risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido/Submission: 25/08/2018
Aceitação/Acceptance: 18/11/2018

Mecanismo de control de congestión para transferencias de datos en Multicast múltiple

Raúl H. Palacios[1, 2], María Balseca[4], Víctor Gallo[3], Nilo Andrade[5], Juan-Carlos Pisco[3], Fabricio Marcillo[2,3]

raulhp@ugr.es, maria.balseca@itsjba.edu.ec, vgallo@uteq.edu.ec, nilo.andrade@uleam.edu.ec, jpisco@uteq.edu.ec, fmarcillo@uteq.edu.ec

[1] Universidad Autónoma del Estado de Hidalgo, Escuela Superior de Huejutla, CP. 43000, Huejutla, México.

[2] Centro de Investigación en Tecnologías de la Información y Comunicación, Granada, CP. 18014, Granada, España.

[3] Universidad Técnica Estatal de Quevedo, Facultad Ciencias de la Ingeniería, CP. 120501, Quevedo, Ecuador.

[4] Instituto Tecnológico Juan Bautista Aguirre, Daule, CP. 090601, Guayas, Ecuador.

[5] Universidad Laica Eloy Alfaro de Manabí, Chone, CP. 130301, Manabí, Ecuador.

DOI: 10.17013/risti.30.62–77

Resumen: En la actualidad es común compartir gran cantidad de información entre los sistemas y los mecanismos de control de congestión son indispensables para evitar la saturación en las redes de datos. Aunque se han propuesto diferentes mecanismos, las investigaciones actuales siguen siendo de interés en este campo de la ciencia. En este artículo, se propone un mecanismo de control de congestión basado en ventana (*window-based*) para transferencias de datos en escenarios de multicast múltiple. El algoritmo propuesto tiene como objetivo principal evitar la pérdida de paquetes manteniendo un equilibrio en la tasa de envío de los emisores. Además, el algoritmo constantemente analiza el estado de saturación del conmutador y de los nodos receptores considerando la tecnología de almacenamiento utilizada. Experimentos y análisis muestran que se consigue un ancho de banda agregado de red considerablemente alto, se evita en medida de lo posible la retransmisión de paquetes perdidos, aun cuando en el sistema de pruebas existe tráfico de red adicional.

Palabras-clave: control de congestión; UDP; múltiple multicast; multicast.

Congestion control mechanism for data transfers in multiple Multicast

Abstract: Currently, it is common to share a lot of information between systems and congestion control mechanisms are essential to avoid saturation in data networks. Although different mechanisms have been proposed, current research follows a focus on this area of science. In this paper, we propose a window-based congestion control mechanism for data transfers in multiple multicast scenarios. The main purpose of the proposed algorithm is to avoid packet loss maintaining a balance in the sending rate of senders. In addition, the algorithm continually analyzes

the state of the saturation of the switch and of the receiver nodes considering the storage technology used. Analysis and experiments show that a considerably high network bandwidth is achieved, retransmission of lost packets is avoided as much as possible, even though there is additional network traffic in the test system.

***Keywords**:* congestion control; UDP; multiple multicast; multicast.

1. Introducción

En el campo de la informática, las comunicaciones permiten el intercambio de datos entre computadores. En los inicios, la comunicación se realizaba entre una única fuente y un único destino. Con la evolución constante de las tecnologías de comunicación se ha alcanzado un progreso notable permitiendo el envío de datos desde una fuente a múltiples destinos simultáneamente, conocido como comunicación multicast. La utilización y aprovechamiento de estas comunicaciones siguen teniendo auge en la actualidad y continúan siendo un tema de interés para la investigación. La comunicación que se realiza en una red de computadores es esencial para los sistemas actuales que requieren transferencias entre estos, por citar un ejemplo se mencionan los sistemas de almacenamiento distribuido. Por otro lado, en (Wittmann & Zitterbart, 2000) se muestra un ejemplo de la contribución de las comunicaciones multicast como éxito importante.

Habilitar el tráfico multicast implica utilizar el protocolo UDP para el transporte de los datos. Esto tiene como desventaja que no existe fiabilidad en la entrega de los datos. A diferencia del protocolo TCP que es un protocolo orientado a conexión y que mantiene sus propios mecanismos (Ait-Hellal & Altman, 2000; Ha, Rhee, & Xu, 2008; Leith, Shorten, & McCullagh, 2008; Xu, Harfoush, & Rhee, 2004) que garantizan la entrega de los datos a todos los destinos. Aunque al utilizar UDP para permitir el tráfico multicast garantiza la entrega a diferentes destinos simultáneamente con menor consumo de recursos de cómputo, existen dos desventajas destacables: 1*) pérdida de paquetes*: es posible la pérdida de paquetes debido a la saturación en la red, además de que se trata de un protocolo no orientado a conexión; *2) congestión*: la falta de mecanismos para ajustar la tasa de envío puede dar lugar a la congestión de la red. Consecuentemente, para el desarrollo de aplicaciones basadas en multicast es necesario considerar aspectos de: fiabilidad, control de flujo, *control de congestión* y gestión de grupos.

