UNIVERSIDAD DE GRANADA

Departamento de Arquitectura y Tecnología de Computadores

Programa Oficial de Doctorado en
Tecnologías de la Información y la Comunicación

Tesis Doctoral

Análisis y Optimización de la Interfaz de Comunicación en Sistemas de Ficheros en Red

Realizada por:

Raúl Hernández Palacios

Granada 2016

Editor: Universidad de Granada. Tesis Doctorales
Autor: Raúl Hernández Palacios
ISBN: 978-84-9163-005-0
URI: http://hdl.handle.net/10481/44276

UNIVERSIDAD DE GRANADA

Departamento de Arquitectura y Tecnología de Computadores

Programa Oficial de Doctorado en
Tecnologías de la Información y la Comunicación

Tesis Doctoral

Análisis y Optimización de la Interfaz de Comunicación en Sistemas de Ficheros en Red

Realizada por:
Raúl Hernández Palacios

Dirigida por:
Antonio Francisco Díaz García
Mancia Anguita López

El doctorando Raúl Hernández Palacios y los directores de la tesis D. Antonio Francisco Díaz García y Dña Mancia Anguita López Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, 2016

Doctorando
Raúl Hernández Palacios

Directores
Dr. D. Antonio F. Díaz García Dra. Dña Mancia Anguita López

- A Yuridia, Yulissa, Raúl Santiago y Raúl Sebastián -

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Esta memoria de tesis, es resultado de mucho tiempo de trabajo y sacrificio. En el recorrido ha sido mucha gente que ha estado en el momento preciso para brindar su apoyo, ayuda y consejos, y me gustaría darles las gracias.

En primer lugar, a mi esposa Yuridia, a mis hijos Olga Yulissa, Raúl Santiago y Raúl Sebastián, por su compañía en esta hermosa aventura, por ser mi soporte diario, por su paciencia interminable, por comprender en todo momento mi ausencia, y por cada vez que llegaba a casa recibirme con un abrazo cálido y un "hola papá!"lleno de euforia.

Una parte importante de mi vida ha estado lejos, en México, pero siempre enviando energías positivas que me fortalecieron para finalizar esta meta, Gracias Familia!.

Por siempre tendré un especial agradecimiento a mis directores de tesis, Dr. Antonio F. Díaz y Dra. Mancia Anguita. Esta meta se ha logrado gracias a su valioso tiempo, ayuda, apoyo, guía constante, comprensión y tantos consejos que he recibido de ustedes. Por todo esto y más, Muchas Gracias!.

Un reconocimiento con afecto para el Dr. Julio Ortega por el incondi-

cional apoyo en las gestiones que he necesitado.

A mis compañeros, colegas y amigos, de despacho y comedor, Juanjo, a mis paisanos Virgilio y Rolando, de Ecuador Gonzalo, Andrés y Pablo, de Cuba Elio y compañía, del CITIC Cristina, Manu y otros más. Gracias por esos momentos de buena charla, motivaciones y apoyo.

A mi suegra, infinitamente agradecido, por dejar su vida durante muchos meses para dedicarla al cuidado de mis hijos.

A José Miguel y colaboradoras del CEIVM-Granada, por el apoyo y facilidades, Raúl y Sebastián lo agradecerán y yo siempre.

A mi amigo y hermano, Waseem y familia, por compartir muchos momentos de alegría en unión de nuestras familias.

A mi paisano, Pancho y familia, los fines de semana que hemos vivido con ustedes han sido muy agradables.

Al PROMEP, por ofrecer los medios económicos necesarios para lograr esta meta.

A los compañeros y amigos de la UAEH, Dr. Aquino, Felipe, Nacho, Erika, Alfredo, Jorge y muchos más que desde lejos me han apoyado con gestiones y me han dado ánimos.

A todos ustedes y más... Muchas gracias!

Resumen

Algunas aplicaciones de computación de altas prestaciones (HPC) requieren una rápida y eficiente distribución de datos entre los nodos de un cluster. Algunos ejemplos de éstas pueden ser: aplicaciones de análisis de información o de almacenamiento de datos.

Los sistemas de almacenamiento en red están adaptándose tanto a los nuevos recursos hardware, entre los que están las tecnologías de almacenamiento de estado sólido SSD, las configuraciones SAN y NAS, y las interfaces de comunicación, como a las necesidades crecientes de capacidad y velocidad que demandan los usuarios y las aplicaciones.

La combinación de almacenamiento y comunicación de datos entre ordenadores abre un inmenso abanico de posibilidades en función de qué se comparte y con quién.

El intercambio de datos entre computadores es el objetivo principal de las comunicaciones en el campo de la informática. La creciente evolución de las tecnologías de comunicación ha logrado alcanzar un progreso notable permitiendo, entre otros, la comunicación de una fuente con múltiples destinos simultáneamente (comunicaciones uno-a-muchos). A esta forma de comunicación se le conoce como comunicación de grupo o alternativamente también conocida como multicast.

Con la intención de conseguir mejoras en la interfaz de comunicación de sistemas de ficheros en red, objetivo principal de esta tesis. En este trabajo se han estudiado diferentes alternativas existentes como técnicas de channel

bonding, SDP en Infiniband, técnicas de compresión de datos y finalmente las comunicaciones basadas en multicast.

El resultado de estos estudios previos nos ha permitido dirigir el enfoque a las comunicaciones multicast. Posteriormente se ha realizado un estudio a fondo de los diversos protocolos multicast fiables que existen en la literatura. Hemos podido conocer en mayor detalle las técnicas principales de tal manera que nos permita desarrollar un protocolo que se pueda adaptar a las necesidades de distribución de datos con fiabilidad.

En la presente tesis se propone un protocolo multicast que ofrece fiabilidad y control de congestión en múltiples trasferencias multicast (comunicación muchos-a-muchos) que puede dar soporte a la distribución de datos de los sistemas de ficheros en red. La propuesta del protocolo se basa en: (1) un nuevo mecanismo de control de congestión que evalúa la información de control de los receptores, reduciendo la velocidad de transferencia del emisor, (2) aprovechar la capacidad de difusión múltiple del conmutador, (3) usar IGMP Snooping, que permite reducir la carga de la red reenviando los paquetes sólo a los puertos donde los receptores se han asociado a la dirección multicast, y (4) utilizar de forma eficiente la tecnología de almacenamiento disponible en los nodos, controlando la velocidad de transmisión de los emisores. La implementación se ha desarrollado a nivel de usuario y utiliza la interfaz UDP. En la evaluación de la propuesta se ha utilizado un cluster con las capacidades de software y hardware necesarias. De forma general, los resultados muestran mejoras en el ancho de banda global, evitando la saturación de la red, disminuyendo o evitando la pérdida de paquetes, y reduciendo la sobrecarga que incluyen las comunicaciones uno-a-uno (unicast) en la transmisión de datos en redes de computadores.

Capítulo 1

Introducción

Los sistemas de almacenamiento en red están adaptándose tanto a los nuevos recursos hardware, entre los que están las tecnologías de almacenamiento de estado sólido SSD, las configuraciones SAN y NAS, y las interfaces de comunicación, como a las necesidades crecientes de capacidad y velocidad que demandan los usuarios y las aplicaciones. En este sentido, aunque se ha avanzado bastante, consideramos que todavía la tecnología de almacenamiento puede aprovechar los recursos disponibles de forma más eficiente. Por ejemplo, los sistemas de almacenamiento para computación en clusters de computadores (*cluster computing*) suelen basarse en modelos centralizados que, aunque simplifican la puesta en marcha, suelen ser un importante cuello de botella. Las prestaciones del sistema de almacenamiento, además de por las propias tecnologías en los dispositivos se ven determinadas por otros elementos cuyos efectos están fuertemente interrelacionados. Entre esos elementos cabe citar los protocolos de comunicación y el diseño de la interfaz de comunicación.

Dado que los sistemas de almacenamiento en red dan soporte a las aplicaciones para la disponibilidad de los datos, en estos entornos es común hacer una replicación de los datos entre los nodos del sistema.

Actualmente las comunicaciones en los sistemas de almacenamiento en

red existentes están basados en comunicaciones unicast. Por lo tanto, las comunicaciones unicast generan una carga elevada de tráfico en la red.

En este trabajo se trata de analizar y optimizar la interfaz de comunicación de los sistemas de almacenamiento en red.

Una de las primeras aproximaciones ha sido el estudio de técnicas de *channel bonding*, donde se pueden aprovechar las diversas interfaces de comunicación presentes en un ordenador de forma que origen y destino utilizan simultáneamente todos los recursos de comunicación. El uso de todas las interfaces posibles de un ordenador implica un incremento en el tráfico de la red y sobrecarga en los nodos. Como consecuencia, la saturación de la red podría ser una causa para no conseguir un rendimiento adecuado en los sistemas de almacenamiento en red.

Además, se ha evaluado el uso de la compresión de datos para reducir el tráfico en la red, lo que puede mejorar las prestaciones globales.

Otra de las alternativas de mejora que se han estudiado ha sido hacer uso de las comunicaciones multicast para utilizar de forma más eficiente la interfaz de comunicación en sistemas de almacenamiento en red. El uso de multicast permite distribuir los datos a través de los nodos del sistema reduciendo la carga de la red. Además, las comunicaciones multicast pueden facilitar el soporte de redundancia de datos con mayor eficiencia.

Dado que las comunicaciones multicast usan UDP como protocolo de transporte, en esta tesis también se propone un esquema de control de congestión para garantizar la entrega de los datos entre los nodos del sistema.

Se pretende abordar dicha mejora estudiando los siguientes elementos:

1. Utilización de protocolos basados en *Reliable Multicast* que permitan el envío simultáneo de mensajes de baja latencia a diferentes destinos físicos.

2. Estudio de modelos que permitan el uso eficiente del soporte hardware basado en IGMP para garantizar el funcionamiento de sistemas con

carga elevada de trabajo.

3. Definir un protocolo fiable basado en multicast para dar soporte al envío de datos con redundancia basada en múltiples servidores.

1.1. Objetivos

El objetivo principal de esta tesis doctoral es la optimización de prestaciones de comunicación en sistemas de almacenamiento en red. Para alcanzar el objetivo se plantean diferentes tareas a realizar:

- Estudio de técnicas de channel bonding y SDP en Infiniband.
- Evaluación de la mejora del interfaz de comunicación basada en técnicas de compresión de datos.
- Analizar las diferentes alternativas de protocolos multicast para Ethernet.
- Diseñar e implementar un protocolo multicast fiable que permita el envío simultáneo de datos a determinados destinos de la red.
- Diseñar y desarrollar un modelo que permita el uso eficiente del soporte hardware del conmutador y que garantice baja tasa de pérdida de paquetes.
- Definir un protocolo multicast fiable que, además de permitir envíos simultáneos y asegurar baja tasa de pérdida de paquetes, utilice eficientemente la tecnología de almacenamiento disponible en la configuración del sistema.

Para lograr estos objetivos se ha utilizado la infraestructura del Departamento de Arquitectura y Tecnología de Computadores. En particular, el cluster Erebo, así como otro cluster del Centro de Supercomputación de

Castilla y León, que cuenta con las funcionalidades tanto de hardware y software para poder evaluar las prestaciones globales que se plantean en el trabajo de tesis.

1.2. Estructura de la Tesis

Esta memoria está organizada en cinco capítulos principales, los cuales se describen a continuación.

En el capítulo 2 se presenta una introducción de los modelos de almacenamiento de datos en red. Se describen algunos ejemplos del almacenamiento de bloques, de ficheros, así como de objetos. Además, se describe de manera general la arquitectura de los sistemas de ficheros distribuidos, de los cluster de computadores y se amplía en mayor detalle las comunicaciones que se utilizan en los sistemas de ficheros distribuidos.

El capítulo 3 se describen algunos aspectos importantes para conseguir un servicio de almacenamiento con buenas prestaciones para aplicaciones con alta necesidad de E/S. Por lo tanto se consideran aspectos de comunicación relacionados con PVFS2, el uso de técnicas de *channel bonding*, SDP en Infiniband y por último una evaluación del uso de técnicas de compresión de datos para conseguir la mejora de la interfaz de comunicación.

El capítulo 4 trata en mayor detalle las comunicaciones multicast con idea de conocer las ventajas generales, y a su vez hacer una comparativa con las comunicaciones unicast que se usan principalmente en los sistemas de almacenamiento en red. En las últimas secciones del capítulo se describen algunos aspectos a considerar para el desarrollo de sistemas de comunicación basados en multicast.

Las comunicaciones multicast permiten la distribución simultánea a diferentes destinos. En el capítulo 5 se realiza un análisis del rendimiento de estas comunicaciones. En primer lugar se mencionan algunos trabajos

relacionados al trabajo de la tesis, donde se demuestran algunas de las principales diferencias. Posteriormente se establece una configuración y se realiza un análisis por simulación. A continuación, se plantea la primera propuesta del protocolo que permite el envío simultáneo a diferentes nodos físicos de la configuración. Las evaluaciones se han realizado en un cluster real.

En el capítulo 6 se presenta una versión mejorada del protocolo aplicado al almacenamiento en red. Se trata de obtener los mejores resultados de acuerdo a la configuración hardware, tanto del conmutador, la red, y de los discos. En los resultados se utilizan los valores óptimos para sacar provecho de las comunicaciones multicast, de tal manera que se consideran diferentes escenarios de configuración entre los nodos del sistema.

Por último, se presentan las conclusiones y principales aportaciones del trabajo de tesis. Así como también, se plantean estrategias para continuar con la mejora del protocolo, de tal manera que sea posible incluirlo en sistemas de ficheros distribuidos, tales como AbFS y HDFS.

Capítulo 2

Antecedentes de almacenamiento en red

La combinación de almacenamiento y comunicación de datos entre ordenadores abre un inmenso abanico de posibilidades en función de qué se comparte y con quién. La incesante mejora de las tecnologías permite desarrollar soluciones más eficientes y complejas lo que obliga a una redefinición continua de los estándares.

En este capítulo se describen las características principales de algunos de los sistemas de almacenamiento en red utilizados en la actualidad, centrándose en los sistemas de ficheros distribuidos. También se describen algunos de los elementos principales en los cluster de computadores relacionados con este trabajo y finalmente se estudian las comunicaciones en sistemas de ficheros distribuidos.

Un elemento común de los sistemas de almacenamiento en red que implementan redundancia es la necesidad del envío de los datos a múltiples destinos. Estos sistemas basan su comunicación en modelos punto a punto (unicast).

2.1. Introducción

Una arquitectura basada en un modelo centralizado en el que un servidor tiene acceso a sus propias unidades de almacenamiento planteaba diversas limitaciones:

- Distancia de conexión: La longitud de los cables SCSI permitía hasta 25 m lo que limitaba a nivel práctico la cantidad de recursos que podían conectarse a un servidor.
- Identificación de los dispositivos: Estos buses específicos tienen limitado los números de dispositivo tanto físicos como lógicos.
- Velocidad de acceso: Los buses específicos no tienen las ventajas de otros buses más orientados a transmisión de datos.
- Tolerancia a fallos: Si el servidor al que están conectados presenta algún fallo de funcionamiento ningún otro servidor puede tener acceso a los datos almacenados en las unidades.

Básicamente los datos se pueden gestionar de 3 formas distintas:

- Almacenamiento de bloques.
- Almacenamiento de ficheros.
- Almacenamiento de objetos.

Los dispositivos de almacenamiento suelen trabajar a nivel de bloque ya que facilita la velocidad de acceso así como su localización. Suelen utilizar un tamaño de bloque entre 512 bytes y 4096 bytes. Por otra parte, a nivel lógico resulta más práctico organizar los datos en ficheros según la información que contienen. Este modelo se ha utilizado (y se sigue utilizando actualmente) ya que resulta muy útil y facilita a las aplicaciones el acceso a los datos.

También se utiliza el almacenamiento de objetos que permite eliminar la jerarquía de nombres que se utiliza en el modelo de ficheros.

Los modelos indicados anteriormente pueden ser aplicados tanto a almacenamiento local como remoto. El uso de redes de transmisión rápidas ha permitido eliminar restricciones (descritas al inicio de la sección) y, en consecuencia, han hecho que surjan diversas soluciones para cada uno de los modelos de almacenamientos indicados anteriormente.

2.1.1. Almacenamiento de bloques en red

El incremento de velocidad de las redes de transmisión ha permitido que la transferencia entre los dispositivos y las unidades de procesamiento no se vea penalizada. Las unidades en red de almacenamiento en bloques se denominan SAN (*Storage Area Network*) y permiten que los sistemas operativos accedan a dichos dispositivos prácticamente de forma transparente. Esta conexión ofrece varias ventajas:

- Escalabilidad: Se pueden formar unidades de almacenamiento más grande (compuestas a su vez por múltiples dispositivos) creando así cabinas de almacenamiento.
- Virtualización: También se puede dotar a dichas unidades de una mayor funcionalidad ya que el almacenamiento físico puede distribuirse de forma dinámica estableciendo volúmenes virtuales.
- Distancia de conexión: Estas cabinas pueden estar separadas a mucha mayor distancia ya que pueden utilizar redes de transmisión que soporten mucha más distancia que los buses específicos.
- Múltiples canales: Otra de las ventajas es que pueden establecerse múltiples canales (*channel bonding*) lo que permite aumentar el ancho de banda o incluso funcionar pese a que algún enlace de red no esté operativo.

- Mejoras tecnológicas: Estos enlaces también pueden cambiar de forma independiente al almacenamiento, por lo que incluso se puede cambiar la interfaz de red de 1G, 10 Gb o Infiniband, aprovechando las mejoras de velocidad y prestaciones sin afectar al resto de recursos.

- Soluciones RAID: Los fabricantes suelen implementar soluciones SAN de almacenamiento compuestos por múltiples discos en configuraciones RAID lo que permite que dichas unidades dispongan de mecanismos independientes de recuperación de datos, consiguiendo incluso que si alguna unidad falla, pueda ser reemplazada en caliente y los datos vuelvan a resincronizarse correctamente sin que externamente pueda observarse algún fallo en el acceso a los datos almacenados.

Los principales sistema de transmisión utilizado en unidades SAN son los siguientes:

2.1.1.1. iSCSI

iSCSI (*Internet Small Computer System Interface*) implementa el envío de órdenes SCSI sobre TCP. IBM desarrolló iSCSI como una prueba de concepto en 1998, y presentó el primer borrador del estándar iSCSI al Internet Engineering Task Force (IETF) en 2000. El protocolo fue ratificado en 2003 y ha sido actualizado por lo que actualmente se utiliza el RFC7143 [67].

iSCSI funciona mediante el transporte de datos a nivel de bloque entre un iniciador iSCSI (*initiator*) en un servidor y un destino iSCSI (*target*) en un dispositivo de almacenamiento. El protocolo iSCSI encapsula las órdenes SCSI y el envío de datos en paquetes TCP. Éstos se envían a través de la red mediante una conexión punto a punto. A su llegada, el protocolo iSCSI los procesa, de forma que las órdenes SCSI lleguen al dispositivo local. Una de las ventajas de este modelo es virtualizar el almacenamiento de forma que se pueden realizar agrupaciones de dispositivos físicos para crear dispositivos

lógicos de diversos tamaños. Al utilizar el estándar de Ethernet, iSCSI no necesita tarjetas ni conmutadores adicionales que podrían tener un mayor coste.

Un aspecto importante son los mecanismos de seguridad que implementa iSCSI. Por una parte se utilizan sistemas de autenticación basados en CHAP (*Challenge-Handshake Authentication Protocol*) que permiten asignar claves de acceso a las unidades de almacenamiento de forma que un iniciador iSCSI pueda acceder a un iSCSI destino. También se pueden utilizar listas de control de acceso (ACL) para limitar los privilegios de acceso, principalmente a la información de la unidad SAN.

iSCSI se ha extendido también sobre Infiniband mediante el protocolo iSER (*iSCSI Extensions for RDMA data transfer*) [55] aprovechando así la mejora de velocidad y latencia que ofrece.

2.1.1.2. Fibre Channel (FC)

Fibre Channel es una tecnología de transmisión de datos entre dispositivos informáticos con velocidades de 4 Gbps (y 10 Gbps en un futuro próximo). FC es especialmente adecuado para la conexión de servidores a dispositivos de almacenamiento compartido y para la interconexión de los controladores de almacenamiento y unidades. FC ofreció una velocidad mayor que SCSI e iSCSI, por lo que se empezó a utilizar como interfaz de transmisión entre servidores y dispositivos de almacenamiento en cluster. Si se utiliza FC con fibra óptica como medio físico permite distancias de hasta 10 Km. Para distancias más cortas incluso puede utilizarse cable coaxial.

FC permite una conexión punto a punto y está diseñado para interoperar con SCSI. Las normas para FC están especificadas por el canal físico de la fibra y el estándar de señalización, y el ANSI X3.230-1994, que es también la norma ISO 14165-1.

2.1.1.3. Otras alternativas

Aunque el protocolo iSCSI es el más utilizado en transmisión de datos sobre IP, existen otras alternativas:

- *ATA over Ethernet* (AoE): protocolo Ethernet SAN que implementa el protocolo de almacenamiento *Advanced Technology Attachment* (ATA) directamente en Ethernet. Sería el equivalente a iSCSI sobre IP pero para unidades ATA.

- *Fibre Channel over Ethernet* (FCoE): se convirtió en un estándar oficial en 2009. Promovido principalmente por Cisco Systems y otros proveedores de redes, trataba de utilizar Ethernet para el transporte de paquetes y reducir la necesidad de tarjetas y conmutadores específicos de FC. Realmente se suele utilizar poco en la conmutación SAN.

- *Fibre Channel over IP* (FCIP): en este caso se realiza tunneling de FC sobre IP, lo que facilita el intercambio de datos a través de una empresa distribuida geográficamente.

- *Internet Fibre Channel Protocol* (iFCP): Es una alternativa al FCIP que combina redes SCSI y FC en Internet.

2.1.1.4. DRBD

Además de los modelos descritos anteriormente orientados a la transferencia de bloques, existe una solución basada en Linux que combina el almacenamiento en bloques y la transmisión de datos permitiendo crear unidades de almacenamiento redundantes con dispositivos físicos que están situados en ordenadores distintos.

El sistema DRBD (*Distributed Replicated Block Device*) [9] es una solución de almacenamiento basada en software que replica el contenido de los dispositivos de bloque (discos, particiones, volúmenes lógicos, etc)

entre nodos. Por lo tanto, DRBD puede ser una solución fiable para dar soporte de alta disponibilidad en los datos de un sistema.

La replicación de datos en DRBD se realiza:

- En tiempo real: La replicación se realiza de forma continua mientras las aplicaciones modifican los datos en el dispositivo.
- De forma transparente: Las aplicaciones no perciben que los datos se almacenan en varios nodos.
- Síncrona o asíncrona: En el caso síncrono, las aplicaciones son notificadas después de que las escrituras se han llevado a cabo en todos los nodos conectados (Protocolo C). Mientras tanto en el caso asíncrono, las aplicaciones son notificadas de las terminaciones de escritura cuando las escrituras se han completado a nivel local, que por lo general es antes de que se han propagado a los otros nodos (Protocolo A y B).

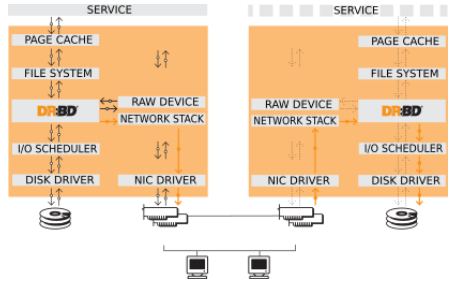

Figura 2.1: Posición de DRBD en la pila de E/S de Linux.

La funcionalidad central de DRBD se implementa como un módulo del kernel de Linux. En concreto, DRBD constituye un controlador para dispositivo de bloque virtual. Por lo tanto DRBD se sitúa en la parte inferior de la pila de E/S del sistema Linux (en este caso) tal y como se muestra en la Figura 2.1.

2.1.2. Almacenamiento de ficheros en red

Un modelo ampliamente utilizado para el almacenamiento de datos es trabajar a nivel de ficheros que generalmente están accesibles mediante un árbol de directorios.

La Figura 2.2 [14] muestra ejemplos de los distintos tipos de sistemas de ficheros.

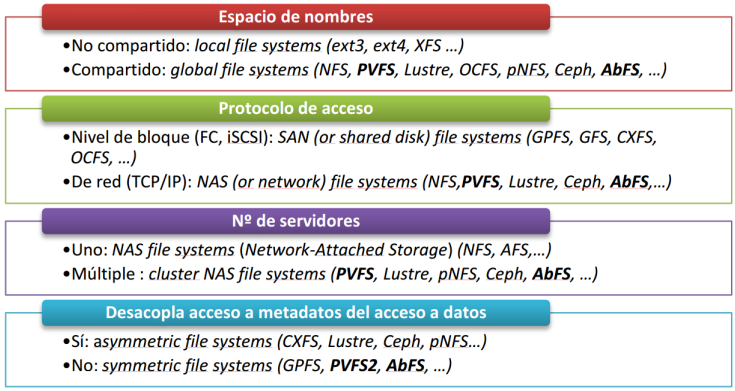

Figura 2.2: Clasificación de sistemas de ficheros con ejemplos.

Los sistemas de ficheros en red permiten compartir ficheros y directorios entre distintos computadores que se encuentran conectadas a una red, de manera que, aunque se acceda desde un computador (cliente) a un disco conectado a otra máquina (servidor), se tiene acceso a los datos de igual forma que si estuvieran almacenados de forma local. Los sistemas de ficheros en red se clasifican en:

- Sistemas de ficheros centralizados (SFC).
- Sistemas de ficheros distribuidos (SFD).
- Sistemas de ficheros paralelos (SFP).

Los sistemas de ficheros centralizados disponen de un servidor que unifica y gestiona todas las operaciones con los ficheros de forma que los clientes

acceden a dicho servidor. Como ejemplos ampliamente conocidos son NFS y CIFS.

NFS (*Network File System*) [73, 19, 88] se ha convertido en el sistema de ficheros en red más ampliamente utilizado por muchos años. NFS implementa un modelo de sistema de ficheros muy parecido a los sistemas Unix. Los archivos son estructurados como una secuencia de bytes y el sistema de ficheros está jerárquicamente estructurado. Un cliente NFS se comunica con el servidor usando llamadas RPC (*Remote Procedure Call*). Las operaciones del sistema de ficheros son implementadas mediante RPC. Inicialmente NFS se ha implementado con soporte UDP, pero en la actualidad se utiliza TCP para garantizar una comunicación segura.

Un sistema de ficheros distribuido se caracteriza por estar formado por un número de computadores que trabajan de forma coordinada con un fin común: que múltiples procesos puedan compartir datos durante un periodo de tiempo de manera segura y fiable [94].

