ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

Evolución de los algoritmos de control de congestión en las distintas variantes del protocolo TCP

Evolution of congestion control algorithms in the different variants of the TCP protocol

Vanesa G. Cadin & Carlos A. Talay
vanesagcadin@hotmail.com, ctalay@uarg.unpa.edu.ar
Instituto de Tecnología Aplicada (ITA – UARG)
Unidad Académica Río Gallegos, Universidad Nacional de la Patagonia Austral
Avenida Gregores y Piloto Lero Rivera s/n – Río Gallegos – Santa Cruz – Argentina.

Recibido: 20/04/2021. Aceptado: 07/07/2021

RESUMEN

En este documento nos centraremos en analizar cómo fueron evolucionando los métodos de control de congestión de la capa de transporte, desarrollados a lo largo del tiempo en las distintas implementaciones que dieron origen a las variantes del protocolo TCP. Se expondrá las 3 líneas fundamentales en que se basa el abordaje del problema de la congestión de datos, ejemplificando con las principales variantes que fueron implementadas en cada una de las variantes de TCP. También se desarrolla brevemente los protocolos de la capa de transporte más actuales, centrándonos en cómo cada uno de ellos utiliza uno de los 3 enfoques mencionados. Por último, dentro de estas variantes, se verá que línea ha sido la que mejor adaptabilidad ha mostrado para el funcionamiento sobre redes de topología híbrida.

Palabras clave: TCP; Algoritmo; Control de congestion.

ABSTRACT

In this document we will focus on analyzing how the transport layer congestion control methods developed over time in the different implementations that gave rise to the TCP protocol variants evolved. The 3 fundamental lines on which the approach to the problem of data congestion is based will be presented, exemplifying the main variants that were implemented in each of the TCP variants. The most current transport layer protocols are also briefly developed, focusing on how each of them uses one of the 3 approaches mention ed above. Finally, within these variants, it will be shown which line has shown the best adaptability for operation over hybrid topology networks.

Keywords: TCP; Algorithm; Congestion control.

1. INTRODUCCIÓN

El protocolo de comunicaciones TCP (*Transmission Control Protocol*) es uno de los protocolos más usados en la capa de transporte. Sus orígenes se remontan a principios de la década del 70, cuando Vint Cerf y Robert Kahn lo proponen como un protocolo de la capa de transporte para acceso al medio de forma fiable y de uso bidireccional. Este protocolo se

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

caracteriza por ser confiable, realizar un control de flujo y poseer un mecanismo de control de congestión de datos.

Si bien en las primeras etapas de las comunicaciones las velocidades de transmisión eran bajas y las redes cableadas, circunstancias que contribuían a que la tasa de pérdida de paquetes fuera baja. A medida que las redes evolucionaron y se transformaran en redes de mayor velocidad de transmisión de datos y topologías híbridas, la pérdida de paquetes conjuntamente con la congestión de datos, se transforma en un problema de compleja solución, lo que dio origen a distintas variantes del protocolo TCP. Estas variantes abordaron la solución al problema bajo tres estrategias, que son: reactivo (basado en pérdidas de paquetes), proactivo (basado en retrasos en la llegada de paquetes) e híbrido (basado en pérdidas de paquetes con estimación de ancho de banda).

A pesar de que el control de congestión no formó parte de la formulación inicial de TCP, constituye un aspecto esencial del protocolo. Un algoritmo de control de congestión acertado mejora notablemente el flujo de datos y por consiguiente throughput (cantidad de tráfico circulante en la red en un momento determinado) asociado a cada conexión TCP.

El documento está estructurado de la siguiente forma: la Sección 2 hablaremos de las principales características del protocolo TCP, congestión y control de congestión; la Sección 3, se describe algunas variantes del protocolo TCP con distintas implementaciones y los respectivos algoritmos utilizados para realizar el control de congestión bajo tres estrategias; la Sección 4, se explica brevemente los protocolos basados en los enfoques actuales para el control de congestión en la capa de transporte; finalmente se presenta algunas conclusiones finales.

2. EL PROTOCOLO TCP (TRANSMISSION CONTROL PROTOCOL)

La capa de Transporte es la responsable del envío del mensaje desde el emisor al receptor, para ello utiliza los servicios de la capa de Red que entrega los datagramas desde el host origen al host destino. Esta capa de Red utiliza los servicios de la capa de Enlace de Datos que entrega tramas entre dos nodos vecinos. TCP es el cuarto nivel de la capa OSI y el tercero en el modelo TCP/IP. Es el protocolo más utilizado en la capa de transporte, y el principal para las aplicaciones de Internet, como el correo electrónico, la transferencia de archivos, la administración remota, navegación web, etc.

En la figura 1 podemos ver, gráficamente, dónde está ubicada la capa de transporte en el modelo ISO OSI y en el STACK TCP/IP.

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

Figura 1: Capa de transporte en el modelo OSI y en el Stack TCP/IP

La primera idea de intercomunicación computadoras se remontan a 1947, en el transcurso del tiempo fueron producto de las investigaciones llevadas a cabo por el informático estadounidense Vinton Cerf como parte de un proyecto dirigido por el ingeniero norteamericano Robert Kahn, entre 1973 y 1974 y, patrocinado por la Agencia de Programas Avanzados de Investigación (*ARPA*) del Departamento Estadounidense de Defensa. Actualmente, Vinton Cerf y Robert Kahn, son considerados los padres de Internet. La especificación de TCP como estándar se publicó en Septiembre de 1981 por Internet Engineering Task Force (*IETF*) y se definió en RFC793 (Postel, 1981), donde ha sido modificado y corregido en el curso de los años para adaptarlo a cada topología de red en el RFC1122. Una de ellas, descrita en la RFC5681, es el algoritmo de control de congestión.

El comienzo del diseño de TCP fue el principio de End-To-End. Establece que los temas relativos a los protocolos de transporte cuando se ponen en ejecución en los ordenadores principales son responsabilidad de ambos extremos de la conexión y, por consiguiente, no deben ser delegados a la red. Dentro de estas responsabilidades se encuentran la de controlar, evitar corrupción de datos y la congestión. TCP es la única capa que se ocupa de la gestión de la congestión lo que no significa que el backbone de Internet no debe preocuparse por esto.
TCP se ha convertido en el estándar de facto de los protocolos de transporte y es empleado en la actualidad por la mayor parte de aplicaciones (Stevens, 1994). El protocolo TCP es un conjunto de algoritmos que envían paquetes a la red sin ningún tipo de reserva previa, pero que son capaces de reaccionar ante determinados eventos en la misma.

La popularidad de TCP se debe en gran parte a las siguientes características:

- Es orientado a la conexión, es decir, por lo que antes de transmitir datos, se asegura de haberse conectado con el otro extremo.
- Es robusto y fiable, debido a que no asume que los protocolos de las capas subyacentes sean confiables y toma la responsabilidad de asegurar una transmisión exitosa.
- Utiliza el estado de la red para modificar la tasa de transmisión de paquetes con el objetivo de utilizar el uso de recursos.
- Implementa mecanismos de control para garantizar que la información llegue de forma completa, ordenada y correcta al receptor.

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

El servicio orientado a conexión, o sea, basa su entrega confiable en un procedimiento conocido como ARQ (*Automatic Repeat reQuest*), en sus distintas variantes, garantiza la integridad de los datos. Mediante el procedimiento mencionado y con la utilización de acuses de recibos conocidos como ACKs (*acknowledgments*) positivos, se logra que, con menos de un ACK (*acknowledge*), por paquete de datos, se pueda confirmar la recepción de información de todo un conjunto de paquetes. Esta técnica se conoce como delayed-ACK (Clark, 1982) y permite lograr un importante aumento de eficiencia en el funcionamiento de la red.

