Transporte de datos Multicast Confiable con Control de Congestión Dinámico

Gabriel Gomiz – Luis Marrone

Laboratorio de Investigación en Nuevas Tecnologías Informáticas
Facultad de Informática - UNLP
Calles 50 y 120, 1900 La Plata, República Argentina
gabriel.gomiz@gmail.com
lmarrone@linti.unlp.edu.ar
http://www.linti.unlp.edu.ar

Resumen La naturaleza de la transmisión multicast hace que sea un mecanismo no orientado a conexión, por lo que la mayoría de los protocolos multicast se desarrollan sobre la capa de transporte UDP y sobre este es necesario construir un esquema de confiabilidad. Para aplicaciones de "streaming" de audio o vídeo, la pérdida ocasional de un segmento es aceptable, pero al transmitir datos críticos (archivos, programas, etc.) es necesario un mecanismo que garantice confiabilidad. Por lo tanto se presenta en esta publicación una propuesta de transmisión multicast confiable con control de congestión, como alternativa a los mecanismos disponibles en la actualidad.

Keywords: multicast, PGM, congestion, ACK

1. Introducción

Muchas aplicaciones necesitan semánticas confiables de transmisión de datos para funcionar correctamente y recibir una secuencia de paquetes sin pérdidas, ni duplicados y en el orden exacto en el que fueron transmitidos. Un protocolo de transporte confiable resuelve estas cuestiones y oculta dicha complejidad proporcionando una abstracción confiable a las aplicaciones.

Todos estos protocolos utilizan las mismas ideas para cumplir con los requisitos de confiabilidad: redundancia para resolver pérdidas de paquetes, almacenamiento (buffers) en los receptores para poder reordenar y el uso de temporizadores y retransmisiones. Debido a que las aplicaciones grupales de comunicación tienen una amplia variedad de requerimientos de confiabilidad, se han desarrollado múltiples protocolos multicast confiables, ninguno de los cuales es un estándar dominante así como lo es TCP para las transmisiones unicast confiables. [13]

Existen, fundamentalmente, dos aproximaciones para los protocolos de Multicast confiable:

1. ACK: hacer que todos los receptores confirmen la recepción de los paquetes
2. NACK: hacer que los receptores solo informen cuando detectan pérdida de paquetes

Si es necesario conocer que absolutamente todos los receptores recibieron todos los datos, entonces es probable que se necesite un protocolo basado en ACKs. Por otro lado, si para la aplicación en cuestión no es importante si alguno de los receptores no recibió la totalidad de los datos, entonces los protocolos basados en NACKs pueden ser mas adecuados ya que escalan mucho mejor.

1.1. Mecanismos basados en ACKs

Debido a que un grupo multicast puede tener un arbitrario número de miembros y a que los protocolos tradicionales utilizan acks para garantizar confiabilidad, esto requeriría que el transmisor pueda manejar un número arbitrario de acks. Desafortunadamente, esto deriva en un problema conocido como "ACK implosion". Este problema ha sido el foco de múltiples investigaciones en los últimos años y existen varias estrategias para solucionarlo o, al menos, intentar disminuirlo.

Para superar el problema de la implosión de las confirmaciones, los protocolos de multicast confiable aplican una aproximación jerárquica en donde el multicast se restringe a una única fuente. Antes de enviar los datos, se establece un árbol de reenvío ("forwarding tree") desde la fuente hasta todos los

miembros del grupo y se determinan puntos de confirmación ("acknowledgement points"). Un punto de confirmación consiste de un router en el árbol de reenvío que acepta realizar copias de caché de los datos y procesar las confirmaciones de los routers o las estaciones que estén por debajo en el árbol. Si se requiere una retransmisión, el punto de confirmación obtiene una copia desde su caché.

Otra estrategia para mejorar los protocolos basados en ACK es la de combinar varios ACKs en un solo paquete.

1.2. Mecanismos basados en NACKs

Muchos esquemas de multicast confiable, en vez de utilizar confirmaciones positivas (ACKs), utilizan confirmaciones negativas (NAKs); esto quiere decir que las estaciones receptoras no responden salvo que algún datagrama se pierda o no sea recibido. Este esquema permite una mejor escalabilidad para el ambiente en el que usamos multicast ya que se pasa la carga de control de errores del transmisor a los destinatarios. La estación que transmite envía los datagramas sin esperar las confirmaciones. La idea es evitar el envío de mensajes de estado (ACKs) cuando todo está normal y así mejorar la eficiencia del protocolo [3].

Para permitir a una estación detectar que un datagrama no ha sido recibido, se debe asignar a cada uno un número único de secuencia. Cuando la estación detecta una pérdida envía un NACK para solicitar la retransmisión del datagrama perdido. El NACK debe ser propagado hacia arriba en el árbol (en dirección a la fuente) hasta que el mismo llega a un punto de confirmación. En este punto se procesa el NACK y se retransmite el datagrama hacia abajo en el árbol.