La diferencia principal entre estos sistemas es que los SFP permiten a los computadores cliente acceder más rápidamente (con menor latencia y mayor productividad) a un único fichero porque se accede en paralelo a trozos del mismo. La Figura 2.3 [14] relaciona distintas clases o tipos de sistemas de ficheros y distintos criterios de clasificación de sistemas de ficheros. En nuestro caso vamos a centrarnos en los sistemas de ficheros distribuidos.

Los sistemas de ficheros distribuidos se utilizan de forma frecuente en aplicaciones que requieren análisis de gran cantidad de datos. Aplicaciones como aerodinámica, análisis probabilístico, predicción del tiempo o geociencias se suelen ejecutar en entornos de computación de altas prestaciones (HPC) en configuraciones con grandes capacidades de cómputo y almacenamiento. Dichas aplicaciones generalmente requieren mayores volúmenes de datos para poder mejorar sus estimaciones por lo que resulta necesario poder proporcionar un almacenamiento eficiente y fácil de utilizar.
Para ello han surgido soluciones específicas para centros de procesamien-

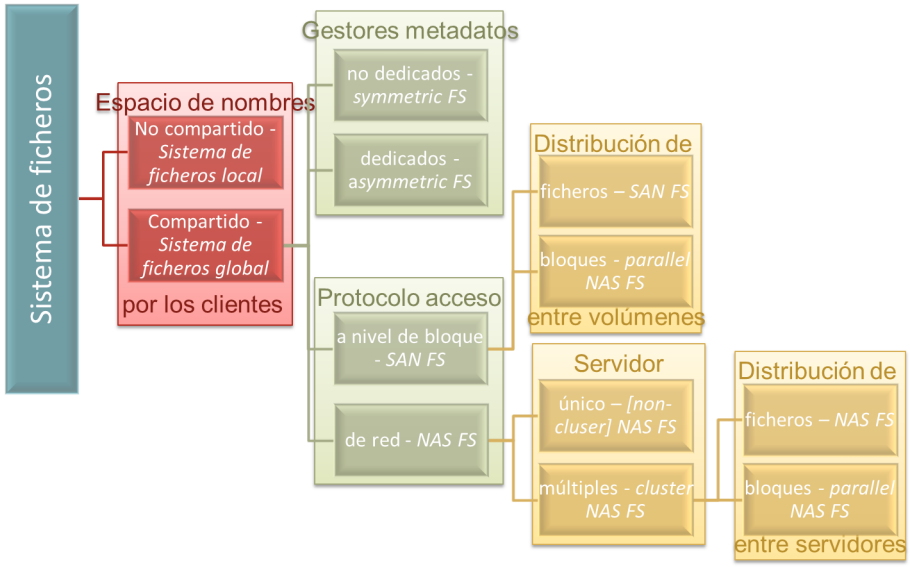

Figura 2.3: Relación entre tipos de sistemas de ficheros y criterios de clasificación.

to de datos donde estos sistemas se encargan de distribuir y almacenar los datos de manera que las aplicaciones puedan hacer uso de ellos de forma transparente. Gran cantidad de ordenadores de la lista TOP500 [93] de los supercomputadores más rápidos del mundo usan sistemas de ficheros distribuidos para ofrecer un almacenamiento persistente, fiable y de rápido acceso a los datos. Actualmente la mayoría de los sistemas de ficheros distribuidos utilizan principalmente TCP como protocolo de comunicación [84][97].

2.1.3. Almacenamiento de objetos en red

Otro sistema de acceso a datos está basado en objetos. Este modelo se utiliza principalmente en el almacenamiento en la nube con estándares como CDMI (*Cloud Data Management Interface*) [12] donde se define una interfaz para que las aplicaciones puedan crear, recuperar y borrar datos que están almacenados en contenedores.

Un importante proveedor de almacenamiento en la nube es Amazon que ofrece diversas soluciones bajo el nombre de S3.

2.2. Sistemas de ficheros distribuidos

Un sistema de ficheros distribuido proporciona un almacenamiento persistente de datos (a veces basado en objetos) donde las aplicaciones y los usuarios pueden controlar desde su creación hasta su destrucción explícita y donde los posibles fallos del sistema quedan ocultos. Dicho sistema consta de recursos o dispositivos de almacenamiento (principalmente discos HDD y SSD) donde los clientes pueden hacer operaciones básicas sobre los ficheros: crear, eliminar, leer o escribir, etc. A diferencia de los sistemas de almacenamiento local, los recursos de almacenamiento y los clientes se encuentran distribuidos en toda la red del sistema. Los ficheros creados en el sistema se comparten de manera unificada y jerárquica entre los usuarios. De tal modo que, los ficheros son almacenados en diferentes recursos de almacenamiento, pero, para el usuario esta distribución es transparente, es decir, los visualiza en una sola ubicación.

En las siguientes subsecciones se describen las principales características de los sistemas de ficheros distribuidos. El enfoque del presente trabajo es mejorar las comunicaciones de los sistemas de ficheros distribuidos, y por lo tanto, se hace un mayor énfasis en las comunicaciones de algunos de los sistemas existentes donde se consideran los protocolos de comunicación y las tecnologías de red.

2.2.1. Arquitectura

Según un estudio realizado en [97] se pueden distinguir dos tipos de arquitecturas para sistemas de ficheros distribuidos. En las arquitecturas *cliente-servidor* muchos servidores gestionan, almacenan y comparten datos y metadatos entre muchos clientes, de tal manera que ofrecen un espacio

de nombre global del sistema. El incrementeo del número de servidores incrementa la capacidad de almacenamiento así como la capacidad de procesamiento. En tanto a las arquitecturas de sistemas distribuidos *basados en cluster*, los datos y metadatos suelen estar separados. Uno o más servidores son dedicados a gestionar los metadatos y otros servidores a almacenar datos. Un sistema con un solo servidor de metadatos se conoce como *centralizado*, mientras que un sistema que distribuye los metadatos en más de un servidor se conoce como *totalmente distribuido*.

2.2.2. Transparencia

El usuario no necesita conocer en detalle el diseño del sistema, cómo y dónde se almacenan los datos, cómo acceder a ellos, los protocolos y tecnologías de red utilizados, y cómo se detectan los fallos. A continuación se explica en detalle los aspectos relacionados para garantizar la transparencia.

2.2.3. Nomenclatura (*Naming*)

Desempeña una función importante, se refiere al mapeo entre objetos lógicos y físicos. Es decir, los usuarios hacen referencia de objetos de datos lógicos representados por nombres de archivos (el usuario hace referencia a un fichero con nombre textual), mientras que el sistema manipula datos almacenados en bloques de discos físicos. El sistema debe respetar las propiedades de:

- Transparencia de ubicación: el usuario debe ver de forma transparente los detalles de cómo y dónde se almacenan los ficheros. De igual forma, es posible que exista más de una copia de un fichero distribuido entre los nodos. Por lo tanto, el mapeo debe informar de la ubicación de cada una de las copias de los ficheros.
- Independiencia de la ubicación: el nombre lógico de un fichero no obliga a cambiar su ubicación en el almacenamiento físico.

2.2.4. Comunicaciones

Dado que un sistema de ficheros distribuido almacena los datos en diferentes servidores implica llevar a cabo una serie de comunicaciones para la distribución. Comúnmente un sistema de ficheros distribuido mantiene un modulo de comunicaciones dedicado para transferir datos entre nodos. Cómo ejemplo podemos encontrar BMI (*Buffered Message Interface*) [28] que consiste en una capa de abstracción de red para E/S paralela. BMI soporta TCP/IP además de diferentes tecnologías de red (Infiniband, Myrinet, etc.). El módulo de comunicaciones de un sistema de ficheros distribuido debe proporcionar entrega de datos rápida y fiable a todos los destinos. Algunos de estos sistemas utilizan mecanimos de comunicación basadas en llamadas RPC sobre el protocolo TCP. Generalmente, los sistemas de ficheros distribuidos utilizan el protocolo TCP en los módulos de comunicación [97]. En tanto, el protocolo UDP prácticamente no se ha explotado en estos escenarios ya que no garantiza la recepción de los datos por lo que resulta necesario añadir mecanismos de control.

2.2.5. Interfaz de acceso a datos y caché

Al igual que un sistema de ficheros local, el sistema de ficheros distribuido debe proporcionar a los clientes la transparencia de las operaciones básicas, crear, escribir, leer y borrar ficheros. El usuario final no debe preocuparse de las tareas que se realizan a bajo nivel de tal manera que las operaciones se puedan desarrollar de forma simple. La interfaz también debe coordinar el acceso a los datos y metadatos de peticiones de diferentes clientes. Algunos de las herramientas para gestionar las operaciones de ficheros se basan en la interfaz de línea de órdenes, tradicionalmente de Unix. PVFS2 [96] utiliza operaciones básicas (cp, rm, mv, etc.). Para realizar las operaciones básicas sobre los ficheros, los usuarios pueden "montar"directorios remotos en el sistema de ficheros local, de tal manera que existe transparencia en la ubicación física de los ficheros. Los usuarios acceden a los ficheros

de manera remota cómo si estuvieran almacenados en los dispositivos de almacenamiento local.

La implementación de caché en un sistema de ficheros distribuido permite (según la ubicación de la caché) reducir el tráfico de la red y/o el uso de CPU. Según [71] un esquema de caché en un sistema de ficheros distribuido debe considerar los siguientes requisitos de diseño:

(a) *Granularidad de los datos almacenados en caché*: La granularidad de los datos almacenados en caché puede variar de partes de un archivo a un archivo completo. Por lo general, más datos se almacenan en caché de lo necesario para satisfacer un único acceso, por lo que muchos accesos pueden ser servidos por los datos almacenados en caché. En [27] se describen las caracteristicas de caché de algunos sistemas de ficheros distribuidos. Una ventaja de utilizar tamaño grande de caché es que puede incrementar la tasa de aciertos al buscar bloques de ficheros y dismuir el tráfico de la red, pero aumenta los problemas de coherencia. Es recomendable que, cuando se implementa memoria de caché grandes, usar bloques grandes (de 8KB o más) y para memoria de caché pequeñas, usar bloques menores de 8KB. En [60] se describen aspectos de diseño de caché con mayor detalle.

(b) *Localización de la caché*: La implementación de caché se puede realizar en el lado de los servidores (reduce los accesos a disco pero incrementa el tráfico de red) o clientes (reduce el tráfico de la red, así como de accesos a disco). Además de que, implementar caché en disco permite dar soporte de confiabilidad, a diferencia de implementar caché en la memoria volatil (los datos almacenados en caché se pierden si el nodo falla). Independientemente de ésta desventaja, [60] describe una serie de ventajas cuando se implementa la caché en la memoria principal dónde hace hincapie al tiempo de acceso reducido. La tendencia actual de los dispositivos de almacenamiento de estado sólido (SSD) puede incrementar las ventajas de implementar caché en los discos.

(c) *Política de modificación*: Los métodos de escritura inmediata (*write-through*) y escritura retardada (*write-back* o *delayed-write*) son las políticas a considerar en el diseño de caché. En el caso de la escritura inmediata, la ventaja principal es la fiabilidad, dado que se pierde poca información cuando un cliente falla. Sin embargo, cada acceso de escritura espera hasta que la información se envía al servidor, por lo tanto el rendimiento de escritura es bajo. Referente a la escritura retardada, las modificaciones se escriben en la caché y posteriormente en los servidores. Aunque presenta problemas de fiabilidad (si un cliente se bloquea los datos no almacenados en disco se pierden) ofrece un par de ventajas, rapidéz en los accesos de escritura y los datos no deben estar escritos en absoluto, es decir, los datos se pueden borrar antes de escribirlos finalmente en los discos de los servidores. Existen versiones del método de escritura retardada que se describen en [60].

(d) *Consistencia de caché*: El cliente determina si la copia de los datos almacenados en la caché local es consistente con la copia maestra en el servidor. Existen dos enfoques para verificar la validez de los datos en cache: iniciado por el cliente (*client-initiated approach*) e iniciado por el servidor (*server-initiated approach*). Algunas de las ventajas y desventajas se describen en [60], y, en [57] se introduce una técnica de implementación de consistencia de caché en sistemas de ficheros distribuidos.

2.2.6. Detección de fallos

El sistema debe proporcionar, de manera transparente para el usuario, la posibilidad de detectar sobrecarga o fallos en los servidores. De tal manera que, en caso de fallo pueda tomar las mejores decisiones para recuperar la funcionalidad del sistema global. El constante intercambio de mensajes entre los nodos del sistema permite establecer mecanismos como *heartbeats* [5] para comprobar la disponibilidad de los servidores. Comúnmente, los

cluster de computadores son la plataforma principal para la ejecución de los sistemas de ficheros distribuidos. Por lo tanto, existen herramientas para monitorizar el estado de los nodos, Ganglia [4] es un ejemplo. A manera de introducción, [110] describe algunas de las técnicas para la gestión de fallos en éste tipo de sistemas.

2.2.7. Tolerancia a fallos

Normalmente los sistemas actuales tienen múltiples servidores, esto permite que, si un servidor falla, otro servidor asuma la carga. Este proceso debe ser transparente para el usuario. Si se emplea el concepto de tolerancia a fallos en un sentido amplio, los fallos en el sistema global se pueden encontrar en la comunicación, en los nodos, almacenamiento, etc. Un sistema tolerante a fallos debe continuar funcionando aún en una forma degradada. El punto más crítico de un sistema son los datos. Por lo tanto, la tolerancia a fallos está muy relacionada con la función de replicación porque el sistema crea copias de datos para proporcionar disponibilidad y ocultar estos fallos a los usuarios. Existen dos enfoques de tolerancia a fallos [84][97]: 1) Fallo como una excepción (*failure as exception*) y 2) fallo como norma (*failure as norm*), en éste segundo enfoque implica la replicación.

2.2.8. Replicación

El mecanismo de replicación de datos permite cumplir la tolerancia a fallos en los sistemas de ficheros distribuidos [31]. Éste mecanismo permite mantener una total disponibilidad de los datos. Básicamente la replicación consiste en realizar una cantidad de copias de datos en diferentes servidores del sistema. Cuando un cliente solicita datos, accede a éstos de manera transparente. Existen dos tipos de datos a ser considerados para replicación: metadatos y datos (o replicación de objeto de datos). La replicación de metadatos es la parte más importante para los sistemas de ficheros distribuido. Los sistemas actuales cuentan con un mecanimo para

garantizar la disponibilidad y capacidad de recuperación de los datos, tales como, servidor de respaldo de metadatos e instantáneas de metadatos con registros de transacción [97]. En cuanto a la replicación de datos existen varios enfoques dependiendo del propósito de las aplicaciones. Para mejorar la tolerancia a fallos, las réplicas se almacenan en diferentes servidores de acuerdo a la política de colocación (*placement policy*). Es decir, las réplicas pueden ser almacenadas en diferentes nodos [72], en diferentes armarios [90], y en diferentes ubicaciones geográficas [9]. Aunque estas políticas de colocación pueden repercutir en la inconsistencia de los datos, si un fallo se produce en cualquier parte del sistema, los datos se encuentran disponibles en cualquier ubicación con réplica.

2.2.9. Sincronización

Considerando la subsección anterior, la replicación de los datos se realiza en diferentes nodos del sistema. Por lo tanto, la sincronización de los datos se debe considerar. Si posteriormente a la replicación inicial se vuelve a escribir datos todas las copias deben actualizarse para ofrecer a los usuarios la versión más reciente de los datos. Tres enfoques principales de sincronización se describen en [35]:

1. **Método síncrono** (*synchronous*): Cualquier petición para modificar datos se bloquea hasta que todas las copias se actualizan. Éste método permite a los usuarios acceder a la última versión de los datos.

2. **Método asíncrono** (*asynchronous*): Permite las peticiones sobre datos modificados, incluso si las copias no se han actualizado. A diferencia del método síncrono, las solicitudes se pueden realizar en un tiempo razonable, pero el problema es que los usuarios pueden acceder a una copia no actualizada.

3. **Método semi-asíncrono** (*semi-asynchronous*): Éste método unifica

los dos anteriores. Es decir, las solicitudes se bloquean hasta que algunas copias (pero no todas) se actualizan.

2.2.10. Balanceo de carga

A nivel de servidor es la posibilidad que debe proporcionar el sistema para balancear la carga de trabajo automáticamente después de agregar o quitar servidores. Por ejemplo, para balancear la carga de los servidores que requiere dar soporte de acceso concurrente desde dos nodos en [10] se emplea un modo "*dual-primario*"(*dual-primary mode*). Lo cual consiste de un nodo primario y un nodo de réplica. De tal manera que el nodo de réplica pasa a ser un segundo nodo primario. Éste modo también es conocido como activo-activo. En cuánto al balanceo de carga a nivel de red, *channel bonding* [2] es un mecanismo que permite, además de balancear el tráfico de la red (*mode 0: Load balancing - round robin*), tolerar fallos en las interfaces de red e incrementar el ancho de banda global del sistema.

2.3. Cluster de computadores

Típicamente, los sistemas de ficheros distribuidos dan soporte de almacenamiento a los sistemas de cluster de computadores. "*Un cluster de computadores se define como un conjunto de computadores conectados por una red de área local (LAN) y organizado de tal manera que trabaje como un único sistema*".

El origen de los cluster de computadores se remonta a principios del año 1960 [79]. Los clusters de computadores pueden ser diseñados con diferentes características de tamaño, arquitecturas y tecnologías, para ofrecer las siguientes prestaciones:

1. *Velocidad de cálculo*: los nodos son conectados para resolver problemas de cálculo complejo.

2. *Capacidad de almacenamiento*: los nodos son conectados para almacenar grandes volúmenes de datos.

3. *Alta disponibilidad*: los nodos son conectados para evitar un sólo punto de fallo. Es decir, el sistema está activo en todo momento.

4. *Balanceo de carga*: los nodos son conectados para balancear la carga de trabajo. Es decir, el trabajo a ejecutar se distribuye entre todos los nodos.

Recientemente, el diseño de cluster de computadores puede incorporar cualquiera de las motivaciones anteriores para atender las demandas de aplicaciones específicas.

En una arquitectura de un sistema de cluster de computadores, se puede distinguir dos partes: la parte hardware y la parte software como se muestra en la Figura 2.4. Según la última lista TOP500 [93] de los supercomputadores, el 85 % se basan en la arquitectura cluster de computadores.

Como se ilustra en la Figura 2.4 la *parte hardware* (*Hardware stack*) corresponde al *Procesador*, *Almacenamiento* y a la *Red de interconexión*.

- **Procesador**: Actualmente la gran mejora de la tecnología permite que casi todos los sistemas cluster se construyan con computadores que incluyen procesadores multi-núcleo. En [93] se puede observar cómo es común encontrar sistemas de hasta 6, 8, 10 y 12 núcleos por socket. Ésto permite que las aplicaciones se ejecuten con mayor rapidez.

- **Almacenamiento**: En [103] se distingue que las dos mejores opciones de almacanemiento en cluster de computadores son: *Sistema de almacenamiento dedicado*[83] y *Sistema de almacenamiento consolidado* [108]. En los cluster de computadores de alto rendimiento se crea, por lo general, un subconjunto de nodos dedicados al almacenamiento de datos. De tal manera que, estos nodos realizan la E/S de datos

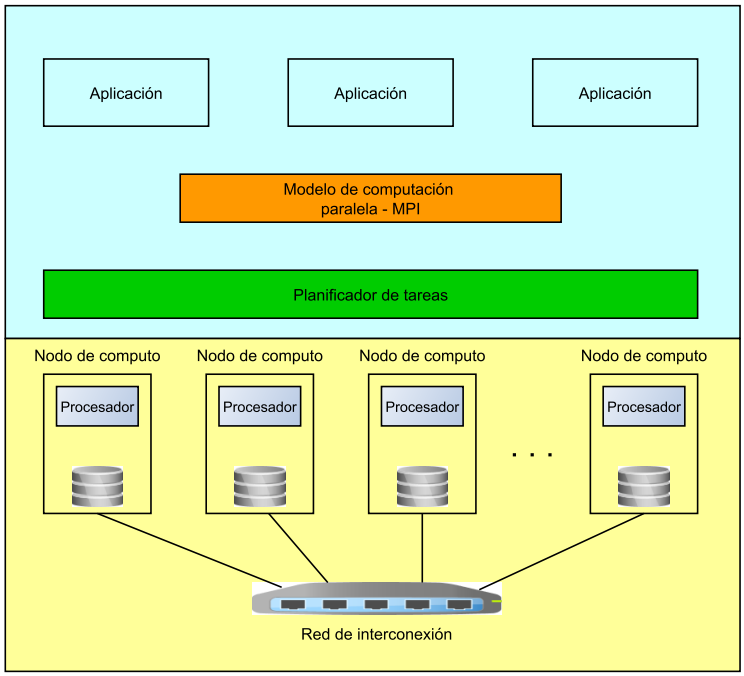

Figura 2.4: Arquitectura de cluster de computadores

mediante un sistema de ficheros distribuido. Algunas veces, los nodos de almacenamiento se conectan al resto a través de una red dedicada (por ejemplo en los *Data center*) para reducir la congestión de la red principal.

- **Red de interconexión**: La red de interconexión es otro componente crítico en cualquier cluster de computadores. Las diferentes aplicaciones tienden a tener diferente demanda del ancho de banda. Como ejemplo particular, podemos citar a los sistemas ficheros distribuidos. Recientemente, en la última estadística de [93] se muestra que un 35 % de los supercomputadores utilizan la familia Ethernet como medio de interconexión. De manera específica, Gigabit Ethernet se ha

convertido en la tecnología mayormente utilizada para la interconexión de computadores. Adicionalmente, permite obtener un gran ancho de banda con un presupuesto razonable. Para aprovechar mejor el rendimiento de Gigabit Ethernet, es necesario diseñar (en las aplicaciones) protocolos de comunicación que permitan hacer una distribución de datos de manera rápida y segura.

De la Figura 2.4, el *planificador de tareas*, el *modelo de computación paralela*, y las *aplicaciones* forman parte de la *parte de software* (*Software stack*) de un cluster de computadores.

- **Planificador de tareas**: También llamado *Gestor de recursos*, se encargan de gestionar una cola de ejecución, planificar la ejecución de las tareas y gestionar los recursos para minimizar costes y maximizar rendimiento de las aplicaciones. El funcionamiento básico consiste en que, los usuarios lanzan trabajos indicando requisitos de memoria, tiempo de procesador y espacio de disco. En cuanto los recursos necesarios estén disponibles, el gestor pone en ejecución el trabajo. Es posible consultar el estado de los trabajos en ejecución, en espera o terminados. Torque [92] y Sun Grid Engine (SGE) [68] son algunos de los gestores más comunes.

- **Modelo de computación paralela**: En arquitecturas cluster de computadores, se utiliza el estándar *Interfaz de Paso de Mensajes (MPI)* [45] como modelo de programación. MPI se ha convertido en el estándar para comunicar nodos que ejecutan los programas en sistemas de memoria distribuida. Éste modelo permite explotar eficazmente el paralelismo de un sistema de cluster de computadores. Los procesos MPI coordinan su progreso mediante el intercambio de mensajes a través de la memoria local y/o la red. Dichos mensajes se pueden enviar punto a punto o de manera colectiva.

- **Aplicaciones**: Las aplicaciones que se ejecutan en los cluster de computadores se pueden identificar como aplicaciones para *computación intensiva* y *uso intensivo de datos*. Inicialmente, la existencia de los cluster de computadores de alto rendimiento se desarrollaron para resolver problemas científicos complejos, tales como la bioinformática [109], la astrofísica [42], y la sismología [16]. Dada la alta complejidad de cálculo, estas aplicaciones requieren gran cantidad de uso de CPU, memoria y recursos de red. A causa de la reciente aparición de aplicaciones de uso intensivo de datos, cada vez más son los sistemas cluster que se utilizan para resolver problemas de gran volúmen de datos, pero en algunos casos, de baja complejidad computacional. Actualmente, Hadoop [50] se ha convertido en la principal solución para el tratamiento de aplicaciones intensivas de datos.

Otro tipo de sistemas cluster que requieren un sistema de comunicaciones eficiente son los cluster de almacenamiento. Un cluster de almacenamiento puede también funcionar de manera independiente para el alojamiento de datos de aplicaciones que no requieren importantes recursos de computación. Estos sistemas dan soporte de almacenamiento a través de los sistemas de ficheros distribuidos. Hay varios tipos de sistemas de almacenamiento para clústers que son ampliamente utilizados por la comunidad científica. Por citar algunos, en la lista del TOP500, gran cantidad de las supercomputadoras [85] más rápidas del mundo utilizan Lustre [26]. Ceph [100] ha sido diseñado como un sistema de almacenamiento de interfaz múltiple. Al igual que HDFS [90] estos sistemas utilizan la técnica de replicación de datos.

El enfoque de la presente investigación se centra en mejorar el interfaz de comunicaciones para sistemas de ficheros distribuidos. El contexto de la investigación se orienta a la distribución de los datos a través de los nodos incluyendo la replicación de los mismos. Por lo tanto, en la siguiente

sección (2.4) se describe, con mayor detalle, algunos sistemas de ficheros distribuidos.

2.3.1. Interconexión de tecnologías de red

El objetivo principal de las comunicaciones es facilitar el intercambio de datos entre sistemas informáticos. Las redes de comunicaciones han evolucionado debido a que los sistemas infórmaticos son más potentes y versátiles [77]. Las redes de interconexión se han diseñado para satisfacer las demandas operativas de diversas áreas de aplicación: Computación de alto rendimiento, almacenamiento E/S, la interconexión de sistemas cluster, etc. Las redes de área local (LAN) son las redes generalmente utilizadas para interconectar computadores autónomos distribuidos a poca distancia, la interconexión de los nodos de un cluster es un buen ejemplo [51]. Las tecnologías de red frecuentemente usadas para interconexión de computadores se muestra en la lista de Noviembre 2015 del TOP500. De los supercomputadores más potentes del mundo un 47,4 % utiliza la familia de interconexión Infiniband (entre las que se encuentran Infiniband FDR, Infiniband QDR e Infiniband FDR14), 23,8 % utilizan 10G Ethernet y el 12,4 % Gigabit Ethernet.