El mecanismo de control de congestión permite prevenir la saturación en la red (específicamente en el conmutador) o en el búfer de los nodos receptores (típicamente debido a los discos lentos - HDD). En un entorno multicast es conveniente conocer información detallada de los receptores, tal como, capacidad actual de búfer de recepción, direcciones multicast desde donde se reciben datos, velocidad de escritura en disco (disco HDD o SSD), etc., con esto se facilita el flujo de datos desde varios emisores y utilizar esta información como medida preventiva para la congestión, de tal manera que sea posible garantizar la fiabilidad en la entrega de los datos a todos los receptores. Como ejemplo, en un sistema de almacenamiento distribuido el control de congestión juega un papel importante debido a que el tráfico generado por estos sistemas suele ser elevado y persistente en la red cuando se trata de envíos muy grandes como es el caso del sistema de ficheros Hadoop Distributed File System (HDFS) diseñado para almacenar conjuntos de datos muy grandes de manera confiable y transmitirlos a un ancho de

banda alto a las aplicaciones de los usuarios (Borthakur, 2008; Shvachko, Kuang, Radia, & Chansler, 2010). Al igual aplicaciones en entornos tales como (Díaz, Muñoz, 2018; Palos-Sánchez, Arenas-Márquez, Aguayo-Camacho, 2017) se podrían beneficiar.

El presente artículo se centra en la implementación de un mecanismo de control de congestión para comunicaciones multicast múltiple evaluado en un entorno real de red de ordenadores, es decir, se evalúa con tráfico real, a diferencia de las diversas propuestas existentes en la literatura de mecanismos de control de congestión que se desarrollan en entornos simulados.

En la siguiente sección se mencionan algunos trabajos que tienen relación con nuestra propuesta. Posteriormente en la sección 3., se describe en mayor detalle el algoritmo propuesto. Seguido de la sección 4 con la evaluación y análisis de la propuesta y, al finalizar se dan las conclusiones observadas en el presente artículo.

2. Trabajo relacionado

En la literatura revisada (Fall & Stevens, 2011; Wittmann & Zitterbart, 2000) se ilustra que los mecanismos de control de congestión son basados en ventana (*window-based*) y en tasa de envío (*rate-based*) y que son ampliamente utilizados por el protocolo TCP, tales como BIC (Xu et al., 2004), CUBIC, Reno, Vegas, por mencionar algunos.

Con el fin de dar a conocer un poco más el campo de aplicación de las comunicaciones multicast, en el presente trabajo se ha analizado una serie de protocolos de comunicación (mejor conocidos como Protocolo Multicast Fiable, RMP – Reliable Multicast Protocol), específicamente se muestra el mecanismo de control de congestión implementado, según (Fall & Stevens, 2011; Wittmann & Zitterbart, 2000), para cada uno de estos protocolos. Como se puede observar en la Tabla 1, los protocolos PGM (Gemmell, Montgomery, Speakman, & Crowcroft, 2003), RMTP (Paul, Sabnani, Lin, & Bhattacharyya, 1997) y TMTP (Yavatkar, Griffoen, & Sudan, 1995) utilizan implementaciones propias o no especifican sobre qué mecanismo han basado la implementación, mientras que RDCM (D. Li et al., 2014) y NORM (Adamson, Bormann, Handley, & Macker, 2009) se enfocan en los principios de BIC y TCP-Friendly, respectivamente, adaptados a comunicaciones multicast. En cambio, los protocolos (no se mencionan en la Tabla 1) CooPNC (Miliotis, Alonso, & Verikoukis, 2014), VCMTP (J. Li & Veeraraghavan, 2012), LBRM (Holbrook, Singhal, & Cheriton, 1995) y en (Kasera, Hjálmtýsson, Towsley, & Kurose, 2000) no especifican implementación de mecanismo de control de congestión.

Protocolo	Control de congestión	Según TCP
RDCM	Window-based	BIC
NORM	Rate-based	TCP-Friendly
PGM	Window-based	TCP-Friendly
RMTP	Window-based	slow-start
TMTP	Window-based Rate-based	-

Tabla 1 – Control de congestión implementado en los RMP.

A continuación, se describe específicamente los algoritmos de control de congestión expuestos en la Tabla 1.

RDCM mantiene un control de congestión ajustando la tasa de envío del emisor donde la tasa de envío no debe ser mayor a la tasa de recepción del lado receptor. Para ello, se implementa un mecanismo de control de congestión basado en ventana donde cada receptor mantiene una ventana de congestión individual. El tamaño de la ventana se actualiza usando el algoritmo BIC (Xu et al., 2004) y la pérdida de paquetes se usa como señal de congestión. Idealmente, cada receptor puede: *a)* detectar inmediatamente pérdida de paquetes en el momento que ocurre, *b)* actualizar la ventana de congestión y enviar al nodo emisor.

Por otro lado, en el protocolo NORM se especifica un mecanismo de control de congestión, NORM-CC, basado en el esquema de TCP-Friendly multicast (TFMCC). La transmisión de datos en NORM se basa en tasa de envío a diferencia de los algoritmos de control de congestión basados en ventana como es el caso de TCP. El funcionamiento de NORM-CC básicamente el receptor que experimenta pérdida de paquetes que corresponde a la tasa de envío calculada más baja se identifica como el receptor actual con mayor limitación de recepción de paquetes (Current Limiting Receiver – CLR). Como medida principal para el funcionamiento del algoritmo de control de congestión, el emisor en NORM transmite mensajes periódicamente que contienen información de control para conocer el estado de los receptores. Estos responden con mensajes de retroalimentación para determinar el estado de congestión y con ello ajustar la tasa de envío de los emisores.

