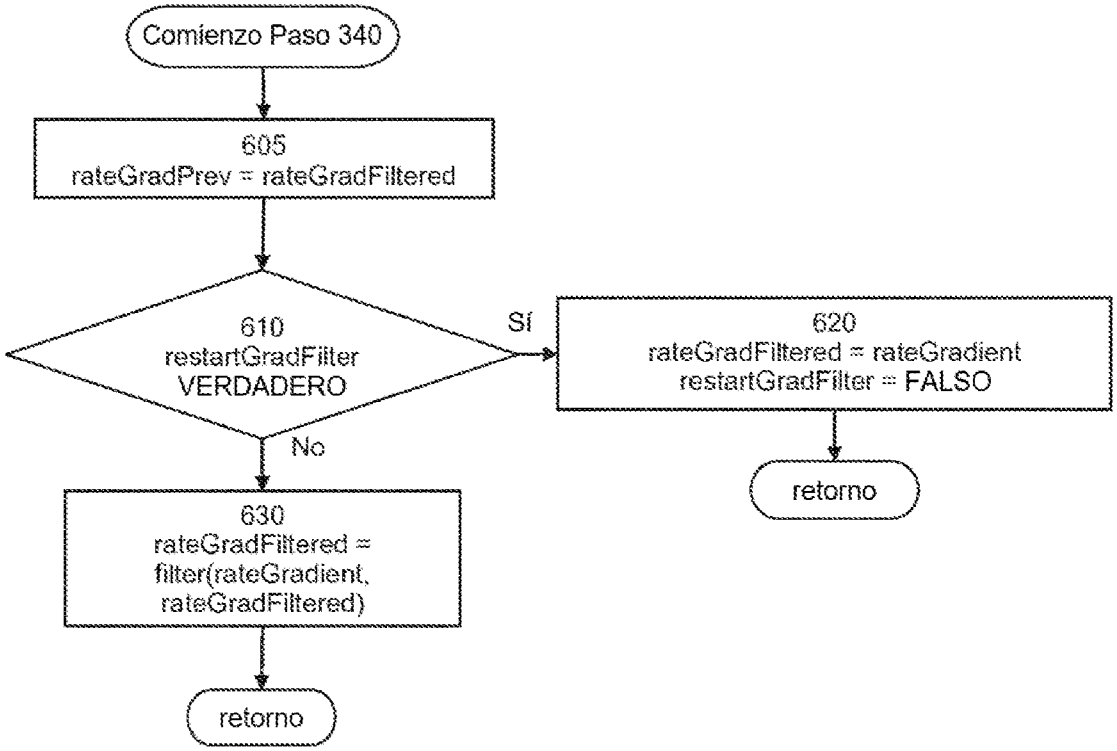
Comienzo Paso 340
605
rateGradPrev = rateGradFiltered
610
restartGradFilter
VERDADERO
Sí
620
rateGradFiltered = rateGradient
restartGradFilter = FALSO
retorno
No
630
rateGradFiltered =
filter(rateGradient,
rateGradFiltered)
retorno