Los mensajes por red requieren un proceso previo de preparación antes de enviarse y de igual forma en la recepción deben ser procesados. Parte de estas funciones básicas corresponden a la *interfaz de red* (NIC) que residen en los nodos. La interfaz de red incluye hardware y software específico para mejorar las operaciones necesarias. La Figura 2.5 muestra un simple ejemplo donde el software de comunicación puede invocar el intercambio de mensajes con las cabeceras apropiadas. De tal manera que las aplicaciones de usuario no se preocupan del procesamiento de los mensajes, debido a que lo realizan las capas inferiores de manera automática. Además, es necesario que los nodos finales agreguen funciones adicionales, a nivel de software, para establecer comunicación entre los dispositivos de comunicación. Por

ejemplo: *a)* definir la *unidad máxima de transferencia* (MTU), típicamente definida a 1500 bytes y que puede escalar hasta 9000 bytes (conocidos como *Jumbo Frames*), *b)* incluir identificadores (ID) en los paquetes, de tal manera que se puedan diferenciar los paquetes que pertenecen a un mismo mensaje, *c)* incluir números de secuencia (*sequence numbers*) en los paquetes para reordenarlos si llegan fuera de orden al destino, así como también puede servir cómo mecanismo de detección de pérdida de paquetes a nivel de aplicación.

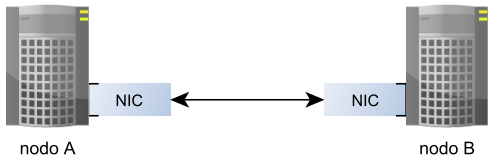

Figura 2.5: Conexión bidireccional de dos interfaces de red.

Por un lado, la tecnología Ethernet ha sido ampliamente utilizada para interconexión de nodos en una red local. Las primeras versiones que datan desde el año 1978 alcanza una velocidad de hasta 10Mbps, aunque en la actualidad existen versiones de 1Gbps y 10Gbps. Hoy día, muchos de los computadores incluyen Ethernet como interfaz de comunicaciónn estándar. Ethernet es una red de conmutación de paquetes que enruta los paquetes a una dirección destino. Ethernet puede ser una buena solución para implementar redes con un alto ancho de banda, baja latencia, y a un costo prácticamente reducido.

En comparación, Infiniband, aunque ofrece velocidades de transmisión de hasta 120Gbps y baja latencia, requiere un incremento considerable en el costo. Éste estándar se ha desarrollado desde octubre del año 2000 por un consorcio de compañías que pertenecen a *Infiniband Trade Association*. Infiniband se puede utilizar como una red para la comunicación entre procesos o como una red para el almacenamiento en servidores de E/S.

Proporciona dos mecanismos para dar soporte a la comunicación a nivel de usuario: funciones send/receive y el acceso directo a memoria remota (RDMA).

2.4. Comunicaciones en sistemas de ficheros distribuidos

En un sistema de ficheros distribuido, los recursos de almacenamiento y los clientes están dispersos en la red. Por lo tanto, el rendimiento de estos sistemas se puede ver afectado por las constantes transferencias de datos entre clientes y servidores, y, en algunos casos entre servidores. A estas transferencias se puede sumar el tráfico generado por las réplicas de datos para satisfacer la demanda de disponibilidad. La capacidad de la red utilizada es una de las métricas usadas para medir el rendimiento de estos sistemas. En cuanto a la red se refiere, para optimizar el rendimiento de los sistemas de ficheros distribuidos es necesario reducir el número de mensajes transmitidos por la red [84]. Otra forma de optimizar el rendimiento es la implementación de caché en clientes o servidores para evitar tráfico adicional [35].

Para establecer la comunicación, gran cantidad de los sistemas de ficheros distribuidos utilizan el método RPC [21, 22]. RPC es un paradigma para proporcionar comunicación entre los programas escritos en un lenguaje de alto nivel a través de una red. De tal manera que hacen que el sistema sea independiente de los sistemas operativos subyacentes. En la arquitectura del método RPC[1](Figura 2.6) se consideran dos protocolos de comunicación, TCP [13] y UDP [81]. TCP es el protocolo de transporte más común para intercambiar datos entre los nodos de un sistema de ficheros distribuido [97]. La ventaja principal de usar TCP se debe a que mantiene sus propios mecanismos de control de congestión para garantizar la entrega de datos.

[1]https://technet.microsoft.com/en-us/library/cc738291(v=ws.10).aspx

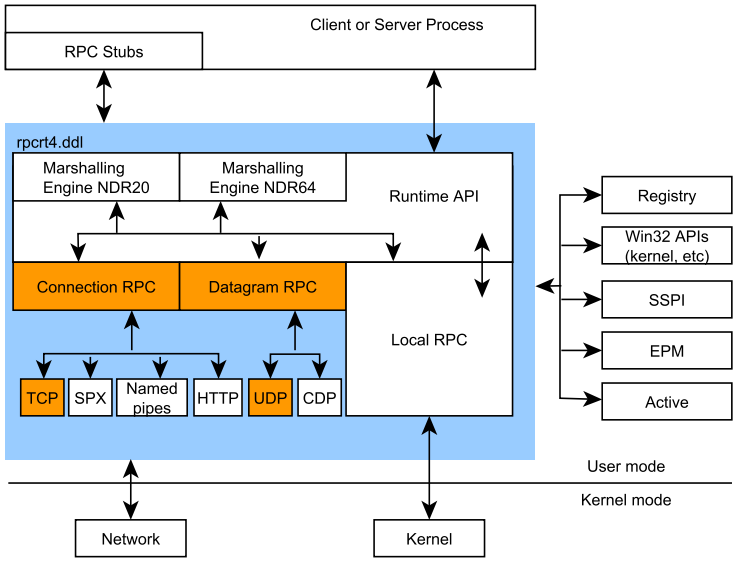

Figura 2.6: Arquitectura del método RPC

En [97] se presenta una taxonomia de algunos sistemas de ficheros distribuidos. Por propio interés de éste trabajo de investigación, en la Tabla 2.1 sólo se consideran algunos aspectos. Como se puede observar, estos sistemas utilizan TCP en el módulo de comunicaciones. Además, en cuanto a la tolerancia a fallos, HDFS, GFS y KFS consideran el enfoque *fallo como norma*, que incurre en realizar replicación de los datos en el lado del servidor.

Tabla 2.1: Algunos aspectos de sistemas de ficheros distribuidos

Sistema	HDFS	PVFS2	Lustre	GFS	Panasas	AFS	KFS
Comunicación	TCP	TCP	-	TCP	TCP	TCP	TCP
Replicación	Serv	No	Serv	Serv	Serv	No	Serv
Tolerancia a fallos	a	b	b	a	b	b	a

a=Fallo como norma b=Fallo como excepción

2.4.1. Hadoop Distributed File System (HDFS)

HDFS es una versión de código abierto de GFS de Yahoo. La principal diferencia de HDFS con otros sistemas es que es altamente tolerante a fallos. Un cluster HDFS consiste de un *NameNode* que es el nodo maestro que gestiona el espacio de nombres del sistema de ficheros y controla el acceso a los ficheros desde los clientes. El *NameNode* tiene un número de *DataNode*s que gestionan el almacenamiento. Los datos de usuario se almacenan en forma de archivos, generalmente un archivo es dividido en uno o más bloques, y estos bloques se almacenan en un conjunto de *DataNode*s.

Todas los protocolos de comunicación en HDFS se colocan en capas en la parte superior del protocolo TCP. El funcionamiento básicamente consiste en que, un cliente establece conexión con el *NameNode* a través de un puerto TCP configurable. El nodo *DataNode* se comunica con el *NameNode* usando el protocolo *DataNode*. Una abstracción RPC envuelve al protocolo *client* y al protocolo *DataNode*. El *NameNode* nunca inicia una conexión RPC, éste solo responde las peticiones RPC usadas por el *DataNode* o clientes.

2.4.2. Google File System (GFS)

El sistema de ficheros GFS se introduce en el año 2003 para satisfacer las crecientes demandas de requisitos de procesamiento de datos de Google. GFS es un sistema de archivos distribuido escalable que fue desarrollado especialmente para aplicaciones distribuidas con gran volumen de datos. Mientras GFS proporciona tolencia a fallos ofrece un alto rendimiento agregado a muchos clientes. Un cluster GFS consta de cientos o miles de nodos de almacenamiento que está construido con hardware de bajo costo.

Los clientes envían peticiones de datos al nodo maestro, que mantiene los metadatos de todos los trozos de los archivos. El nodo maestro devuelve al cliente los metadatos solicitados. Posteriormente, el cliente tiene acceso para conectar con el servidor que mantiene los datos para establecer transferencias. Todas las comunicaciones entre nodo maestro y los nodos

clientes se realizan usando TCP.

2.4.3. Andrew File System (AFS)

AFS fue desarrollado como parte del proyecto conocido como Andrew. Inicialmente se desarrolló para computadores que ejecutan sistemas operativos como BSD UNIX y Mac. AFS proporciona un acceso transparente a ficheros compartidos de forma remota. Éste sistema es implementado con dos componentes de software que existen en los procesos UNIX, llamados Vice y Venus. Actualmente, la investigación de AFS se realiza con el proyecto Open AFS. Las recientes versiones soportan plataformas como: Linux, Apple Mac OSX, Sun Solaris, entre otros.

AFS es implementado sobre TCP/IP. El protocolo RPC llamado *Rx*, que es implementado para AFS, es usado para comunicaciones entre dos nodos. El protocolo de comunicación es optimizado para redes de área amplia (WAN por sus siglas en inglés), por lo tanto no existen restricciones sobre la ubicación geográfica de los servidores y clientes que participan en la comunicación.

2.4.4. Lustre File System

Lustre [87] es un sistema de ficheros paralelo que gestiona los archivos almacenados internamente como objetos. Presenta a los clientes una semántica POSIX estándar y acceso concurrente de lectura y escritura para los objetos compartidos. Un sistema de archivos Lustre tiene cuatro unidades funcionales.

- Meta data server (MDS) para almacenar los metadatos.
- Object storage target (OST) para guardar los datos.
- Object storage server (OSS) para gestionar los OSTs.
- Clientes que acceden y utilizan los datos.

Los OSTs son dispositivos de bloques. Un MDS, OSS, y un OST pueden residir en el mismo nodo o en nodos diferentes. Lustre no administra directamente los OSTs, y delega esta responsabilidad en los OSSs para asegurar la escalabilidad para grandes clusters y supercomputadores.

2.4.5. Abierto File System (AbFS)

AbFS es un sistema de ficheros distribuido y paralelo con las siguientes características:

- Almacenamiento robusto de datos y metadatos.
- Elevado rendimiento en operaciones con metadatos y datos.
- Almacenamiento paralelo basado en striping.
- Aprovechamiento de las capas orientadas al almacenamiento en red (soporte iSCSI).
- Diseño más modular con mejor diferenciación entre las funciones de cliente y servidor.
- Mayor número de elementos en caché, tanto en clientes como servidores para mejora de las prestaciones.
- Monitorización y generación de eventos(*traps*) SNMP.

Una de las características de AbFS2 es su capacidad para poder utilizar diverso tipo de recursos de almacenamiento (tanto unidades SAN como DAS) y redes de comunicación en clientes y servidores. A continuación se muestran diversas configuraciones:

Almacenamiento sólo DAS

En el modelo convencional de un cluster, clientes y servidores se encuentran dentro de la misma red y pueden compartir sus recursos. Generalmente esta configuración permite que todos los clientes puedan comunicarse entre sí.

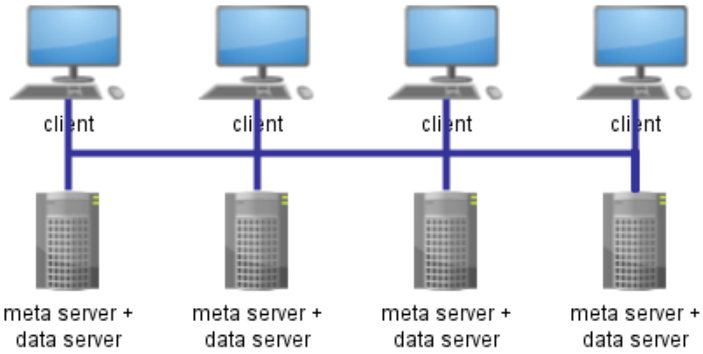

Figura 2.7: Configuración AbFS2 utilizando almacenamiento DAS

Este modelo puede plantear diversas alternativas, como mayor o menor número de servidores de metadatos y datos (pueden ser distintos), o bien incluir clientes que no tienen acceso directo entre si (ej. a través de un NAT real o una máquina virtual).

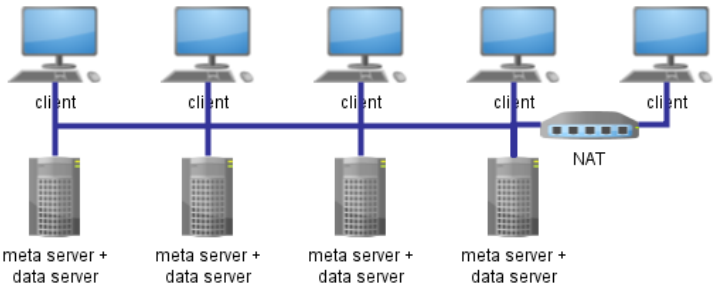

Figura 2.8: Configuración con acceso de un cliente a través de NAT

Este modelo también permite la redundancia de datos y metadatos entre los servidores lo que permite la tolerancia a fallos en los servidores.

Almacenamiento SAN compartido entre servidores

El almacenamiento SAN permite crear configuraciones más complejas

basadas en recursos de almacenamiento en red con unidades que ya incluyen cierta redundancia (unidades RAID). En este modelo hay dos posibilidades, si lo clientes pueden acceder directamente a las unidades de almacenamiento o no.

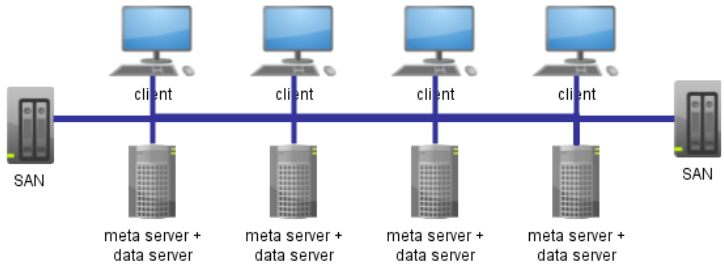

Figura 2.9: Configuración utilizando almacenamiento SAN

Puede haber topologías que interconexión que permitan mayor ancho de banda entre las SAN y los servidores (ej. Fibre Channel o Infiniband) a los que los clientes no puedan acceder de forma directa. En ese caso, los servidores ofrecen a los clientes el espacio de almacenamiento.

Modelo global de funcionamiento

AbFS2 utiliza un modelo distribuido y simétrico de servidores en el que cada nodo aporta ciertos recursos. Las principales ventajas de este sistema son:

- Repartir la carga de trabajo entre los servidores evitando cuellos de botella.
- Utilizar la capa iSCSI (o ISER) de transferencia de datos cuando sea posible.

A grandes rasgos, el modelo cliente-servidor también facilita la división de tareas de cada elemento así como simplifica los estados posibles de funcionamiento. La siguiente figura muestra la relación entre los diversos elementos de AbFS2.

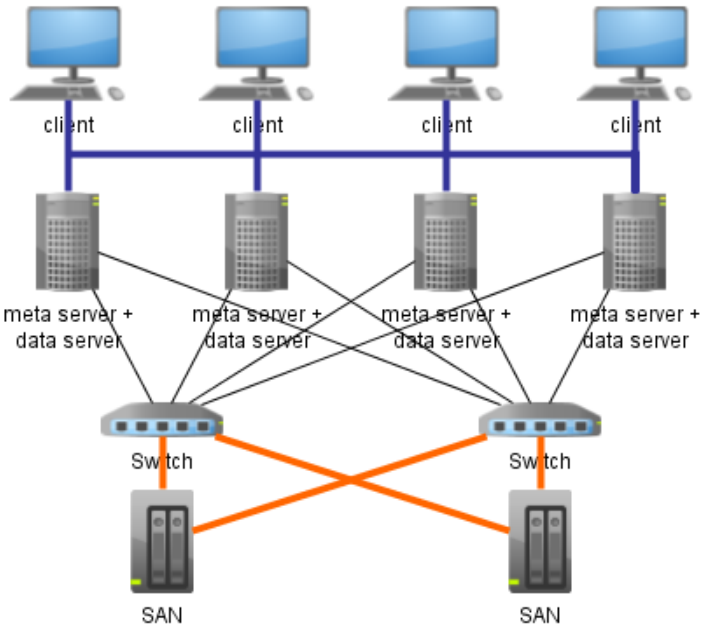

Figura 2.10: Utilizando almacenamiento SAN a través de los servidores

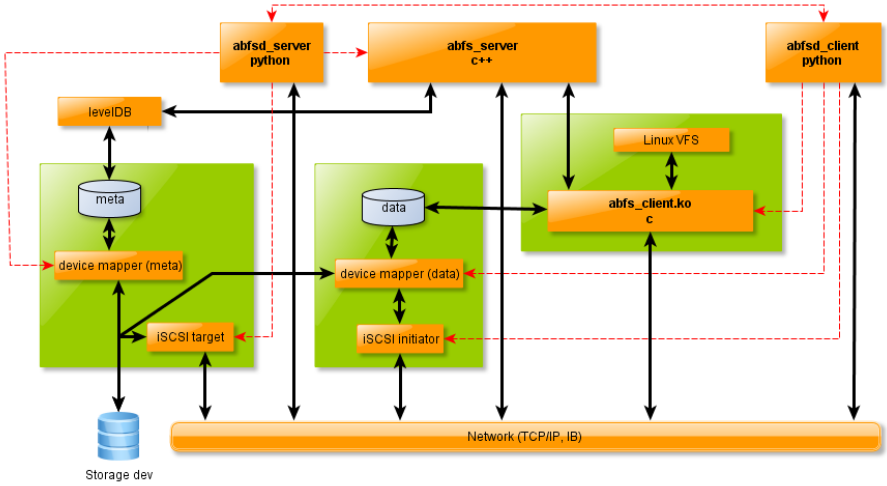

Figura 2.11: Relación entre los diversos elementos de AbFS2

Capítulo 3

Estudio de mejora de la interfaz de comunicaciones aplicado a sistemas ficheros en red.

En este capítulo se describen algunos aspectos importantes para conseguir un servicio de almacenamiento con buenas prestaciones para aplicaciones con alta necesidad de E/S. Por lo tanto se consideran aspectos de comunicación relacionados con PVFS2, el uso de técnicas de *channel bonding*, SDP en Infiniband y por último una evaluación del uso de técnicas de compresión de datos para conseguir la mejora de la interfaz de comunicación.

3.1. Introducción

Para conseguir un servicio de almacenamiento con buenas prestaciones para aplicaciones con elevados requerimientos de E/S es recomendable

utilizar un sistema de ficheros paralelo para el acceso a los discos de los nodos del cluster. Un sistema de ficheros paralelo permite que varios nodos (clientes) puedan acceder en paralelo a un mismo fichero y a múltiples ficheros. Dicho acceso paralelo se consigue distribuyendo el fichero entre los discos de los nodos. Este modelo distribuido implica que las comunicaciones afectan directamente en las prestaciones globales del sistema.

3.2. Almacenamiento paralelo a través de PVFS2

Parallel Virtual File System version 2 (PVFS2) [96] es un sistema de ficheros paralelo diseñado para Cluster de Linux, aunque es portable también a otras arquitecturas. El objetivo fundamental del diseño de PVFS2 es proporcionar escalabilidad y el acceso de alto rendimiento a los datos para las aplicaciones científicas. Incorpora optimizaciones y las características que puedan hacerlo útil también para otras cargas de trabajo. PVFS2 se ha diseñado para servir como instrumento para la investigación de E/S y como una herramienta de nivel de producción para el uso de la comunidad de cómputo de altas prestaciones.

PVFS2 es el sucesor original del sistema de ficheros PVFS [86] desarrollado en la Universidad de Clemson. El proyecto de PVFS comenzó a mediados del año 90, y desde entonces se ha venido convirtiendo en una herramienta estándar para la comunidad científica que aborda la implementación de aplicaciones de altas prestaciones.

PVFS2 utiliza una arquitectura cliente/servidor (Figura 3.1), tanto el demonio del servidor y las librerías del cliente residen totalmente en espacio de usuario. Puede existir cualquier número de servidores, y cada servidor puede proporcionar ya sea metadatos, datos de ficheros, o ambos. Estos datos se distribuyen de acuerdo a las reglas que seleciona el usuario. El esquema por defecto es la división de los datos (*stripe data*) de manera uniforme y similar al funcionamiento de discos RAID.

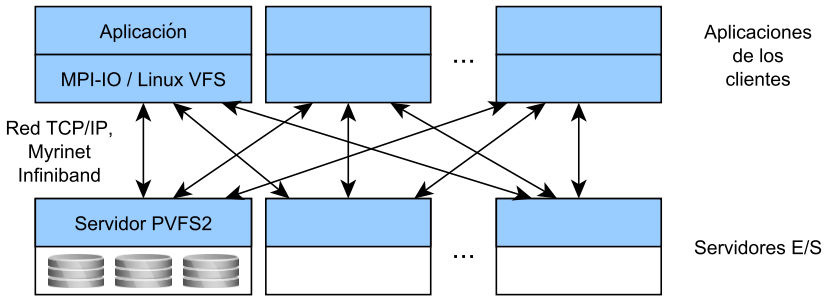

Figura 3.1: Arquitectura PVFS2.

Los componentes software de PVFS2 se muestran en la Figura 3.2. BMI, Flow y Trove son los elementos principales para la gestión de los datos en PVFS2.

- BMI: es la capa de abstracción de red que permite el funcionamiento de diferentes redes.
- Flow: se encarga de poner los datos en la red o gestionar para pasarlos al disco.
- Trove: almacena los datos en los discos físicos de los servidores.

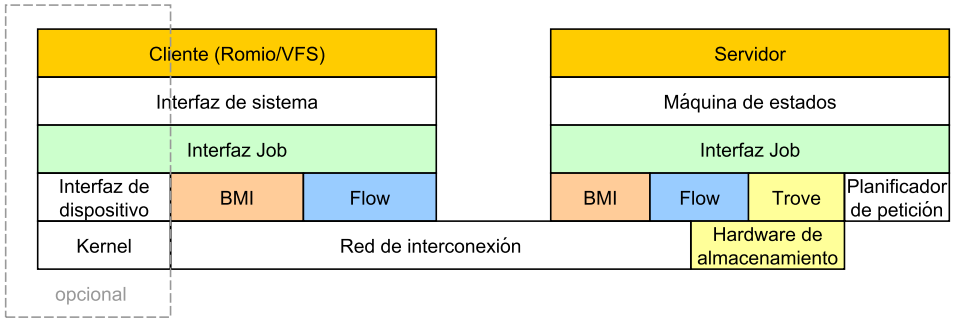

Figura 3.2: Componentes PVFS2.

Dado el interés de la temática de la tesis. A continuación de describe en más detalle las características principales del modulo de comunicación de PVFS2.

BMI se ha desarrollado para formar la base de comunicaciones de red en el sistema de ficheros PVFS2. El funcionamiento básico consiste en publicar una operación y posteriormente comprobar el estado de la misma hasta que haya terminado. BMI se ha implementado como una biblioteca a nivel de usuario que incluye diferentes módulos para redes usadas en computación de alto rendimiento incluyendo: TCP/IP, InfiniBand [28, 15], Myrinet [28, 23] y Portals [33].

El modelo de comunicaciones en BMI se encarga del paso de mensajes para proporcionar fiabilidad, orden en los mensajes y un control de flujo. Las operaciones de comunicación en BMI son no bloqueantes, de tal manera que el usuario primero envía un mensaje *post* y posteriormente realiza una comprobación del estado de la operación.

La arquitectura del módulo BMI 3.3 incluye la capa de *control de métodos* para dar soporte a los diferentes tipos de redes. Además, esta capa se encarga de hacer el vínculo entre el módulo BMI y los módulos superiores de PVFS2. El control de métodos mantiene una lista de direcciones de red para identificar a los nodos que pueden contener información específica de un protocolo en particular. De la Figura 3.3 cada método proporciona una entrega fiable y ordenada, así como un control de flujo para el protocolo que gestiona. Cada método se encarga de mantener el estado de las operaciones que se ejecutan a través de colas de operación doblemente enlazadas del kernel de Linux. El conjunto de operaciones son privadas para cada método.

Tanto la interfaz BMI como la interfaz de cada método incluye una serie de funciones a nivel de usuario. De tal manera que es posible el uso de la biblioteca para otros fines de comunicación.

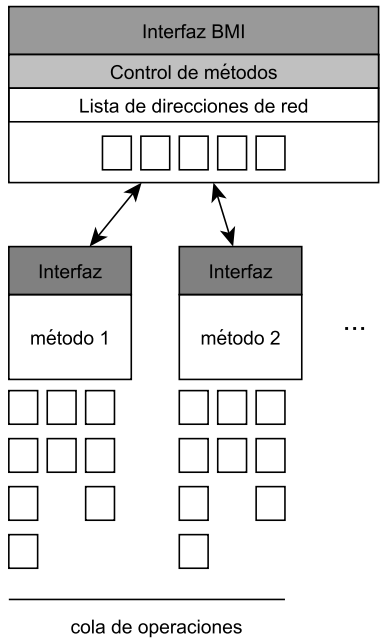

Figura 3.3: Arquitectura BMI.

3.3. Incremento de ancho de banda a través de Channel Bonding

En los sistemas basados en conmutadores es posible establecer varias comunicaciones simultáneas. Por lo tanto se pueden utilizar varias tarjetas de red en un mismo ordenador para ampliar el ancho de banda disponible. El uso de *channel bonding* permite escalar las comunicaciones con la mejora del ancho de banda a una relación prestaciones/coste mejor que otras redes como Myrinet, Infiniband, etc.

La idea principal de *channel bonding* es utilizar varias interfaces de red como si fueran una sola. *Bonding* está disponible por defecto en el código fuente del Kernel de Linux.

Considerando un ejemplo de configuración de *channel bonding*, la Figura 3.4 representa un nodo que tiene cuatro interfaces de red físicas. El caso de la interfaz eth0 y eth1 forman el enlace virtual bond0, mientras que la interfaz eth2 y eth3 forman el bond1, de tal manera que ambos enlaces virtuales pertenecen a subredes diferentes. El ancho de banda agregado teórico por cada enlace virtual podría ser de hasta 2 Gbps suponiendo que cada enlace físico tiene una máxima velocidad de transferencia de 1 Gbps.

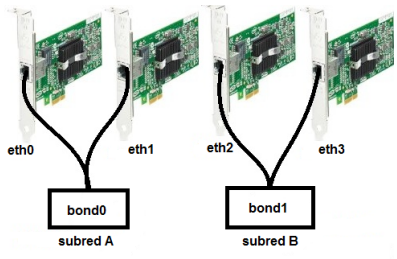

Figura 3.4: Configuración *channel bonding*.