Cada punto de confirmación utiliza el mismo esquema para asegurarse de que tiene una copia de todos los datagramas de la secuencia. Eventualmente, al seguir cada uno de los puntos de confirmación, llegaremos a la fuente de la transmisión.

Con esta aproximación, el diseño de la topología y los puntos de confirmación son cruciales para el éxito del esquema de multicast confiable. Sin los suficientes puntos de confirmación, un datagrama perdido puede causar una implosión de NACKs. En particular, si un router tiene demasiados descendientes, un datagrama perdido puede causar que el mismo sea sobrecargado con pedidos de retransmisión. Desafortunadamente, la selección automática de los puntos de confirmación es un problema complejo y muchos protocolos multicast confiables requieren configuración manual. Por esto, la transmisión multicast se aplica mejor a servicios que tienden a existir por largos períodos de tiempo, topologías que no cambian rápidamente y situaciones donde los routers intermedios acuerdan servir como puntos de confirmación [4].

Como vemos, los mecanismos basados en NACKs pueden sufrir también problemas de implosión, los cuales son resueltos con técnicas de supresión de NACKs y/o soluciones basadas en temporizadores, como hace por ejemplo el protocolo SRM.

1.3. Grupo de trabajo RMT

La IETF formó el grupo RMT (Reliable Multicast Transport Working Group) para establecer un conjunto estándar de protocolos de transporte multicast confiables. Este grupo creó un conjunto de directivas que los desarrolladores de los protocolos propuestos deberían satisfacer. Una vez implementados, se espera que sean de uso general para una amplia variedad de aplicaciones.

Es claro que diferentes aplicaciones van a requerir diferentes aspectos de confiabilidad. A diferencia del modelo de transporte unicast, donde TCP es prácticamente universal, es poco probable que un único protocolo de transporte multicast confiable sea aceptable para todo tipo de aplicaciones. El foco del grupo RMT apunta a tres tipos de módulos:

- "Nack Oriented Reliable Multicast Protocol" (NORM) [5] que utiliza confirmaciones negativas (NAKs) para garantizar confiabilidad. Este protocolo evita el problema de la implosión de los NAKs haciendo que todos los receptores también escuchen y procesen los NAKs de otras estaciones y/o utilizando demoras probabilísticas para evitar NAKs redundantes. NORM puede trabajar en modelos multicast con un único transmisor y también con múltiples transmisores. Este protocolo seguramente será mas útil en aplicaciones de transferencia de archivos.

Pros: simplicidad; el emisor no necesita conocer la cantidad de receptores; el emisor no necesita llevar cuenta del estado de los receptores; escalabilidad para aplicaciones y topologías de red donde es prohibitivo construir una infraestructura de entrega multicast sobre la capa básica IP multicast. [12]

Cons: como varios de los protocolos basados en NAKs, es difícil para el emisor saber cuando puede liberar los buffers de transmisión; se necesitan mecanismos adicionales a nivel de sesión si el emisor necesita saber si un determinado receptor ha recibido todos los datos.

- "TRee ACKnowledgement based protocol" (TRACK) [6] que utiliza un árbol para controlar la retro-alimentación y las *reparaciones*(retransmisión de paquetes perdidos). Este protocolo tiene la característica de devolver al transmisor una confirmación de cada receptor. Esto puede ser importante, por ejemplo, en aplicaciones financieras donde el receptor *debe pagar* por los datos. En este modelo el transmisor necesita una confirmación de que puede *cobrar* a los receptores. TRACK puede no ser una buena alternativa en redes donde el camino de retorno del receptor al transmisor no sea confiable.
- "Asynchronous Layered Coding" (ALC) [7] que utiliza técnicas de forward error-correction (FEC) y no requiere de retro-alimentación.

2. Alternativas de transporte multicast

2.1. Multicast IP

Pros: estándar y altamente disponible. Respeta la filosofía *extremo a extremo.*
Cons: no tiene garantías de confiabilidad, no es *amigable* con los routers

2.2. Scalable Reliable Multicast (SRM)

Scalable Reliable Multicast (SRM)[8] es un marco de trabajo de multicast confiable para usar a nivel aplicación y orientado a sesiones. En este marco de trabajo todo el tráfico es multicast. Este marco de trabajo puede ser resumido describiendo sus 3 tipos de paquetes:

- **latidos (heartbeats):** cada miembro envía periódicamente un heartbeat que incluye el número de secuencia del último paquete enviado. Estos paquetes son utilizados por los demás miembros para poder detectar la pérdida de paquetes, comparando el número de secuencia del heartbeat con el número de secuencia del último paquete recibido.
- **NACKs:** cuando se detecta un paquete perdido, se envía un paquete NACK a todos los miembros. Como todos los miembros ven este NACK, todos pueden participar en la reparación de los datos.
- **reparación (repair):** cada miembro de la sesión mantiene un caché con los últimos paquetes de datos recibidos (y enviados) y si reciben un NACK de un paquete que tienen en el caché, retransmiten el paquete a todo el grupo como una reparación.