Dado que el protocolo TCP utiliza los servicios del protocolo IP y este proporciona un servicio no orientado a la conexión (de mayor esfuerzo) para la entrega de datagramas a través de Internet, es TCP el que debe proporcionar fiabilidad, mediante un mecanismo de control de flujo entre los hosts extremos en una conexión a través de la implementación de un control de flujo basado en ventana deslizante. Con tal fin, dispone de una serie de mecanismos para actuar en caso de que se pierdan los segmentos. El tamaño de la ventana de transmisión determina cuántos datos pueden estar en tránsito, es decir, cuántos datos pueden haber sido enviados por el transmisor sin que aún se haya recibido su asentimiento. Cuando se tienen en tránsito tantos datos como indica la ventana de transmisión, el emisor deja de enviar nuevos segmentos. Cada vez que se recibe un asentimiento, y si el valor de la ventana de transmisión lo permite, TCP transmite nuevos datos. El valor de la ventana de transmisión se determina por el mínimo de dos límites, uno que está impuesto por el receptor e indica el tamaño del buffer de recepción disponible, para evitar saturar el receptor, y otro impuesto por el propio emisor, denominado ventana de congestión, conocido como Congestion Window (*CWND*), que tiene el objetivo de evitar el envío de más datos que los que la red soporta (Afanasyev et al., 2010).

La evolución del control de la congestión de TCP comienza a mediados de los años '80. Hasta ese momento, el control de flujo de transmisión basado en ventanas deslizantes había funcionado bastante bien, pero con la popularización de Internet la congestión pasó a ser un problema. El control de la congestión trata de determinar que el transmisor no envíe más datos que los que la red es capaz de manejar. Por lo tanto, la congestión ocurre cuando las demandas de tráfico exceden la capacidad de red disponible (incluyendo todos los enlaces y enrutadores desde la fuente al destino). Los algoritmos para control de congestión son un factor clave que juega un papel fundamental en el nivel de rendimiento y el comportamiento de la cantidad de flujo de datos dentro de las redes. Se definen como el conjunto de técnicas, metodologías y procesos que buscan determinar dinámicamente el ancho de banda y la latencia de la red. A partir de los datos, modifican la tasa de envío de paquetes en forma dinámica y su rendimiento de acuerdo a los cambios de tráfico para evitar el colapso de la subred.

2.1. Control de congestión

La congestión es un problema omnipresente en todas las redes de paquetes de datos en general y en Internet (basada en protocolo IP) en particular, no proporciona un servicio de información extremo a extremo sobre el estado de congestión de la red, es la capa de transporte junto al protocolo TCP quien implementa este mecanismo de control de la congestión extremo a extremo entre dos procesos que se están ejecutando en equipos distintos
Se considera a la congestión como uno de los problemas principales que afronta la transmisión de datos, saturando los recursos de la red y degradándose su utilización. Por este motivo, uno de los aspectos más importantes, desde el punto de vista del rendimiento o eficacia y confiabilidad del protocolo de TCP, es el control de la congestión.

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

2.2. Congestión

La congestión de red es el fenómeno producido cuando a la red, o parte de ella, se le inyecta más tráfico del que puede procesar. De manera sencilla, la congestión se da cuando la demanda es mayor a los recursos disponibles. Cuando se habla de recursos se refiere a ancho de banda, tiempos de respuesta, tamaño de los buffers, nivel de procesamiento de los procesadores de los elementos de red, etc.

Hay que destacar que la congestión tiene dos efectos: el primero, cuando la congestión empieza a producirse, el tiempo de transmisión a través de la red aumenta. El segundo, conforme la congestión se hace más severa, los nodos de la red descartan paquetes (Kaur y Singh, 2017).
Si se produce una congestión, las transferencias de paquetes se retrasan y descartan, debido a esto, algunos protocolos o aplicaciones intentan retransmitir datos. Los usuarios realizan la misma acción o solicitan los mismos datos una y otra vez. En este caso, la proporción de datos válidos está disminuyendo y, al final, se produce un colapso de la congestión y es difícil utilizar los recursos de la red. Por lo tanto, debemos controlar esta congestión para mejorar la calidad del servicio de la red.

Pero aun así el control de la congestión es difícil de realizar (Turkovic et al., 2019), debido a las siguientes razones:

- Internet está diseñado para ser autónomo.
- Internet es muy grande y todavía se está expandiendo.
- Sin control centralizado.
- No hay forma de controlar el comportamiento de cada usuario.
- Es difícil determinar cuántos usuarios/aplicaciones comparten la red exactamente.
- Es difícil determinar exactamente el origen de la congestión.
- Es difícil determinar con exactitud la capacidad de las redes.
- Es difícil determinar exactamente cuántas redes están congestionadas.
- Es difícil determinar exactamente por qué se pierden los paquetes.

2.3. Control de congestión en TCP

La técnica de control de congestión basada en ventanas utilizada por TCP, intentará regular la tasa de envío de datos ajustando el tamaño de la ventana para evitar la congestión de la red y, al mismo tiempo, proporcionar una parte justa del ancho de banda de la red a todas las conexiones.

TCP define un parámetro interno llamado CWND, en base al cual se implementa el control de congestión. Dependiendo del valor que tome CWND, el host fuente podrá estimar el número de paquetes, de los cuales no se ha recibido un acuse de recibo, y que pueden tener en tránsito en un momento determinado sin que se produzca congestión. TCP supone que hay congestión cuando se pierde un paquete. TCP utiliza dos mecanismos llamados, incremento aditivo (*additive increase*) y decremento multiplicativo (*multiplicative decrease*). El primero, incremento aditivo, aumenta la CWND cuando el nivel de congestión disminuye y, decremento multiplicativo reduce la CWND cuando el nivel de congestión aumenta. TCP interpreta los valores del tiempo de espera, conocidos como timeouts, como signo de congestión. Cada vez que ocurre un timeout, el host origen pone su CWND a la mitad del valor que tenía previamente. Esta división se corresponde con la parte de “decremento multiplicativo” de este mecanismo. La CWND no puede caer debajo del tamaño de un único segmento TCP, dado por el parámetro de TCP tamaño máximo de segmento, más conocido por su nombre en inglés Maximum Segment Size o MSS. Cada vez que un host origen envía

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

con éxito todos los paquetes que entran en la CWND, se aumenta la CWND en el tamaño equivalente a un paquete. Esto sería el mecanismo de incremento aditivo.

En pocas palabras, el control de congestión en TCP cuando detecta una pérdida de paquetes no puede enviar paquetes nuevos sin recuperar los paquetes perdidos, esto se logra manipulando dinámicamente el tamaño de la ventana.

Se han implementado varias técnicas de control de congestión en TCP para limitar la velocidad de envío de datos que ingresan a Internet mediante la regulación del tamaño de la CWND. Las técnicas que utiliza, son los algoritmos Slow Start, Congestion Avoidance, Fast Retransmit y Fast Recovery. Las primeras dos fases del mecanismo de control de la congestión, Slow Start y Congestion Avoidance, regulan la cantidad de paquetes inyectados en la conexión y son responsables de la detección de la congestión, en pocas palabras, controlan las transmisiones. Mientras que los otros dos algoritmos, Fast Retransmit y Fast Recovery, reaccionan para superar la congestión y proporcionar un mecanismo que acelera la recuperación de la conexión.

La figura 2 muestra el comportamiento típico del control de congestión en TCP.

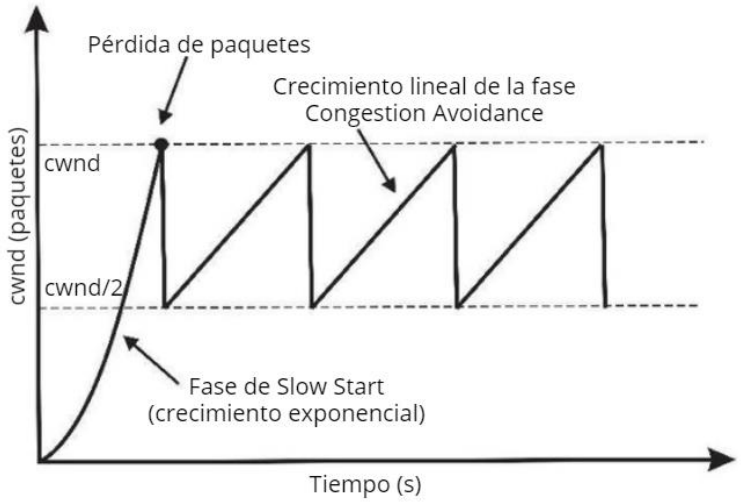

Figura 2: Dinámica del control de congestión de TCP

Ante estas características, se han desarrollado una variedad de implementaciones que tratan este problema de diferente forma, permitiendo que el protocolo TCP se adapte a todo tipo de redes. Algunos se han diferenciado en mínimas expresiones, mientras que han supuesto cambios drásticos en la gestión de la ventana.