Se han desarrollado algunas implementaciones de *channel bonding* con el fin de mejorar las prestaciones de la red. Entre las que cabe citar el proyecto CLIC [43], donde se demuestra la ganancia del ancho de banda proporcionado por una, dos y tres tarjetas de red, además se puede evaluar la reducción de la latencia cuando se utilizan dos y tres tarjetas. También demuestran que el uso de *channel bonding* es práctico para tamaños de mensajes elevados. El motivo de este comportamiento se debe a que los paquetes se agrupan, y las mismas rutinas de interrupciones en el receptor pueden atenderlas de una vez, reduciéndose los tiempos de procesamiento.

Otra implementación de *channel bonding* fue desarrollado en el proyecto ClusterNFS [95] para el sistema de ficheros NFS. La idea principal del diseño de ClusterNFS fue explotar completamente la infraestructura de hardware, utilizando múltiples interfaces de red.

La configuración de *channel bonding* permite establecer parámetros de unión de la interfaz. Por mencionar los más importantes para ofrecer un

equilibrio de carga y alta disponibilidad se consideran:

- *arp_interval*: específica en milisegundos la frecuencia en que ocurre la supervisión ARP, esto permite evitar la degradación del rendimiento de la red en caso de que falle un enlace de red.

- *downdelay*: especifica en milisegundos el tiempo de espera después de que falla un enlace, si se llega al valor de configuración la interfaz de red que falla se deshabilita.

- *miimon*: especifica en milisegundos con qué frecuencia se supervisa el enlace para comprobar que la interfaz está activa. Esto es importante si se requiere una alta disponibilidad.

- *mode*: permite especificar la política de unión de una interfaz al módulo *bonding* principal. Los principales modos de unión (aunque existen otros más):

    1. *balance-rr* o 0: establece una política round-robin para tolerancia a fallos y balanceo de carga. Las transmisiones son recibidas y enviadas secuencialmente en cada interfaz esclava vinculada comenzando por la primera disponible.

    2. *active-backup* o 1: establece una política de respaldo activa para tolerancia a fallos. Las transmisiones son recibidas y enviadas a través de la primera interfaz esclava disponible. Se utiliza otra interfaz esclava si falla la interfaz esclava activa.

- *primary*: Especifica el nombre de la interfaz, por ejemplo *eth0*, del dispositivo principal. En este caso *eth0* corresponde a la interfaz primaria de las interfaces de unión y permanece en ese rol mientras no falle.

3.4. Socket Direct Protocol en Infiniband

Se han realizado algunos estudios, tal como [75] para determinar en que medida es posible el incremento del ancho de banda en la transferencia de datos. De tal manera que, en [75] es posible reducir la sobrecarga de la comunicación, o permite el uso de las interfaces de red para obtener un alto ancho de banda y baja latencia como lo es Infiniband.

Infiniband permite mejorar el ancho de banda y reducir la latencia que suele ser necesario en la computación de alto rendimiento. Para mantener una compatibilidad con el software, en [20] se ha propuesto la solución *Socket Direct Protocol* (SDP) sobre Infiniband. SDP se ha propuesto con el fin de permitir que las aplicaciones basadas en sockets puedan aprovechar las características proporcionadas por la arquitectura Infiniband.

El diseño principal de SDP se basa en dos objetivos básicos: *a)* mantener la semántica tradicional de socket SOCK_STREAM como se aplica comúnmente en TCP/IP; *b)* soporte de *byte-streaming* a través de un protocolo de paso de mensajes, incluyendo las transferencias de datos de sin acceder al núcleo y de las transferencias de datos de copia cero (*zero-copy*). En la Figura 3.5 se muestra la ubicación de SDP respecto a la pila de red.

Los beneficios particulares de SDP se enfocan a: obtener un mejor rendimiento, disminuir la latencia y reducir la utilización de la CPU basados en los modos de transferencia de datos utilizados, la copia de búfer (*BCopy*), copia cero (*ZCopy*) tanto para la lectura como para la escritura.

El paquete de software *Open Fabrics Enterprise Distribution* (OFED) se compone de varios módulos de software que se incluyen en el Kernel de Linux entre los que se incluía SDP desde la versión 1.4.1, pero desde la versión 3.5 se ha eliminado el módulo del Kernel, así como las bibliotecas *libsdp* y *sdpnetstat* que daban soporte de funcionalidad en el espacio de usuario. Además, las implementaciones de SDP no daban soporte para redes Gigabit, siendo nuestro principal interés para intentar explotar al máximo el rendimiento de estas redes.

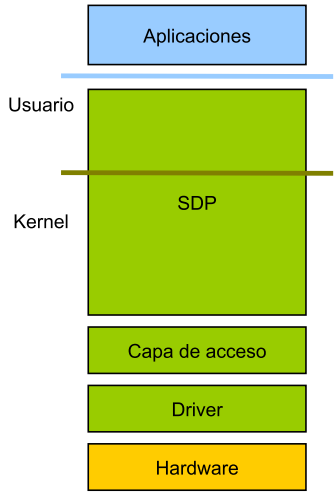

Figura 3.5: SDP en la pila de red.

3.5. Otras alternativas para mejora de la interfaz de comunicaciones

Otras propuestas para mejorar el rendimiento de comunicación en aplicaciones que requieren el intercambio de datos se basan en una combinación de optimizaciones de flujo de datos y de compresión. Por lo tanto, las aplicaciones no necesitan ser modificadas.

De forma particular, en [37] se propone una interfaz a nivel de usuario que mejora la velocidad de transferencia de datos en aplicaciones que requieren gran cantidad de información. Típicamente la información se almacena en los dispositivos de almacenamiento central y se distribuye entre los nodos, como se muestra en la Figura 3.6. Mientras tanto en la Figura 3.7 se presenta el modelo propuesto que consiste en que, un proceso en el servidor recibe la solicitud de otros nodos y clientes que almacenan los datos localmente, por lo tanto se procesan de forma local. En la propuesta, un elemento interesante es el uso de técnicas de compresión de datos para

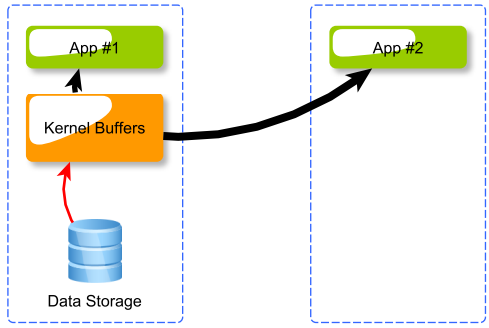

Figura 3.6: Flujo de datos habitual en la transferencia de archivos de datos.

mejorar las tasas de transferencia de datos. La compresión de flujos de datos se realiza desde el almacenamiento de datos, las interfaces de red y desde los búfer temporales. Por lo tanto, la compresión de datos puede reducir considerablemente el tráfico de datos por la red, lo que puede mejorar las prestaciones de la red y en consecuencia permite mejorar de forma indirecta la calidad de servicio (QoS) de la red.

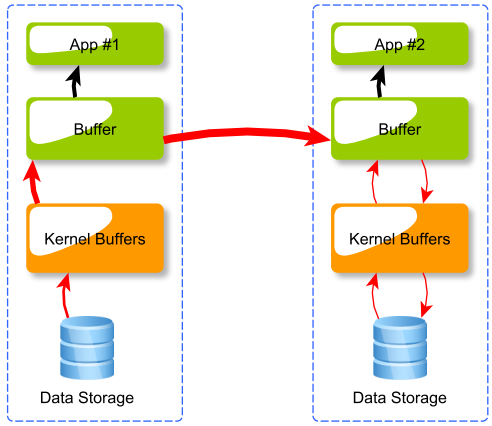

Figura 3.7: Flujo de datos de la propuesta en [37].

Para validar el uso de la compresión, primero se determina la relación

entre el factor de compresión $C < 1$, la velocidad de descompresión U_{CS} y la aceleración de la transferencia de datos S_{DT}. Si se tiene un tamaño de bloque B y una transferencia de datos de disco D_S, el tiempo de transferencia del bloque corresponde a T_B.

$$T_B = \frac{B}{D_S} \tag{3.1}$$

Además, si se utiliza la compresión, el tiempo para transferir el bloque sería T_{BC}, por lo tanto, sólo se necesita transferir una parte de B, pero se tiene que añadir un tiempo adicional para descomprimir los datos.

$$T_{BC} = \frac{B \cdot C}{D_S} + \frac{B}{U_{CS}} \tag{3.2}$$

Se define la velocidad de descompresión relacionada con la transferencia de datos $U_{CSX} = U_{CS}/D_S$, por lo tanto, la aceleración de transferencia de datos cuando se utiliza compresión es $S_{DT} = T_B/T_{BC}$, de tal forma que:

$$S_{DT} = \frac{1}{C + \frac{1}{U_{CSX}}} \tag{3.3}$$

Esta ecuación es similar a la ley de Amdahl, pero la diferencia es que el segundo factor depende del tamaño de todo el bloque B. La Figura 3.8 muestra el S_{DT} obtenidos con diferentes factores de compresión C y la velocidad de descompresión U_{CSX} relacionada con la velocidad de datos.

En esta figura, los valores S_{DT} entre 0 y 1 no representan aumento de velocidad, por lo que no es útil utilizar la compresión. El área punteada representa una aceleración entre 1 y 2, y así sucesivamente. En una visión general, se puede observar que el ancho de banda del disco limita la transferencia de datos, mientras que una descompresión rápida mejora las transferencias de datos y reduce las asignaciones en disco.

Ahora, en este trabajo se usa SDP en Infiniband para aprovechar un canal rápido reduciendo el cuello de botella que se puede encontrar en el

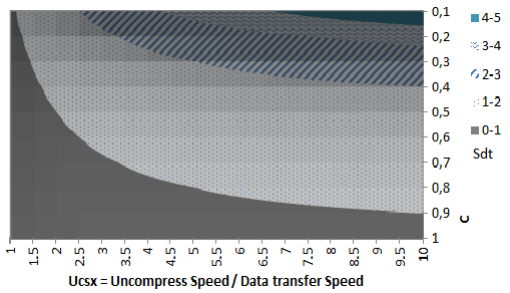

Figura 3.8: Aceleración en la transferencia de datos localmente.

disco, por lo tanto, también se evalúa la penalización de la transmisión de datos con y sin compresión.

Siendo así, se considera I_{BX} = (Velocidad de transferencia de datos Infiniband / transferencia de datos de disco) = I_{BS}/D_S, además se estudia la penalización debido al disco. El tiempo de transmisión a través de Infiniband y sin transferencia de disco sería:

$$T_{IB} = \frac{B}{I_{BX} \cdot D_S} \tag{3.4}$$

Y, con acceso a disco, entonces T_{IBD}:

$$T_{IBD} = \frac{B}{D_S} + \frac{B}{I_{BX} \cdot D_S} \tag{3.5}$$

En este caso, consiste en una penalización $P_D/T_{IB}/T_{IBD}$, donde $P < 1$:

$$P_D = \frac{1}{1 + I_{BX}} \tag{3.6}$$

Y, usando la compresión, entonces T_{IBDC}:

$$T_{IBDC} = \frac{B \cdot C}{D_S} + \frac{B \cdot C}{I_{BX} \cdot D_S} + \frac{B}{U_{CS}} \tag{3.7}$$

Y, $U_{CS} = U_{CSX} D_S$. Penalización $P_{DC} = T_{IB}/T_{IBDC}$, donde $P < 1$:

$$P_{DC} = \frac{U_{CSX}}{(C \cdot I_{BX} + C)\, U_{CSX} + I_{BX}} \tag{3.8}$$

Un valor P bajo significa un rendimiento pobre. En la Figura 3.9 se puede observar el impacto del uso de diferentes factores de compresión C y velocidad de descompresión de factor U_{CSX}.

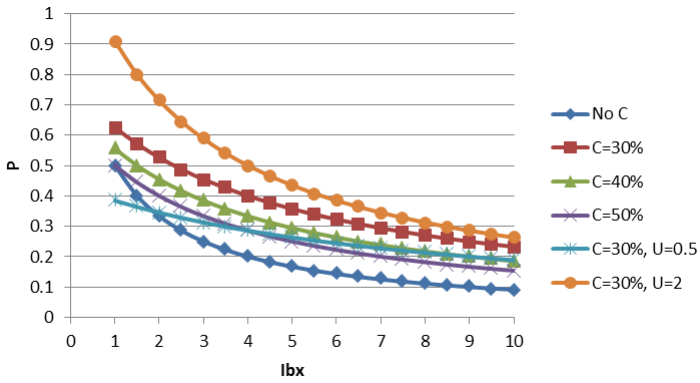

Figura 3.9: Penalización obtenida en la transferencia de datos.

Si la velocidad de descompresión es tan rápida como la velocidad Infiniband, siempre se obtendrán mejores resultados con el uso de la compresión, e incluso si la velocidad de descompresión alcanza el 50 % de velocidad de Infiniband. En casos reales, se puede obtener esta velocidad de descompresión que es incluso más rápido que la transferencia de datos con Infiniband (depende de la velocidad de datos Infiniband y del protocolo utilizado).

En general, es aconsejable el uso de la compresión de cara a reducir el tráfico de datos por la red de tal manera que apoye a la mejora de prestaciones de la red para satisfacer las demandas de QoS.

En [37] se analizan diferentes operaciones para validar la importancia del uso de la compresión. Los factores de evaluación principales comprenden: el factor de compresión, la velocidad de compresión y la velocidad de

descompresión. Los resultados obtenidos depende del tipo de datos, de modo que se han evaluado operaciones con 6 tipos de datos diferentes que se utilizan en algunas aplicaciones HPC (HBSM, DLSPI, NOAA, GENOME, XML, IEEE doble precisión):

- HBSM: Harwell Boeing Sparse Matrix. Estos ficheros se utilzian para almacenar matrices dispersas de gran tamaño. El fichero de ejemplo es una matriz de 44609 por 44609 con 1029655 elementos, de un chasis de automóvil [1] del software MCS/NASTRAN y los programas de ingeniería estructural de Boeing ATLAS [7].

- DLSPI: (Database for Large-scale Peptide Identification). Estos datos se han obtenido de un espectrómetro de masas [3]. El fichero contiene unan base de datos microbiológica que contiene codificado una secuencia de proteínas de 2.65 millones de elementos.

- NOAA: Estos ficheros corresponden a:ISH(Integrated Surface Hourly) / ISD(Integrated Surface Database) del National Climatic Data Center (NOAA) de [8].

- GENOME: Estos datos se han obtenido de proyecto genoma humano: Dec. 2013 (GRCh38/hg38) (hg38, GRCh38 Genome Reference Consortium Human Reference 38 (GCA_000001405.2)), obtenidos de [6].

- XML: Grandes volúmenes de datos con formato de intercambio eXtensible Markup Language obtenidos de benchmarks XML [11].

- IEEE: Estos datos contienen números aleatorios de punto flotante almacenados en formato IEEE754 de doble precisión.

De forma particular se ha utilizado el algoritmo LZ4 debido a que ofrece mejor rendimiento para el contexto de evaluación. La Figura 3.10 muestra el factor de compresión obtenido para diferentes tipos de datos que dependen del tamaño de bloque de datos. El tamaño de bloque se establece

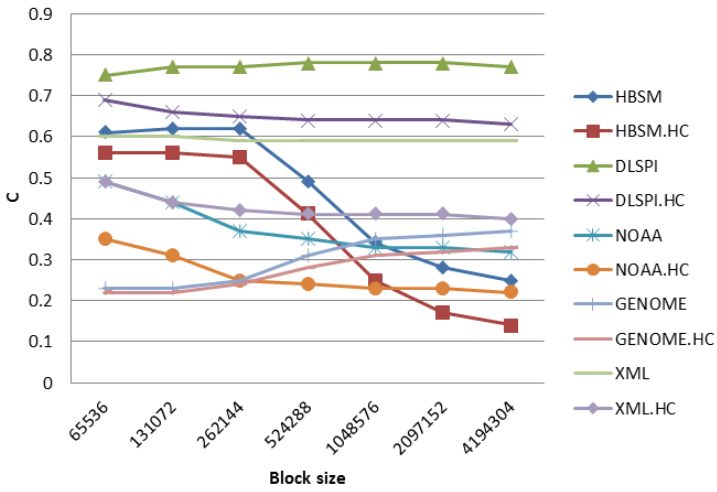

Figura 3.10: Factor de compresión con diferentes tipos de datos.

desde 64KB hasta 4MB, de forma general un tamaño de bloque grande ofrece mejor relación de compresión. En las pruebas realizadas, se evalúa la compresión normal y compresión elevada (HC). Además, HC ofrece una mejor relación de compresión pero la velocidad de compresión se reduce drásticamente, pero una menor relación de compresión ofrece una mejor velocidad de descompresión, tal y como se muestra en la Figura 3.11. En general, la relación de compresión obtenida es inferior a 0,5 y la velocidad de descompresión es mejor que 400-500MB/s.

También se ha evaluado la tasa de transferencia de datos sobre Infiniband. La Figura 3.12 corresponde a la velocidad de transferencia con diferentes tipos de datos cuando los datos no son almacenados en la caché del servidor. Por lo tanto la velocidad de transferencia del disco puede limitar el rendimiento global. Cuando se utiliza la compresión de datos, el ancho de banda de transferencia de datos puede mejorar hasta 220 %, mientras que la no compresión obtiene 24MB/s.

Por otro lado, la Figura 3.13 muestra cuando el servidor usa la caché para

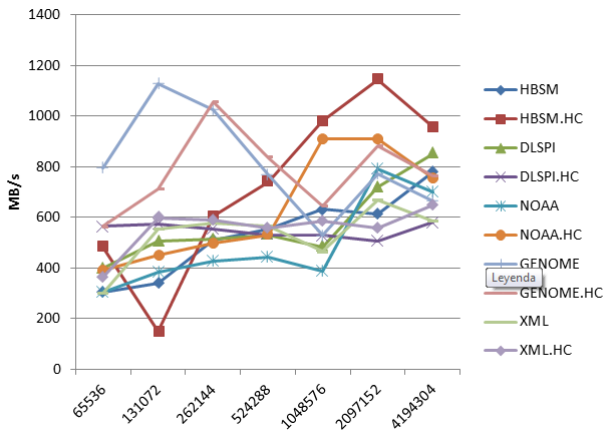

Figura 3.11: Factor de descompresión con diferentes tipos de datos.

almacenar los datos temporalmente. Infiniband incrementa la velocidad de transferencia de datos hasta 169MB/S, pero cuando se usa la compresión, la velocidad de transferencia puede mejorar aún más hasta 335MB/s, aproximadamente 200 %.

Además de las alternativas de mejora mencionadas anteriormente se ha estudiado la técnica multicast como un mecanismo que puede beneficiarse de los recursos de la interfaz de comunicaciones aplicado a sistemas de ficheros en red. Dada las ventajas que aporta multicast en el sentido de la disminución del tráfico generado en las comunicaciones, en el siguiente capítulo se estudian todos los aspectos para hacer abordar el objetivo principal de la presente tesis.

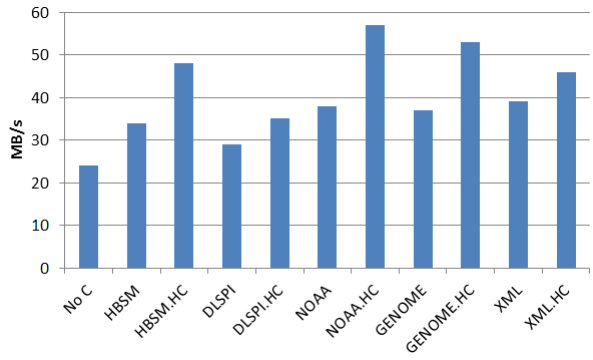

Figura 3.12: Velocidad de transferencia de datos sin caché en servidor.

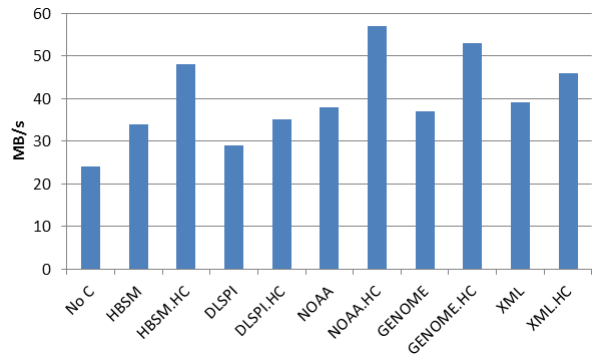

Figura 3.13: Velocidad de transferencia de datos con caché en servidor.

Capítulo 4

Comunicaciones Multicast

Este capítulo trata en mayor detalle las comunicaciones multicast con idea de conocer las ventajas generales, y a su vez hacer una comparativa con las comunicaciones unicast que se usan principalmente en los sistemas de almacenamiento en red. En las últimas secciones del capítulo se describen algunos aspectos a considerar para el desarrollo de sistemas de comunicación basados en multicast.

4.1. Introducción

El intercambio de datos entre computadores es el objetivo principal de las comunicaciones en el campo de la informática. Inicialmente la comunicación implementada permitía sólo la trasferencia entre un único fuente y un único destino (comunicación uno-a-uno). La creciente evolución de las tecnologías de comunicación ha permitido alcanzar un progreso notable permitiendo la comunicación de un fuente con múltiples destinos simultáneamente (comunicaciones uno-a-muchos y muchos-a-muchos). A esta forma de comunicación se le conoce como comunicación de grupo o alternativamente también conocida como multicast. La utilización y aprovechamiento de estas comunicaciones sigue siendo un tema de interés

para los investigadores.

En la actualidad, la comunicación en una red de ordenadores es esencial para las aplicaciones, en especial para las aplicaciones distribuidas, tal como los sistemas de ficheros distribuidos. La comunicación multicast ha estado en el centro de interés en el área de Internet y ha contribuido en algunos éxitos importantes (ver, por ejemplo [101]).

En infraestructura de red con IP (*Internet Protocol*), se usa IP multicast como método para comunicaciones uno-a-muchos y muchos-a-muchos. Los datos se envían sólo una vez, aunque lo reciban un número elevado de destinos. Los conmutadores de la red se encargan de replicar los datos (paquetes) por las salidas del conmutador necesarias para que alcancen todos los nodos del grupo de destinos. IP multicast utiliza una dirección de grupo IP multicast. Lo utiliza tanto el fuente como los destinos. [82].

Una videoconferencia es un claro ejemplo para mostrar las ventajas que puede ofrecer la comunicación multicast sobre la comunicación uno-a-uno. Una de las principales ventajas es la eficiencia alcanzada para llegar a todos los miembros de un grupo al mismo tiempo [101].

4.2. Tipos de comunicación

Para alcanzar los objetivos planteados en éste trabajo ha sido necesario analizar los tipos de comunicación existentes. El modelo TCP/IP ofrece diversos tipos de comunicación, dónde, en algunos casos permite la fiabilidad en la entrega de los datos.

El enfoque principal de desarrollar un sistema de comunicaciones en esta investigación se deriva de hacer envíos simultáneos a diferentes receptores, de tal modo que, con esta propuesta se logre disminuir el tráfico en la red. En este contexto se pueden diferenciar diversos tipos de comunicación en función del número de emisores y receptores involucrados en la comunicación.

Básicamente podemos encontrar los siguientes tipos de comunicación:

- Unicast.
- Multicast.
- Broadcast.

Aunque en la literatura existen algunos otros tipos de comunicación, en este trabajo sólo nos enfocamos en los tipos anteriormente mencionados y que se describen en las siguientes subsecciones.

4.2.1. Comunicación Unicast

La comunicación unicast es una comunicación uno-a-uno o punto-a-punto, por lo tanto, se puede utilizar para la comunicación en aplicaciones cliente/servidor en las que hay exactamente un emisor y un receptor. Estas comunicaciones están principalmente dirigidas por el emisor de datos identificando la dirección IP del receptor. De tal modo que, los paquetes unicast usan la dirección del dispositivo de destino para la entrega de los datos, además estos datos pueden pasar por una interconexión de redes.
Este tipo de comunicación es la forma más común y eficiente de la comunicación con un único nodo. La comunicación no involucra otros nodos que no se han identificado dentro del paquete enviado por el emisor [76]. La Figura 4.1 muestra un ejemplo donde *nodo1* establece conexión con su dirección IP como origen para iniciar envío de datos a la dirección IP de *nodo2* como el destino.

En este tipo de comunicaciones se requieren transmisiones de control (reconocimientos positivos o ACK) desde el receptor hacia el emisor para comprobar la entrega de los datos. La comunicación unicast se utiliza únicamente para la comunicación entre dos nodos. Por lo tanto, si se pretende usar unicast para dar soporte de comunicación multicast

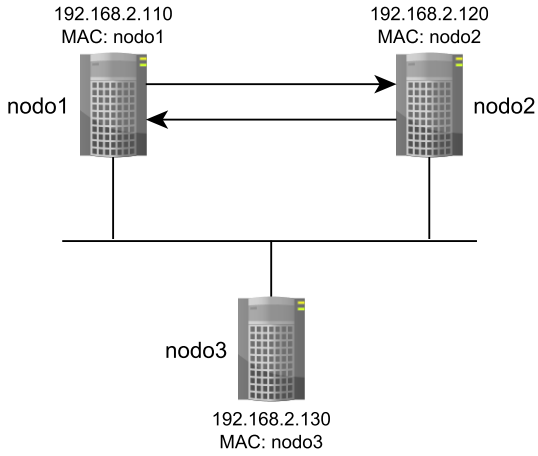

Figura 4.1: Comunicación unicast

será necesario establecer un total de canales de comunicación del orden de $\frac{n(n-1)}{2}$ para un grupo de tamaño n[101].

4.2.2. Comunicación Multicast

Mientras que unicast permite el envío de datos entre un emisor y un receptor, las comunicaciones multicast permiten el envío de datos desde un emisor a muchos receptores (uno-a-muchos), o desde muchos emisores a muchos receptores (muchos-a-muchos) si la gestión de los grupos se realiza de forma adecuada. Los envíos a muchos receptores se realiza de forma simultánea, de tal manera que se puede reducir el número de canales de comunicación que se ha descrito en la subsección 4.2.1. Para identificar los grupos multicast, se utiliza una clase de dirección IP específica que se describe en la sección 4.4. Por ejemplo, en la Figura 4.2 el *nodo1* envía datos a los nodos *nodo2* y *nodo3* que se han asociado a la dirección multicast 239.5.5.0. En el ejemplo, el *nodo1* sólo envía un mensaje a la dirección multicast sin la necesidad de establecer comunicación con cada

nodo receptor. Además el *nodo4* no recibe el tráfico porque no se ha asociado a la dirección multicast.