Mientras en el protocolo PGM se presenta un esquema de control de congestión, PGMCC, para comunicaciones multicast de tasa de envío única (single-rate) compatible con TCP. PGMCC es un mecanismo de control de congestión basado en ventana y tiene como objetivo principal hacer que el emisor no transmita más rápido que el receptor más lento dentro del sistema. PGMCC funciona de tal manera que el emisor monitorea continuamente la información recibida en mensajes NAK desde los receptores. La información que proporcionan los receptores es fundamental para PGM y consta de: *a)* identidad del receptor, *b)* último número de secuencia recibido, y *c)* tasa de pérdida de paquetes en el nodo local.

En cuanto al protocolo RMTP, utiliza un mecanismo de control de congestión basado en ventana dónde el emisor utiliza una ventana de congestión *cong_win* para recudir la velocidad de transmisión cuando existe congestión en el sistema. El mecanismo implementado utiliza un procedimiento de ajuste lineal de ventana llamado *slow-start* que se utiliza en implementaciones de TCP. Básicamente, el emisor usa un inicio lento en espera de reconocimientos positivos (ACK) desde los receptores.

En el protocolo TMTP utiliza una combinación de técnicas basadas en tasa de envío y en ventana. El componente basado en tasa de envío prohíbe a los emisores enviar datos a una tasa mayor que la tasa máxima de transmisión predefinida. La tasa máxima de transmisión se define al momento de crear el grupo multicast y nunca cambia. Mientras que, en el mecanismo basado en ventana disminuye la tasa de envío dividiendo el tamaño de la ventana y retrasando las retransmisiones el mayor tiempo posible, lo cual aumenta la posibilidad de recibir un acuse ACK de recibido.

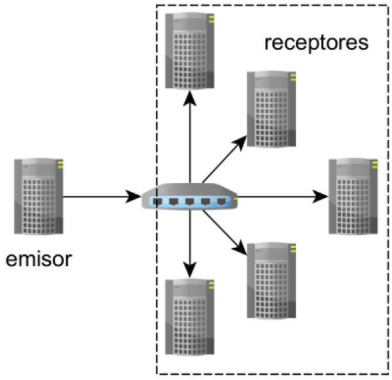

Figura 1 – Configuración básica para envíos multicast

Como se puede observar en el análisis previo, las implementaciones se realizan siguiendo los principios de los algoritmos de control de congestión utilizados en TCP. Al mismo tiempo, las implementaciones tienen un enfoque de transmisión de datos en el que un emisor envía datos a diferentes receptores, tal como se muestra en la Figura 1., y en ocasiones utilizan una configuración en árbol para hacer la distribución de los datos (D. Li et al., 2014). En cambio, en el escenario de estudio para evaluar nuestra propuesta se busca utilizar al máximo todos los recursos existentes en el sistema de tal manera que los nodos tengan las funciones de emisor/receptor. Por lo tanto, nuestro mecanismo de control de congestión analiza constantemente el estado de los nodos, del conmutador y considera la tecnología de almacenamiento utilizada para ajustar la tasa de transferencia desde los emisores.

3. Mecanismo de control de congestión

Típicamente, los sistemas actuales de almacenamiento distribuido se ejecutan en entornos de clúster de computadores, donde las transferencias de datos son elevadas. Aunado a esto, cuando existen múltiples transferencias multicast simultáneamente, la pérdida de paquetes se podría originar cuando:

- El búfer del conmutador se satura debido al tamaño pequeño de estos (en nuestro caso 1 Mbyte).
- El sistema operativo no tiene la capacidad de gestionar los paquetes recibidos.
- La aplicación no puede procesar todo el tráfico multicast cuando se utilizan discos lentos.

Siendo así, en el presente trabajo se propone un mecanismo de control de congestión basado en ventana para escenarios donde se requiere hacer múltiples transferencias multicast. Además, el mecanismo propuesto monitorea constantemente el estado del búfer del conmutador y la tecnología de almacenamiento utilizada en el sistema, tal como se muestra en la Figura 2, en la que se asume que, en la configuración un nodo receptor podría recibir datos desde uno, dos o tres emisores simultáneamente.

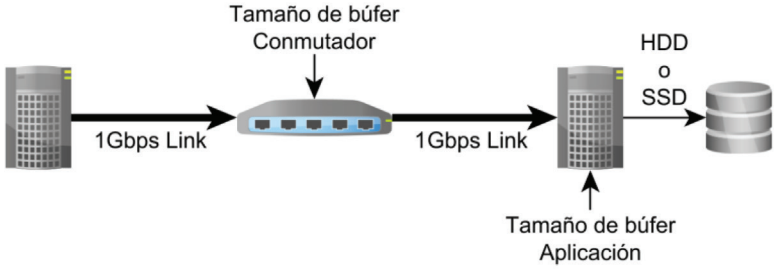

Figura 2 – Posibles puntos de congestión en transferencias multicast

3.1. Detalles de la implementación

La implementación se ha desarrollado en lenguaje C en la distribución CentOS de Linux. Se ha desarrollado la codificación del cliente y del servidor donde reside el mecanismo de control de congestión propuesto. Ambas aplicaciones utilizan sockets UDP para habilitar el tráfico multicast. Básicamente los clientes, reciben datos y los escriben en disco, y cada determinado tiempo envían información de control para ajustar la tasa de envío por el mecanismo de control de congestión propuesto en el lado de los emisores.

La información de control se envía desde los receptores a los emisores a través de un paquete Broadcast (ver Figura 3).