3. DISTINTOS TIPOS DE ABORDAJES EN EL PROBLEMA DE CONTROL DE CONGESTIÓN EN EL PROTOCOLO TCP

A continuación, se analizarán los tres tipos de metodologías de control de congestión utilizados por el protocolo TCP. Para una referencia de cómo será el desarrollo de estos métodos, presentamos el siguiente gráfico:

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

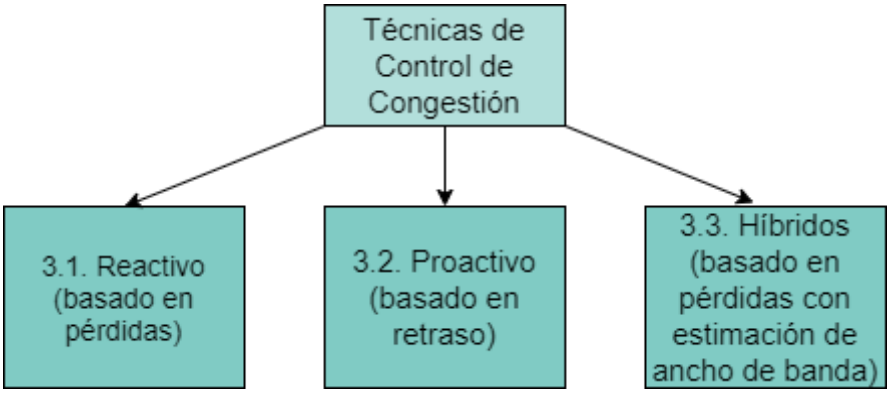

Figura 3: Metodologías de control de congestión que utiliza el protocolo TCP

3.1. Reactivo (basado en pérdidas)

Los protocolos reactivos no toman ninguna acción a menos que, y hasta que, el problema realmente ocurra. Intentan resolver el problema, pero no consideran por qué ocurre. Si entienden que hay congestión en la red accionan sus algoritmos de control de congestión. Sin embargo, existen situaciones en las cuales la pérdida de paquetes puede tener otro origen que no es la congestión y, por tanto, no deberían disparar sus mecanismos de control de congestión.

Se permite que la CWND crezca siempre que regresen los ACK, lo que indica la capacidad permitida en la red hasta el punto en que comienzan a aparecer ACK duplicados, y significa una pérdida de paquete debido a congestión o error de canal. En ese punto, se toman acciones correctivas principalmente reduciendo la CWND y el umbral de inicio lento en diferentes cantidades con la intención de permitir que la red salga del estado congestionado. Los protocolos reactivos con miras a maximizar el rendimiento siempre llevan la red a la capacidad máxima, después de lo cual todas las conexiones sufren el daño colateral causado por la sobreestimación de la capacidad del canal.

El principal problema de los protocolos reactivos es que la CWND de todos los flujos que participan en la red va aumentando hasta el punto en que la capacidad del búfer en los routers se desborda y todos los flujos experimentan la pérdida de paquetes originada por la congestión. En este momento, la CWND de todos los flujos se reduce a la mitad y mediante la retransmisión rápida, la red sale del estado de congestión. El punto es que, con el objetivo de maximizar la capacidad disponible, la red se lleva periódicamente a su capacidad máxima y luego todos los flujos sufren el daño colateral creado por algunos de los flujos demasiado ambiciosos.

Dentro de este grupo reactivo encontramos como protocolo actual a TCP Wave. De manera resumida, utiliza el mecanismo de control de congestión basada en ráfagas, como ventaja tiene equidad y eficiencia y como desventaja tiene un tiempo completo de ida y vuelta, conocidos como RTT (*Round Trip Time*), altamente volátil.

3.1.1. TCP Tahoe

Después de observar una serie de colapsos de congestión en 1980, TCP Tahoe es la primera variante de TCP en incorporar los mecanismos de control de congestión y representa la segunda generación de versiones de TCP, que incluye dos nuevas técnicas, prevención de congestión y transmisión rápida. Este algoritmo fue desarrollado por Jacobson y Karels en 1986. Basado en el mismo concepto presentado por Jacobson y Karels, luego se introducen

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

muchos más algoritmos. TCP Tahoe tiene, por tanto, los mecanismos Slow Start, Congestion Avoidance, Fast Retransmit. No obstante, el principal inconveniente que presenta es el del Slow Start, concretamente en enlaces de retardo elevado, provocando el bajo rendimiento del protocolo.

Se basa en un principio de conservación de paquetes, es decir, si la conexión se ejecuta de manera estable y a la capacidad de ancho de banda disponible, el flujo de paquetes es lo que un físico llamaría "conservador": un nuevo paquete no se coloca en la red hasta que salga un paquete antiguo. Por cada paquete enviado en la red por una fuente, se espera que se transmita un ACK desde el destino confirmando la entrega de paquetes transmitidos. La fuente controla la tasa de envío de paquetes usando la CWND, que determina la cantidad de paquetes que la fuente puede enviar. El destino también anuncia a la fuente la cantidad de datos que está dispuesto a almacenar en búfer para la conexión llamada ventana de recuperación (*rwnd*). Al usar estas dos variables, la fuente puede transmitir los datos hasta la cantidad máxima de CWND o rwnd. La transmisión de datos entre el origen y el destino depende de los valores mínimos comparativos de CWND o rwnd. Este principio es un elemento central, tanto de Slow Start como de Congestion Avoidance.

Sin embargo, hay ciertos problemas que deben resolverse para garantizar este equilibrio entre paquetes entrada - salida.

- La conexión no llega al equilibrio, o
- Un remitente inyecta un paquete nuevo antes de que salga un paquete antiguo, o
- No se puede alcanzar el equilibrio debido a los límites de recursos a lo largo del camino.

Cuando se establece una nueva conexión, TCP desconoce la capacidad disponible de la red y, por lo tanto, el tamaño óptimo de la CWND. Para obtener su valor ideal, la estrategia es comenzar a enviar datos a tasas cada vez más altas hasta que se produzcan pérdidas de paquetes. Tahoe sugiere que siempre que una conexión TCP se inicia o se reinicia después de la pérdida de un paquete, debe pasar por un procedimiento llamado Slow Start. El motivo de este procedimiento es que una ráfaga inicial puede abrumar la red y es posible que la conexión nunca se inicie. El algoritmo Slow Start sugiere que se incremente el tamaño de la CWND en el mismo número de segmentos con acuse de recibo, por cada ACK recibido con éxito. Al recibir un ACK, se puede enviar el doble de la cantidad de datos que fueron reconocidos por ese ACK (política de aumento multiplicativo). TCP empieza la comunicación probando la red, enviando un solo segmento, y fija el tamaño de la CWND a un segmento. Es decir, en el primer RTT la ventana aumenta el valor de CWND a dos, con lo que se envían dos segmentos, luego aumenta a cuatro, se envían cuatro segmentos y así sucesivamente utilizando múltiplos de dos. Por lo tanto, esto provoca que se duplique la tasa de envío por cada RTT y que aumente exponencialmente la CWND hasta que perdemos un paquete que es un signo de congestión, como podemos observar en la Figura 4. A partir de esto, disminuimos nuestra tasa de envío y reducimos la CWND a uno para garantizar la liberación de recursos de red.

La ventana del receptor es lo suficientemente grande y el crecimiento de forma exponencial de CWND provocado por el algoritmo Slow Start, tiene como consecuencia el crecimiento de la CWND a tal punto que comenzará a descartar paquetes.