Para minimizar el número de NACKs y reparaciones, estas dos operaciones se preceden por un "exponential back-off". Esto significa que en vez de enviar el paquete directamente, el cliente debe esperar un tiempo aleatorio (basado en la distancia al nodo en cuestión) y si en este tiempo otro nodo envía el paquete correspondiente, este cancela su propia transmisión.

SRM está diseñado para cumplir con la definición minimal de multicast confiable, esto es, entrega eventual de todos los datos a todos los miembros del grupo, sin asegurar un determinado orden de entrega. Los autores creen que si fuera necesario asegurar un determinado orden de entrega, sería fácil de agregar una capa sobre SRM que de esto se ocupe.

SRM está fuertemente basado en el modelo de entrega de grupo que es la pieza central del protocolo multicast IP. En multicast IP, las fuentes de datos simplemente envían a la dirección multicast del grupo sin necesitar conocimiento previo de los miembros del grupo.

Para poder recibir datos enviados al grupo, los receptores anuncian que están interesados a través de un mensaje *multicast join* en la subred local y no necesitan conocimiento previo de los miembros del grupo y/o transmisores activos. Cada receptor se une o deja el grupo de manera individual, sin afectar la transmisión de los datos a cualquier otro miembro.

SRM extiende el concepto de grupo multicast maximizando la información y los datos compartidos entre todos los miembros, y fortalece la individualidad de la pertenencia haciendo a cada miembro responsable por la correcta recepción de sus propios datos.

Finalmente, SRM intenta seguir los principios centrales del diseño de TCP/IP. Primero, SRM requiere sólo el modelo básico de entrega del protocolo IP – mejor esfuerzo con posible duplicación y/o re-ordenamiento de paquetes – y construye la confiabilidad en una base punto-a-punto. No se requiere ningún cambio o soporte especial de la red IP que existe por debajo. En segundo lugar, de una forma similar a TCP que setea temporizadores y ventanas de control de congestión, los algoritmos de SRM dinámicamente ajustan sus parámetros de control basándose en la performance observada dentro de una sesión. Esto permite a las aplicaciones que utilizan el marco de trabajo de SRM a adaptarse a un rango amplio de tamaños de grupo, topologías y anchos de banda de los enlaces mientras mantiene una performance alta y robusta. [8]

Pros: útil para replicar datos con poca probabilidad de cambios. No sufre de implosiones de ACK o NAK.

Cons: limitado a escalabilidad de mejor esfuerzo; no puede llevar cuenta de la pertenencia a los grupos; el throughput es inestable si la tasa de fallos de la red no es desestimable.

2.3. Reliable Multicast Transport Protocol (RMTP)

RMTP [9] provee una entrega ordenada y completa de una secuencia de datos desde una fuente a un grupo de receptores. RMTP está basado en una jerarquía de múltiples niveles en la cual los receptores se agrupan en jerarquías de regiones locales, con un receptor designado (DR) en cada región. RMTP define DRs que reúnen los mensajes de estado de los nodos de la región RMTP local y proveen las reparaciones (retransmisiones de datos perdidos), si estos están disponibles. Los receptores dirigen los mensajes administrativos usando unicast hacia el DR. De esta manera, el DR provee recuperación local y consolidación del tráfico de control al siguiente DR de la jerarquía si los datos solicitados no estuvieran disponibles.

RMTP provee entrega confiable de una única fuente a un grupo de receptores sin saber exactamente la identidad de los mismos. Los receptores envían sus mensajes de estado periódicamente al transmisor/DR que retransmite los paquetes que se puedan haber perdido. Dado que los ACKs se envían periódicamente desde los receptores, aunque uno de estos ACKs se pierda, el transmisor/DR no debe realizar nada especial, porque otro ACK reflejando el estado actualizado será generado por el mismo receptor. Por lo tanto, la generación periódica de mensajes de estado provee al protocolo RMTP de una característica inherente de tolerancia a fallos y no debe ser considerado como una sobrecarga ya que simplifica la recuperación de errores.