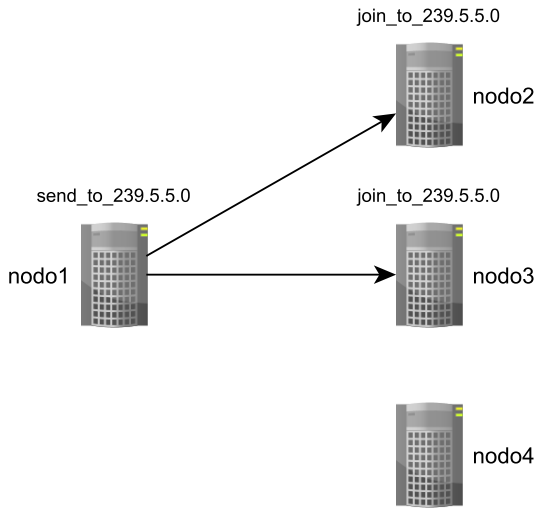

Figura 4.2: Comunicación multicast

Para habilitar comunicaciones multicast, en la capa de red se utiliza el protocolo TCP/IP. De forma más específica esto implica implementar protocolos una capa por encima del protocolo UDP. La desventaja de UDP es que no garantiza la entrega de los datos. Por lo tanto, es necesario agregar mecanismos de detección de pérdida y retransmisión de datos. En la actualidad los conmutadores que conectan los nodos de una red tienen soporte para administrar los grupos multicast. Estos grupos multicast puede crecer o disminuir dinámicamente. Los nodo se unen (*join*) a un grupo multicast si están interesados en recibir tráfico dirigido a la dirección multicast de dicho grupo y lo deja (*leave*) cuando dejan de estar interesados. El *Internet Group Management Protocol* (IGMP) permite llevar a cabo la comunicación entre los nodos y los conmutadores de la red.

4.2.3. Comunicación broadcast

La comunicación broadcast es comparable con la comunicación multicast ya que existe un solo emisor. En cambio, con broadcast un solo mensaje se entrega a todos los potenciales receptores (por ejemplo, en una subred), mientras que con multicast solo lo reciben los nodos interesados en el tráfico. La manera más común de lograr la comunicación broadcast es utilizar una dirección de difusión especial, en la cual se indica al mecanismo de comunicación que el mensaje debe ser entregado a todos los nodos de la subred. En la Figura 4.3 se muestra un ejemplo de comunicación broadcast donde el emisor envía un único mensaje a todos los nodos de la misma subred que el emisor. La dirección IP 255.255.255.255 es comúnmente utilizada para la comunicación broadcast. Al enviar un mensaje broadcast, el emisor no necesita conocer el número de receptores.

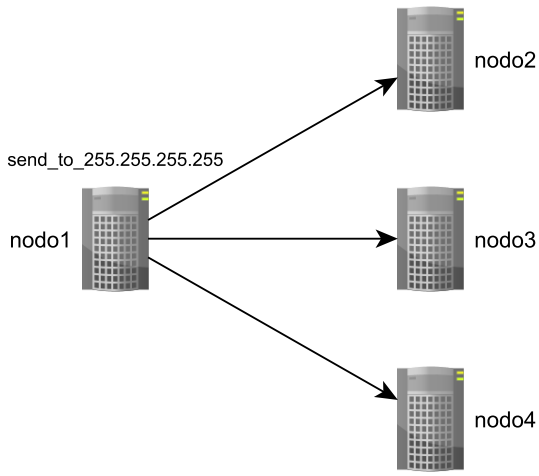

Figura 4.3: Comunicación multicast

Broadcast es menos eficiente porque ocupa más infraestructura de la red al enviarlo a todos los nodos quieran o no quieran los datos, y también porque el emisor no ha identificado el conjunto de receptores. Además, los

envíos broadcast puede ser ineficiente porque los nodos reciben el mensaje en la capa de red, esto implica hacer una interrupción para procesar el mensaje y pasarse a la capa superior, incluso puede resultar que ninguno de los nodos de la subred estén interesados en el mensaje. Un claro ejemplo del uso de broadcast se puede encontrar en el protocolo de resolución de direcciones o *Address Resolution Protocol* (ARP) [80].

4.3. Multicast vs Unicast

Multicast se refiere a la entrega de datos de forma simultánea a un grupo de nodos receptores como destino, desde un emisor como origen. Por el contrario, en unicast un emisor se comunica con un único nodo receptor de destino. De tal manera que con unicast, si un emisor necesita comunicarse con 3 nodos receptores, tiene que establecer 3 canales de comunicación. En cambio, multicast permite crear un sólo canal de comunicación para los 3 nodos receptores. Si se requiere implementar aplicaciones basadas en unicast el propio protocolo TCP, al ser un protocolo orientado a la conexión, permite mantener un control de flujo y de congestión para la entrega fiable de los datos, además mantiene un orden en la entrega de estos. Aunque, también es posible hacer implementaciones unicast basadas en UDP, es responsabilidad del programador en este caso dar soporte de fiabilidad en la entrega de datos según las características de las aplicaciones. UDP permite aprovechar las comunicaciones multicast, que, por lo contrario a unicast, proporcionan soporte de envíos simultáneos. Las desventajas más destacables si se usa UDP en implementaciones multicast son las siguientes:

- *Pérdida de paquetes*: Se pueden perder paquetes debido a que se trata de un protocolo que no mantiene una conexión con el receptor, Por lo tanto es necesario desarrollar aplicaciones multicast con entrega fiable de datos. En la actualidad, el tema “Reliable multicast”sigue siendo un área de interés para la investigación.

- *Congestión*: La falta del uso de ventanas de envío como en TCP y de mecanismos para ajustar las tasas de envío puede dar lugar a la congestión de la red. Como consecuencia, estos aspectos también deben ser considerados siguiendo las necesidades de transmisión de las aplicaciones.

Por otra parte, los sistemas distribuidos pueden aprovechar ciertas características destacablemente importantes de multicast, por mencionar algunas:

- *Mejora la eficiencia*: permite mejor uso ancho de banda de la red disponible y reduce considerablemente la carga de los dispositivos de red y nodos fuente y destino.

- *Optimiza el rendimiento*: permite eliminar la redundancia del tráfico al disminuir los canales para el envío de datos.

4.4. IP Multicast

El direccionamiento de tráfico multicast se realiza mediante una IP especial. Mientras que para la comunicación punto a punto se utilizan direcciones IP de la clase A, B y C. En cambio, para establecer comunicaciones multicast se utiliza la dirección IP de la clase D. En la Figura 4.4 se describen las clases de direcciones IP que componen el conjunto de direcciones de la pila TCP/IP.

La clase de una dirección IP se determina a partir de los bits del orden superior. De la Figura 4.4, en la clase A, B y C la sección *red* corresponde a la identificación de la red. El rango de direcciones de red de la clase A comprende de 1.0.0.0 hasta 127.225.255.255, de la clase B de 128.0.0.0 hasta 191.255.255.255, de la clase C de 192.0.0.0 hasta 223.255.255.255. Para estas clases la sección *nodo* es para administrar las subredes y los nodos finales.

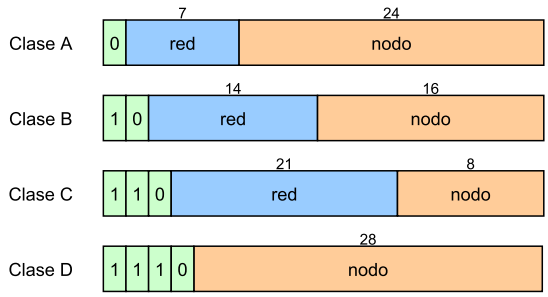

Figura 4.4: Tipos de direcciones IP

Tabla 4.1: Asignación de direcciones IP

Dirección IP	Asignación
224.0.0.1	Todos los sistemas en la subred
224.0.0.2	Todos los enrutadores en la subred
224.0.0.4	Todos los enrutadores DVMRP
224.0.0.5	Todos los enrutadores OSPF
224.0.0.9	Todos los enrutadores RIP2
224.0.0.13	Todos los enrutadores PIM
224.0.0.15	Todos los enrutadores CBT

El rango de direcciones multicast comprende de 224.0.0.0 hasta 239.255.255.255. El rango de direcciones de 224.0.0.0 hasta 224.0.0.255 está reservado para asignaciones permanentes de diferentes aplicaciones, en las que se incluyen los protocolos de ruteo.

Algunas direcciones multicast actualmente asignadas se mencionan en la Tabla 4.1.

El alcance de los paquetes IP viene dado por el campo TTL (*Time to Live*) de la cabecera del paquete. TTL es un mecanismo que contabiliza

los *saltos* y determina el alcance de la red que el paquete puede atravesar. Inicialmente se define un valor TTL en la aplicación. Por cada *salto* que el paquete realiza el valor TTL se decrementa una unidad, causando la pérdida del paquete cuando el valor TTL se establece a 0 ya que se ha alcanzado el número máximo de saltos para llegar al destino. Típicamente, el valor asignado para comunicaciones multicast en una red local, se establece un TTL de 1.

La unión de un nodo a un grupo multicast se inicia desde el receptor utilizando el protocolo IGMP. Actualmente existen tres versiones de este protocolo (IGMPv1 [34], IGMPv2 [40] e IGMPv3 [56]) que permite la gestión de los grupos multicast desde los enrutadores o desde los conmutadores con soporte IGMP Snooping. Por ejemplo, en la Figura 4.5, el *nodo B* envía al conmutador un mensaje join con la dirección multicast del grupo al que desea asociarse, en este caso la dirección 239.5.5.0. Cuando el conmutador recibe esta petición registra el puerto al que está conectado el nodo B en una tabla de entradas multicast. Una vez realizado el registro de éste puerto, el *nodo B* puede recibir el tráfico multicast que el *nodo A* envía a la dirección multicast 239.5.5.0. Para abandonar el grupo multicast, el *nodo B* únicamente tiene que enviar un mensaje *leave* al conmutador en el que se indica. Cuando el conmutador registra el puerto del *nodo B* asociado a la dirección 239.5.5.0 inicializa un temporizador configurable, que representa el tiempo que tiene el *nodo B* para permanecer dentro del grupo. De esto se deduce que, a nivel de aplicación, es necesario mantener un mecanismo que permita que el *nodo B* se mantenga permanentemente dentro del grupo multicast de interés.

El mapeo de las direcciones multicast en un entorno IPv4 a nivel de red se realiza sobre las direcciones físicas que corresponde al tipo de red que se utiliza. Por ejemplo, en el caso de direcciones unicast, a nivel de red se obtiene la dirección física asociada a la dirección IP mediante el uso del protocolo ARP. Mientras que, para el caso específico de direcciones físicas

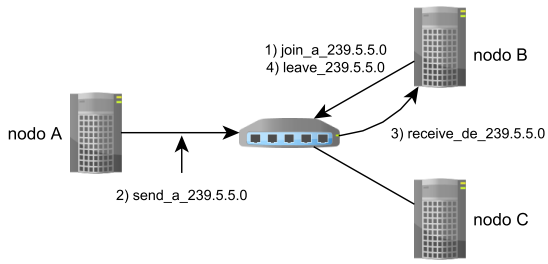

Figura 4.5: Funcionamiento básico del protocolo IGMP (se indica el orden de las acciones)

Ethernet, en [34] se definen procedimientos para obtener las direcciones físicas de direcciones IP multicast. Dado que las redes Ethernet son las más comunes, el mapeo de direcciones multicast se lleva a cabo como se describe a continuación. En primer lugar se asignan a los 24 bits de mayor peso de la dirección MAC los valores 01:00:5E. El bit posterior siempre lleva un valor de 0 y los 23 bits de menor peso restantes contienen el valor de los 23 bits de menor peso de la dirección multicast IPv4. Por ejemplo la dirección IP multicast 239.5.5.0 utilizada en el ejemplo de la Figura 4.5, se correspondería con la dirección física Ethernet 01:00:5E:05:05:00.

4.5. Aspectos importantes para multicast

En comunicaciones multicast, el intercambio de datos se realiza entre más de dos nodos de una red. En esta sección se explican algunos mecanismos relacionados con la comunicación especialmente relevantes en comunicaciones multicast, en los que se incluyen: fiabilidad, control de flujo y congestión y gestión de los grupos multicast. En entornos multicast, en ocasiones es más complejo implementar estos mecanismos que para la comunicación unicast. Lo anterior debido a las características particulares de cada nodo dentro de un grupo y, de la cantidad de grupos dentro de la

red.

4.5.1. Fiabilidad

En [101] se puede encontrar una definición tradicional de este término: *"un servicio fiable es aquel en el que todos los datos se entregan al receptor en el orden correcto, sin errores y sin ninguna duplicación. Si no es posible proporcionar un servicio fiable, por ejemplo, a causa de un fallo de enlace, generalmente se informa al usuario y la comunicación finaliza"*. Como describe Wittmann, los mecanismos utilizados para proporcionar servicios fiables se basan en la suposición de que existe sólo un emisor y un receptor. Un protocolo multicast fiable debe asegurar que todos los nodos receptores reciben todos los datos desde los emisores. Esta fiabilidad en la transmisión resulta útil, por ejemplo, en los sistemas de ficheros distribuidos. Los datos y las posteriores actualizaciones deben enviarse a todos los nodos de almacenamiento con el fin de asegurar que la consistencia de los datos se mantiene. Por lo tanto, el protocolo tiene que garantizar la entrega fiable a todos los receptores.

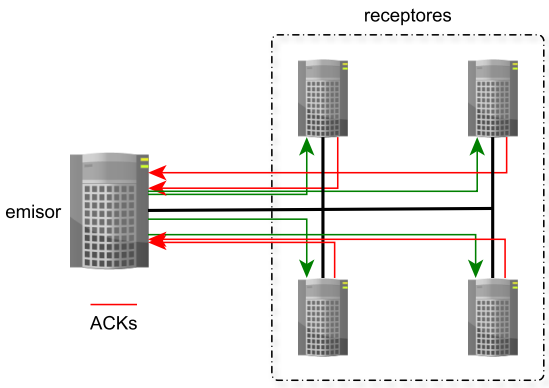

Figura 4.6: Saturación por ACKs en emisor (las líneas en verde significa el envío de datos y en rojo los ACK).

El termino fiabilidad también está relacionado con los mecanismos de detección y recuperación de errores, y por lo tanto deben ser considerados para el desarrollo de un protocolo fiable. Un mecanismo básico para la detección de errores consiste en asignar números de secuencia a los paquetes que se envían a los receptores, de tal modo que los receptores mantengan un mecanismo de comprobación de secuencia para detectar pérdida de paquetes. En cuanto al mecanismo de recuperación de errores, en las implementaciones unicast y en algunos casos multicast cada receptor envía un reconocimiento positivo (ACK) por cada paquete que recibe dando lugar al problema conocido como *implosión de ACK*. Por ejemplo, en la Figura 4.6 se muestra un ejemplo en el que un emisor puede aumentar la carga de trabajo causada por los ACK enviados desde los receptores. En el ejemplo se ilustra un emisor puede ser saturado como consecuencia del envío de un sólo paquete de datos. En entornos donde existen diversos grupos multicast, el incremento del envío de ACK puede ser considerablemente elevado. Como la probabilidad de fallo es baja, sería más eficiente usar para conseguir fiabilidad reconocimientos negativos (NAK) en lugar de reconocimientos positivos (ACK), así se evitaría la implosión de ACK. Este mecanismo permite disminuir la carga de trabajo de los emisores y al mismo tiempo reducir el tráfico de la red, debido a que los receptores únicamente envían al emisor un NAK por cada paquete que no se recibe o que se llega con errores. En la Figura 4.7 se ilustra la disminución del tráfico generado para mantener la recuperación de paquetes perdidos. Por este motivo se prefiere el uso de NAK frente a ACK en aplicaciones o protocolos basados en UDP, especialmente cuando se utilizan comunicaciones multicast. (en el Capítulo 6 de [101] se describen algunos protocolos).

4.5.2. Control de flujo y de congestión

Es necesario implementar mecanismos de control de flujo y congestión para regular la tasa de envío de datos entre los nodos que participan en

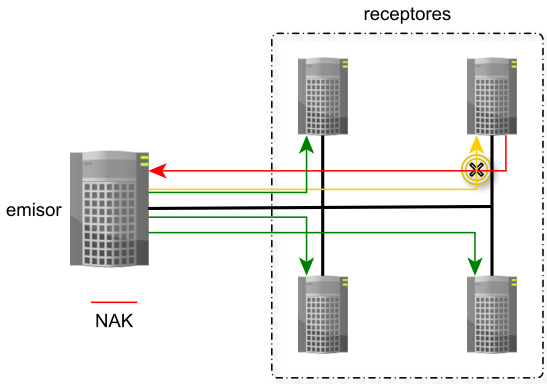

Figura 4.7: Mecanismo de recuperación con NAK (las líneas en verde significa el envío de datos, en amarillo un paquete que no se ha recibido o llega con error y en rojo los paquetes NAK).

la comunicación. Los mecanismos implementados para unicast (basadas en TCP) están diseñados para la comunicación entre dos nodos. Aunque existen implementaciones de estos mecanismos para UDP (sea para comunicaciones unicast o multicast) es necesario estudiar las necesidades de las aplicaciones para intentar adaptar estos requerimientos a las nuevas implementaciones. Los mecanismos de detección y recuperación de errores están relacionados con el control de flujo de los datos. El mecanismo de control de flujo permite gestionar la tasa de envío de datos del nodo emisor hacia el (los) receptor (es) con la intención de evitar que un emisor rápido sature a un receptor lento. Existen dos enfoques que se comparan sistemáticamente en [98], unos *basados en el emisor* y otros *en el receptor*. En el segundo enfoque, recae en el receptor la responsabilidad de informar al emisor para ajustar la tasa de envío de acuerdo a su estado de saturación en el búfer de recepción. En esa comparación se manifiesta que el enfoque *basado en el receptor* mantiene un mejor rendimiento que el enfoque opuesto. Normalmente, el enfoque *basado en el receptor* utiliza paquetes NAK debido a que permite evitar la saturación por paquetes ACK. En [39] [101] se describen los mecanismos

basado en ventana (*window-based*) y *basado en tasa de envío* (*rate-based*) utilizados para ajustar el control de flujo de datos y que son ampliamente utilizados por el protocolo TCP.

El mecanismo de control de congestión puede ayudar a prevenir la saturación de la red o de los búfer de los nodos receptores. Por lo tanto, mantiene una estrecha relación con el control de flujo. Este mecanismo permite regular la tasas de envío de datos de los emisores permitida según el estado de la red. En un entorno multicast, este mecanismo requiere de información detallada de los receptores (capacidad de los búfer de recepción, direcciones multicast a las que se ha asociado, etc.) para facilitar el flujo de datos desde varios emisores. Esta información se puede utilizar como medida preventiva para evitar la congestión, tanto de la red como en los nodos receptores. De tal manera que el protocolo garantice la fiabilidad en la entrega de los datos a todos los receptores de forma transparente. En un sistema de ficheros distribuido, el control de congestión tiene 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.

4.5.3. Gestión de grupos multicast

Antes de hacer el envío de datos desde un emisor a una dirección multicast, es necesario que exista la unión de los nodos de la red interesados en recibir el tráfico. Un nodo puede unirse a diferentes direcciones multicast y el emisor puede no pertenecer al grupo multicast donde envía los datos. El protocolo IGMP permite gestionar, dada una dirección IP multicast, la pertenencia de un nodo a un determinado grupo multicast. Además, a nivel de conmutador las implementaciones de IGMP Snooping en estos dispositivos facilita conocer el estado de uso de direcciones multicast y de los nodos que pertenecen a esta. Un conmutador que implementa IGMP Snooping escucha los mensajes IGMP enviados por los nodos de la red y proporciona una transmisión selectiva de tráfico multicast basado en la

información de dirección multicast que contiene cada mensaje. Por tanto, la gestión adecuada de los grupos multicast bajo IGMP Snooping permite hacer una distribución de datos evitando que el tráfico se convierta a envíos similares a broadcast.

4.5.4. Ancho de banda y latencia

Las aplicaciones basadas en multicast tienen requisitos comunes con las aplicaciones unicast. Principalmente cuando se trata de transferir datos de gran tamaño, de tal manera que puede impactar en el consumo de ancho de banda de la red y que además deben mantener una baja latencia. Algunas aplicaciones tienen requisitos de retardo estrictos mientras que otros no. Algunas aplicaciones consumen un importante ancho de banda, como por ejemplo las aplicaciones de transferencia de ficheros, mientras que otras mantienen un bajo uso de ancho de banda [65].

En general, las aplicaciones se deben diseñar de tal manera que se puedan adaptar a la variabilidad del estado de la red, principalmente cuando se experimentan momentos de congestión. Además, deben ser adaptables a las condiciones de la red, en el sentido de que es posible que algunas aplicaciones unicast hagan uso del ancho de banda común. Aunque, en un entorno de cluster de computadores los nodos suelen tener una arquitectura homogénea, también es posible que algunos nodos se encuentren más saturados que otros; es decir, puede existir diversidad en la capacidad de procesamiento. Es importante que las aplicaciones tengan esto cuenta. De igual forma, la tecnología de almacenamiento puede afectar el funcionamiento de las aplicaciones. Si en los nodos se consideran dispositivos de almacenamiento suficientemente rápidos que puedan atender las demandas, por ejemplo, de las transferencias de grandes volúmenes de datos, puede ser posible aprovechar de forma óptima el ancho de banda global de la red.

En entornos de cluster de computadores la latencia de la red puede ser muy baja, de tal manera que puede no ser perceptible por las

aplicaciones. Sin embargo, la tecnología de almacenamiento puede penalizar el rendimiento global de las aplicaciones, como ocurre en el siguiente ejemplo. Supongamos un nodo receptor que recibe datos desde dos direcciones multicast, se crean entonces dos hebras de recepción, una por cada dirección multicast. Supongamos además que cada hebra, al mismo tiempo, debe escribir los datos en disco para garantizar fiabilidad. Bajo estas condiciones, el incremento de la latencia de la escritura en disco es considerablemente alta, hasta el punto que el ancho de banda de escritura en disco puede disminuir por debajo del 50 % (en el capítulo 6 se hace un amplio análisis y se muestran algunos resultados importantes).

4.6. Aplicaciones multicast

Actualmente el IP multicast juega un papel importante en el entorno de Internet. Las comunicaciones multicast permite a los desarrolladores añadir, a las aplicaciones o protocolos, mayor funcionalidad sin impactar de forma importante en la red. El desarrollo de aplicaciones o protocolos multicast es aparentemente simple. A nivel de datagrama es posible que cualquier aplicación pueda enviar datos a un dirección multicast. Simplemente a nivel de aplicación es necesario aumentar el valor TTL de tal manera que los datagramas tengan la facilidad de atravesar los enrutadores hasta llegar al destino. Para recibir los datagramas multicast, basta con habilitar la unión a una dirección multicast de forma transparente mediante un informe de pertenencia al grupo multicast común. Sin embargo, la habilitación del soporte multicast en aplicaciones y protocolos es un reto importante, principalmente cuando: el envío de flujos de datos es contante, se requiere una entrega fiable de datos, y hay que gestionar un elevado número de grupos multicast. Estos aspectos requieren pues una consideración especial en este trabajo.

Las aplicaciones multicast se desarrollan con el soporte de la capa de

transporte del protocolo UDP. Aunque, TCP proporciona un servicio a las aplicaciones, tal como, recuperación ante la pérdida de paquetes, corrección de errores, una entrega ordenada, etc. Sin embargo, TCP únicamente proporciona servicios de comunicación unicast. En cambio, UDP, aunque proporciona servicios mínimos, por ejemplo, detección de errores (si un paquete se detecta con error, simplemente se descarta), da soporte para las comunicaciones multicast. Por lo tanto, las aplicaciones multicast se deben ejecutar sobre UDP, como ilustra la Figura 4.8. Para dar fiabilidad a las aplicaciones multicast es necesario establecer mecanismos, como los descritos en la sección 4.5.

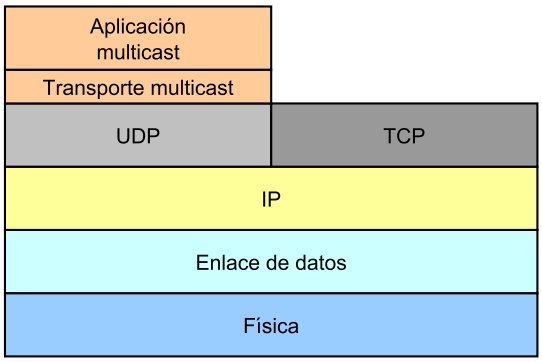

Figura 4.8: Aplicaciones multicast sobre UDP.

En [70] se hace una clasificación de algunas aplicaciones multicast donde se hace distinción entre aplicaciones multimedia y aplicaciones de manipulación de datos, y entre aplicaciones que envían datos en tiempo real y lo contrario a éstas (Figura 4.9). En esta figura, la columna de la izquierda incluye las aplicaciones de transmisión de datos en tiempo real. Las aplicaciones con restricciones de tiempo real requieren baja latencia, lo que supone un mayor reto en sistemas de ficheros con requisitos de fiabilidad debido a que los mismos datos deben ser distribuidos a múltiples nodos. En cuanto a la columna de la derecha de la misma figura, describe las

aplicaciones donde no es necesaria una transmisión en tiempo real. En este caso, aunque no tienen requisitos estrictos de latencia, usualmente necesitan fiabilidad y escalabilidad.

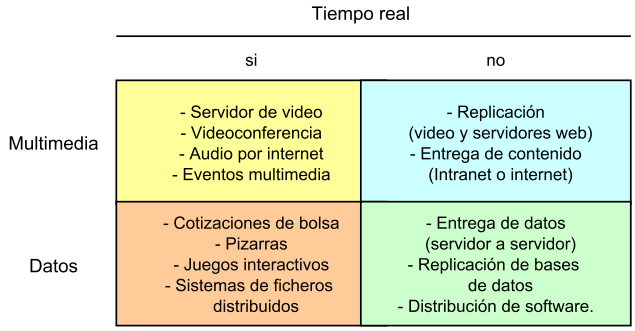

Figura 4.9: Clasificación de aplicaciones multicast.

Las *aplicaciones multimedia en tiempo real*, por su parte, no requieren una fiabilidad estricta. En este caso se requiere asegurar que llegan los datos, no importa que lleguen con retraso, y que los posibles errores no degraden el flujo multimedia a un nivel que pueda ser percibido por el ser humano [70]. En la Figura 4.9, en particular el recuadro de *aplicaciones de datos en tiempo real*, se ha agregado los sistemas de ficheros distribuidos. Conmúnmente estas aplicaciones requieren hacer la entrega en tiempo real para realizar las operaciones sobre los datos (HDFS puede ser un caso particular, donde el sistema Hadoop primero envía los datos a los nodos de almacenamiento y posteriormente lanza las tareas a realizar por el paradigma MapReduce). Por otra parte, hay *aplicaciones multimedia o de datos* que requieren una transferencia fiable pero no en tiempo real. Por ejemplo, el sistema DRBD, aunque en esencia hace manipulación de datos, en algunos casos puede ser que la transmisión de datos a los nodos secundarios no sea en tiempo real (de forma particular en DRBD se refiere a los protocolos A y B).