Cuando la CWND alcanza el valor de la variable umbral denominado ssthresh (*umbral de Slow Start*), TCP abandona la fase de Slow Start y entra en una fase Congestion Avoidance

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

para encontrar la capacidad disponible de la red y no saturarla. En la Figura 5 se observa la evolución de la ventana de transmisión, que, como se ha mencionado, es el mínimo entre la CWND y el tamaño del buffer de recepción. Por lo tanto, el valor de la ventana de transmisión en cada momento da una indicación de la tasa de transmisión de TCP, ya que, en ausencia de errores se va a transmitir durante un RTT como máximo tantos segmentos como indique la ventana de transmisión. Se abandona el estado de Congestion Avoidance cuando el emisor TCP detecta un evento de congestión por expiración de timeout o por la recepción consecutiva de tres ACK duplicados. En el primer caso, se reduce el valor de CWND a 1 y se define el valor de ssthresh a la mitad del valor alcanzado por CWND o al menos dos segmentos. En caso contrario, se reconfigura la variable ssthresh al valor CWND/2, divide a la mitad el valor de CWND y le suma 3 de los ACK duplicados recibidos luego, establece el valor de ssthresh a la mitad del valor de CWND y se invoca a la estrategia Fast Retransmit.

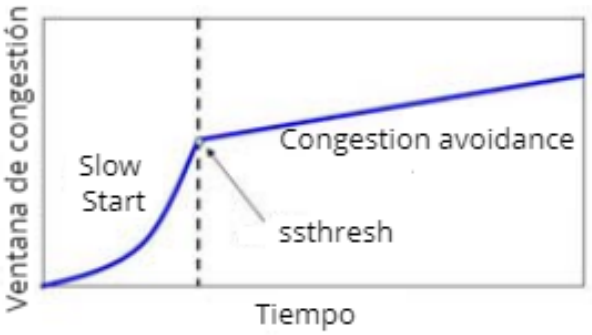

Figura 4: Evolución de la ventana de congestión en TCP.

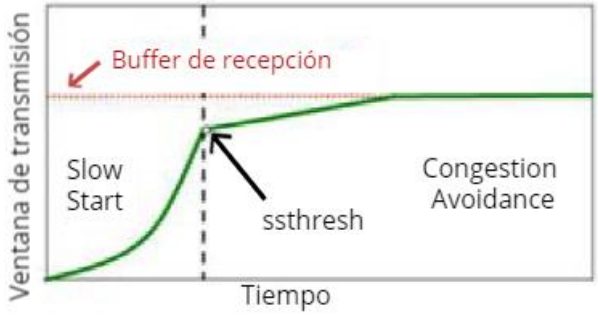

Figura 5: Evolución de la ventana de transmisión en TCP

La fase de Fast Retransmit permite que un emisor conozca que se ha perdido un segmento incluso antes de que venza el temporizador de retransmisión. Cuando se recibe un segmento fuera de orden, el receptor genera lo que se denomina un ACK duplicado, es decir, vuelve a asentir los mismos datos que ya asintió anteriormente. Lo que hace es retransmite el segmento, donde no se requiere tiempo de espera para que expire el temporizador de retransmisión, solo asume en que en cuanto llegan al emisor tres asentimientos duplicados (cuatro ACKs idénticos) son un buen indicador de un segmento perdido, como se puede observar en la Figura 6. Luego, el emisor establece ssthresh en la mitad de la CWND actual y ajusta el valor de la CWND igual a uno, reiniciando la fase Slow Start.

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

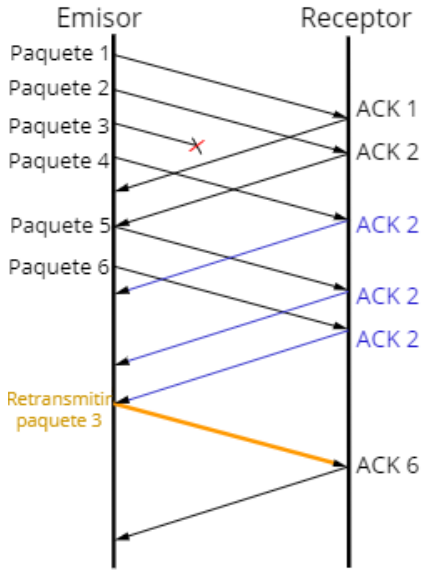

Figura 6: Cómo se detecta congestión en Fast Retransmit

3.1.2. TCP Reno

TCP Reno es la tercera versión de la primera serie desarrollada. En 1981 está disponible la primera versión de TCP/IP (4.2BSD). En 1986 se presenta la versión 4.3BSD, que mejora el rendimiento y en 1988 tenemos la variante 4.3BSD-Tahoe. Posteriormente en 1990 se desarrolla la versión a la cual nos referiremos: 4.3BSD-Reno (Moraru et al., 2003). En comparación con el TCP Tahoe, cuando se detecta congestión, este reduce la CWND a un segmento lo que dispara el algoritmo de Slow Start, mientras que TCP Reno dispara el algoritmo de Fast Recovery, reenviando el paquete perdido y reduce el umbral a la mitad, evitando disparar el algoritmo de Slow Start utilizando reconocimientos adicionales para temporizar posteriores paquetes.

Cuando en Fast Recovery llega el tercer ACK duplicado actualiza el valor de ssthresh a la mitad de flightsize (número de segmentos en tránsito, es decir que han sido enviados pero aún no han sido asentidos) asumimos que el segmento se perdió y lo retransmitimos inmediatamente, sin aguardar el tiempo de espera, y se iguala la CWND a ssthresh más 3 veces el tamaño máximo de segmento (el valor de este tamaño es un parámetro configurable). Cada vez que llegue un nuevo ACK duplicado, se incrementa la CWND en el valor del tamaño máximo de segmento. En caso de que el número de segmentos en tránsito sea menor que el valor de la ventana de transmisión, se transmite un nuevo segmento. Cuando llegue el ACK que asienta otro segmento, o sea, un ACK no duplicado, se actualiza CWND con el valor de ssthresh. A continuación, se prosigue en la fase Congestion Avoidance donde la reducción de la CWND al valor justo después de entrar en la recuperación es una manera simple y confiable para asegurar el estado de salida de Fast Recovery.

En la figura 7 podemos observar la variación del tamaño de la CWND para TCP Tahoe vs. TCP Reno.

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

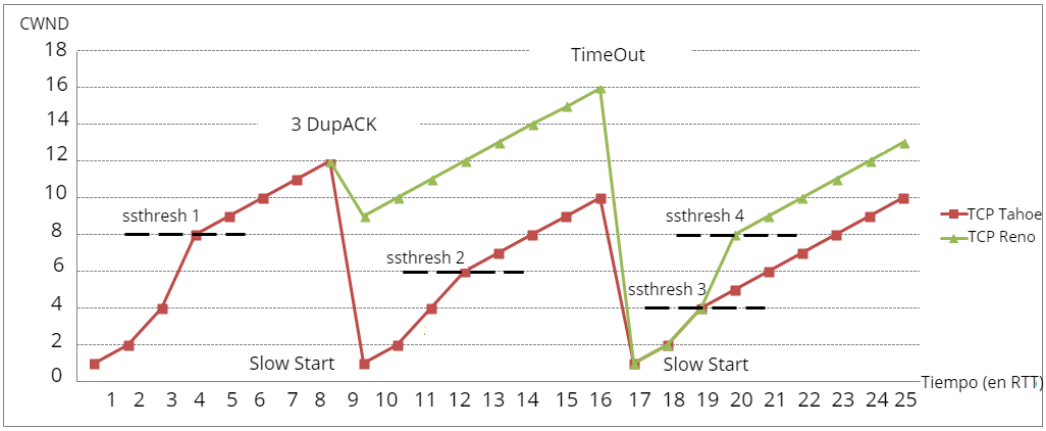

Figura 7: CWND TCP Tahoe vs. TCP Reno

En la figura anterior podemos ver que cuando la detección de la pérdida es por la recepción de tres ACK duplicados (3 DupACK), TCP Tahoe disminuye radicalmente la tasa de envío y el ancho de banda se reduce a la mitad, mientras que la reducción de la tasa de TCP Reno es significativamente menor. Después de ver la pérdida de un paquete, el TCP Tahoe aumenta el número de segmentos linealmente en lugar de exponencialmente. En el caso que la pérdida se detecte por la expiración de un timeout, la respuesta de TCP Reno es idéntica a la de TCP Tahoe.