Fig. 6

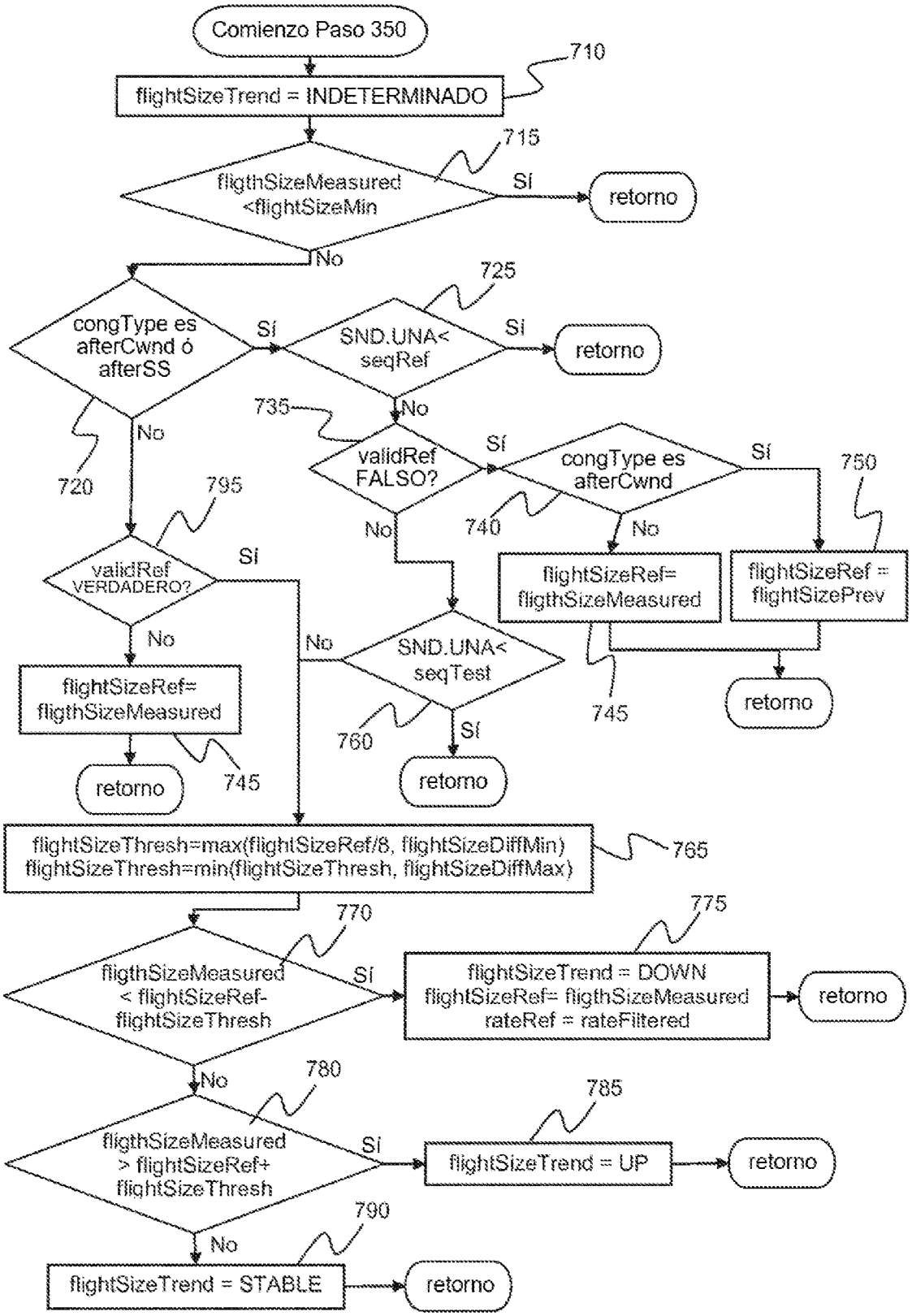
Comienzo Paso 350
710
flightSizeTrend = INDETERMINADO
715
fligthSizeMeasured <flightSizeMin
Sí
retorno
No
725
congType es afterCwnd ó afterSS
Sí
SND.UNA< seqRef
Sí
retorno
No
735
No
720
validRef FALSO?
Sí
congType es afterCwnd
Sí
750
795
740
No
No
validRef VERDADERO?
Sí
flightSizeRef= fligthSizeMeasured
flightSizeRef = flightSizePrev
No
No
SND.UNA< seqTest
745
retorno
flightSizeRef= fligthSizeMeasured
760
Sí
retorno
745
retorno
flightSizeThresh=max(flightSizeRef/8, flightSizeDiffMin)
flightSizeThresh=min(flightSizeThresh, flightSizeDiffMax)
765
770
775
fligthSizeMeasured < flightSizeRef- flightSizeThresh
Sí
flightSizeTrend = DOWN
flightSizeRef= fligthSizeMeasured
rateRef = rateFiltered
retorno
No
780
785
fligthSizeMeasured > flightSizeRef+ flightSizeThresh
Sí
flightSizeTrend = UP
retorno
790
No
flightSizeTrend = STABLE
retorno