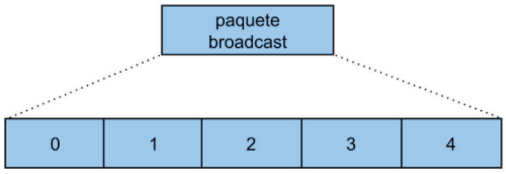

Figura 3 – Campos incluidos en el paquete broadcast con información de control

Los campos de la Figura 3 se describen a continuación para una mejor comprensión del funcionamiento del mecanismo de control propuesto.

1. *marca de tiempo*: almacena la marca de tiempo cuando el algoritmo prepara el paquete Broadcast a enviar.
2. *último paquete recibido* (*upr*): almacena la secuencia del último paquete recibido.
3. *total de paquetes recibidos* (*tpr*): contabiliza el total de paquetes recibidos en el socket.
4. *total de paquetes perdidos* (*tpp*): contabiliza el total de paquetes perdidos.
5. *bytes escritos en disco* (*bed*): almacena el total de bytes escritos en el disco.

El mecanismo de control de congestión es usado para prevenir o evitar la congestión de la red. Para lograrlo se apoya del uso de:

- *reconocimientos negativos (NAK)*: de tal manera que los receptores solo envían solicitud de retransmisión cada vez que se pierde un paquete.
- *ventana de congestión*: se utiliza una ventana de congestión adaptativa de acuerdo a la configuración del sistema, primordialmente cuando los receptores se asocian a diferentes direcciones multicast (descrito en la sección 3).

Debido al tamaño reducido del búfer del conmutador, éste podría ser la principal causa de pérdida de paquetes cuando los emisores envían datos a la máxima capacidad de envío (en nuestro caso se ha utilizado una red Gigabit que puede alcanzar una tasa de envío de hasta 120MB/s en cada emisor). En el algoritmo, para resolver esta situación se aprovecha la información de control que se envía desde los receptores a los emisores, de tal manera que se calcula la diferencia *upr* (número secuencia del *último paquete recibido* en el receptor) y *upe* (número de secuencia del *último paquete enviado* en el emisor). La siguiente ecuación simple ilustra el cálculo de la diferencia mencionada: ΔT_x $R_x = upr - upe$ y se calcula en el lado del emisor.

El mecanismo de control de congestión se calcula en el lado del emisor y de manera periódica recibe información de control desde los receptores. Cada paquete con información de control (Figura 3) se computa en tiempo real por cada hebra de recepción en cada receptor y se envía a los emisores y la información solo se procesa en los emisores correspondientes. Una vez que la información sea procesada en el mecanismo de control de congestión de los emisores, se calcula lo siguiente considerando el número de emisores:

1. tasa máxima de envío de datos (o tasa de envío crítica) sin pérdida de paquetes, la tasa de envío se ajusta dinámicamente y se incluye un retardo de tiempo T_{CWND} entre cada CWND.
2. tamaño de ventana de congestión (CWND) óptimo.
3. ancho de banda de red y de escritura en disco de los receptores.

En cuanto al funcionamiento del algoritmo propuesto en este artículo, la Figura 4 ilustra la función básica que se realiza en la propuesta. Como se ha mencionado anteriormente, el mecanismo de control de congestión se ejecuta en el lado de los emisores y recibe como parámetros de entrada la información de control que se envía desde los emisores. Al momento de iniciar se crea la función manejador_brodcast() que se utiliza para recibir la información de los receptores, con la información se calcula: el número de emisores (*n_emisores <- sumatoria_i*) en cada receptor y el ancho de banda agregado (*BWr <- sumatoria_BWi*) que en la *marca de tiempo* recibe el receptor, valores óptimos para *CWND* y *Tcwnd*. Si el número de emisores en determinado receptor es 1 (*n_emisores = 1*) el algoritmo asigna los valores más óptimos de *CWND* y para calcular *Tcwnd* se asigna *T_1* si *buffer < 80%* (es decir, si el búfer de recepción tiene menos del 80% de ocupación), en caso opuesto se *Tcwnd = T_1w*.