El enfoque jerárquico utilizado en RMTP junto con la decisión de diseño de no llevar cuenta explícitamente de los receptores, dan un alto grado de escalabilidad al protocolo. Si algunos receptores están muy lejos del transmisor, este no debe preocuparse por estos, ya que el DR correspondiente será responsable de manejar los ACKs y de retransmitir los paquetes perdidos. Además, una característica es clave para la escalabilidad de RMTP: la información de estado que se mantiene en el transmisor es independiente del número

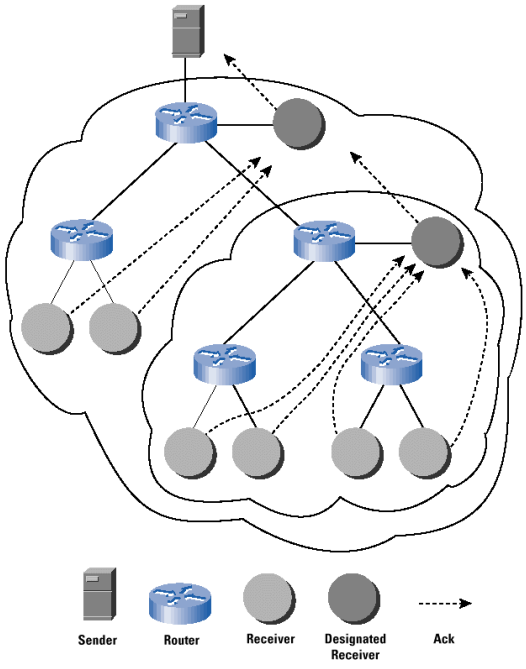

Figura 1. [10]

de receptores. El precio que RMTP paga por esta escalabilidad es el caché adicional que tienen que mantener tanto el transmisor como los DRs.

Pros: su estructura jerárquica brinda un comportamiento predecible; se adapta automáticamente a fallos en los servidores o a servidores que no pueden alcanzarse. Útil para transmisión de archivos multicast.

Cons: comparte las mismas características que SRM. Requiere que los DRs se definan estáticamente.

2.4. Pragmatic General Multicast (PGM-OPGM-PGMCC)

Pragmatic General Multicast (PGM) es un protocolo de transporte multicast confiable diseñado principalmente para aplicaciones que requieran transmisión de datos multicast ordenada y libre de duplicados desde múltiples transmisores hacia múltiples receptores.

La principal ventaja de PGM sobre los protocolos multicast tradicionales es que para cada receptor en el grupo, PGM garantiza que el mismo recibe todos los paquetes a partir de las transmisiones y posibles retransmisiones, o es capaz de detectar una pérdida de paquetes de datos que es irrecuperable.

PGM se diseñó específicamente como una solución para aplicaciones multicast cuyos requerimientos de confiabilidad sean básicos. Su meta central de diseño es la simplicidad de operación siempre teniendo en cuenta la escalabilidad del protocolo y la eficiencia en el uso de la red.

En PGM, no existe la noción de pertenencia a un grupo; se provee una entrega multicast confiable de datos dentro de una ventana de transmisión. El árbol de PGM se construye utilizando elementos de red (NEs) que manejan PGM sobre el árbol multicast existente. Una fuente PGM envía a los receptores paquetes de datos multicast (ODATA) en una secuencia. Cuando los receptores detectan paquetes que faltan en la secuencia, envían un paquete unicast a su padre en el árbol PGM. Los padres confirman la recepción del NAK a todos sus hijos con una confirmación multicast del NAK (paquete conocido como NCF). Los paquetes de reparación (RDATA) son generados por la fuente o por un reparador local designado (DLR) en respuesta a un NAK; esto significa que los NEs nunca guardan ODATA o proporcionan paquetes de reparación. Un paquete de reparación es el paquete perdido reenviado o un paquete FEC, dependiendo de los parámetros de la sesión. Antes de enviar un NAK, los receptores ejecutan un back-off aleatorio y suprimen el NAK si reciben el correspondiente NCF, los datos o los datos de reparación.

En cuanto a implementaciones de PGM es importante considerar **OpenPGM** . Es una implementación de código abierto del protocolo PGM que facilita el desarrollo de aplicaciones distribuídas escalables que intercambian datos a través de la red y con asistencia de la misma. OpenPGM es un transporte de datos, no impone ningún formato de mensaje o lógica de comunicación de alto nivel. OpenPGM puede ser ejecutado en varias plataformas de hardware y de software permitiendo así una plataforma de red heterogénea.

Otra optimización que realizan los receptores es demorar intencionalmente por un tiempo mínimo aleatorio la emisión de un NAK para evitar que varios receptores emitan el mismo NAK simultáneamente. Opcionalmente, es posible utilizar FEC (Forward Error Correction) donde se utilizan códigos de bloque lineales para generar h paquetes de paridad a partir de k paquetes originales de datos, de forma que tomando k paquetes cualquiera, del total de (h+k) paquetes, se pueden decodificar los k paquetes originales.

En el modo básico de operación de PGM simplemente se realiza un límite en la tasa de trasferencia del emisor. Para realizar control de congestión el emisor debe recibir información acerca del estado de la transmisión. Para este fin, PGM soporta 3 tipos opcionales de feedback al emisor:

1. carga del peor enlace (medido por los NEs)
2. carga del peor camino punto-a-punto (medido por los NEs)
3. carga del peor camino punto-a-punto (medido por los receptores)

Sin embargo, PGM no especifica como se ajustará esta tasa; el esquema de control de congestión se deja a criterio de la implementación.