Fig. 7

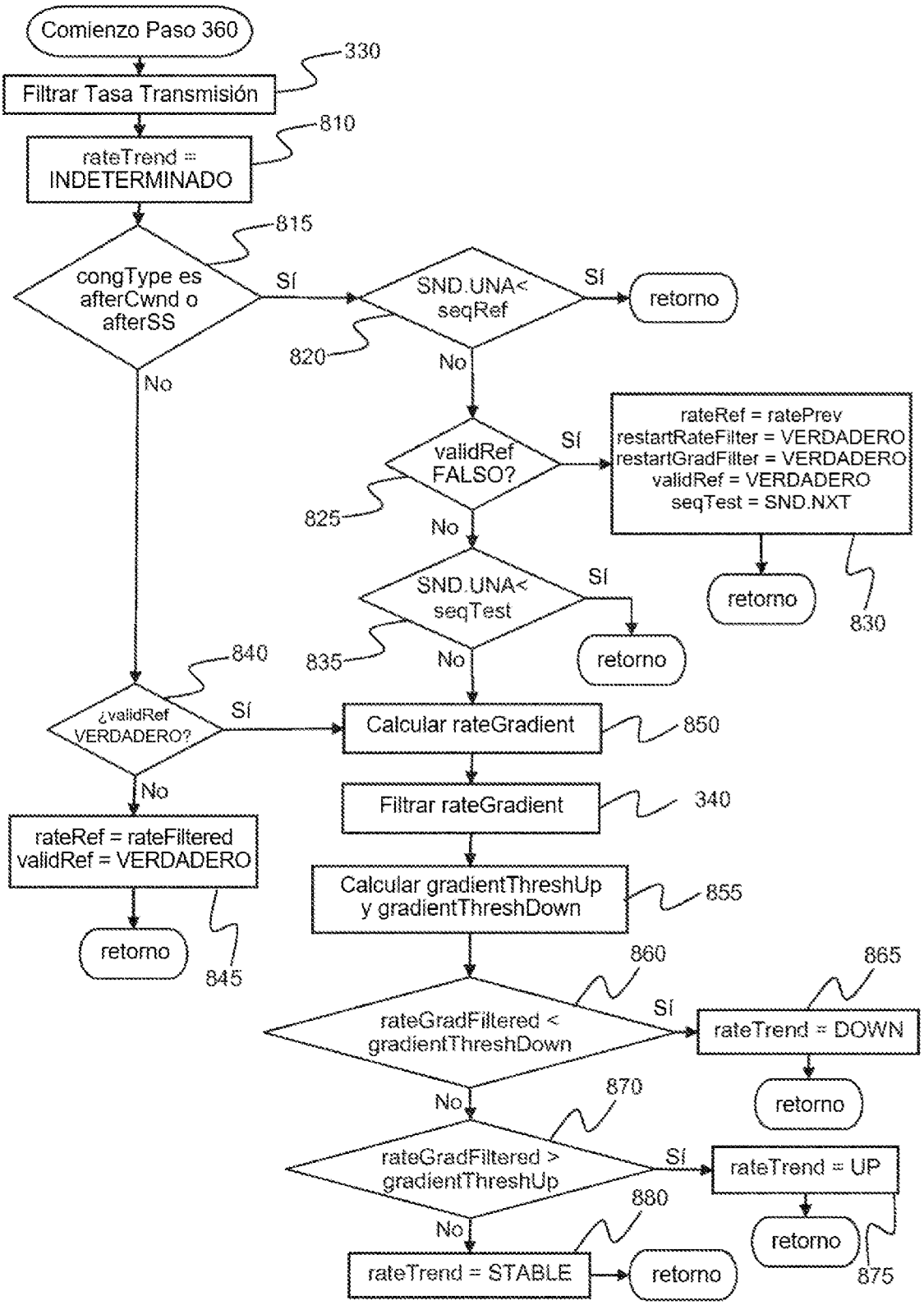
Comienzo Paso 360
330
Filtrar Tasa Transmisión
810
rateTrend = INDETERMINADO
815
congType es afterCwnd o afterSS
Sí
SND.UNA< seqRef
Sí
retorno
820
No
No
validRef FALSO?
Sí
rateRef = ratePrev
restartRateFilter = VERDADERO
restartGradFilter = VERDADERO
validRef = VERDADERO
seqTest = SND.NXT
825
No
retorno
830
SND.UNA< seqTest
Sí
retorno
835
No
840
¿validRef VERDADERO?
Sí
Calcular rateGradient
850
No
Filtrar rateGradient
340
rateRef = rateFiltered
validRef = VERDADERO
Calcular gradientThreshUp y gradientThreshDown
855
retorno
845
860
865
rateGradFiltered < gradientThreshDown
Sí
rateTrend = DOWN
retorno
No
870
rateGradFiltered > gradientThreshUp
Sí
rateTrend = UP
retorno
875
No
880
rateTrend = STABLE
retorno