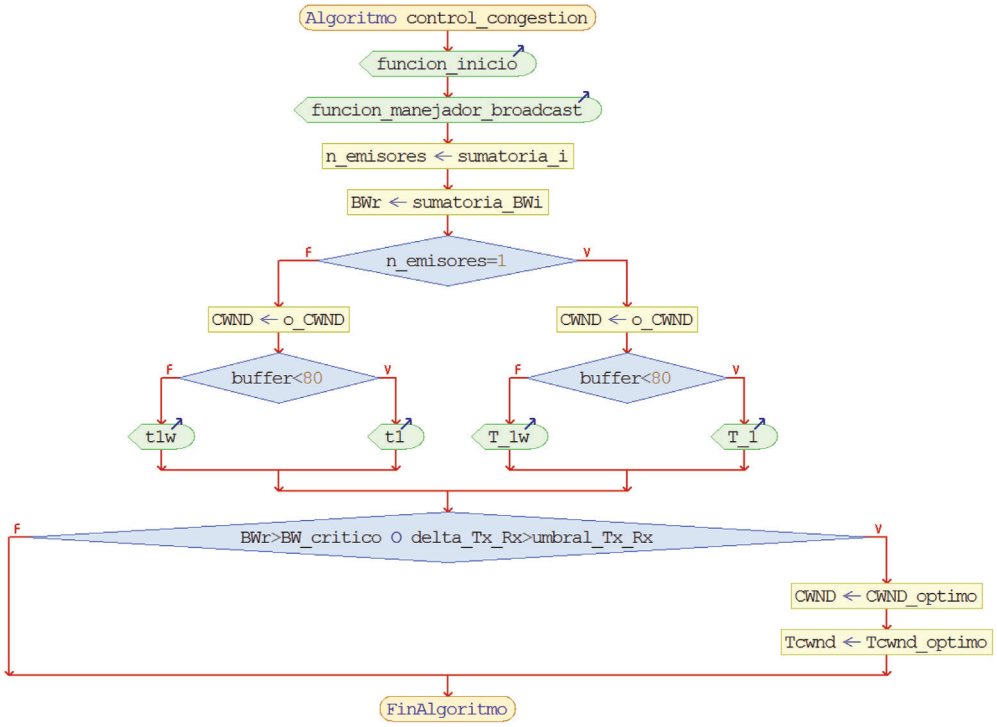

Figura 4 – Diagrama de flujo básico del funcionamiento del mecanismo de control de congestión

Cuando *n_emisores* > *1* el algoritmo asume que existen 2 o hasta 3 emisores (3 emisores es el máximo permitido en este estudio), por lo cual, la probabilidad de congestión es mayor. Consecuentemente, el algoritmo asigna un nuevo valor de *CWND*. Aunado a esto, el algoritmo calcula un nuevo valor para *Tcwnd*, de tal manera que, si *buffer* < *80%*, *Tcwnd* = *t1* y en caso contrario se asigna *t1w*. Siendo así, *Tcwnd* (Ver Figura 5) es el retardo que agrega el algoritmo entre cada ventana de envío y, se usa *T_1* y *t1* son los retardos utilizados para conseguir una tasa máxima de envío de datos desde los emisores, mientras que *T_1w* y *t1w* se utilizan para enviar datos a una tasa mínima de envío.

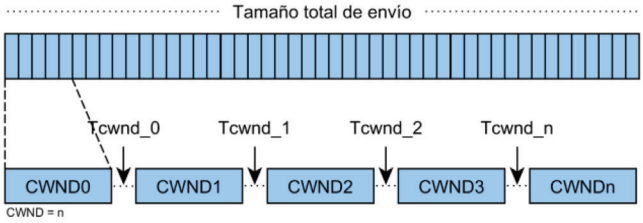

Figura 5 – Asignación de *Tcwnd* según cálculo realizado en el algoritmo de control de congestión

Con el fin de detectar congestión en el conmutador, el algoritmo compara el ancho de banda de los receptores con el ancho de banda crítico, es decir, el ancho de banda de recepción máximo permitido en los receptores ($BWr > BW\_critico$) y la diferencia $\Delta T_x\_R_x$ con el $umbral\_T_x\_R_x$ ($\Delta T_x\_R_x > umbral\_T_x\_R_x$). Si una de las dos condiciones se cumple, el algoritmo asigna nuevos valores de *CWND* y *Tcwnd* para disminuir la tasa de envío y con ello evitar congestión.

Con esta medida, el mecanismo propuesto permite reducir considerablemente la pérdida de paquetes porque el control de congestión puede ralentizar la tasa de envío desde los emisores. Como idea principal, se pretende evitar la pérdida de paquetes para impedir las retransmisiones que en muchas ocasiones puede ser innecesaria si el algoritmo tiene la funcionalidad adecuada.

4. Evaluación del algoritmo

En la literatura es común encontrar propuestas que son evaluadas en entornos simulados. En nuestro caso, con el fin de mostrar resultados reales, se ha evaluado el algoritmo en un entorno de clúster de computadores donde se realizan transferencias por otras aplicaciones principalmente con conexiones TCP.

4.1. Configuración

La configuración se distingue de diferentes grupos multicast donde la cantidad se determina por el número emisores. Es decir, cada emisor envía datos a una determinada dirección multicast, y forma un grupo con hasta tres nodos receptores (se determina el valor de tres receptores por cuestiones de diseño del escenario de evaluación).

Los nodos se interconectan por una red de 1Gbps a través de un conmutador Dlink DGS-1248T con un búfer de 1Mbyte y tiene la capacidad de reenviar hasta 71.4 Mpaquetes/s. La configuración de cada nodo consta de 2 CPU Intel Xeon E5320 a 1.87GHz, Memoria RAM de 8GB, Sistema Operativo Linux distribución CentOS 6.6. En cuanto a los discos de almacenamiento se incorporan discos de estado sólido SSD Kingston de 120GB, Firmware 521ABBF0 y una tasa de escritura máxima de 450MB/s.

4.2. Experimentos y resultados

Se ha determinado que cada nodo se asociará a tres direcciones multicast, de tal manera que constantemente recibe paquetes desde tres diferentes emisores. Lo anterior supone un evento crítico debido a la poca capacidad del búfer del conmutador e incrementa la probabilidad de pérdida de paquetes.

Según un estudio previamente realizado para definir el tamaño de paquete para este escenario, se define como *tamaño_paquete* = 8KB. Además, se ha estudiado para obtener el mejor tamaño de CWND, siendo el valor óptimo *CWND = 3*. Con estos valores alcanzados, el algoritmo de control de congestión prácticamente evita la pérdida de paquetes con un ancho de banda considerablemente aceptable según el escenario de estudio.