Como trabajos relacionados, se han propuesto varios esquemas de control de congestión complementarios a PGM. Uno de éstos es llamado **PGMCC** [11]; esquema de control de congestión de tasa única cuyos objetivos se centran en la escalabilidad, estabilidad y rápida respuesta a la variación de las condiciones de la red.

En el esquema de tasa única que usa PGMCC, todos los receptores obtienen la misma tasa de datos y el emisor se adapta al receptor más lento del grupo. Se conoce que estos esquemas tienen la limitación de que un único receptor lento puede tirar abajo la tasa de envío para todo el grupo. El comportamiento es muy similar al control de congestión de TCP, ya que utiliza un ciclo de control basado en ventana que se ejecuta entre el emisor y un receptor seleccionado, llamado "ACKER". El rol del ACKER es proveer al emisor de una rápida retroalimentación de la misma forma que lo hace un receptor en TCP. La implementación de esta variante requiere modificaciones tanto en los emisores como en los receptores.

3. Nuestra propuesta - PGMCOOP

La utilización de PGM en esquemas 1 a N incorpora confiabilidad a las transmisiones, pero, al utilizar una tasa de transferencia fija, no permite aprovechar el ancho de banda completo que hubiera disponible en la red, ya que no posee un mecanismo para poder incrementarlo, así como tampoco disminuye el mismo en el caso de existir congestión.

Aquí se presenta una variante del protocolo, que llamaremos PGMCOOP, cuya novedad con respecto a PGM es incorporar un mecanismo de ajuste dinámico de la tasa de transferencia del emisor, en función de los NAKs / tasa de NAKs que va recibiendo. La simplicidad del diseño y la implementación es uno de los objetivos fundamentales de esta variante, y para cumplir con esto se buscó una estrategia donde solo sea necesario modificar la lógica de emisión y que el esquema sea transparente para los receptores que podrán seguir trabajando con el protocolo PGM puro.

Una estrategia que se utilizó antes de desarrollar PGMCOOP, fue usar un protocolo multicast desarrollado ad-hoc que llamamos internamente MDP. Se implementó sobre multicast UDP simple y tenía 2 etapas:

1. Envío de datos usando IP multicast con tasa fija de transferencia.
2. Fase de reparación posterior con estrategia unicast para cada receptor que necesite reparaciones.

El comportamiento del protocolo MDP con respecto a la cantidad de bloques perdidos es proporcional al crecimiento de la tasa de transferencia. A medida que incrementamos la misma y nos acercamos al límite de potencia de procesamiento de los receptores, los bloques perdidos se incrementan. Este comportamiento evidencia dificultades generadas por el uso de una parametrización estática (tasa de transferencia) y por no disponer de un mecanismo de control de congestión, además de la ineficiencia generada a partir de potencialmente repetir la transmisión de las mismos bloques de reparación en un esquema unicast. PGMCOOP intenta solucionar estos inconvenientes.

Para el control de congestión se prefirió utilizar una estrategia diferente a la de PGMCC, realizando modificaciones mínimas a PGM para poder cambiar la tasa de transferencia en forma dinámica y de esta forma intentar mejorar su comportamiento. Debido a que este protocolo será utilizado en múltiples escenarios que poseen distintas características de equipamiento de red y emisores y receptores con características totalmente heterogéneas, es necesario que el mismo pueda adaptarse a cada red regulando la tasa de transmisión para no sobrecargar a los receptores, evitando así que se produzcan demasiados NAKs y se deba realizar un alto número de retransmisiones.

Se adoptó una estrategia de "congestion avoidance", con un esquema basado en el algoritmo de incremento aditivo y reducción multiplicativa (AIMD). Esta estrategia evita la congestión combinando el crecimiento lineal de la ventana de congestión con una reducción exponencial de la tasa de transferencia cuando ésta es detectada.

Se utilizan los NAKs como indicadores de congestión, el emisor reducirá la tasa de transferencia por un factor multiplicativo; por ejemplo dividir a la mitad la ventana de transmisión. En ausencia de NAKs la ventana se incrementa linealmente. El resultado de esto es una curva con forma de dientes de sierra ya que este algoritmo va probando la red para detectar el ancho de banda disponible, de manera similar a lo que ocurre con el algoritmo para evitar congestión en TCP.