Fig. 8

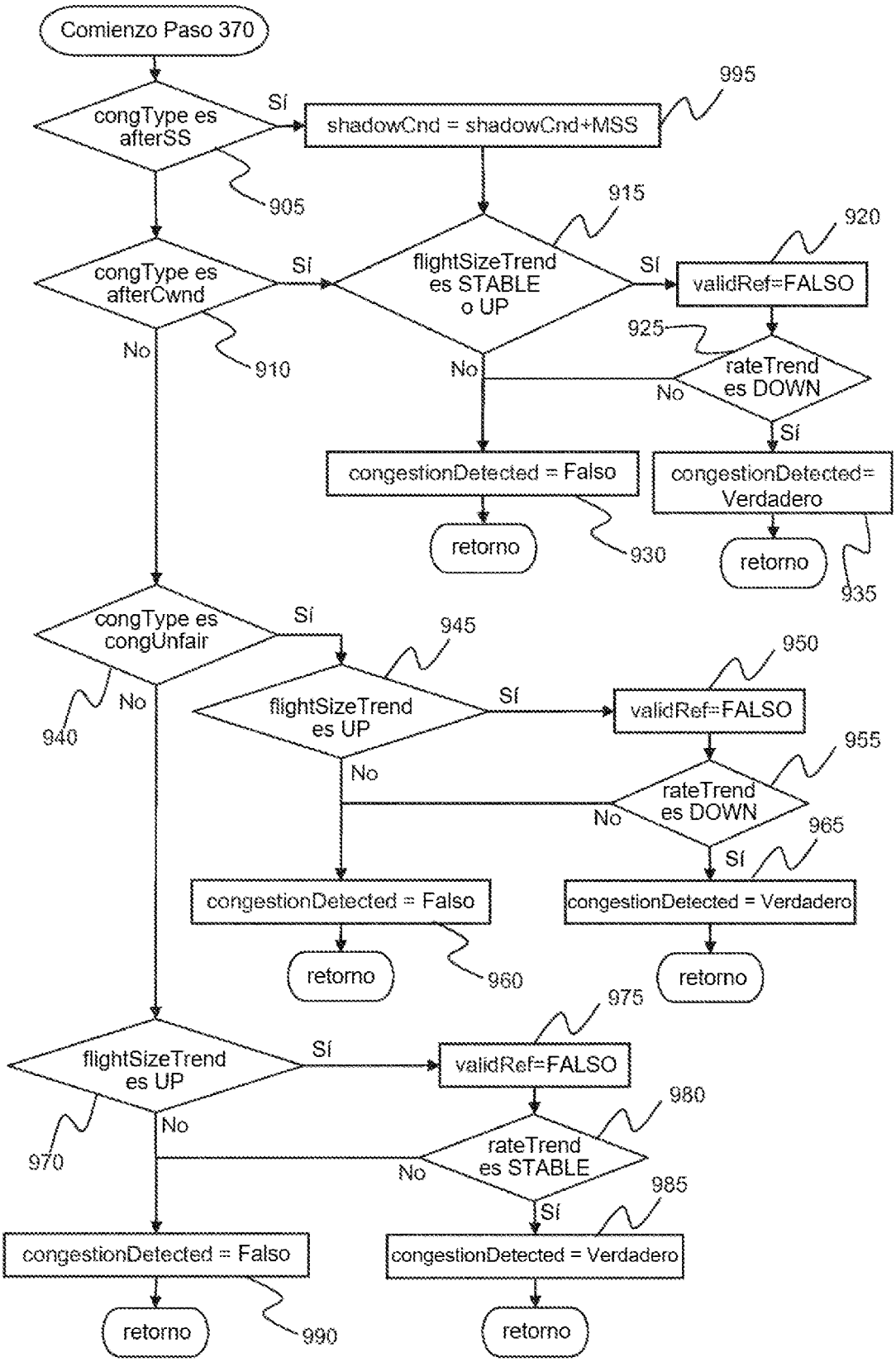
Comienzo Paso 370
congType es afterSS
Sí
shadowCnd = shadowCnd+MSS
995
905
congType es afterCwnd
Sí
flightSizeTrend es STABLE o UP
915
Sí
validRef=FALSO
920
925
No
910
No
rateTrend es DOWN
No
Sí
congestionDetected = Falso
congestionDetected= Verdadero
retorno
930
retorno
935
congType es congUnfair
Sí
945
950
940
No
flightSizeTrend es UP
Sí
validRef=FALSO
No
955
rateTrend es DOWN
No
965
Sí
congestionDetected = Falso
congestionDetected = Verdadero
retorno
960
retorno
975
flightSizeTrend es UP
Sí
validRef=FALSO
980
970
No
rateTrend es STABLE
No
985
Sí
congestionDetected = Falso
congestionDetected = Verdadero
retorno
990
retorno