Los mecanismos Fast Retransmit y Fast Recovery son eficaces cuando se pierde un único segmento. Sin embargo, cuando se pierden varios segmentos consecutivos, el algoritmo no se recupera con facilidad ya que son difíciles de detectar y su rendimiento es casi el mismo que el de TCP Tahoe en condiciones de alta pérdida de paquetes. Esto se debe a que después de la recuperación de la pérdida de un segmento, CWND generalmente queda reducida a la mitad del valor que tenía, con lo que en caso de que el asentimiento que llegue sea parcial, es decir, que no asienta a todos los segmentos que se habían enviado antes de entrar en Fast Recovery, la ventana es tan pequeña que no se pueden enviar nuevos segmentos, por lo que se ha de esperar al vencimiento del temporizador para continuar con la retransmisión. Existe una versión de TCP que modifica el comportamiento de Fast Recovery, denominada NewReno (Henderson et al., 2012).

3.1.3. TCP New Reno

TCP New Reno es una ligera modificación, definida en el RFC3782 (Floyd et al., 2004) sobre TCP Reno, más específicamente en el algoritmo Fast Recovery mejorando su respuesta en caso de múltiples pérdidas. Al igual que TCP Reno, New Reno también entra en retransmisión rápida cuando recibe múltiples paquetes duplicados, sin embargo, se diferencia de TCP Reno en que no sale de la Fast Recovery hasta que se produzca la confirmación (*ACK*) de todos los segmentos de la CWND. Más formalmente, TCP New Reno utiliza una variable de estado adicional para registrar el número de secuencia más alto del último segmento TCP enviado antes de entrar a la fase Fast Recovery. Esta última procede como en TCP Reno, sin embargo, cuando se recibe un ACK nuevo, hay dos casos:

- La recepción de un nuevo ACK de datos significa que todos los paquetes enviados antes de la detección de errores fueron entregados con éxito y cualquier nueva pérdida reflejaría un nuevo evento de congestión, entonces sale de la fase de Fast Recovery y establece el valor umbral de CWND y continúa evitando la congestión como TCP Tahoe.

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

- Si el ACK es un ACK parcial, confirma la recuperación de sólo el primer error e indica más pérdidas en la ventana original de paquetes. Sale de la recuperación rápida cuando se reconocen todos los datos de la ventana.

TCP New Reno resuelve el problema de bajo desempeño ante eventos de múltiples pérdidas en la CWND, pero como contrapartida emplea un tiempo excesivo en el proceso de recuperación, pues le toma un RTT recuperar cada uno de los segmentos perdidos en la CWND. Por lo tanto, el ancho de banda disponible no se utiliza de manera efectiva.
La versión TCP New Reno se propone una modificación al algoritmo de Recuperación Rápida de forma que en caso de que existan varias pérdidas por ventana se soluciona el problema de TCP Reno.

3.1.4. Opción TCP SACK y TCP FACK

TCP SACK (*Selective Acknowledgement*) es una opción extendida de TCP New Reno. Soluciona los problemas que enfrentan TCP RENO y TCP New-Reno, a saber, detección de múltiples paquetes perdidos y retransmisión de más de un paquete perdido por RTT. Partiendo de la mejora que supone el uso de SACK, surgió TCP FACK, que hace referencia a Forward Acknowledgement, que funciona alrededor de los problemas de detección de múltiples paquetes perdidos y retransmisión por RTT.

SACK conserva las propiedades de TCP Tahoe y TCP Reno de ser robusto en presencia de paquetes fuera de orden y utiliza tiempos de espera de retransmisión como método de recuperación de último recurso. Requiere que los segmentos no se reconozcan de forma acumulativa, sino que deberían reconocerse de forma selectiva. Por lo tanto, cada ACK tiene un bloque que describe qué segmentos están siendo reconocidos. Por consiguiente, el remitente tiene una imagen de qué segmentos han sido reconocidos y cuáles aún están pendientes. Siempre que el remitente ingresa a la fase de Slow Start, inicializa una tubería variable que es una estimación de la cantidad de datos pendientes en la red, y también establece CWND a la mitad del tamaño actual. Cada vez que recibe un ACK reduce la tubería en 1 y cada vez que retransmite un segmento lo incrementa en 1. Siempre que la tubería es más pequeña que la CWND, comprueba qué segmentos no se han recibido y los envía. Si no hay ningún segmento pendiente, envía un nuevo paquete (Lar et al., 2011). De modo que se puede enviar más de un segmento perdido en un RTT.

Desafortunadamente, el mecanismo SACK tiene serias limitaciones en su forma actual. La especificación TCP restringe la longitud del campo de opciones a 40 bytes. Un cálculo simple revela que la opción SACK puede contener como máximo cuatro bloques de paquetes de datos recibidos en orden. Este escenario empeora si queremos utilizar opciones de TCP adicionales, que reducen el espacio para los pares de números de secuencia que se incluyen en SACK. En el peor de los casos, cuando se pierden todos los demás paquetes, este límite se excede justo después de que se reciben los primeros 4 paquetes (por lo tanto, se pierden 4 paquetes). No obstante, es poco probable que ocurra esta situación del peor de los casos en las redes cableadas, ya que durante los eventos de congestión generalmente se caen paquetes consecutivos, las pérdidas aleatorias en las redes inalámbricas pueden mostrar patrones que se aproximan al peor de los casos. Esta observación muestra que SACK no es una solución universal al problema de pérdidas múltiples.

Aunque SACK proporciona al receptor capacidades de notificación ampliadas, no define ningún algoritmo de control de congestión en particular. FACK (Mathis y Mahdavi, 1996) define procedimientos de recuperación utiliza información adicional disponible en SACK

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

para manejar la recuperación de errores (control de flujo) y el número de paquetes pendientes (control de velocidad) en dos mecanismos separados. El primero, control de flujo, utiliza ACK selectivos para indicar pérdida y proporciona un medio para la retransmisión oportuna de paquetes de datos perdidos. Debido a que los paquetes de datos retransmitidos se notifican como perdidos para al menos un RTT y una pérdida no puede recuperarse instantáneamente, el remitente FACK debe retener, como mínimo, la hora de la última retransmisión. Por último, control de la velocidad, a diferencia de TCP Reno y TCP New Reno intentan estimarlo asumiendo que cada ACK duplicado recibido representa un segmento que ha abandonado la red. En lugar de la técnica de inflación de la CWND, el FACK mantiene tres variables de estado especiales:

H: el número de secuencia más alto de todos los paquetes de datos enviados.
F: el número de secuencia más hacia adelante de todos los paquetes de datos reconocidos.
R: el número de paquetes retransmitidos.

TCP FACK realiza el sencillo cálculo de (H−F+R) para saber cuántos paquetes están pendientes de enviar. Esta relación, en términos de robustez de las pérdidas de ACK, proporciona una estimación fiable de los paquetes de datos pendientes en la red. El destino puede utilizar esta expresión para decidir si envía o no una nueva porción de datos. Más formalmente, los datos se pueden enviar cuando el cálculo el número de paquetes de datos pendientes está por debajo de un límite permitido por CWND.

Las ventajas de FACK han sido ampliamente reconocidas desde hace mucho tiempo y FACK ha sido una parte integrada del Kernel de Linux desde la versión 2.1.92. Debido a que FACK modifica solo las reacciones en la fase de recuperación, las características de estado estacionario de efectividad y equidad son exactamente las mismas que para TCP Reno.