3.1. Diseño

Cada sesión de envío del emisor trabajará enviando datos a los receptores via paquetes de datos (ODATA) con PGM, con una ventana de transmisión cwnd, que le permitirá enviar sin restricciones hasta agotada dicha ventana. Luego del envío de cada bloque, el emisor deberá chequear cual es la tasa

actual de recepción de paquetes NAK. Si la tasa de NAKs es menor al umbral nak_th durante toda la duración de la ventana cwnd, entonces se ampliará la ventana de transmisión utilizando el factor lineal aditivo (a). Por el contrario, si se detecta una tasa de NAKs mayor al umbral nak_th en ese período, reduciremos la ventana de transmisión multiplicando la misma por el factor de decremento (b) y el emisor entrará en un estado de control de congestión (CC). En el estado de control de congestión, el emisor podrá seguir enviando datos, pero sin incrementar la ventana de transmisión cwnd y deberá controlar luego de la transmisión de cada paquete la tasa actual de NAKs; es decir que dentro del estado de control de congestión el límite cwnd no se utilizará; el emisor, sin embargo, utilizará una ventana de control de congestión cc_cwnd. Esta ventana se usa para "limpiar" el estado de congestión ya que el emisor deberá controlar que no haya recibido ningún paquete NAK durante esta ventana para poder salir del estado de control de congestión. Una vez que el emisor no haya recibido ningún NAK durante una ventana del tamaño definido para el control de congestión se podrá volver al estado normal de envío. Los siguientes parámetros que controlarán la transmisión serán configurables por parte del usuario/desarrollador que utilice las funciones del protocolo:

cwnd: tamaño inicial de ventana de transmisión
cc_cwnd: ventana de transmisión en la que no se deben recibir NAKs para salir de estado CC
nak_th: umbral límite de la tasa de NAKs
a: factor de incremento lineal ($a > 0$)
b: factor de decremento exponencial ($0 < b < 1$)
r: tasa de transferencia inicial de transmisión (valor predeterminado: 100.000 bps)

Algoritmo de envío A continuación se presenta, en lenguaje de pseudo-código, el algoritmo que se utilizará en la solución:

```
 1: mientras existan datos para enviar
 2:     pgm_send(sock, buffer)
 3:     si estado==CC
 4:         si nak_rate < nak_threshold
 5:             si se completó cc_cwnd
 6:                 estado=NORMAL
 7:             sino
 8:                 avanzar ventana cc_cwnd
 9:         sino
10:             reiniciar ventana cc_cwnd
11:     sino
12:         si nak_rate > nak_threshold
13:             estado=CC
14:             cwnd = cwnd * b
15:             cc_cwnd = cc_cwnd * b
16:             continuar
17:     si se completó cwnd
18:         si estado==NORMAL y nak_rate < nak_threshold
19:             cwnd = cwnd + a
20:             cc_cwnd = cc_cwnd + a
21:     sino
22:         avanzar ventana cwnd
23: fin bucle
```

De esta manera, el emisor podrá ir incrementando la tasa de transferencia linealmente, cada vez que complete una ventana cwnd sin que la tasa de NAKs haya superado el umbral. Cuando la tasa de NAKs sea mayor al umbral configurable, deberá reducir la tasa de transferencia y luego entrará en la fase de control de congestión. Cuando haya pasado una ventana cc_cwnd completa donde la tasa de NAKs sea menor a la mitad del umbral definido, entonces podrá salir del estado de control de congestión y volver a trabajar con la ventana cwnd. Este proceso se repetirá sucesivamente y la tasa de transferencia debería converger al ancho de banda disponible en la red para la transmisión multicast.

3.2. Implementación

Los receptores solo implementarán la función normal de recepción con la notificación de los NAKs respectivos a los paquetes faltantes que hayan detectado como cualquier receptor normal PGM. Este diseño también tiene la ventaja de que no es necesario modificar la lógica interna del protocolo PGM o sus funciones de bajo nivel. El esquema de control de congestión se puede construir, por ejemplo, sobre la implementación OpenPGM del protocolo PGM, utilizando sus primitivas.

Se realizaron 2 modificaciones a la librería OpenPGM para poder satisfacer las necesidades de nuestra implementación: una función para poder consultar los contadores internos de la librería (mas precisamente el valor de los NAKs recibidos en el emisor para servir de dato de entrada del algoritmo de control de congestión) y un método para poder modificar la tasa máxima de transferencia que regula el throughput de envío del emisor dentro de una sesión determinada.

4. Escenario de pruebas

La investigación y desarrollo realizados surgen de la necesidad de mejorar el módulo de comunicaciones de un sistema de sucursales, que fué realizado con desarrollo propio y a medida, por la Gerencia de Sistemas de la Cooperativa Obrera Ltda. Este módulo es necesario para el envío de comandos de control a las cajas, para el envío de datos y para monitoreo del estado de cada una de ellas. En la Fig. 2 se muestra el esquema tipo y sobre el cual se realizaron las pruebas.

En particular el requerimiento de comunicaciones implementado, fue un mecanismo para poder enviar tablas de datos desde el SERVER a todas las cajas y además el envío de nuevas versiones de la aplicación (archivos de varios MBs), que por motivos de performance se decidió hacer el envío bajo multicast. Utilizar multicast en una red LAN nos daría la ventaja de realizar solo una transmisión total y, con un bajo número de reparaciones, ocupar muchos menos tiempo el ancho de banda de la red de la sucursal (la cual en muchos casos está limitada a equipamiento no switcheado y con bajas velocidades $\approx 10\,Mb/s$); esta ventaja sería más evidente sobre la solución unicast a medida que creciera el número N de cajas y se reduciría considerablemente el tiempo total de transmisión.