Fig. 9

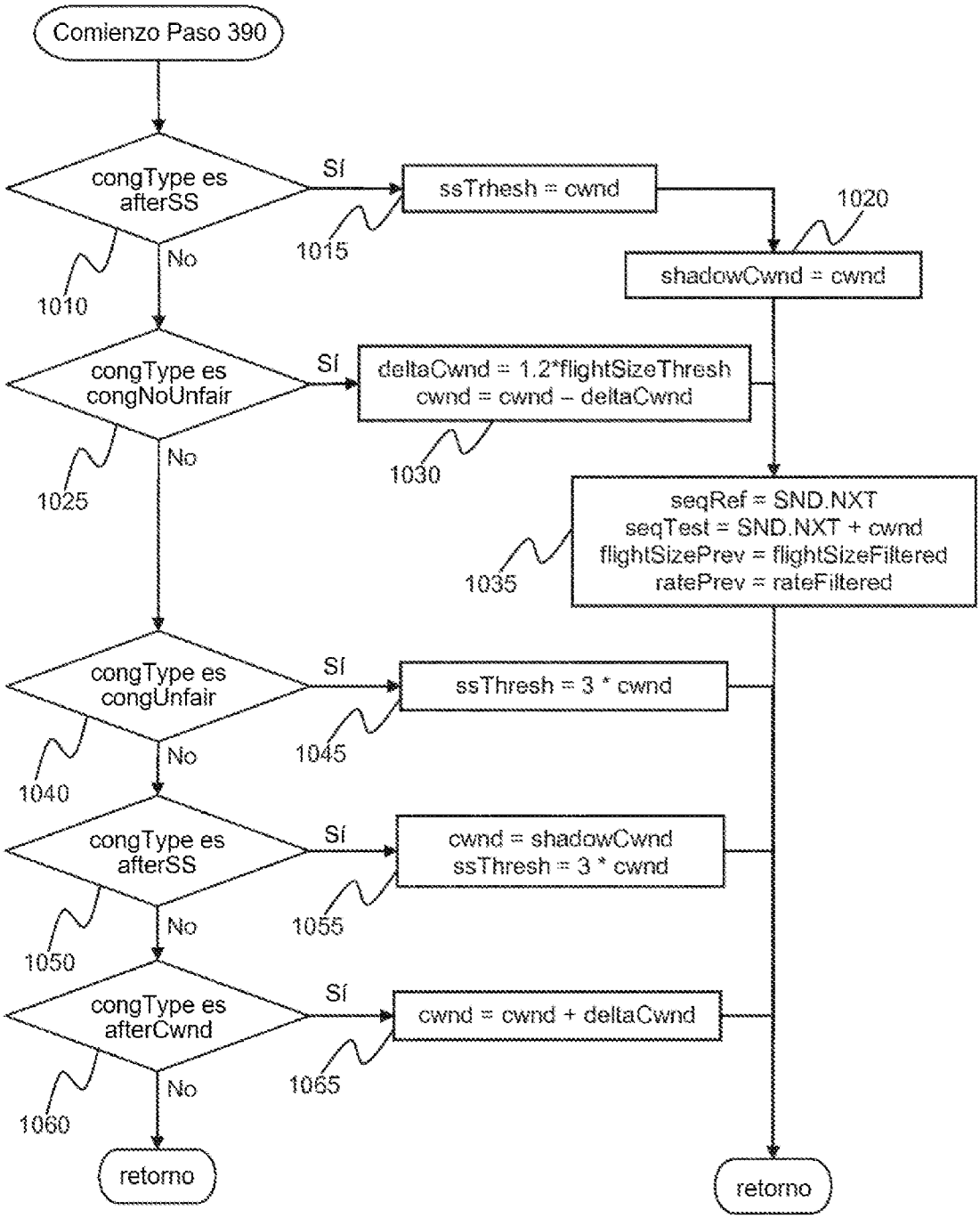
Comienzo Paso 390
congType es afterSS
Sí
ssTrhesh = cwnd
1015
1020
shadowCwnd = cwnd
No
1010
congType es congNoUnfair
Sí
deltaCwnd = 1.2*flightSizeThresh
cwnd = cwnd – deltaCwnd
1030
No
1025
seqRef = SND.NXT
seqTest = SND.NXT + cwnd
flightSizePrev = flightSizeFiltered
ratePrev = rateFiltered
1035
congType es congUnfair
Sí
ssThresh = 3 * cwnd
1045
No
1040
congType es afterSS
Sí
cwnd = shadowCwnd
ssThresh = 3 * cwnd
1055
No
1050
congType es afterCwnd
Sí
cwnd = cwnd + deltaCwnd
1065
No
1060
retorno
retorno