En [82] se hace una descripción de tres categorías de aplicaciones

multicast que las diferencia totalmente de las aplicaciones unicast.

- *Uno a muchos* (1-M): Un solo nodo envía datos a dos o más receptores.
- *Muchos a muchos*(M-M): Cualquier número de nodos envían datos al mismo grupo multicast, además de recibir de esa dirección.
- *Muchos a uno*(M-1): Cualquier número de nodos receptores que envían datos a un emisor a través de unicast o multicast.

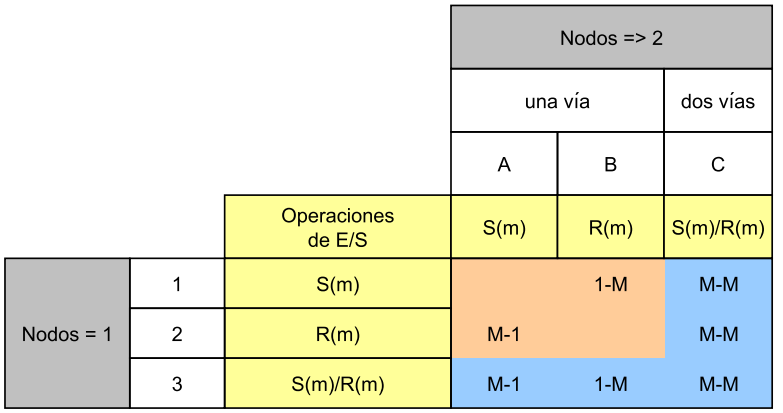

Figura 4.10: Aplicaciones multicast en relación a la E/S y a la distribución de datos.

De [82] se extrae parte de una clasificación que ha sido adaptada a las necesidades de este trabajo con el fin de simplificar la explicación de los tipos de aplicaciones que hacen uso de comunicaciones multicast. En la Figura 4.10 (donde *S(m)* se refiere a envío multicast y *R(m)* a recepción multicast) se define el tipo de aplicaciones en términos de la combinación de mecanismos de comunicación que utilizan, es decir la relación de la E/S que representan. Por ejemplo, a nivel de IP, la multidifusión de E/S siempre es 1-M o M-M, mientras que para unicast siempre se utiliza 1-1.

4.6.1. Aplicaciones uno a muchos (1-M)

Las comunicaciones 1-M tienen un solo emisor y muchos receptores simultáneos. Por ejemplo, la relación B1 de la Figura 4.10, muestra la clásica relación para la comunicación 1-M. En la Figura 4.11 se hace una representación de la distribución básica en este tipo de aplicaciones.

Algunos ejemplos de este tipo de aplicaciones pueden ser: radio por internet, vídeo bajo demanda, videoconferencia, etc. Por lo tanto, las aplicaciones 1-M se caracterizan por el envío de datos de *una vía*. Es decir, si se establece esta configuración, un emisor no puede recibir el tráfico de datos generado por él mismo hacia la dirección multicast específica.

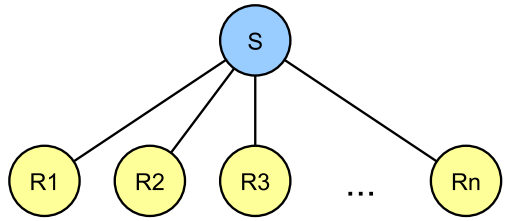

Figura 4.11: Aplicaciones multicast uno a muchos.

4.6.2. Aplicaciones muchos a muchos (M-M)

En las aplicaciones muchos a muchos, dos o más receptores también pueden actuar como emisores (relación C1, C2, C3 de la Figura 4.10). De tal manera que las aplicaciones M-M se pueden caracterizar por el envío de datos de *dos vías*. En este entorno un nodo emisor puede recibir los datos que el mismo a enviado a una dirección multicast específica. Con este esquema de comunicación, cada nodo que ejecuta una aplicación M-M puede recibir datos desde múltiples emisores y al mismo tiempo también enviar datos. Por lo tanto, esto puede plantear un reto importante en la gestión de las comunicaciones. De igual forma, este mecanismo permite que el propio

nodo emisor, reciba los mismos datos que envía a la dirección multicast. Esto puede suponer en algunos casos que se reciban datos que en realidad no se van a usar. La Figura 4.12 muestra un ejemplo generalizado del mencionado entorno de comunicación.

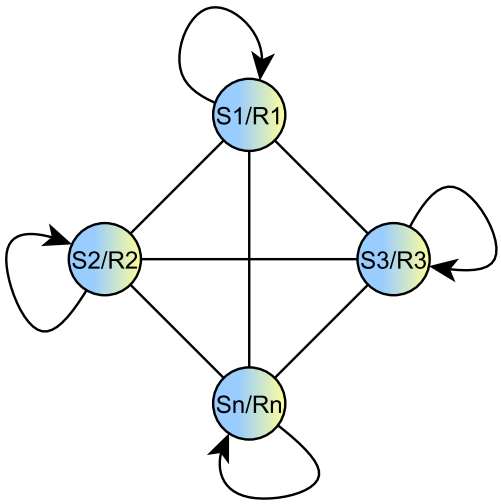

Figura 4.12: Aplicaciones multicast muchos a muchos.

4.6.3. Aplicaciones muchos a uno (M-1)

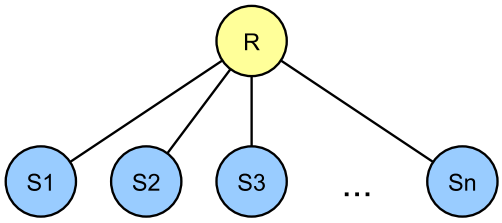

Figura 4.13: Aplicaciones multicast muchos a uno.

Las aplicaciones muchos a uno pueden ser de una sola vía (al igual

que las aplicaciones uno a muchos) o de dos vías si se utiliza un protocolo de petición/respuesta. Esto consiste en que, cualquiera de los emisores o receptores puede generar una solicitud. Las aplicaciones muchos a uno se diferencia de las aplicaciones 1-M y M-M en que no representa un mecanismo de comunicación en la capa IP, de tal modo que, las aplicaciones M-1 tienen múltiples emisores y un receptor [82]. La Figura 4.13 ilustra las características esenciales en la comunicación.

Capítulo 5

Análisis de rendimiento en comunicaciones Multicast

Las comunicaciones multicast permiten la distribución simultánea a diferentes destinos. En este capítulo se realiza un análisis del rendimiento de estas comunicaciones. En primer lugar se mencionan algunos trabajos relacionados al trabajo de la tesis, donde se demuestran algunas de las principales diferencias. Posteriormente se establece una configuración y se realiza un análisis por simulación. A continuación, se plantea la primera propuesta del protocolo que permite el envío simultáneo a diferentes nodos físicos de la configuración. Las evaluaciones se han realizado en un cluster real.

5.1. Introducción

Muchas aplicaciones de computación de altas prestaciones (HPC) requieren una rápida y eficiente distribución de datos entre los nodos de un cluster. Algunos ejemplos de éstas pueden ser: aplicaciones de *análisis de datos* y aplicaciones de *almacenamiento de datos*. Además, algunas

aplicaciones necesitan disponer de redundancia de datos para garantizar una alta disponibilidad en las aplicaciones. Con el fin de mantener redundancia de datos, estos deben ser enviados a múltiples servidores. Para poder implementar este mecanismo es común usar comunicaciones basadas en TCP o unicast para realizar la difusión de la comunicación, como por ejemplo en [9][90]. TCP es un protocolo orientado a conexión de tal manera que asegura fiabilidad a las aplicaciones.

Para establecer una conexión TCP se usa el procedimiento conocido como "negociación en tres pasos"(*three-way handshake*) y para la desconexión se utiliza el procedimiento "negociación en cuatro pasos"(*four-way handshake*). Además, TCP mantiene un temporizador de retransmisión (*timeout*) por cada conexión. Este temporizador se usa cuando TCP espera recibir un reconocimiento positivo (ACK) del receptor. Cuando una conexión se establece, se configuran algunos parámetros (por ejemplo, número de secuencia) para asegurar la entrega ordenada de los datos. Por lo tanto, en todas las transferencias TCP usa ACK para asegurar la entrega de los datos y establecer comunicación con cada nodo. Como ejemplo particular, si un nodo quiere enviar datos a n nodos en TCP, este nodo debe establecer n canales de comunicación unicast.

Tal y como se describe en [58], con el uso de TCP, el tiempo de comunicación se incrementa cuando se tiene un número alto de nodos receptores, a diferencia de las comunicaciones multicast basadas en ACK, donde el tiempo de transferencia y la sobrecarga del procesador se pueden reducir considerablemente. En este trabajo, sin embargo, se usa un protocolo basado en NAK (reconocimiento negativo, que se describe más adelante) para prevenir el problema de implosión de comunicaciones basadas en ACK.

Las comunicaciones basadas en multicast son considerablemente útiles porque permiten enviar datos a múltiples nodos receptores haciendo uso del soporte hardware de los conmutadores, proporcionados por el sistema, para el envío de datos de forma simultánea. En sistemas cluster, los

conmutadores configurados con IGMP Snooping permiten un uso eficiente de comunicaciones multicast, reduciendo la utilización del ancho de banda [64]. Cuando IGMP Snooping no se usa, el conmutador distribuirá los datos por todos sus puertos, por lo que el tráfico generado será muy similar al obtenido usando broadcast, tal y como se muestra en [66].

En comunicaciones multicast, la transferencia de datos fiable no esta garantizada. El envío de datos puede ser masivo y simultáneo, esto puede ocasionar saturación en algunos nodos receptores provocando pérdidas de datos. En consecuencia, las aplicaciones que hacen uso de estos datos se pueden ver afectadas. Considerando este aspecto, y dada la importancia de que los datos deben estar siempre disponibles, es muy importante desarrollar un mecanismo de control de congestión para transferencias múltiple multicast (multi-multicast) de tal manera que este mecanismo pueda prevenir o minimizar al máximo la pérdida de paquetes.

5.2. Trabajo relacionado

En la literatura se han propuesto algunas taxonomías de protocolos multicast fiables o RMP (*Reliable Multicast Protocol*) por sus siglas en Inglés ([74] [17]). Estas taxonomías se basan en características que deben ser consideradas para implementar protocolos multicast fiables. Teniendo en cuenta los criterios de clasificación usados en estas taxonomías, para implementar un protocolo que pueda adaptarse a las condiciones globales de la red de datos, deberían considerarse los siguientes aspectos: *propagación de datos*, *mecanismo de control de congestión*, *confiabilidad*, y *proceso de reparación*.

La Tabla 5.1 caracteriza algunos protocolos multicast propuestos en la literatura, incluyendo la propuesta en este trabajo, la clasificación de la tabla se hace valorando los aspectos de:

- **Propagación de datos (*Data propagation*):** Es la forma en la que

los datos se transmiten a los receptores. La difusión basada en árbol es la opción más usada en escenarios de Internet.

- **Confiabilidad (*Reliability mechanism*):** Este mecanismo garantiza al emisor que debe tener en cuenta si debe reenviar datos que no fueron recibidos de forma correcta por uno o más receptores. Éste mecanismo puede estar basado en el uso de un reconocimiento positivo (ACK) enviado por el receptor al emisor por cada paquete recibido correctamente, basado en en el uso de un reconocimiento negativo (NAK) enviados por el receptor al emisor si no recibe correctamente un paquete, o en una implementación basada en una combinación de ACK-NAK.

- **Reparación de pérdida (*Repair loss*):** Los datos no recibidos por el receptor deben ser reenviados desde el emisor. La mayoría de las implementaciones usan TCP unicast para el reenvío de datos perdidos desde los emisores a cada receptor. En la literatura también se pueden encontrar algunas implementaciones basadas en multicast.

- **Control de congestión (*Congestion control*):** Es necesario implementar un mecanismo que permita prevenir la saturación en los receptores y/o en la red ajustando la tasa de envío en los emisores.

- **Escenario objetivo (*Target scenario*):** Escenario dónde se evalúa cada protocolo.

En [58] se puede ver una comparación de mecanismos confiables para comunicaciones multicast, tales como: basados en ACK, basados en NAK, y en una combinación ACK-NAK. Tras las evaluaciones, los resultados en [58] (así como en [106]) muestran que la implementación basada en NAK es la más recomendable para obtener mejor rendimiento.

El protocolo propuesto en este trabajo de investigación se basa en reconocimientos negativos (NAK) para asegurar la entrega de datos y utiliza

Tabla 5.1: Clasificación de los protocolos actuales de multicast.

RMP	Propagación de datos	Mecanismo fiabilidad	Recup.	Control congestión	Uso
Delta-RMP	MM	NAK	Uc	W(Delta-CC)	CPD
RDCM[61]	Mc(árbol)	NAK	Uc	W(BIC)	CPD
CooPNC[69]	Mc	MACK	NS	NS	RI
NORM[18]	Mc(árbol)	NAK	NS	R(TCP-F)	WAN
[54]	Mc(árbol)	NAK	Mc	No	WAN
PGM[44]	Mc(árbol)	NAK	NS	R	WAN
VCMTP[62]	Mc	NAK	Uc	No	VLAN
RMTP[78]	Mc(árbol)	ACK	Uc	W	WAN
TMTP[107]	Mc(árbol)	ACK-NAK	Uc	W	WAN
LBRM[52]	Mc(árbol)	ACK-NAK	Uc	NS	WAN

Uc=Unicast Mc=Multicast MM=Multi-Multicast NS=Not Specified W=Window TCP-F=TCP-Friendly R=Rate RI=Redes Inalámbricas CPD=Centros de Procesamiento de Datos

información de control de los receptores para minimizar la congestión. Además, se utiliza el soporte IGMP Snooping que proporcionan los conmutadores con el fin de enviar los datos desde los servidores a grupos específicos de receptores, evitando de esta manera envíos como *broadcast* y *multiples broadcast* cuando se requiere hacer múltiples envíos multicast simultáneamente. IGMP Snooping permite asegurar que el tráfico multicast de un grupo específico sólo se recibe por los nodos interesados en ese tráfico.

Tal y como se puede observar en la Tabla 5.1, la mayor parte de los protocolos utilizan un enfoque de distribución de datos a los receptores basado en árbol. A diferencia, en nuestro enfoque, IGMP Snooping nos permite evitar el uso de un enfoque basado en árbol.

Con el fin de conocer las implementaciones basadas en comunicaciones multicast, en la literatura se han encontrado algunos desarrollos con

características particulares.

Por ejemplo en [61] se propone un protocolo multicast fiable para redes de centro de datos, conocido como RDCM (*Reliable Data Center Multicast*). El objetivo principal de RDCM es minimizar el impacto de la pérdida de paquetes sobre el rendimiento multicast, mediante el aprovechamiento de los potentes recursos de interconexión en los centros de procesamiento de datos (o *data centers*). Sin embargo, el protocolo propuesto en esta tesis ajusta la tasa de envío de los emisores para reducir la pérdida de paquetes con un mecanismo de control de congestión adaptativo.

De igual forma, [54] propone un protocolo multicast fiable que permite la escalabilidad. Este protocolo utiliza múltiples canales multicast para el envío de datos. Dicho enfoque usa un canal multicast para transmitir paquetes originales, es decir, por primera vez, y la retransmisión de los paquetes se realiza por un canal multicast adicional. Es por esto que el protocolo se describe como un protocolo con soporte de múltiples canales multicast. A diferencia, en nuestra propuesta consideramos el envío simultáneo desde varios emisores, de tal manera que la capacidad de ancho de banda de la red sea compartida eficientemente.

VCMTP (*Virtual Circuit Multicast Transport Protocol*) en [62] se refiere a un protocolo con soporte de circuitos virtuales. VCMTP usa reconocimientos negativos y conexiones unicast TCP para el envío de NAKs y las retransmisiones. Además, VCMTP utiliza funciones de control de errores y control de flujo.

Además de las redes cableadas, las redes inalámbricas también se han estudiado en la literatura. En [89] hacen uso de un enfoque de petición de retransmisiones basado en NAK, pero las retransmisiones de datos se realizan a través de un canal multicast. En consecuencia, es posible que estas retransmisiones las reciban nodos receptores que no requieren de estos datos. Este tráfico adicional puede afectar de forma negativa al rendimiento global del sistema.

Otro entorno multicast estudiado se encuentra en [69] donde se propone un protocolo multicast para redes wireless de corto alcance (CooPNC). El protocolo proporciona un esquema de cooperación indirecta en la capa de control de acceso al medio (MAC) . El objetivo principal de CooNPC es preservar una alta fiabilidad con una alta eficiencia energética, y un intercambio de información de control bajo.

5.3. Análisis de tráfico multicast

Para poder aprovechar los beneficios del uso de comunicaciones basadas en multicast, en esta sección se analiza, mediante simulación, las prestaciones globales de estas comunicaciones. En primer lugar, el escenario propuesto, permite crear diversos grupos de receptores, los nodos pueden recibir tráfico multicast desde diferentes direcciones multicast a las que se han asociado previamente. Es muy importante que el conmutador tenga soporte específico del protocolo IGMP para gestionar eficientemente los grupos multicast. Como se ha mencionado en secciones anteriores, el soporte IGMP Snooping permite gestionar el tráfico multicast, de tal manera que el tráfico se distribuye solo a grupos específicos de receptores interesados en recibir datos de determinadas direcciones, tal y como se muestra en la Figura 5.1. A diferencia de la Figura 5.2, donde todos los nodos en la red reciben el tráfico multicast. Por lo tanto, el consumo de ancho de banda y el uso de memoria para el procesamiento de paquetes se reduce considerablemente.

5.3.1. Configuración

En este primer análisis la aplicación requiere que múltiples servidores trabajen en paralelo. Cada servidor envía información en tiempo real a un grupo de servidores. En la configuración inicial, cada grupo está compuesto por 4 nodos, 1 emisor y 3 receptores.

En una configuración sin IGMP Snooping, el conmutador se satura

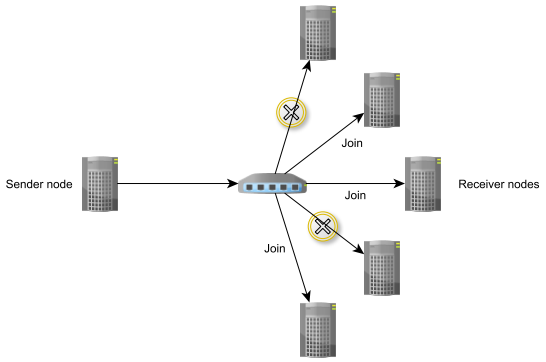

Figura 5.1: Envío de datos *con* soporte IGMP Snooping

rápidamente porque éste intenta enviar datos simultáneamente a todos los destinos. Sin embargo, con el uso de IGMP Snooping el conmutador de manera selectiva solo envía datos a receptores específicos.

A manera de ejemplo, si N es el número de servidores, G es el número de grupos, R es el número de receptores por grupo. Asumiendo que el número de nodos conectados en el conmutador es el número de servidores N y que cada servidor envía X bytes simultáneamente a cada grupo, el tráfico multicast T generado, según una simple fórmula sería:

En una configuración con IGMP Snooping:

$$T = X.G.R \tag{5.1}$$

En una configuración sin IGMP Snooping:

$$T = X.G.N \tag{5.2}$$

En este caso la relación N/R determina una mejora por diversas razones; **a**) en el caso anterior la relación $N/R = 4$, por ejemplo, el sistema puede procesar 400 % más de tráfico, **b**) si se aumenta el número de servidores de manera significativa, el rendimiento global podría mejorar, **c**) los nodos que

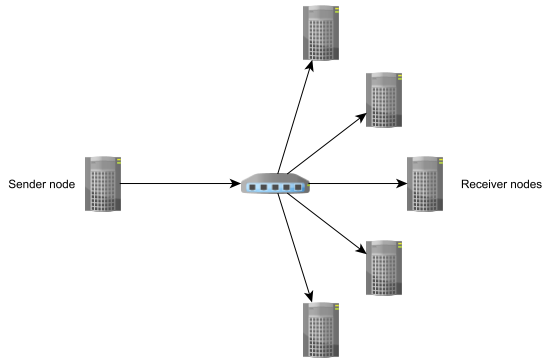

Figura 5.2: Envío de datos *sin* soporte IGMP Snooping

no pertenecen a un grupo, no necesitan recibir el tráfico, por lo que reduce el tiempo de procesamiento de paquetes no deseados.

En este análisis se estudia la pérdida de paquetes en comunicaciones multicast. Esto es importante porque los nodos que no reciben todos los datos pueden funcionar de manera incorrecta. Dado que la razón de este trabajo está enfocado a la distribución de datos en sistemas de ficheros distribuidos, es muy importante garantizar que todos los nodos en el sistema recibirán todos los datos. En una configuración de cluster de computadores, en un escenario de múltiples transmisiones multicast simultáneamente, la pérdida de paquetes se puede producir cuando:

- El búfer interno del conmutador se satura.
- El sistema operativo no puede gestionar todos los paquetes recibidos o la aplicación no puede procesar todo el tráfico multicast.

A pesar de que se hace uso de una configuración con soporte IGMP Snooping, que permite mejorar el rendimiento y poder hacer uso eficiente de la comunicación, es necesario implementar un esquema de control de congestión eficiente para tratar de prevenir la pérdida de paquetes. El

problema del envío simultáneo desde diferentes emisores es que puede causar que algunos receptores se saturen más que otros, por lo que, si reciben el tráfico por igual, estos receptores probablemente pierden datos.

5.3.2. Control de congestión

Aunque IGMP Snooping permite mejorar el rendimiento de la red no garantiza fiabilidad en la entrega. Por lo tanto, se propone un esquema de control de congestión para asegurar fiabilidad en la entrega de los datos a todos los receptores. CUBIC [48] es el mecanismo de control de congestión para entornos de red de alta velocidad. Este mecanismo se usa actualmente en el kernel de Linux para comunicaciones TCP, además de ser una versión mejorada de BIC [105]. La función de crecimiento de ventana de CUBIC se rige por una función cúbica (5.3) en términos del tiempo transcurrido desde el último evento de pérdida.

$$W_{cubic} = C(t - K)^3 + W_{max} \tag{5.3}$$

Donde C es el factor de escala, t es el tiempo transcurrido desde la última reducción de ventana, W_{max} es el tamaño de ventana justo antes de la última reducción de ventana, y $K = \sqrt[3]{W_{max}\beta/C}$, donde β es un factor de disminución aplicado a la reducción de ventana en el momento de un evento de pérdida. A partir del estudio experimental en [59] en este escenario de simulación, se propone un mecanismo de control de congestión para comunicaciones multicast basado en las principales características de CUBIC.

5.3.3. Simulador

A pesar de que existen diversos simuladores de tráfico de red, el interés en esta sección es hacer una simulación para evaluar:

- El tráfico generado por el conmutador.

- Cómo llegan los paquetes a los receptores.
- Cómo y cuándo ocurre la pérdida de paquetes.

Se ha diseñado un simulador donde se consideran los elementos descritos anteriormente. En este caso, la simulación se puede modelar por eventos y se ha elegido Simpy [91] como motor de simulación. Simpy es un framework de simulación de eventos discretos basado en Python, que puede ser adaptado a nuestro modelo. Se ha creado a partir de medidas experimentales obtenidas en un entorno real.

Los conmutadores usados en sistemas de cómputo suelen implementar dos tipos de conmutación en el nivel de enlace de la red: *cut-through* y *store-and-forward* [102]. *Cut-through* se usa en conmutadores de alto rendimiento, permite comenzar a procesar un paquete antes de que se haya recibido en su totalidad. En cambio, *store-and-forward*, que es la alternativa más usada implementa un esquema que puede procesar un paquete cuando ha sido completamente recibido. En nuestro modelo se ha definido un sistema de colas de E/S virtual que evita la copia del paquete y permite reenviar paquetes a cada puerto cuando sea necesario.

El modelo de servidor incluye dos etapas que trabajan en una configuración productor-consumidor. Para la recepción, el servidor ejecuta hebras específicas para cada grupo multicast y usa un búfer por grupo (este búfer puede crecer dinámicamente, hasta un máximo limitado por la memoria principal de los servidores, porque el uso de memoria virtual basada en intercambio (*swap*) penaliza drásticamente el rendimiento). Con esta configuración, diferentes grupos tienen diferentes búfer y no se afectan uno al otro. Esta parte se define como productor porque procesa los paquetes de forma asíncrona. El consumidor está recibiendo y liberando datos del búfer de recepción. Si un consumidor se sobrecarga, el búfer de recepción no tendría suficiente espacio y podría existir una pérdida de paquetes. El mecanismo de control de congestión trata de evitar que el búfer se quede sin

espacio para procesar paquetes. El esquema de control de congestión se basa en CUBIC, que se adapta a los requerimientos iniciales de la simulación.

5.3.4. Evaluación de rendimiento

Con el fin de obtener un rendimiento óptimo, es necesario enviar el máximo número de paquetes que la red puede procesar, y al mismo tiempo, tratar de evitar la pérdida de paquetes porque esto puede penalizar drásticamente el rendimiento global del sistema. Para obtener un modelo más cercano a un entorno real, se ha analizado el tráfico entre diferentes nodos en un cluster de ordenadores con una red Gigabit. Estos nodos están conectados por un conmutador Dlink DGS-1248T. Este conmutador tiene un búfer de paquetes de 1 Mbyte y tiene la capacidad de reenviar hasta 71.4 Mpaquetes/s. Los nodos servidores están configurados de la siguiente manera: con CPU de 2 x 1.87GHz Intel Xeon E5320, Memoria RAM de 4GB, Disco Local de 512GB y Sistema Operativo CentOS 6.6.

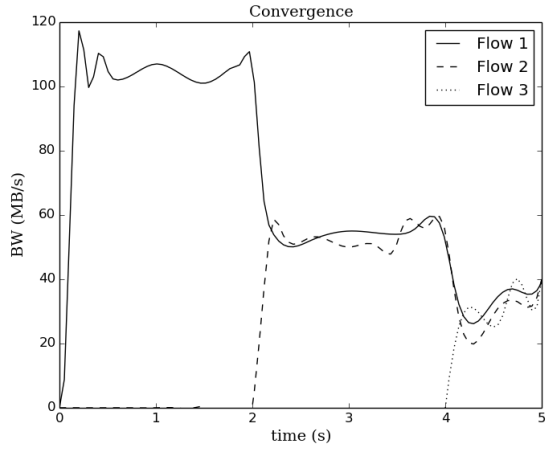

Figura 5.3: Ancho de banda obtenido con un consumidor rápido.