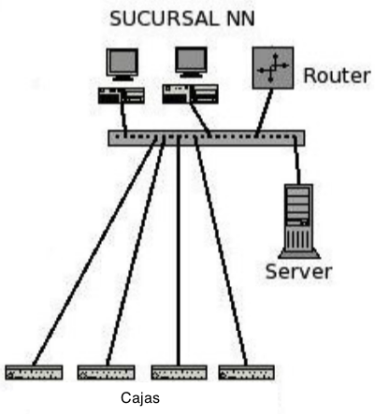

Figura 2.

4.1. Transmisión

En la etapa de preparación se realizan algunas tareas previas a la transmisión y que simplificarán los pasos posteriores. Una vez que se tiene el archivo final a transmitir, se calcula el tamaño total del mismo (file_size). La transmisión se realiza en bloques. Estos bloques permiten la reparación posterior, ya que los receptores pedirán a la fuente los bloques no recibidos/perdidos haciendo referencia por número de bloque. En esta etapa, se calcula el número total de bloques (blocks) en función de file_size y el tamaño de bloque (MAX_DATA_SIZE). Como último paso de esta etapa, se calcula un valor de hash (md5sum) de los contenidos del archivo. Este hash se utiliza para chequear la consistencia del archivo recibido en los receptores y decidir si se utilizará luego en capa de aplicación o deberá ser descartado por errores en algún bloque.

Se dividieron las pruebas en dos grupos de escenarios: aquellos 3 escenarios que tienen redes de 10 Mb/s con repetidores (hubs), escenarios A, B y C y los otros 3 que poseen el segmento de red implementado con switches de 100 Mb/s y tecnología full duplex, escenarios D, E y F. En el caso de 10 Mb/s se realizaron pruebas con tasas iniciales de transferencia de 200, 400, 600 y 800 KB/s. Para cada combinación de escenario y tasa inicial de transferencia se realizaron 5 pruebas de transferencia completas con cada uno de los 3 protocolos. Esto permitió analizar los resultados tomando promedio de

la tasa de transferencia y del tiempo de transmisión y luego se calculó varianza y desviación estándar de estos resultados usando la tasa de transferencia como variable aleatoria. Para los escenarios de 100 Mb/s se utilizó el mismo esquema de pruebas, pero usando como tasas de transferencia iniciales los siguientes valores: 2000, 4000 y 6000 KB/s.

Los protocolos de prueba fueron MDP desarrollado internamente y descripto brevemente en la Sección 3, PGM y nuestra propuesta PGMCOOP.

5. Resultados obtenidos

En principio, el análisis de los resultados del protocolo experimental desarrollado en el presente trabajo puede considerarse positivo. Como vemos en el gráfico que tomamos a modo de ejemplo de uno de los escenarios planteados, en la mayoría de las pruebas realizadas, PGMCOOP fue el protocolo que logró las mejores tasas de transferencia. Ver Fig. 3. El escenario es el C con 8 cajas.

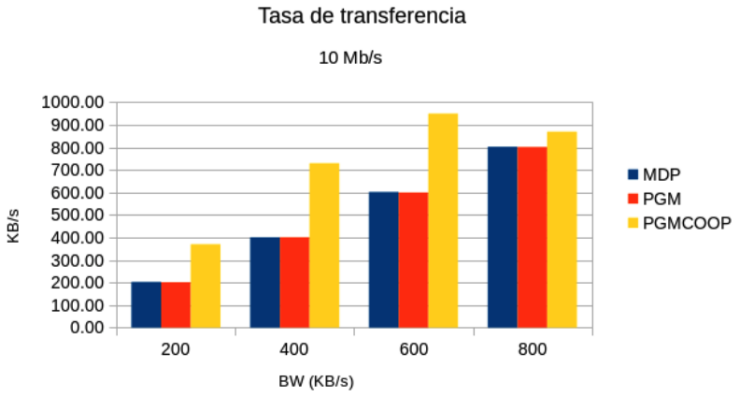

Figura 3.

Las pruebas se realizaron utilizando parámetros de crecimiento de la tasa de transferencia y tamaños de ventanas muy conservadores. Por eso, por ejemplo para tasas iniciales de 200 KB/s, solo se pudo lograr en promedio alrededor de 320 KB/s, pero podrían utilizarse parámetros que permitan un comportamiento mucho mas agresivo del protocolo.

Los mejores rendimientos de PGMCOOP se obtienen utilizando como tasa de transferencia inicial el 50 % del ancho de banda disponible en el segmento de red.