4.3 Congestión en conmutador: $\Delta T_x\_R_x$

Como primer análisis realizado, y de acuerdo a la configuración planteada, además considerando los dispositivos de almacenamiento en los receptores se llevó a cabo la evaluación del conmutador, de tal manera que el algoritmo de control de congestión propuesto determine el mejor funcionamiento de acuerdo a la tasa de envío registrada en los receptores. En la Figura 6 se puede visualizar la predicción del algoritmo que calcula el estado de saturación del conmutador cuando los emisores envían datos a la máxima capacidad posible. El porcentaje de pérdida de paquetes alcanza los máximos valores (casi el 100%) cuando la tasa de envío de los emisores es alrededor de 120MBps (en la Figura 6 se muestra la sumatoria de ancho de banda que registra un receptor) de tal manera que los receptores registran una mínima tasa de recepción de paquetes. En función de que el algoritmo detecta la pérdida de paquetes con información de control de los receptores, los emisores disminuyen la tasa de envío hasta alcanzar la tasa de envío óptima y así disminuir la pérdida de paquetes a causa de saturación en el conmutador. Mientras el valor ΔTx_Rx sea menor, no existe congestión en el conmutador.

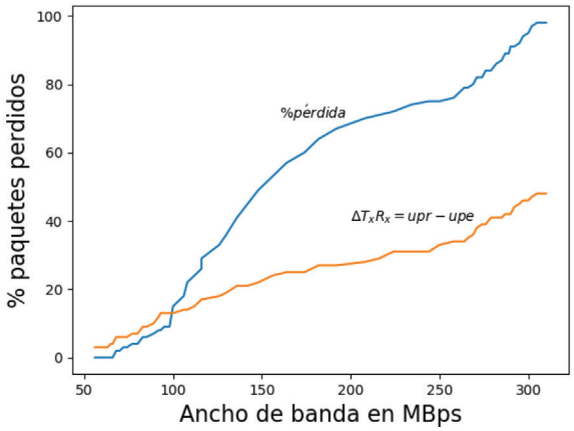

Figura 6 – Valores para ΔTx_Rx respecto al ancho de banda de recepción

4.4. Ganancia de ancho de banda $\sum BW_r + BW_{env}$

De acuerdo a los resultados observados en la Figura 6 y, considerando que el algoritmo, en primer lugar, tiene como objetivo evitar la pérdida de paquetes, a continuación se presentan resultados de ganancia de ancho de banda.

En la Figura 7 se muestra el ancho de banda obtenido por un receptor considerando la sumatoria de recepción desde todos los emisores a los que se ha asociado para recibir información. En la Figura 7 se representa:

- BW_{max}, máximo ancho de banda teórico disponible en el nodo para envío y recepción.

- BW_{env}, máxima tasa de envío como nodo emisor.
- $\sum_r BW_r$, sumatoria del ancho de banda de recepción, en este caso, simultáneamente el nodo recibe información desde 3 emisores con una tasa de envío de ±22MB/s.
- $\sum_r BW_r + BW_{env}$, total de ancho de banda que consume el nodo en envío y recepción simultáneamente.

En la evaluación se ha verificado la tasa de pérdida de paquetes en donde predomina una tasa del 0% con algunos valores que alcanzan el 0.001%. En la evaluación aquí presentada, cada nodo consume aproximadamente ±80MB/s de ancho de banda considerando que en el entorno de evaluación se ejecutaban otras aplicaciones que consumían ancho de banda de la red.

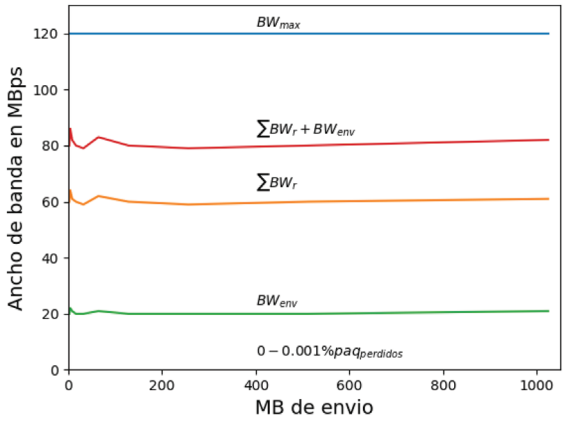

Figura 7 – Ancho de banda en receptor con diferentes tamaños de envío

4.5. Ancho de banda de emisor/receptor y pérdida de paquetes

Según los valores óptimos obtenidos en la subsección 4.2, en la Figura 8 se representa el ancho de banda $\Sigma BW_{emisores}$ como la capacidad máxima de envío de los emisores (para mayor ilustración, se muestra la sumatoria de dos emisores) y por otro lado se obtiene el ancho de banda agregado ΣBW_r del receptor. En este caso el algoritmo, agrega un retardo T_{cwnd} hasta evitar la pérdida de paquetes (en los estudios realizados la tasa de pérdida de paquetes no superaba el 0.001%) ver Figura 9.

En la evaluación, se ha conseguido evitar la pérdida de paquetes a una tasa de envío de los emisores de 20MB/s y una tasa de recepción agregado de 40MB/s como se observa en la Figura 8.