Fig. 10

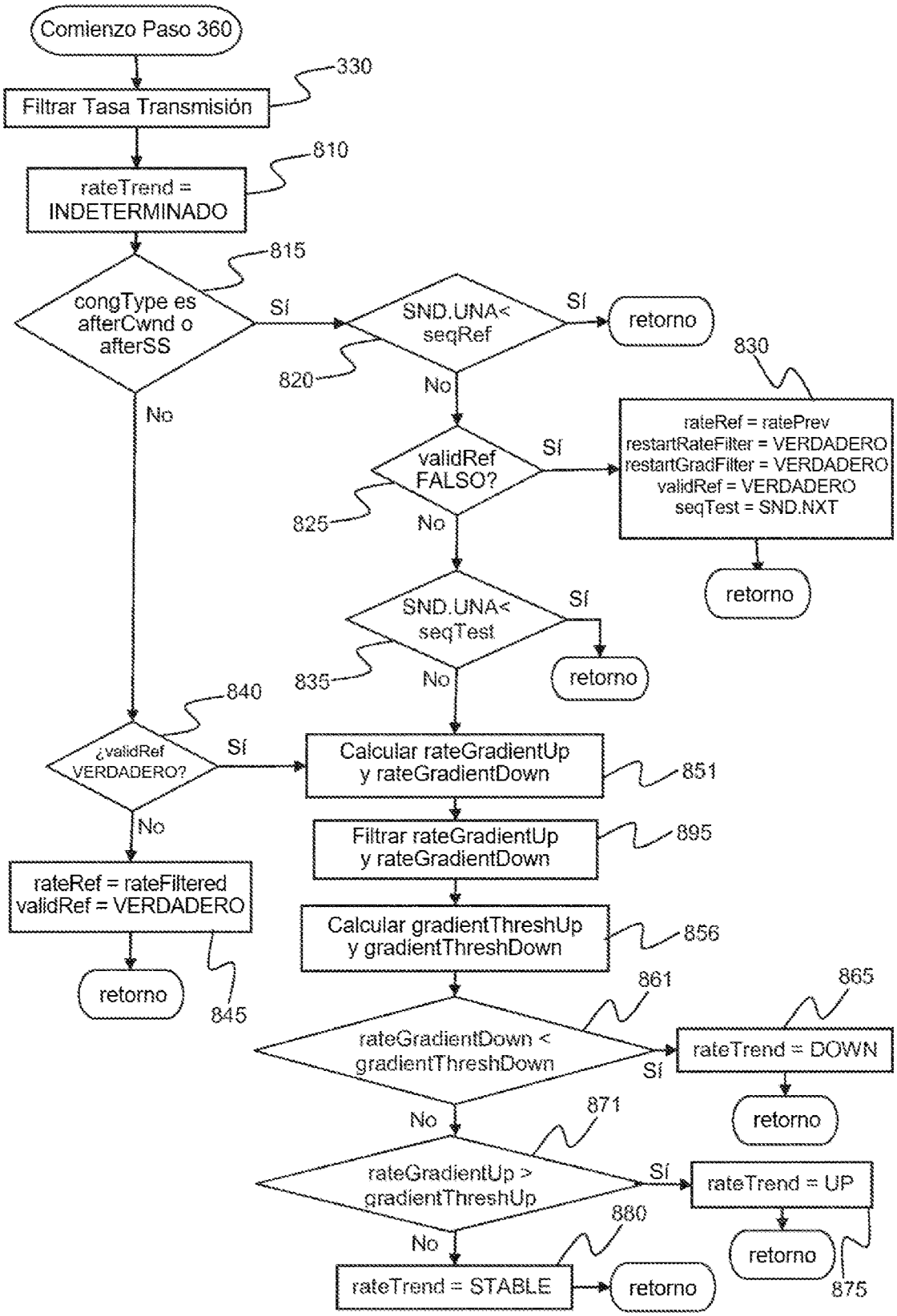
Comienzo Paso 360
330
Filtrar Tasa Transmisión
810
rateTrend = INDETERMINADO
815
congType es afterCwnd o afterSS
Sí
No
820
SND.UNA< seqRef
Sí
retorno
No
830
rateRef = ratePrev
restartRateFilter = VERDADERO
restartGradFilter = VERDADERO
validRef = VERDADERO
seqTest = SND.NXT
retorno
validRef FALSO?
Sí
825
No
SND.UNA< seqTest
Sí
retorno
835
No
840
¿validRef VERDADERO?
Sí
Calcular rateGradientUp y rateGradientDown
851
No
Filtrar rateGradientUp y rateGradientDown
895
rateRef = rateFiltered
validRef = VERDADERO
Calcular gradientThreshUp y gradientThreshDown
856
retorno
845
861
865
rateGradientDown < gradientThreshDown
Sí
rateTrend = DOWN
retorno
No
871
rateGradientUp > gradientThreshUp
Sí
rateTrend = UP
No
880
retorno
875
rateTrend = STABLE
retorno