Este comportamiento se produce porque la estrategia de aumentar la tasa de transferencia termina siendo contraproducente para estos casos donde la misma está cerca del límite de capacidad de los receptores.

Con la estrategia implementada, el protocolo pierde un intervalo de tiempo inicial hasta que se dispara el control de congestión, se disminuye la tasa de transferencia y el envío se estabiliza. Pero este proceso tiene un impacto en los resultados, suficiente para que un envío estable usando PGM a 800 KB/s termine dando mejores resultados.

En el escenario C, PGMCOOP claramente fue el protocolo de mejores resultados, en 200 y 400 KB/s casi duplicando las tasa de transferencias de MDP y PGM y en 600 KB/s logrando un 50 % mas.

De estos resultados se puede concluir que resulta adecuada la estrategia de usar ventanas de transferencia para observar la tasa de NAKs que ingresan y subir la tasa de transferencia mientras que la misma no supere el umbral configurado.

En la mayoría de los casos de prueba de los escenarios de 100 Mb/s, también es el protocolo que logró los mejores resultados, pero no supera a los otros 2 protocolos tan claramente como en los casos anteriores. Al utilizar tasas de transferencia iniciales mas exigentes, se llega mas rápidamente a los límites de capacidad de los receptores. Ver Fig. 4. El escenario considerado es el F con 20 cajas.

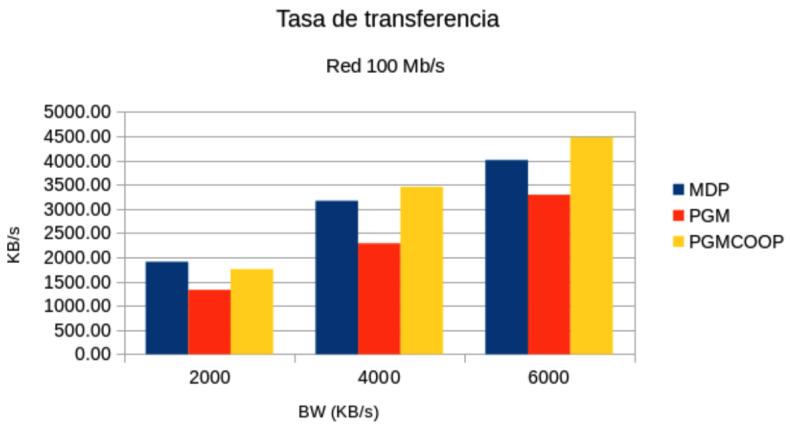

Figura 4.

Referencias

1. Banikazemi, Mohammad.:IP Multicasting: Concepts, Algorithms, and Protocols. Survey Paper, Ohio State University (1997)
2. Davies, Joseph.:Understanding IPv6, (2003)
3. Diot, Cristophe; Dabbous, Walid; Crowcroft Jon.: Multipoint Communication: A Survey of Protocols, Functions and Mechanisms, (2004).
4. Comer, Douglas.: Internetworking with TCP/IP, (2000)
5. Adamson, B., Bormann, C., Handley, M., Macker, J.: NACK-Oriented Reliable Multicast (NORM) Transport Protocol, RFC 5740, IETF, (November 2009)
6. Whetten, B., Chiu, D., Paul, S., Kadansky, M., Taskale, G.:TRACK Architecture, A Scalable Real-Time Reliable Multicast Protocol, Internet Draft, Internet Engineering ask Force, (July 2000).
7. Luby, M., Watson, M., Vicisano, L.: Asynchronous Layered Coding (ALC) Protocol Instantiation, RFC 5775, (April 2010)
8. Floyd, S., Jacobson, V., Liu, C., McCanne, S., Zhang, L.: A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, ACM SIGCOMM 95, (August 1995).
9. Shiroshita, T., Sano,T., Takahashi,O., Yamanouchi, N.:Reliable Multicast Transport Protocol, INTERNET-DRAFT, IETF, (September, 1997)
10. Paul,S.,Sabnani, K. K., Lin, J. C. -H.,Bhattacharyya, S.: Reliable multicast transport protocol (RMTP), IEEE Journal on Selected Areas in Communications (Volume:15 , Issue: 3), (1997-2002).
11. Rizzo, L.,Vicisano, L., Handley, M.: PGMCC single rate multicast congestion control: Protocol Specification, https://tools.ietf.org/html/draft-ietf-rmt-bb-pgmcc-03.txt, (2004).
12. Adamson B., Bormann C., Handley M., Macker, J.: NACK-Oriented Reliable Multicast (NORM) Protocol Building Blocks, (2001)
13. Speakman, T., Crowcroft, J., Gemmell, J., Farinacci, D. , Lin, S., Leshchiner, D., Luby, M., Montgomery, T. , Rizzo, L., Tweedly, A., Bhaskar, N., Edmonstone, R., Sumanasekera, R., Vicisano, L.: PGM Reliable Transport Protocol Specification, RFC 3208, (2001)