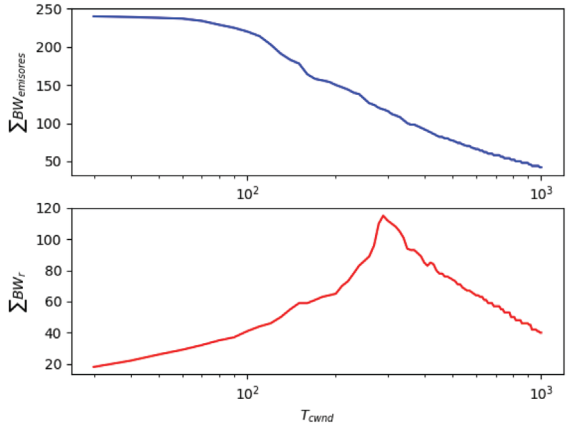

Figura 8 – Ancho de banda de 2 emisores y receptor para CWND = 3

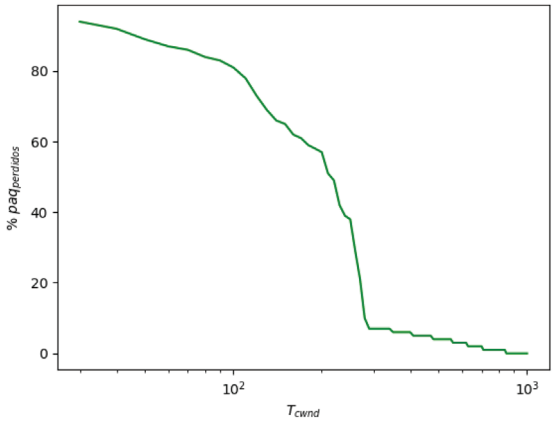

Figura 9 – Pérdida de paquetes para CWND = 3

De igual manera, se ha realizado un análisis con tamaño de ventana CWND=4 con el fin de verificar el comportamiento del algoritmo y del ancho de banda obtenido tanto en emisor y receptores. Aumentar el tamaño de ventana implica un ligero incremento en el tráfico de la red y, para evitar la pérdida de paquetes es necesario agregar un retardo T_{cwnd} mayor y así ralentizar por un lado la tasa de envío. En la Figura 10 se puede observar la caída en la tasa de envío en emisores y consecuentemente en la tasa de recepción. Y, en la Figura 11 se observa el porcentaje de tasa de pérdida de paquetes con el tamaño de ventana CWND=4 utilizado en estos resultados.

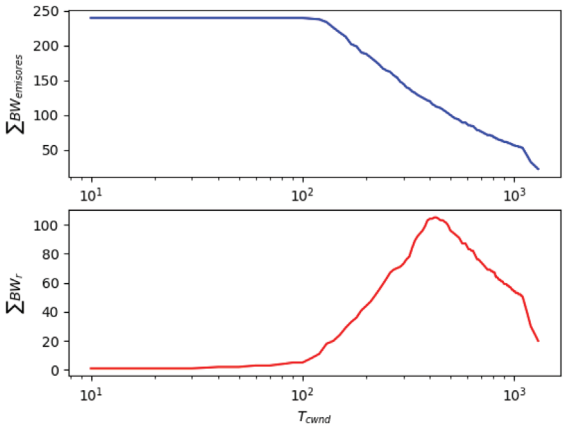

Figura 10 – Ancho de banda de 2 emisores y receptor para CWND = 4

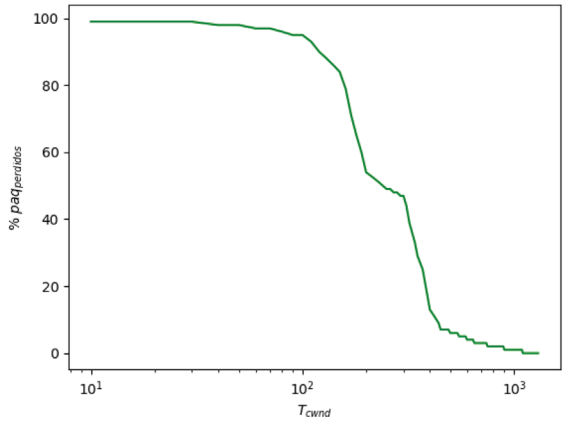

Figura 11 – Pérdida de paquetes para CWND = 4

4.6. Ancho de banda en disco BW_{disco}

Respecto a los discos utilizados, y siguiendo como enfoque principal de aplicación donde un sistema de almacenamiento distribuido garantice la disponibilidad de los datos para las aplicaciones, es necesario asegurar la escritura de los datos en el disco. Con la función *fdatasync()* se puede lograr tal fin. De tal manera que, se agrega un parámetro *sync_size = 32 paquetes* a considerarse en el mecanismo de control de congestión. Es decir, la función *fdatasync()* escribe los datos en disco cada 256KB. El máximo ancho de banda promedio en los discos SSD utilizados alcanzan hasta 120MB/s para nuestra configuración, ver Figura 12.

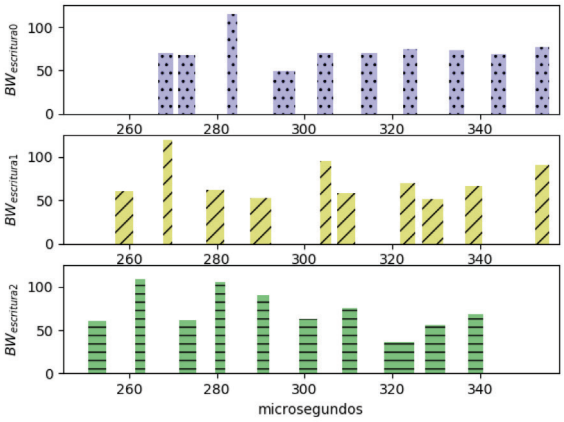

Figura 12 – Evaluación de escrituras simultáneas en disco SSD

Como se puede observar, el comportamiento de los dispositivos de almacenamiento, en este caso discos SSD dan soporte al algoritmo de control de congestión evitando la saturación y por consecuencia pérdida de paquetes en los receptores al momento de realizar las escrituras simultáneas de los paquetes de datos recibidos.