3.2. Proactivo (basado en retraso)

El protocolo proactivo busca desarrollar una estrategia que permita evitar que el tráfico llegue a una situación de congestión. La evolución de la CWND y el rendimiento de una conexión TCP proactiva se modelan como funciones del ancho de banda, el retraso, el tamaño del paquete, el número de flujos TCP simultáneos y el factor de penalización. Este protocolo intenta anticipar la sobreestimación de la capacidad de la red y comenzar a tomar acciones correctivas para evitar la incipiente fusión congestiva de la red. Deben emplear un enfoque muy cooperativo al tratar con la utilización de la capacidad y la red debe funcionar por debajo de su capacidad máxima. Hay que tener en cuenta que debe dejarse algo de espacio para los flujos incontrolables que en cualquier momento pueden obstruir parte de la capacidad de la red. El enfoque del protocolo debe estar más orientado a lograr una maximización general del rendimiento para todas las conexiones que prevalecen en la red en lugar de simplemente maximizar el flujo de una conexión.

El emisor intenta ajustar la CWND de manera proactiva a una tasa de envío óptima, de acuerdo con la información recopilada a través de la retroalimentación, que puede indicar indirectamente la condición de la red. De esta forma, el emisor reacciona de manera inteligente al estado de la red o a las pérdidas de paquetes. Se pueden emplear diferentes estrategias en el diseño para que el emisor pueda deducir la condición de la red.

Los esquemas proactivos, en general, exhiben una manera más efectiva para manejar las pérdidas aleatorias, dada la forma de ajuste del tamaño de CWND. Sin embargo, estos esquemas pueden llevar a problemas relacionados con la equidad con otros tráficos en la red.

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

Dentro del grupo proactivo encontramos, como protocolo actual, a TCP Nimbus y TCP LEDBAT (*Low Extra Delay Background Transport*). De manera resumida, TCP Nimbus utiliza en el mecanismo de control de congestión cola explícita y tráfico cruzado modelado y como ventaja tiene agresividad escalable para lidiar con CUBIC. Por otro lado, TCP LEDBAT utiliza en el mecanismo de control de congestión una demora adicional en los flujos de alta prioridad y como ventaja no afecta a otros flujos.

3.2.1. TCP Dual

TCP Tahoe ha prestado un gran servicio a la comunidad de Internet al resolver el problema del colapso de la congestión. Sin embargo, esta solución tiene el inconveniente desagradable de forzar la red con fases periódicas de gran amplitud. Este comportamiento induce cambios periódicos significativos en la velocidad de envío, el tiempo de ida y vuelta y la utilización del búfer de la red, lo que genera una variabilidad en las pérdidas de paquetes.

Wang y Crowcroft (1992) presentaron TCP DUAL en 1992, que refina el algoritmo Congestion Avoidance. Intenta mitigar los patrones oscilatorios en la dinámica de la red mediante el uso de un mecanismo de detección de congestión proactivo junto con reacciones más suaves a los eventos detectados. Más específicamente, introduce el retraso en la cola como parámetro de predicción del estado de congestión de la red.

Supongamos que las rutas no cambian durante la transmisión y que el receptor reconoce cada paquete de datos inmediatamente. Entonces podemos considerar el valor mínimo de RTT observado por el remitente (RTT_{min}) como una buena indicación de que el camino se encuentra en un estado libre de congestión. Si hacemos una suposición más de que un aumento del RTT solo puede ocurrir debido al aumento de la utilización del búfer, la diferencia entre el valor RTT medido y el valor mínimo de RTT (retraso en la cola Q = RTT - RTT_{min}) se puede ver como un indicador del nivel de congestión en el camino.

Para cuantificar el nivel de congestión, TCP Dual además mantiene un valor RTT máximo observado durante la transmisión (RTT_{max}.). La diferencia entre máximo y mínimo de RTT se considera una medida del nivel máximo de congestión (es decir, el máximo retraso en la cola $Q_{max} = RTT_{max} - RTT_{min}$). Finalmente, una fracción del retardo máximo en la cola ($Q_{thresh} = \alpha \cdot Q_{max}$, $dónde\ 0 < \alpha < 1$) sirve como un umbral, que, cuando se supera, indica la congestión estado de la red.

En la propuesta de Wang y Crowcroft (1992), el umbral de retraso en TCP Dual se selecciona como la mitad del retraso máximo de cola ($Q_{thresh} = Q_{max}/2$), y la estimación de la congestión se realiza una vez por período RTT en función del valor RTT promedio ($Q = RTT_{avg} - RTT_{min}$). Si se supera este umbral ($Q > Q_{thresh}$), la CWND disminuye en $1/8^{th}$ (es decir, política de reducción multiplicativa aplicada). Como podemos ver en la dinámica teórica de la CWND de TCP DUAL (Figura 8), la efectividad mejora mucho en comparación con Tahoe (es decir, gráficamente, el área sombreada es proporcionalmente mayor). Sin embargo, hay una serie de compensaciones.

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

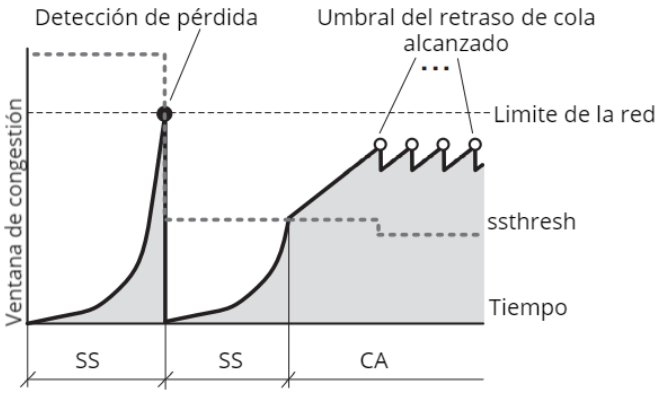

Figura 8: Dinámica de CWND de TCP DUAL
(SS: fase de Slow Start, CA: fase de Congestion Avoidance)

Si el punto de saturación de la red se estima incorrectamente, el flujo no puede utilizar los recursos de red disponibles de manera justa y eficaz. Por un lado, en el caso de que el umbral está subestimado (p. ej., observado RTT_{max} no es el máximo real) los recursos de la red estarán infrautilizados. Por otro lado, la sobreestimación del umbral puede causar una distribución injusta de recursos entre diferentes flujos TCP DUAL. Por ejemplo, si un flujo Dual ya está transmitiendo datos cuando aparece un nuevo flujo Dual, el nuevo flujo observará un valor RTT_{min} más alto y sobrestimará el umbral de retraso de la cola. El flujo con el umbral de cola inferior (el antiguo flujo) tiene una mayor probabilidad de predecir el estado de congestión y desencadenar la reducción de la CWND, mientras que el otro flujo continuará con el crecimiento de la CWND sin notar nada anormal. Por tanto, el nuevo flujo puede capturar potencialmente una mayor parte de los recursos de la red.

3.2.2. TCP Vegas

TCP Vegas propone mecanismo para evitar la congestión que difiere del utilizado por TCP Reno. Mientras que TCP Reno requiere tener la pérdida de paquetes para censar el ancho de banda disponible, TCP Vegas tiene como objetivo detectar la congestión en sus etapas iniciales y reducir su tasa de envío de datos (*throughput*) para prevenir que haya pérdida de paquetes. TCP Vegas se basa en el hecho de que las medidas proactivas para enfrentar la congestión son mucho más eficaces que las reactivas. La idea es controlar y mantener el tamaño adecuado de la ventana de manera de no llegar a la perdida de paquetes para señalar que hay congestión. Realiza un seguimiento de cuándo se envió cada segmento y también administra el tamaño de la ventana observando el RTT de los paquetes que el emisor envió con anterioridad. Si el RTT crece, TCP Vegas reconoce que la red comienza a estar congestionada y reduce su CWND. Por el contario, si los valores de RTT decrecen, el emisor determina que la red no está congestionada e incrementa el tamaño. Para evitar las oscilaciones, se define un intervalo en donde se mantiene el tamaño de la ventana.