Fig. 11

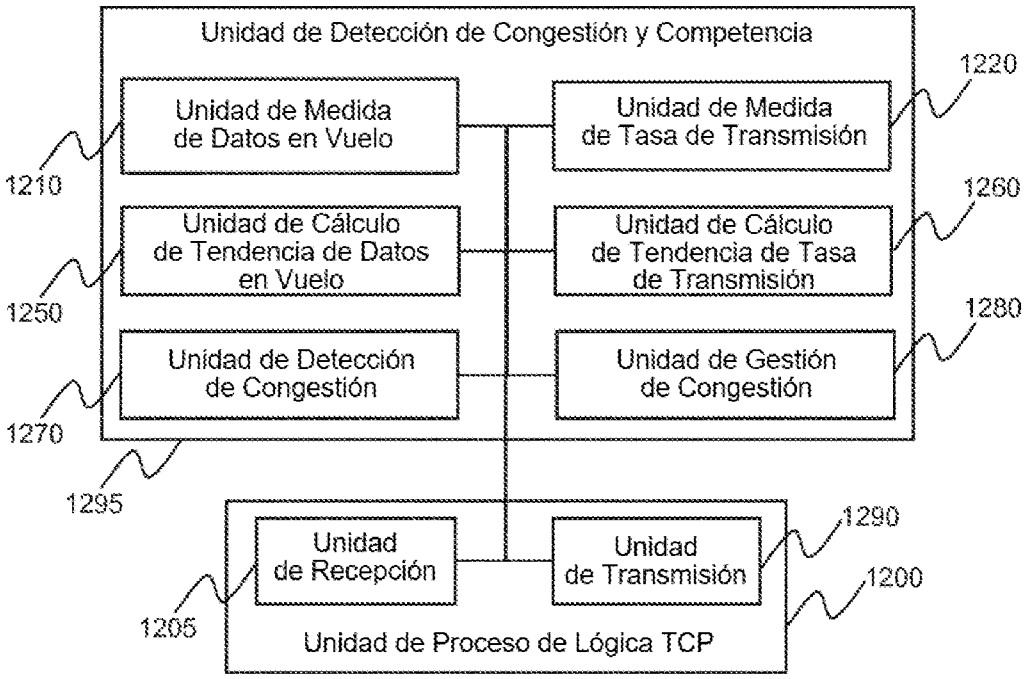
Unidad de Detección de Congestión y Competencia
Unidad de Medida de Datos en Vuelo
Unidad de Medida de Tasa de Transmisión
Unidad de Cálculo de Tendencia de Datos en Vuelo
Unidad de Cálculo de Tendencia de Tasa de Transmisión
Unidad de Detección de Congestión
Unidad de Gestión de Congestión
Unidad de Recepción
Unidad de Transmisión
Unidad de Proceso de Lógica TCP
1210
1220
1250
1260
1270
1280
1295
1290
1200
1205