En el primer escenario, la Figura 5.3[1], se muestra que es posible obtener el máximo ancho de banda cuando un consumidor es capaz de procesar rápidamente los datos. En este caso el principal interés es inicialmente enviar tantos paquetes como sea posible, por lo tanto un modelo de inicio lento (slow start) no es adecuado. En esta configuración es importante evaluar qué ocurre cuando dos (t=2) y tres (t=4) grupos multicast tratan de enviar datos a los mismos receptores. Es muy importante lograr un balanceo entre los grupos para obtener un uso óptimo de los recursos de red.

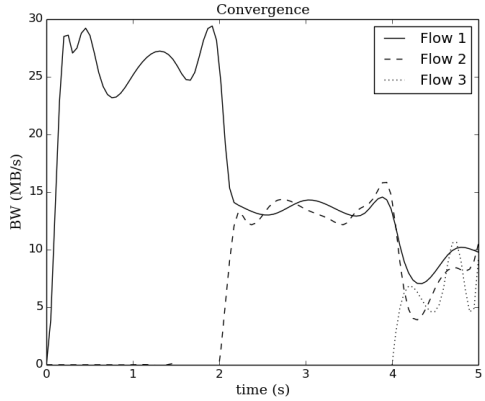

Figura 5.4: Ancho de banda obtenido con un consumidor sobrecargado.

De otra manera, cuando el consumidor es más lento (hasta 30 MB/s) y capaz de producir un colapso, el sistema debe ajustar la tasa de consumo y de procesamiento de paquetes como sea posible. En este entorno, es necesario estudiar lo que ocurre cuando dos y tres grupos multicast envían datos a los mismos receptores. La Figura 5.4[2] muestra cómo el control de congestión puede rápidamente responder a los cambios en el sistema.

[1]Flow1, Flow2 y Flow3 corresponde a los emisores.
[2]Flow1, Flow2 y Flow3 corresponde a los emisores.

5.4. Protocolo Multi-Multicast

De acuerdo a los resultados obtenidos en la simulación y, dado que en sistemas distribuidos, un nodo puede enviar información en tiempo real a un grupo de receptores. Por ejemplo, el mecanismo de replicación en HDFS, donde, por defecto se realizan tres copias de un bloque de fichero en diferentes nodos del sistema para mantener redundancia de los datos.

En esta sección se presenta el protocolo Delta-RMP que incluye un esquema de control de congestión que intenta evitar la pérdida de paquetes. La propuesta se basa en: (1) un nuevo mecanismo de control de congestión que supervisa la información enviada por los receptores para ajustar la tasa de envío de los emisores, (2) el aprovechamiento del soporte de difusión de datos del conmutador, y (3) el uso de IGMP Snooping, que permite reenviar datos solo a los nodos receptores que se han unido a una dirección multicast específica.

5.4.1. Diseño

Debido a que la implementación se ha realizado para un configuración de cluster de computadores, donde, los nodos pueden recibir tráfico desde varias direcciones multicast el diseño permite crear diferentes grupos con hasta tres receptores. La configuración analizada tiene 12 servidores y 12 grupos multicast, y cada grupo está compuesto de 4 nodos, 1 emisor y 3 receptores. En esta configuración, la interconexión de los nodos se hace mediante un conmutador, por lo tanto, no se requiere de una configuración en árbol como los protocolos que se describen en la Tabla 5.1 de la Sección 5.2. En la Figura 5.5[3] se muestra la interconexión de los nodos en la configuración, cada estilo de línea representa los diferentes grupos multicast.

En la subsección 5.3.1 se han mencionado las posibles causas de pérdida de paquetes en una configuración cluster. En un entorno de múltiples

[3]Cada estilo de línea representa a cada grupo multicast.

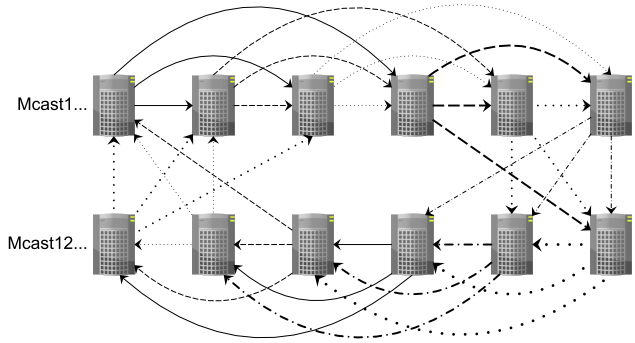

Figura 5.5: Datos transferidos entre diferentes grupos multicast.

comunicaciones multicast simultáneas es posible la rápida saturación del conmutador si las tasas de envío son muy elevadas, además de que, la misma aplicación en los receptores no podría gestionar todas las recepciones aunque tengan búfer de recepción independientes para cada dirección multicast a la que se han asociado. Por lo tanto, también se propone un mecanismo de control de congestión, Delta-CC, para estas comunicaciones.

5.4.2. Implementación

La implementación se ha desarrollado en espacio de usuario bajo la distribución CentOS de Linux. Todo el desarrollo se ha realizado en lenguaje C. Se distinguen dos partes: el cliente, Figura 5.6, y el servidor, Figura 5.7. Ambas partes de la arquitectura utilizan sockets UDP que permiten las comunicaciones multicast.

En el lado del cliente se incluye una interfaz que recibe todos los paquetes desde diferentes direcciones multicast, por ejemplo, cada cliente recibe paquetes desde diferentes nodos físicos y envía información de control a esos nodos emisores. Cuando un cliente se asocia a una dirección multicast, éste crea una hebra. Esta hebra se encarga de:

a) almacenar en un búfer todos los paquetes recibidos.

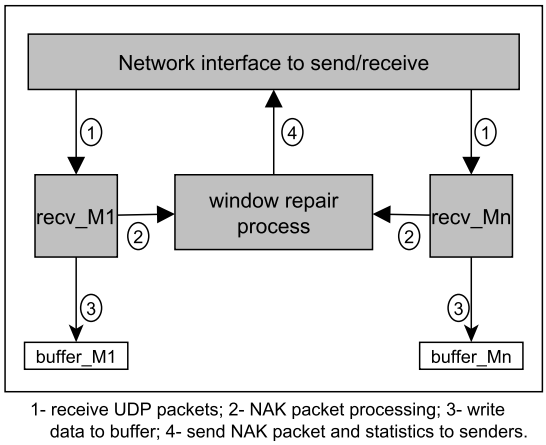

Figura 5.6: Diseño de la arquitectura cliente. Los números indican el orden en el que se realizan las acciones.

b) detectar los paquetes perdidos, el número de secuencia de cada paquete recibido se compara con el contador de paquetes en el receptor. Si el número de secuencia es mayor que el contador de paquetes, el algoritmo detecta la pérdida de paquetes y se notifica al módulo *window repair process*, este módulo solicita, con un paquete NAK, la retransmisión de las secuencias de paquetes perdidos mediante un canal TCP dedicado.

c) obtener estadísticas (marca de tiempo(timestamp), última secuencia recibida, total de paquetes recibidos, total de paquetes perdidos) y los procesa en el modulo *window repair*. El módulo *window repair process* obtiene todas las estadísticas de cada hebra. Por cada hebra se envía un mensaje con esta información de control a los nodos emisores. La información de control de las estadísticas son necesarias para ajustar la tasa de envío por el mecanismo de control de congestión, Delta-CC, en el lado de los emisores.

En cuanto al lado del servidor, incluye una interfaz para enviar paquetes a una sola dirección multicast y, recibir mensajes de control desde los

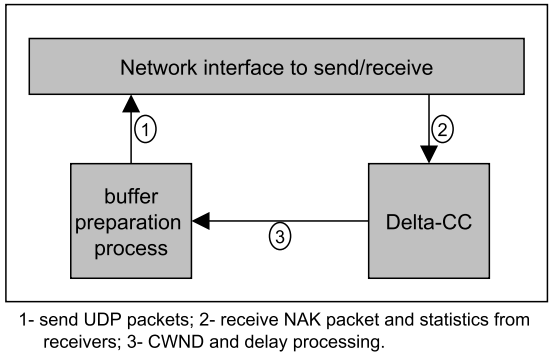

Figura 5.7: Diseño de la arquitectura servidor.

receptores asociados a esa dirección multicast. El mecanismo de control de congestión Delta-CC inicializa parámetros antes de enviar paquetes multicast. Principalmente, en la inicialización se considera un tamaño óptimo de ventana de congestión (CWND) y un retardo de tiempo a usar entre cada CWND para ajustar la tasa de envío. El módulo *buffer preparation process* obtiene la información desde el Delta-CC y ajusta el tamaño de paquete a usar en los próximos envíos. Este mecanismo permite reducir considerablemente la pérdida de paquetes porque Delta-CC puede ralentizar la tasa de envío desde los emisores. El total de datos a transmitir son fragmentados en paquetes UDP de 8 o 32 KB de acuerdo al tráfico de la red y si existen envíos simultáneos de varios emisores. El algoritmo asigna un número de secuencia a cada paquete antes ser enviado. Este número de secuencia es usado para contabilizar la cantidad de datos que se han enviado en un determinado tiempo.

5.4.3. Mecanismo de control de congestión: Delta-CC

En la subsección 5.3.1 se comentan las causas por las que se pueden perder paquetes. Como se ha comentado, la pérdida de paquetes puede ser especialmente significativa en un entorno con múltiples comunicaciones

multicast. Se hace necesario añadir un mecanismo de control de congestión en estos entornos, aquí se propone un mecanismo basado en ventana. La Figura 5.8 muestra un ejemplo en el que dos nodos envían datos de forma simultánea a un nodo receptor. El mecanismo de control de congestión debe dar soporte de fiabilidad en la entrega de datos.

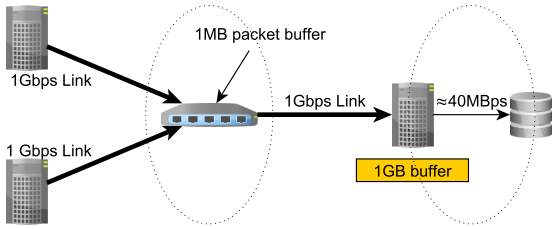

Figura 5.8: Escenario con dos emisores y un receptor.

En primer lugar, el sistema realiza un proceso de calibración para calcular:

- La tasa máxima de envío sin pérdida de paquetes (throughput crítico) para uno o más emisores. El retardo de tiempo entre cada CWND se determina por la velocidad de envío de los emisores.
- Tamaño óptimo de ventana CWND cuando uno o más emisores envían de forma simultánea.
- Ancho de banda de los receptores.

Usualmente el conmutador no tiene información para determinar el comportamiento dinámico del tráfico de paquetes. Cuando sólo un emisor envía paquetes, el conmutador los reenvía a los puertos correspondientes para entregar los paquetes a los nodos interesados en el tráfico. Pero, cuando más de un emisor trata de enviar paquetes a un mismo puerto de destino, el conmutador comienza a almacenarlos debido al limitado ancho de banda. Se ha estudiado la forma de evitar la saturación usando la diferencia entre el

número de secuencia del último paquete recibido por el receptor y el último número de secuencia enviado por el emisor, esta diferencia se ha llamado *Delta_Tx_Rx*. En la Figura 5.9 se puede observar cómo al disminuir la tasa de envío de los emisores el valor *Delta_Tx_Rx* disminuye considerablemente, esto significa que el conmutador funciona de forma correcta y la pérdida de paquetes se puede reducir o evitar. El valor *Delta_Tx_Rx* se calcula por una hebra en el lado del emisor con la información de control recibida desde los receptores. En el lado del receptor, la pérdida de paquetes puede ser prácticamente nula con éste mecanismo.

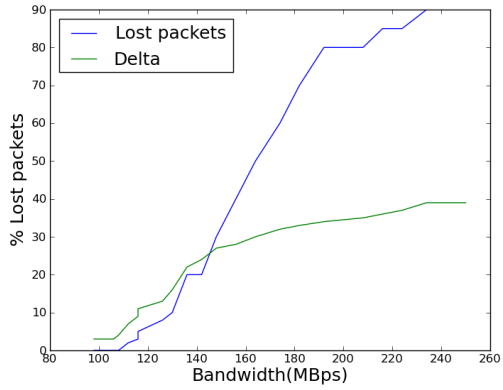

Figura 5.9: Valor *Delta* y pérdida de paquetes con diferentes ancho de banda.

Con el fin de dar a conocer en mayor detalle el esquema de control de congestión Delta-CC, a continuación se describe el funcionamiento del algoritmo.

Periódicamente, el receptor envía información de control a los emisores, donde se incluye: marcas de tiempo, última secuencia recibida, total de paquetes recibidos, y total de paquetes perdidos.

En el lado emisor se crean dos hebras:

- La hebra0 envía paquetes UDP con un tamaño de CWND y espera

un retardo T_{CWND}.

- La hebra1 recibe la información de control desde los receptores, y calcula el tamaño de CWND y T_{CWND} que la hebra0 debe utilizar en los próximos envíos.

Con esta información el algoritmo obtiene:

- El número de emisores: de acuerdo al ancho de banda recibido por cada canal multicast, el emisor calcula el ancho de banda que el cliente está recibiendo.

- Detección de congestión: si el ancho de banda del receptor excede el ancho de banda crítico o *Delta_Tx_Rx* es más grande que el *umbral_Tx_Rx*, el sistema asume que más de un emisor envía datos y que es necesario disminuir la tasa de envío.

- CWND: El CWND usado con un emisor es de 30 paquetes y con más de un emisor, es de 2 ó 3 paquetes.

- T_{CWND}:

  - Con un emisor, se usa T_1 (el retardo mínimo entre cada CWND) si el búfer de recepción está por debajo del 80 % de ocupación y de otra manera se usa T_{1w} (el máximo retardo entre cada CWND). El valor T_1 permite enviar a la máxima velocidad de envío de un emisor, mientras que T_{1w} permite enviar a la mínima velocidad del receptor más lento.
  - Con más de un emisor, si el búfer de recepción está por debajo del 80 % de ocupación, se calcula t para obtener un ancho de banda proporcional al ancho de banda crítico, si no, el algoritmo considera el máximo ancho de banda del receptor más lento.

En la siguiente sección se realizan algunos experimentos para demostrar que con esta propuesta es posible reducir el tráfico TCP unicast siempre y cuando el mecanismo de control de congestión Delta-CC se mantenga con un rendimiento óptimo.

5.5. Evaluación

En este apartado se describe la configuración utilizada para validar el funcionamiento del protocolo, los experimentos y los resultados obtenidos para los escenarios de estudio.

5.5.1. Configuración

La configuración consta de diferentes grupos multicast. El número de grupos multicast se determina por el número de servidores. Por ejemplo, si la configuración permite N servidores, G es el número de grupos multicast. El número de receptores en cada grupo multicast es determinado por la redundancia r. Por lo tanto, $receptores = G \,.\, r$. Por ejemplo, la configuración de la Figura 5.5 tendría 36 receptores.

5.5.2. Experimentos y resultados

Para obtener un rendimiento óptimo, los emisores deben enviar el máximo número de paquetes que la red puede procesar, y, al mismo tiempo, tratar de evitar la pérdida de paquetes para el correcto funcionamiento de las aplicaciones y evitar la penalización del rendimiento global del protocolo. Los resultados obtenidos en este análisis han sido de una configuración real en un cluster de computadores en que los nodos están interconectados por una red Gigabit. Los nodos en la configuración se interconectan con un conmutador Dlink DGS-1248T con un búfer de paquetes de 1 Mbyte y es capaz de reenviar hasta 71.4 Mpaquetes/s, y se ha usado la versión 1 de IGMP Snooping. La configuración de los nodos consta de: CPU de 2 x

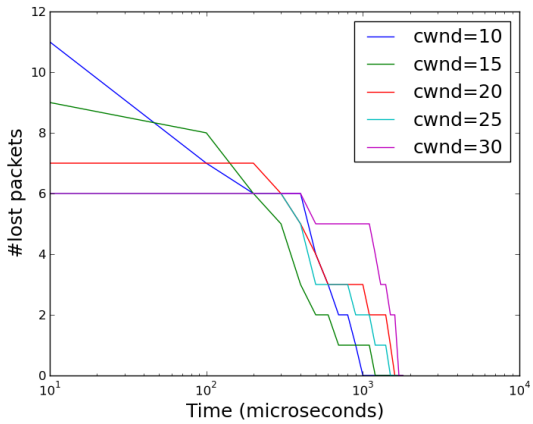

Figura 5.10: Pérdida de paquetes con diferentes tamaños de CWND

1.87GHz Intel Xeon E5320, Memoria RAM de 4GB, Disco Local de 512GB y Sistema Operativo CentOS 6.6.

En la evaluación se han considerado dos escenarios. En el primer escenario un emisor envía paquetes de 32KB a todos los receptores. El nodo receptor almacena los datos en el búfer de recepción con una elevada tasa de pérdida de paquetes cuando el emisor envía datos a la máxima velocidad que permite la red. Para reducir la pérdida de paquetes, el algoritmo agrega un pequeño retardo entre cada CWND antes de hacer el envío. La tasa de pérdida de paquetes puede ser reducida hasta 0. La Figura 5.10 muestra la relación de pérdida de paquetes, mientras que la Figura 5.11 el ancho de banda obtenido de acuerdo al retardo introducido entre cada CWND para diferentes tamaños de CWND. Como se muestra en los resultados, es posible conseguir una pérdida de paquetes 0 con un ancho de banda hasta de 100 MB/s, y con un tamaño de CWND de 30 paquetes.

En el segundo escenario, dos emisores envían datos simultáneamente a los receptores, sin un tiempo de retardo entre cada tamaño de CWND y para un tamaño de paquete grande (entre 16 y 32 KB). En este caso, el búfer

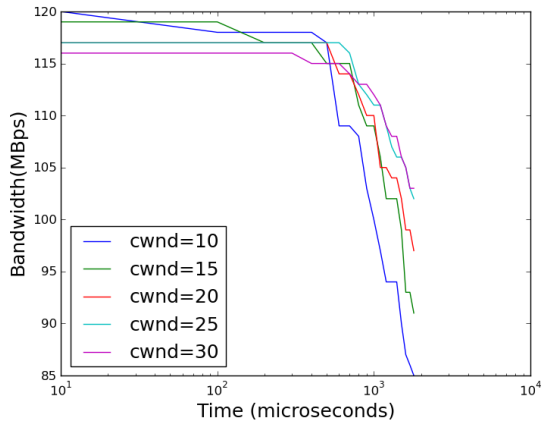

Figura 5.11: Ancho de banda obtenido con diferentes tamaños de CWND

del conmutador se satura rápidamente debido a que el propio conmutador no tiene la capacidad de almacenar y reenviar los paquetes a los puertos de destino. Para realizar estos experimentos ha sido necesario incluir barreras de sincronización en los envíos de los nodos emisores. Con estos tamaños de paquete (entre 16 y 32 KB), el conmutador se congestiona rápidamente porque el ancho de banda de los emisores alcanza hasta 200 MB/s. En este caso, la solución óptima es disminuir el tamaño de paquete a 8KB y usar un tamaño de CWND más pequeño, de tal modo que la tasa de envío de los emisores disminuya para evitar la congestión del conmutador. Los experimentos utilizan un tamaño de CWND de 2, 3 y 4 paquetes de 8KB.

En la Figura 5.12 se muestran los resultados de ancho de banda de emisor y receptor obtenidos para un tamaño de CWND de 2 paquetes. Como se muestra en los resultados, la media de ancho de banda obtenido en el emisor está por debajo de los 100MB/s y la pérdida de paquetes es igual a 0. En tanto, en la Figura 5.13 la media del ancho de banda obtenido en el receptor está ligeramente por encima de los 100MB/s con una pérdida de paquetes igual a 0. En la Figura 5.14 la media de ancho de banda del

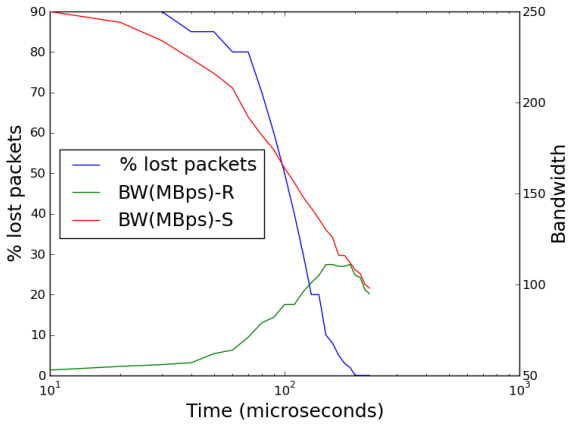

Figura 5.12: Ancho de banda y pérdida de paquetes de emisor y receptor con CWND=2

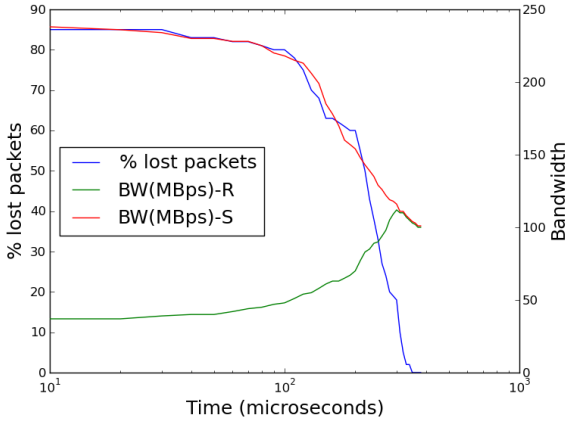

Figura 5.13: Ancho de banda y pérdida de paquetes de emisor y receptor con CWND=3

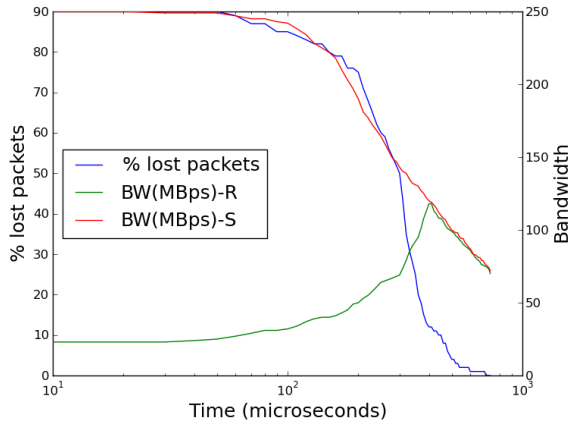

Figura 5.14: Ancho de banda y pérdida de paquetes de emisor y receptor con CWND=4

receptor está sobre los 70MB/s para conseguir una pérdida de paquetes igual a 0. Estos resultados se han obtenido con el objetivo de obtener una pérdida de paquetes igual a 0. En consecuencia, con un tamaño de CWND de 2 paquetes, se consigue una pérdida de 0 con un tiempo de retardo menor, esto significa que la tasa de envío de los emisores se reduce hasta evitar la congestión en el conmutador. Siguiendo esta comparativa, se puede deducir que los tamaños de CWND óptimos son de 2 y 3 paquetes (Figuras 5.12 y 5.13) porque con un tamaño de CWND de 4 paquetes o superior el ancho de banda del receptor disminuye considerablemente hasta obtener una pérdida de paquetes igual a 0.

Con el fin de detectar el momento en que los emisores deben reducir la tasa de envío, el algoritmo calcula la diferencia (*Delta*) entre el último número de secuencia recibido en el receptor y el último número de secuencia enviado por el emisor. Este mecanismo permite que el algoritmo ajuste la tasa de envío del emisor. Como se ha observado anteriormente, en la Figura 5.9 se muestra la relación del valor *Delta* y la pérdida de

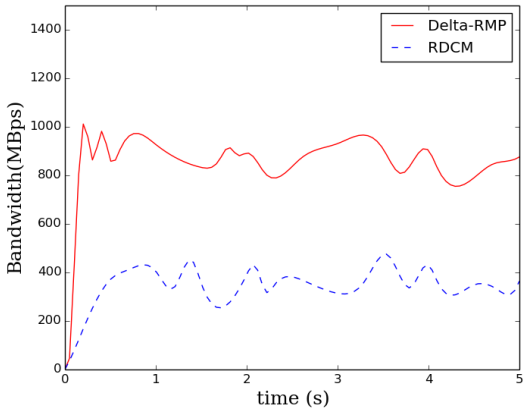

Figura 5.15: Comparativa del protocolo Delta-RMP con el protocolo RDCM.

paquetes de acuerdo a la tasa de envío del emisor. Como se puede observar, cuando la tasa de envío es alta, el valor *Delta* es grande y la pérdida de paquetes es alta. A manera de comparativa, en la Figura 5.15 se muestra la representación del ancho de banda efectivo del protocolo propuesto Delta-RMP y el protocolo RDCM más reciente encontrado en la bibliografía con características similares a la propuesta en este capítulo. RDCM tiene un bajo rendimiento debido a que mantienen una configuración en árbol para las retransmisiones entre los nodos.

A continuación se trata de mostrar cómo el protocolo Delta-RMP puede ajustar el ancho de banda a los valores óptimos para minimizar o evitar la pérdida de paquetes. En la Figura 5.16, se muestra cómo el mecanismo de control de congestión ajusta la tasa de envío de acuerdo a la evolución del tiempo, cuando sólo transmite un emisor es posible usar la máxima tasa de envío, y, como se puede observar en el tiempo 2 y 4, el protocolo puede ajustar la tasa de envío cuando dos y hasta tres emisores envían de forma simultánea. La Figura 5.17 muestra el ancho de banda cuando los receptores tienen ocupado un 80 % del búfer. De tal modo que, la tasa de envío de los

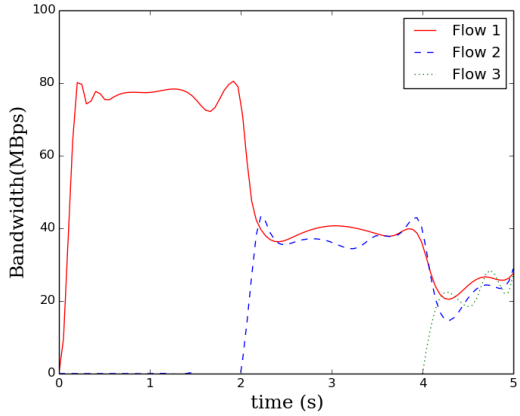

Figura 5.16: Ancho de banda con alta tasa de envío

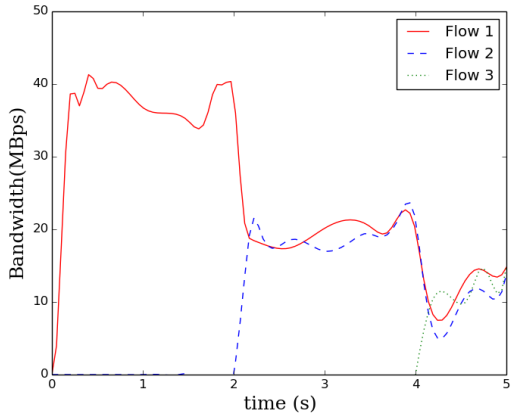

Figura 5.17: Ancho de banda con tasa limitada de envío

emisores se ajusta de acuerdo a las condiciones que presenta el sistema.