5. Conclusiones

En las redes de datos, con el crecimiento de tráfico de información se hace necesario implementar mecanismos que aseguren la entrega a todos los destinatarios. Los enfoques que existen en la literatura no consideran una orientación para implementarse en sistemas de almacenamiento distribuido. Por tal motivo, en el presente trabajo se presentó el diseño y evaluación de rendimiento de un nuevo algoritmo de control de congestión basado en ventana (window-based) para comunicaciones de datos en entornos multicast múltiple donde los nodos del sistema actúan como emisores y receptores de datos. El algoritmo constantemente evalúa el estado de saturación del conmutador y de los nodos receptores considerando la tecnología de almacenamiento utilizada, en este caso discos SSD. Se realizaron evaluaciones de rendimiento en un entorno real simulando una capa de almacenamiento que provee de información a transmitir. En los resultados, el algoritmo consigue mantener una nula o reducida pérdida de paquetes al mismo tiempo que el ancho de banda de red agregado se mantiene estable y considerablemente alto sin consumir el ancho de banda disponible por las interfaces de red.

Referencias

Adamson, B., Bormann, C., Handley, M., & Macker, J. (2009). NORM: *Nack-Oriented Reliable Multicast Transport Protocol,* DOI: 10.17487/RFC5740

Ait Hellal, O., & Altman, E. (2000). Analysis of TCP vegas and TCP reno. *Telecommunication Systems, 15*(3-4), 381–404. DOI: 10.1023/A:1019159332202

Borthakur, D. (2008). *HDFS architecture guide. Retrieved from: https://hadoop. apache.org/docs/r1.2.1/hdfs_design.pdf.*

Díaz, O., & Muñoz, M. (2018). Implementación de un enfoque DevSecOps+ Risk Management en un Centro de Datos de una organización Mexicana. *RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação*, (26), 43–53. DOI: 10.17013/ risti.26.43-53

Fall, K. R., & Stevens, W. R. (2011). *TCP/IP illustrated, volume 1: The protocols*. Boston: Addison-Wesley.

Gemmell, J., Montgomery, T., Speakman, T., & Crowcroft, J. (2003). The PGM reliable multicast protocol. *IEEE Network, 17*(1), 16–22. DOI: 10.1109/MNET.2003.1174173

Ha, S., Rhee, I., & Xu, L. (2008). CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review, 42*(5), 64–74. DOI: 10.1145/1400097.1400105

Holbrook, H. W., Singhal, S. K., & Cheriton, D. R. (1995). Log-based receiver-reliable multicast for distributed interactive simulation. *ACM SIGCOMM Computer Communication Review, 25*(4), 328–341. DOI: 10.1145/217391.217468

Kasera, S. K., Hjálmtýsson, G., Towsley, D. F., & Kurose, J. F. (2000). Scalable reliable multicast using multiple multicast channels. *IEEE/ACM Transactions on Networking, 8*(3), 294–310. DOI: 10.1109/90.851976

Leith, D. J., Shorten, R. N., & McCullagh, G. (2008). Experimental evaluation of cubic-TCP. In *Proceedings of the 6th International Workshop on Protocols for Fast Long-Distance Networks* (PFLDnet 2008), 5–7.

Li, D., Xu, M., Liu, Y., Xie, X., Cui, Y., Wang, J., & Chen, G. (2014). Reliable multicast in data center networks. *IEEE Transactions on Computers, 63*(8), 2011–2024. DOI: 10.1109/TC.2013.91

Li, J., & Veeraraghavan, M. (2012). A reliable message multicast transport protocol for virtual circuits. In *International Conference on Communications, Mobility, and Computing (CMC), 119–123.*

Miliotis, V., Alonso, L., & Verikoukis, C. (2014). CooPNC: A cooperative multicast protocol exploiting physical layer network coding. *Ad Hoc Networks, 14*, 35–50. DOI: 10.1016/j.adhoc.2013.11.004

Palos-Sánchez, P. R., Arenas-Márquez, F. J., & Aguayo-Camacho, M. (2017). La adopción de la tecnología cloud computing (SaaS): efectos de la complejidad tecnológica vs formación y soporte. *RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação*, (22), 89–105. DOI: 10.17013/risti.22.89-105

Paul, S., Sabnani, K. K., Lin, J., & Bhattacharyya, S. (1997). Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in Communications, 15*(3), 407–421. DOI: 10.1109/49.564138

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The hadoop distributed file system. In *IEEE 26th Symposium On Mass Storage Systems and Technologies (MSST), 2010, (pp.* 1-10). DOI: 10.1109/MSST.2010.5496972

Wittmann, R., & Zitterbart, M. (2000). *Multicast communication: Protocols, programming, & applications*. Amsterdam: Elsevier.

Xu, L., Harfoush, K., & Rhee, I. (2004). Binary increase congestion control (BIC) for fast long-distance networks. In *INFOCOM 2004. Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies, (pp. 4* 2514-2524). DOI: 10.1109/INFCOM.2004.1354672

Yavatkar, R., Griffoen, J., & Sudan, M. (1995). A reliable dissemination protocol for interactive collaborative applications. *Proceedings of the Third ACM International Conference on Multimedia, (pp.* 333-344). DOI: 10.1145/217279.215288