Fig. 12

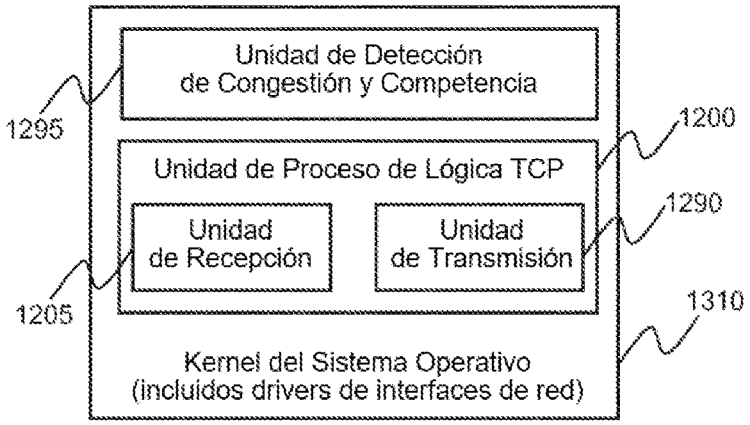
Unidad de Detección de Congestión y Competencia
Unidad de Proceso de Lógica TCP
Unidad de Recepción
Unidad de Transmisión
Kernel del Sistema Operativo (incluidos drivers de interfaces de red)
1295
1200
1290
1205
1310

Fig. 13

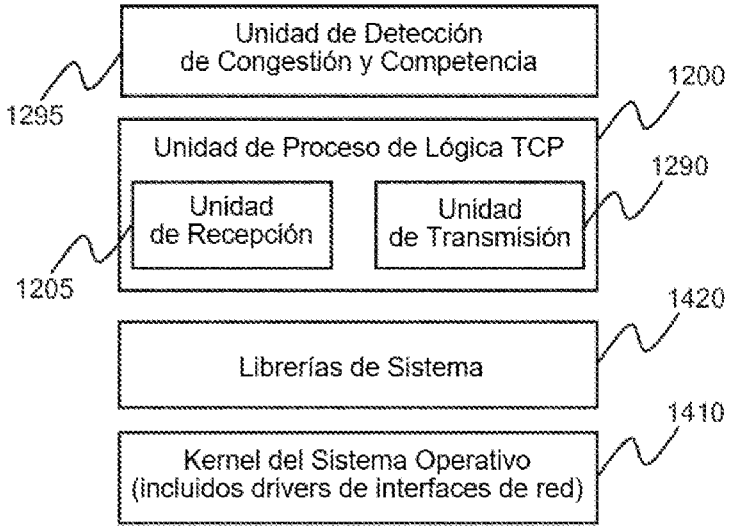
Unidad de Detección
de Congestión y Competencia
1295
1200
Unidad de Proceso de Lógica TCP
1290
Unidad
de Recepción
Unidad
de Transmisión
1205
1420
Librerías de Sistema
1410
Kernel del Sistema Operativo
(incluidos drivers de interfaces de red)

Fig. 14

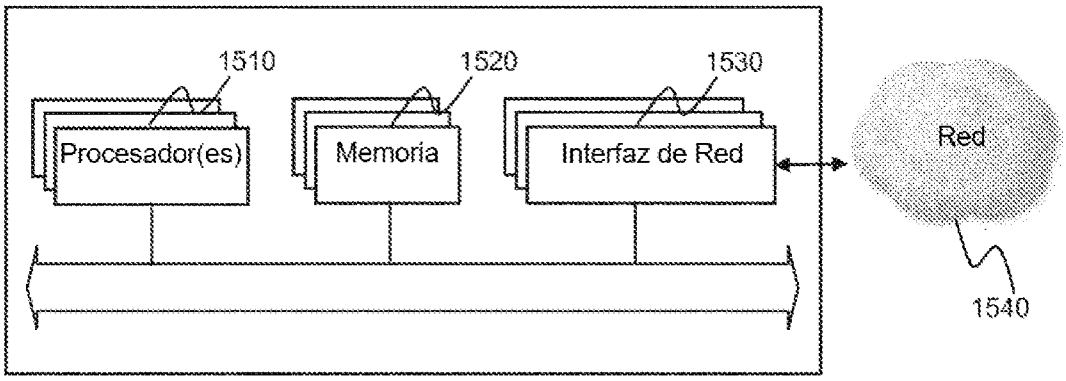
1510
1520
1530
Procesador(es)
Memoria
Interfaz de Red
Red
1540

Fig. 15