Se diferencia de los otros algoritmos durante su fase de Slow Start. El motivo de esta modificación es que cuando se inicia una conexión por primera vez no se tiene idea del ancho de banda disponible y es posible que durante el aumento exponencial se dispare el ancho de banda en una gran cantidad y así induzca congestión. Con este fin, TCP Vegas aumenta exponencialmente solo cada dos RTT, entre eso, calcula el envío real a lo esperado y cuando la diferencia supera un cierto umbral, sale de la fase de Slow Start y entra en la fase de Congestion Avoidance (Chowdhury y Alam, 2019).

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

3.3. Híbridos (basado en pérdidas con estimación de ancho de banda)

La tercera línea desarrollada para realizar el control de congestión se basa en una solución híbrida, en ella se plantea la filosofía del control de la congestión analizando las pérdidas de paquetes con una estimación simultánea del ancho de banda.

Esta variante nace principalmente por el desarrollo de redes de tipo inalámbricas y eventualmente redes híbridas. Es por ello que estas soluciones están basadas sobre la base de evitar la degradación del rendimiento debido a las características del enlace inalámbrico. Son soluciones que mantienen la idea de host-to-host y al mismo tiempo proporcionan cierto nivel de resistencia a las pérdidas de paquetes no relacionadas con la congestión (Kim, 2016).

Para lograr este objetivo, este algoritmo de desarrollo de control de congestión utiliza la métrica de retraso para fines específicos que no están directamente relacionados como señal de congestión. Estas técnicas utilizan la señal de retardo para calcular un tamaño óptimo de la CWND después del evento de pérdida de paquetes, pero también para diferenciar entre pérdidas de paquetes relacionadas con la congestión y las que no se deben a la congestión. La técnica de estimación de ancho de banda propuesta sentó las bases para que el emisor pudiera deducir entre una pérdida relacionada con la congestión y una pérdida no relacionada (aleatoria) sin ningún soporte de la red. Se identificaron algunas debilidades en este enfoque, tales como, la sobreestimación del ancho de banda, lo que dio lugar a la evolución y los refinamientos de las variantes del protocolo TCP que trabajan con esta idea, y que intentan mitigar los problemas descubiertos. Dentro de esta línea tenemos como principal desarrollo el protocolo TCP Westwood, al que siguieron con algoritmos más refinados las variantes Westwood+, ABSE (*Adaptive Bandwidth Share Estimation*), CRB (*Combined Rate and Bandwidth estimation*), BBE (*Bottleneck and Buffer Estimation*), BR (*Bulk Repeat*) y A (*Agile – Probing*).

3.3.1. TCP Westwood

TCP Westwood (Mascolo et al., 2001) mantiene el concepto extremo a extremo de TCP e independiente de la red y, al mismo tiempo, puede mejorar significativamente la eficiencia de la transferencia de datos en redes propensas a errores. Es un nuevo protocolo TCP con una modificación del lado del emisor del esquema de control de congestión de ventanas, parte del algoritmo de TCP New Reno, para que estime el ancho de banda disponible de la red de manera dinámica, midiendo y promediando la tasa de retorno de ACK recibidos. La idea innovadora clave es estimar continuamente, en el remitente TCP, la tasa de paquetes de la conexión mediante el monitoreo de la tasa de recepción ACK. La tasa de conexión estimada se utiliza para calcular la CWND y la configuración del umbral de Slow Start después de un episodio de congestión, es decir, después de un tiempo de espera o 3 confirmaciones duplicadas. Restablecer la ventana para que coincida con el ancho de banda disponible hace que TCP Westwood sea más resistente a pérdidas esporádicas debido a problemas de canal inalámbrico. De hecho, el rendimiento no es muy sensible a los errores aleatorios, mientras que TCP Reno es igualmente sensible a la pérdida aleatoria y la pérdida de congestión y no puede discriminar entre ellos. De ahí la tendencia de TCP Reno a reaccionar de forma exagerada a los errores, lo que lleva a una reducción de ventana innecesaria ya que "ciegamente" la reduce a la mitad después de tres ACK duplicados. En particular, TCP Westwood introduce un mecanismo de recuperación más rápida para evitar una reducción demasiado conservadora de la CWND después de un episodio de congestión al tener en cuenta la estimación de extremo a extremo del ancho de banda disponible. La ventaja del mecanismo propuesto es que el remitente TCP se recupera más rápido después de pérdidas, especialmente en conexiones con tiempos de ida y vuelta grandes, o en enlaces inalámbricos

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

donde las pérdidas esporádicas se deben a enlaces poco fiables en lugar de a congestión. Las modificaciones propuestas siguen el principio de diseño de un extremo a otro del TCP. Solo requieren ligeras modificaciones en el lado del remitente y son compatibles con versiones anteriores (Pawale et al., 2020).

4. ENFOQUES ACTUALES PARA EL CONTROL DE CONGESTIÓN EN LA CAPA DE TRANSPORTE

Actualmente, la investigación y desarrollo de los algoritmos de control de congestión se centran en tres enfoques principales:

- Apuntar a la baja latencia y la utilización de ancho de banda completo.
- Espacio de usuario, que usan UDP.
- La adopción de soluciones de múltiples rutas (*Multipath*) en la capa de transporte.

Dentro de estas ideas podemos encuadrar el desarrollo de los protocolos más destacados para los enfoques mencionados.

4.1. Bottleneck Bandwidth and Round -Trip (BBR)

TCP BBR desarrollado por Google y se puede clasificar cómo basado en latencia con estimación de ancho de banda. Se pretende diferenciar entre la pérdida de paquetes y la aparición de congestión en la red. La pérdida de paquetes puede ocurrir antes de que se sature la red y no haya suficiente capacidad, sin embargo, la congestión se detecta cuando la red está ya desbordada, produciendo grandes retardos.

El objetivo principal del algoritmo BBR es asegurar que el cuello de botella no se congestione, alcanzando así el máximo rendimiento con un retardo mínimo. Por lo tanto, BBR hace una estimación de la cantidad de datos que puede haber en la red BDP[1] a partir de la capacidad disponible (cuello de botella) y el retardo mínimo. El control de la transmisión de información se hace a partir de la estimación del ancho de banda disponible en cada instante. No se utiliza una CWND o el recuento de ACKs para saber la disponibilidad de la red, pero existe un límite en la cantidad de datos que pueden estar fluctuando de un extremo a extremo: 2 – BDP. A diferencia de los protocolos tradicionales que se basan en la pérdida de paquetes como señal de congestión, BBR estima la capacidad del canal disponible para determinar la cantidad de datos que enviar. Propone no tratar a la perdida de paquetes como sinónimo de congestión, sino que apunta a la baja latencia y la completa utilización de ancho de banda.

4.1.1. TCP Low Latency (LoLa)

TCP LoLa es un nuevo control de congestión basado en retardos que admite tanto un retardo de cola bajo como una alta utilización de la red en redes de área amplia de alta velocidad. Esto es particularmente útil para mezclas de tráfico que consisten en demanda de ancho de banda y retraso flujos sensibles (por ejemplo, transferencia de archivos larga y “web 2.0” interactiva tráfico). TCP LoLa mantiene el retraso de la cola en el enlace del cuello de botella bajo alrededor de un valor de umbral objetivo fijo. Este valor objetivo es independiente del número de flujos que comparten el cuello de botella. TCP LoLa logra una alta utilización de enlaces y logra convergencia a la equidad incluso entre flujos con diferentes tiempos de ida y vuelta, debido a su novedoso mecanismo llamado “equilibrio de flujo justo” (Hock et al., 2017).

[1] El bandwidth delay product es la cantidad máxima de información que está en la red en un tiempo dado. Se modifica este valor para que la capacidad de la red no sea infinita.

ICT-UNPA-279-2021
ISSN: 1852-4516
Aprobado por Resolución N° 0333/21-R-UNPA
http://doi.org/10.22305/ict-unpa.v13.n1.793

Tiene sistemas de control frente a pérdidas y controles de flujo en una misma conexión y congestión. Estos mecanismos permiten llevar a cabo conexiones estables. Surge con la finalidad de reducir la latencia en las comunicaciones HTTP a través de Internet. Disminuye el número de paquetes que tienen que intercambiarse dos equipos para iniciar una conexión, y por tanto, permite que las conexiones se establezcan más rápido.

QUIC tiene como principal objetivo reducir los tiempos de ida y vuelta. Implementa, como mecanismo de control de la congestión, CUBIC, siendo posible combinar otros, como TCP New Reno y BBR (Kakhki et al., 2019).

4.2. Multipath TCP (MPTCP)

MPTCP es una extensión recientemente estandarizada del protocolo TCP permitiendo que una conexión TCP envíe datos a través de cualquier número de interfaces sin dejar de brindar el mismo servicio a la aplicación de manera que se mantenga la equidad con otros flujos competidores, promete un uso más eficiente de los recursos de red y resistencia ante fallas de red. Logra esto mediante la combinación de varias conexiones TCP, denominadas subflujos en la arquitectura de referencia.

En los centros de datos, los servidores multi-homed, la multiplicidad de rutas de red y el ancho de banda agregado muy alto se están convirtiendo en la norma y los protocolos de transporte que podrían maximizar el uso de los recursos de la red y minimizar el tiempo de finalización de los flujos, al mismo tiempo que son justos con los flujos de TCP existentes, están muy investigados (Nguyen et al., 2019).

5. CONCLUSIONES

TCP es el principal protocolo de la capa de transporte y el que se encarga de transportar la mayor parte del tráfico de Internet, por lo que el rendimiento de Internet depende de cómo funciona TCP. Desde que empezó a ser más fácil el acceso a Internet y utilizado para un uso general, la cantidad de usuarios y de tráfico comenzaron a crecer de manera exponencial. Las características de rendimiento de una versión particular de TCP se definen, en gran medida, por el algoritmo de control de congestión que implementa y constituyen una forma efectiva de reducir sobrecargas temporales en la red. El problema que intenta resolver el control de la congestión, es el uso inteligente de los recursos disponibles en las redes (anchos de banda, capacidades de almacenamiento, etc.) y/o tráfico excesivo.

En este documento hemos presentado una recopilación de los principales enfoques que se adoptaron para la implementación de los algoritmos de control de congestión, para protocolos de la capa de transporte, fundamentalmente las variantes de TCP.

Como se observa, los desarrollos se dieron como consecuencia de las tres líneas conceptuales que se plantearon para evitar la congestión de datos. Así mismo, dentro de estas líneas, se plantearon soluciones en respuesta a los requerimientos que fueron apareciendo y los desafíos planteados por las tecnologías emergentes. Un caso particular fue la aparición de las redes inalámbricas, que dieron origen a todo el desarrollo de dispositivos móviles, pero que a su vez plantearon nuevos y complejos desafíos.

Hoy en día, la variedad de escenarios no nos permite asegurar que una de estas tecnologías se imponga por sobre el resto, de hecho, aún siguen apareciendo propuestas que mejoran los

actuales protocolos. Inclusive tampoco podemos asegurar que no surjan nuevos paradigmas de comunicaciones que planteen nuevos desafíos, por lo que podemos afirmar con cierta seguridad, que la evolución de los algoritmos de control de congestión es tema que está abierto y seguramente depararán nuevas propuestas que intenten dar respuesta a los nuevos requerimientos.

Actualmente no existe ningún enfoque de control de congestión para TCP que pueda aplicarse universalmente a todos los entornos de red. Una de las principales causas es la amplia variedad de entornos y por consiguiente los diferentes puntos de vista con respecto a qué métrica de evaluación de red considerar para obtener una mejor respuesta al fenómeno de la congestión. Si bien esto es un problema que conspira para lograr una propuesta de control de congestión que sea definitiva, deja un espacio para seguir proponiendo innovaciones y a exploración de futuras líneas de investigación.

6. REFERENCIAS

AFANASYEV, A., TILLEY, N., REIHER, P. and KLEINROCK, L. (2010, July) “Host-to-Host Congestion Control for TCP”, IEEE Communications Surveys & Tutorials, Volume 12(3), page 304-342. https://doi.org/10.1109/SURV.2010.042710.00114

CHOWDHURY, T., ALAM, M. J. (2019) “Performance Evaluation of TCP Vegas over TCP Reno and TCP NewReno over TCP Reno”. JOIV: International Journal on Informatics Visualization, vol. 3, no 3, p. 275-282. http://dx.doi.org/10.30630/joiv.3.3.270

CLARK, D. (1982, July) “RFC 813: Window and Acknowledgment Strategy in TCP”.

FLOYD, S., HENDERSON, T. and GURTOV, A., (2004) “RFC3782: The NewReno modification to TCP’s fast recovery algorithm”.

HENDERSON, T. and FLOYD, S. (2012, April) “RFC 6582: The NewReno Modification to TCP's Fast Recovery Algorithm”.

HOCK, M., Neumeister, F., Zitterbart, M. and Bless, R. (2017) “TCP LoLa: Congestion control for low latencies and high throughput”. En 2017 IEEE 42nd Conference on Local Computer Networks (LCN). IEEE, p. 215-218. https://doi.org/10.1109/LCN.2017.42

“IETF, Internet Engineering Task Force”, [En línea]. Available: http://www.ietf.org. [Último acceso: Marzo 2021].

KAKHKI, A. M., Jero, S., Choffnes, D., Nita-Rotaru, C. and Mislove, A. (2019, July) “Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols”. Communications of the ACM, Vol. 62 No. 7, Pages 86-94. https://doi.org/10.1145/3330336

KAUR, H. and SINGH, G. (2017) “TCP congestion control and its variants” Advances in Computational Sciences and Technology Volume 10 (6), p.1715-1723.

KIM, B.H. (2016) “Techniques for End-to-End Tcp Performance Enhancement Over Wireless Networks”, Publicly Accessible Penn Dissertations.

LAR, S., LIAO, X., GUO, S. (2011) “Modeling TCP NewReno slow start and congestion-avoidance using simulation approach”. International Journal of Computer Science and Network Security, vol. 11, no 1, p. 117-124.

MASCOLO, S., CASETTI, C., GERLA, M., SANADIDI, M.Y. and WANG, R. (2001) “TCP Westwood: Bandwidth estimation for enhanced transport over wireless links”, ACM MOBICOM, p. 287–297. https://doi.org/10.1145/381677.381704

MATHIS, M. y MAHDAVI, J. (1996) "Forward acknowledgement: refining TCP congestion control", en Proc. conferencia sobre aplicaciones, tecnologías, arquitecturas y protocolos para comunicaciones informáticas (SIGCOMM), Nueva York, NY, EE.UU., págs. 281–291.

MORARU, B., COPACIU, F., LAZAR, G. and DOBROTA, V. (2003) "Practical Analysis of TCP Implementations: Tahoe, Reno, New-Reno", pp. 125–138.

NGUYEN K, GOLAM KIBRIA M, ISHIZU K, KOJIMA F, SEKIYA H. (2019) "An Approach to Reinforce Multipath TCP with Path-Aware Information". Sensors, 19(3):476. https://doi.org/10.3390/s19030476

PAWALE, S. S., VANJALE, S. B., JOSHI, S. D. and PATIL, S. H. (2020, November) "Performance Improvement of TCP Westwood by Dynamically Adjusting Congestion Window in Wireless Network". Journal of University of Shanghai for Science and Technology. ISSN: 1007-6735, Volume 22, Issue 11, p. 166 – 177.

POSTEL, J. (1981) "Transmission Control Protocol", RFC793.

STEVENS, R. (1994) "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley.

TURKOVIC, B., KUIPERS, F. A., UHLIG, S. (2019) "Fifty shades of congestion control: A performance and interactions evaluation". arXiv preprint arXiv:1903.03852.

WANG, Z. and CROWCROFT, J. (1992) "Eliminating periodic packet losses in 4.3– Tahoe BSD TCP congestion control algorithm" ACM Computer Communication Review, vol. 22, no. 2, pp. 9–16. https://doi.org/10.1145/141800.141801