Capítulo 6

Protocolo Delta-RMP con soporte de diferentes tecnologías de almacenamiento.

En el presente capítulo se muestra una versión mejorada del protocolo aplicado al almacenamiento en red. Se trata de obtener los mejores resultados de acuerdo a la configuración hardware, tanto del conmutador, la red, y de los discos. En los resultados se utilizan los valores óptimos para sacar provecho de las comunicaciones multicast, de tal manera que se consideran diferentes escenarios de configuración entre los nodos del sistema.

6.1. Introducción

Es importante en los sistemas de ficheros distribuidos garantizar la entrega de los datos a todos los nodos de almacenamiento. En la actualidad es posible aprovechar los recursos hardware disponible en los cluster de

computadores (procesamiento, almacenamiento, red, etc.). Las aplicaciones pueden hacer uso de los dispositivos de almacenamiento en los nodos del cluster de computadores [36]. Estos dispositivos normalmente están conectados por redes de transmisión con un elevado ancho de banda capaz de procesar grandes cantidades de información, de hasta Gb/s. Desde décadas atrás hasta hoy día han aparecido diversos modelos de almacenamiento en red [29] [25] [100] [36] [90]. De forma particular, estas implementaciones utilizan para el envío de datos entre los nodos del sistema el modelo de comunicación TCP/IP. Por ejemplo, TCP/IP es usado para distribuir datos de forma paralela a todos los nodos del almacenamiento en PVFS2. En otros sistemas, tales como los que se pueden ver en [100] y [90] se transmiten y replican datos con este mismo modelo de comunicación.

En el diseño de la capa de comunicaciones de un sistema de almacenamiento en red es común utilizar el protocolo TCP Aunque TCP asegura la entrega de todos los datos usando control de flujo y congestión, presenta el problema de que, si la pérdida de paquetes es alta, el rendimiento de la red se degrada drásticamente, como se describe en [32]. Por consiguiente, para obtener mejoras importantes, en la actualidad se desarrollan protocolos para evitar, en gran medida, los problemas de rendimiento de TCP. Este nuevo enfoque de desarrollo de capa de comunicaciones para sistemas distribuidos se basan en el protocolo UDP [38]. Las comunicaciones basadas en UDP permiten enviar datos con menos carga de las cabeceras que las comunicaciones basadas en TCP. Sin embargo, UDP puede penalizar seriamente la fiabilidad del envío, debido a que no mantiene mecanismo de control de flujo y congestión que permitan evitar la pérdida de paquetes. Aunque presenta esta desventaja, a nivel de aplicación es posible considerar la implementación de estos mecanismos para la entrega eficiente de datos.

Con el desarrollo de protocolos basados en UDP es posible distribuir datos a grupos específicos de receptores. Como se ha comentado en otras secciones, las comunicaciones multicast permiten la distribución de los

mismos datos en un solo envío hacia distintos receptores distribuidos geográficamente o dentro de una red local. En la literatura actual se pueden encontrar implementaciones de protocolos multicast diseñados para entornos WAN, de hasta cientos o miles de receptores. Esto implica un incremento de latencia en la comunicación debido a que las comunicaciones deben atravesar diversos enrutadores para llegar a los receptores. La principal ventaja del uso de comunicaciones multicast es que puede mejorar la transferencia de datos cuando se requiere transferir los mismos datos desde un destino a múltiples nodos receptores, además puede reducir la sobrecarga de la red significativamente. Sin embargo, las comunicaciones multicast no garantizan la entrega de los datos a todos los receptores. Adicionalmente, si el tráfico de la red es alto, la probabilidad de pérdida de paquetes también se incrementa drásticamente. En este entorno de comunicación, los principales factores de pérdida de paquetes pueden ser: el búfer del conmutador o el bus de E/S del disco duro cuando un receptor recibe transferencias más rápido que la velocidad de escritura del disco y el bus intermedio de almacenamiento temporal de datos está al menos un 80 %.

Las aplicaciones que se ejecutan en un entorno de cluster deben satisfacer dos condiciones primordiales para aprovechar todos los recursos hardware: a) distribuir datos tan rápido como las condiciones de la red lo permiten, y b) asegurar la entrega de los datos a todos los receptores.

En este capítulo se presenta una mejora del protocolo propuesto y presentado en el Capítulo 5, lo cual consiste en aprovechar el almacenamiento disponible en los nodos para asegurar la calidad de servicio, QoS (*Quality of Service*), de las aplicaciones. El protocolo Delta-RMP permite replicar datos a diferentes grupos multicast. Al mismo tiempo se hace un análisis sobre la influencia de la tecnología de almacenamiento en el rendimiento global del protocolo.

6.2. Trabajos previos

En la computación en cluster, para asegurar la calidad del servicio, las aplicaciones requieren almacenar los datos en diferentes nodos de la arquitectura. Muchos de los beneficios de la computación cluster se han analizado en [99] y [49]. En [99] y [49] se mencionan dos de las características principales que se consideran en este capítulo: alta disponibilidad y alto rendimiento. Alta disponibilidad se enfoca principalmente en almacenar datos en nodos específicos de la arquitectura. La replicación de datos en diferentes nodos es el mecanismo más común para garantizar alta disponibilidad. Por otro lado, los sistemas de almacenamiento en red también deben garantizar alto rendimiento. Tal y como se menciona en [99], el rendimiento se mide como el tiempo promedio para satisfacer la petición del cliente. El tiempo promedio incluye el tiempo de comunicación de red. Este tiempo se ve afectado por el tráfico en la red.

6.2.1. RMP propuestos

En la literatura especializada se pueden encontrar propuestas de protocolos multicast fiables para entornos de red. En el Capítulo 4 se hace un resumen de algunos protocolos, además se describen los mecanismos de control de congestión y fiabilidad que utilizan. Algunos de los protocolos usan un solo canal multicast para transferir datos a los diferentes receptores de un grupo específico. Además, el protocolo propuesto por Kasera [54] utiliza un canal multicast para las retransmisiones en caso de pérdidas de paquetes independiente del canal utilizado para la transferencia de los datos.

El protocolo UDT propuesto en [47] se basa en UDP. UDT es un protocolo fiable para el transporte de datos a nivel de aplicación, que permite la distribución de datos en aplicaciones de procesamiento intensivo de datos sobre redes WAN rápidas. El protocolo UDT está orientado a la conexión. En consecuencia, no es posible la transferencia de los mismos datos desde

un emisor a diferentes nodos de forma simultánea.

La descripción anterior de los protocolos no refleja el uso de las tecnologías de almacenamiento existentes. Es decir, sólo son dedicados a la transferencia de los datos. Por otra parte, en este capítulo se plantea el uso eficiente del almacenamiento de los nodos de un cluster. Este planteamiento se sigue con el uso de múltiples transferencias simultáneamente.

6.2.2. Comunicación en cluster

Cuando se mide el rendimiento de aplicaciones paralelas que se ejecutan en cluster de computadores, el sistema de comunicación es comúnmente uno de los principales factores que limitan la velocidad de ejecución de las aplicaciones, principalmente cuando se utilizan redes comerciales con poco ancho de banda.

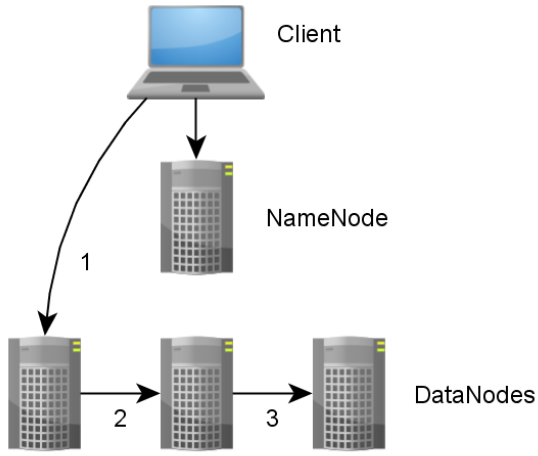

Figura 6.1: Modelo de replicación en HDFS.

En [97] [30] [84] se han estudiado algunas de las características de los sistemas de almacenamiento en red. En estos trabajos se puede observar que el protocolo TCP sigue siendo usado en la entrega de los datos de una

manera fiable. En el Capítulo 3 se han descrito algunas de las desventajas de TCP, así como los beneficios que se podrían obtener si se utiliza UDP, en particular multicast, en la capa de comunicaciones de los sistemas de almacenamiento en red.

En la Figura 6.1 se muestra el modelo de replicación de HDFS y en la Figura 6.2 para DRBD se puede distinguir en que medida, las comunicaciones multicast, pueden ayudar a disminuir el número de canales de comunicación y al mismo tiempo el tráfico redundante.

En el sistema de ficheros distribuido Hadoop todos los protocolos de comunicación se basan en TCP/IP. Un cliente establece conexión a un puerto TCP configurable con el *NameNode*, para este establecimiento de conexión se utiliza el protocolo *ClientProtocol*, mientras que la comunicación entre el *NameNode* y el *DataNode* se inicia por el *DataNode* mediante el protocolo *DataNode*. El protocolo RPC engloba tanto al protocolo *ClientProtocol* como al *DataNode* usados en HDFS. Por otro lado, para replicar un bloque de datos (comúnmente de 64MB, aunque es configurable) de un fichero con un factor de replica 3, el cliente hace conexión con el *NameNode* para recuperar la lista de *DataNodes* a ser usados para las réplicas. En primer lugar, el cliente envía datos al primer DataNode de la lista devuelta por el NameNode, el primer *DataNode* empieza a recibir bloque de datos de 4KB y, cuando cada bloque se escribe en disco este *DataNode* envía cada bloque al segundo DataNode de la lista. De igual forma, el segundo DataNode cada vez que escribe en disco cada bloque recibido, lo envía al tercer *DataNode*. Por lo tanto, el modelo de replicación de datos en HDFS, requiere tantos canales de comunicación dado el factor de réplica. Las comunicaciones para las réplicas en HDFS se realizan mediante el protocolo *DataTransferProtocol*. Por otra parte, para realizar las réplicas de datos con el modelo *Three-way* que se describen en la Figura 6.2 se requieren al menos dos conexiones TCP.

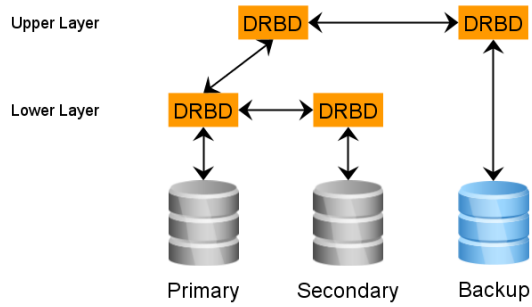

Figura 6.2: Modelo de replicación *Three-Way* en DRBD.

6.3. Diseño del sistema Delta-RMP

6.3.1. Motivación

Tanto la tecnología de cluster de computadores como los sistemas de almacenamiento en red tienen un importante rol en HPC. De tal modo que permite otros modelos como la computación en la nube, centros avanzados de procesamiento de datos o computación intensiva de datos, entre otros. Se han realizado grandes esfuerzos para mejorar el rendimiento global de la computación cluster, por ejemplo, a nivel de aplicación [41], consumo energético [24] [104] y a nivel de capa de comunicación [38] [46] [53]. Sin embargo, las comunicaciones multicast no se han explotado en ningún sentido, habitualmente, en las redes cluster se utiliza la comunicación basada en TCP cuando se emplea la tecnología Ethernet.

En este ámbito, y debido a los beneficios que multicast ofrece, se ha continuado con la mejora del protocolo propuesto en Capítulo 5 de esta tesis. Básicamente el modelo de comunicación se refiere a un modelo de múltiples transferencias multicast que permite almacenar los datos recibidos en cada uno de los receptores. Lo hace de forma que, el protocolo puede distinguir la tecnología de almacenamiento utilizada (sea disco HDD o SDD) para adaptar las tasas de transferencia de los emisores y así evitar o disminuir la

pérdida de paquetes por el efecto de una baja tasa de escritura.

6.3.2. Diseño de la arquitectura

Para obtener mayor ventaja de las comunicaciones multicast en la configuración es posible crear múltiples grupos multicast con varios receptores en cada grupo, de tal manera que los nodos pueden recibir el tráfico desde más de una dirección. El sistema se ha diseñado de manera de que los nodos puedan transmitir datos a otros nodos de tal forma que cada receptor debe distinguir los datos recibidos desde cada dirección multicast. Al mismo tiempo, cada nodo debe almacenar los datos recibidos considerando la disponibilidad del búfer de recepción y si en el tiempo está recibiendo de una o más direcciones multicast. En este último aspecto se considera la tecnología de almacenamiento usada, HDD o SSD. La Figura 6.3 muestra un ejemplo básico de la configuración.

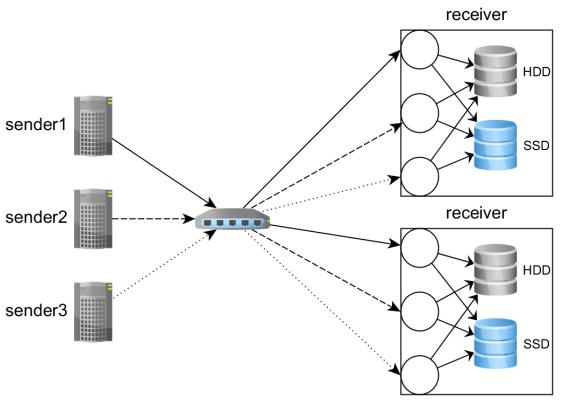

Figura 6.3: Transferencia de datos con distintos grupos multicast y tecnología de almacenamiento.

En un entorno de múltiples transferencias multicast es posible la pérdida de paquetes, las principales consecuencias se describen en la subsección 5.3.1.

Además, en la configuración se utiliza IGMP Snooping para reducir

la carga de la red. IGMP Snooping se incorpora actualmente en los conmutadores y su uso permite mejorar el rendimiento global y hacer un uso eficiente del ancho de banda de la red.

6.3.3. Detalles de la implementación

Los principales detalles de la implementación se han descrito en la subsección 5.4.2. En los siguientes apartados se proporcionan más detalles del cliente y servidor, así como del organigrama.

6.3.3.1. Formato de paquetes

Principalmente, en el protocolo se definen tres tipos de paquetes: paquete de datos (Figura 6.4) usado en el lado del emisor, paquete de retransmisión (Figura 6.4(b)) y paquete broadcast (6.4(c)), usados en el lado del receptor. El rol de cada uno de los campos de las figuras mencionadas anteriormente se describen a continuación.

El rol de los campos (a, b, ..., f) de las Figuras 6.4(a)(b) es el siguiente (ver también la Figura 6.5):

(a) *mc_addr*: Dirección multicast. Cada nodo emisor envía datos sólo a una dirección multicast específica.

(b) *id_block_t*: ID del bloque total. El emisor aleatoriamente asigna un ID a cada bloque total de datos a enviar.

(c) *tt_block*: Tamaño total de bloque. Este campo incluye la carga de datos total a enviar. La carga se divide en bloques de tamaño específico.

(d) *id_block*: ID de bloque. El sistema asigna un número de secuencia a cada bloque que se prepara para envío.

(e) *t_block*: Tamaño de bloque. Tamaño en bytes de cada bloque (en algunas secciones se le llama "paquete").

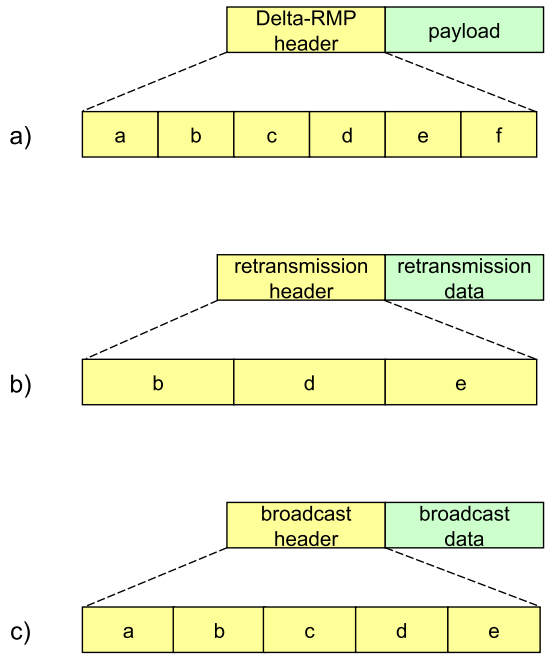

Figura 6.4: Cabecera de los paquetes Delta-RMP: a) paquetes de datos, b) petición de retransmisión y c) paquete broadcast.

(f) *n_blocks*: Número de bloques. Se refiere al tamaño total de bloques para enviar. Es decir: *n_blocks = tt_block / t_block*.

La siguiente descripción de campos (a, b, c) corresponde a la Figura 6.4(c):

(a) *timestamp*: Almacena la marca de tiempo cuando el algoritmo prepara el paquete broadcast para enviar.

(b) *lrs*: Almacena la última secuencia recibida (**l**ast **r**eceived **s**equence) en el socket.

(c) *trp*: Contabiliza el total de paquetes recibidos (**t**otal **r**eceived **p**ackets).

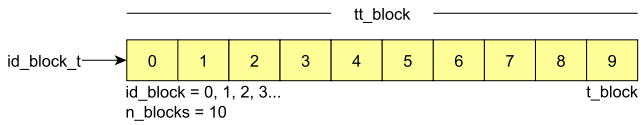

Figura 6.5: Cabecera del paquete de datos en detalle.

(d) *tlp*: Contabiliza el total de paquetes perdidos (**t**otal **l**ost **p**ackets).

(e) *bwd*: Almacena el total de bytes escritos en disco (**b**ytes **w**ritten on **d**isk).

6.3.3.2. Organigrama de Delta-RMP

En la Figura 6.6 se describe la secuencia de los diagramas de flujo del emisor y del receptor.

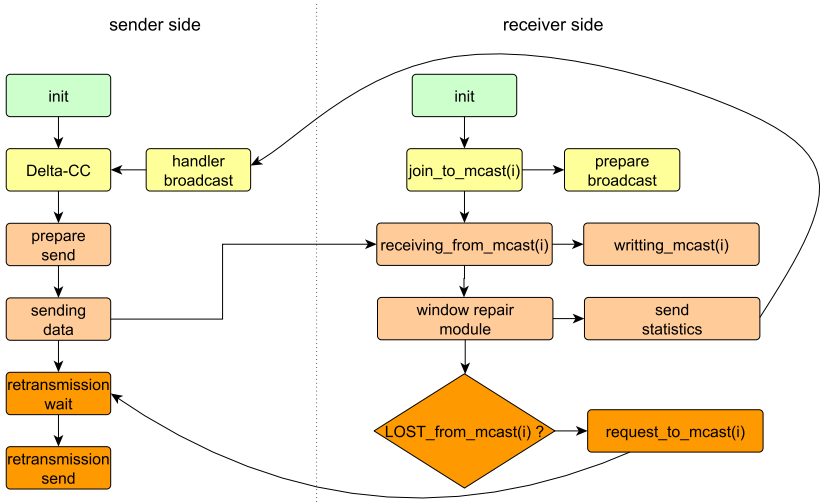

Figura 6.6: Delta-RMP: state flow diagram.

En la etapa de inicialización, el receptor crea dos hebras cuando se asocia

a una dirección multicast. Además, prepara la comunicación del módulo que se encarga de enviar las estadísticas al (los) emisor(es) de donde recibe datos. El cliente incluye una interfaz (primera hebra) para recibir datos desde diferentes emisores. Es decir, el cliente recibe datos desde nodos físicos distintos y periódicamente envía estadísticas a cada uno de estos nodos.

En el Algoritmo 1 se proporcionan más detalles acerca del funcionamiento del receptor en el momento que se asocia a una dirección multicast.

Una vez que el cliente se ha asociado a un dirección multicast y empieza a recibir paquetes, cada paquete recibido se almacena temporalmente en el búfer intermedio e incrementa el contador de paquetes recibidos (líneas 5-6). Debido a que cada paquete recibido incluye el número de secuencia (campo (d) *id_block* de la Figura 6.4(a)(b)), el identificador de bloque se compara con el contador de paquetes (línea 8), si *id_block* > *counter_packets* se notifica al *window repair module* (*wrm*) (línea 9). Como consecuencia a la detección de pérdida, se envía un paquete NAK por un canal TCP dedicado al emisor con la información del paquete perdido (línea 10). Finalmente, *window repair module* obtiene todas las estadísticas (line 14) (que se han descrito en la Figura 6.3(c)) y las envía a los emisores.

La segunda hebra únicamente escribe los datos recibidos desde el búfer temporal al disco e informa del número de bytes escritos en el momento de registrar las estadísticas.

Específicamente, el *window repair module* obtiene todas las estadísticas desde cada hebra y cada hebra se encarga de enviar un mensaje a todos los emisores. Las estadísticas son usadas para prevenir la congestión en el esquema de control de congestión Delta-CC. La congestión se previene en los emisores. Con este mecanismo se puede definir el tamaño de paquete y de la ventana a utilizar según la carga del receptor.

Algoritmo 1 Funcionamiento del receptor en la inicialización y recepción de datos.

```
1: init() {crea thread0 y thread1}
2:
3: thread0: {tarea}
4: while (1) do
5:    buffer = received_packet_i
6:    counter_packets++
7:    {detecta pérdida paquetes}
8:    if id_block > counter_packets then
9:       notifica_a_wrm()
10:      envía_NAK()
11:   else
12:      continue
13:   end if
14:   obtiene_estadísticas()
15: end while
16:
17: thread1: {tarea}
18: escribe_en_disco()
```

En el lado del emisor se incluye una interfaz para enviar datos a una dirección multicast específica. Además, se establece una interfaz (*handler broadcast module*, Figura 6.6) para recibir información que incluye las estadísticas desde los receptores asociados a la dirección multicast. En la inicialización, y antes de realizar el envío de datos, Delta-CC establece algunos parámetros, principalmente, inicializa los valores óptimos para el tamaño ventana de congestión (CWND) y el tiempo de retardo a utilizar entre cada CWND. Una vez que se han negociado los valores óptimos y el protocolo empieza a enviar datos, el *handler broadcast module* temporalmente recibe estadísticas desde los receptores. El módulo Delta-CC se encarga de analizar estas estadísticas y reestablecer (si es necesario) los parámetros para ajustar la tasa de envío. En la sección 6.4 se realiza una

evaluación para demostrar en qué medida este mecanismo puede reducir considerablemente la pérdida de paquetes debido a que el Delta-CC reduce la tasa de envío desde los emisores. A consecuencia de que UDP no establece mecanismos para evitar la congestión, al reducir la tasa de envío, el protocolo Delta-RMP ocupa sólo un porcentaje del ancho de banda disponible de la red. Por lo tanto, es posible la transferencia de datos de otras aplicaciones. La carga total de datos a enviar desde el emisor se fragmenta en paquetes UDP de acuerdo al tráfico de la red. A cada paquete UDP se le asigna un número de secuencia antes de ser enviado. Este número de secuencia también es utilizado para hacer un seguimiento de la cantidad de datos que se ha enviado.

6.3.4. Delta-CC

El mecanismo de control de congestión Delta-CC es usado para prevenir o evitar la congestión de la red. Básicamente, este objetivo se puede lograr con el uso de reconocimientos negativos (NAK) y con ventanas de congestión. Un receptor genera un paquete NAK para solicitar al emisor la retransmisión de un paquete cuando el receptor ha detectado que éste no ha llegado. El uso de NAK para mantener fiabilidad reduce el número de paquetes emitidos por el receptor comparado con el uso de ACK. Por otro lado, la ventana de congestión se utiliza para calcular la capacidad máxima en la transferencia de datos, con el objetivo de evitar la pérdida de paquetes que llevaría a la trasferencia de paquetes NAK. La ventana de congestión se calcula de acuerdo al número de direcciones multicast a las que un receptor se ha asociado. Por ejemplo, si un receptor se ha asociado a tres direcciones multicast, el mecanismo de control de congestión establece valores óptimos de ventana de congestión de acuerdo al comportamiento de los receptores de estas tres direcciones..

El conmutador puede ser la causa de pérdida de paquetes. Si los emisores comienzan a enviar paquetes con una alta tasa de envío, el

conmutador comienza a almacenarlos e intenta reenvíar estos paquetes a los puertos correspondientes para distribuirlos a los receptores. En este caso, si los emisores simultáneamente envían datos, el búfer del conmutador podría no tener la capacidad suficiente para almacenar todos los paquetes que llegan por cada puerto. Por lo tanto, el conmutador puede ser potencialmente la causa de pérdida de paquetes debido a que normalmente no tiene información para regular el comportamiento dinámico del tráfico de paquetes. Por otra parte, si los emisores envían datos a la máxima capacidad de envío (en el caso de Gigabit puede ser de hasta 120MB/s), el sistema puede experimentar una elevada pérdida de paquetes. Para resolver este problema, y para aprovechar el uso de números de secuencia, es posible usar la diferencia $lpr - lps$, donde lps es el número de secuencia del último paquete recibido en el lado del receptor y lps es el número de secuencia del último paquete enviado por el emisor. En el Capítulo 4 se describe la manera en que se ha calculado el valor $Delta\_Tx\_Rx$ que corresponde a la diferencia $lpr - lps$ mencionada anteriormente.

El mecanismo Delta-CC se ejecuta en el lado del emisor y periódicamente recibe información de control (estadísticas) desde los receptores. Cada paquete de información de control (Figura 6.3(c)) se captura en tiempo real por cada hebra en el lado del receptor y se envía a los emisores. Una vez la información de control se encuentre en el modulo Delta-CC, éste calcula (teniendo en cuenta el número de emisores):

1. La tasa máxima de envío de datos (o tasa de envío crítica) sin pérdida de paquetes, la tasa de envío se ajusta dinámicamente incluyendo un retardo de tiempo T_{CWND} entre cada $CWND$

2. El tamaño de ventana de congestión óptimo (CWND)

3. El ancho de banda de los receptores en lo que respecta a los datos que recibe a través de la red, y el ancho de banda de escritura en disco de los receptores cuando se utiliza discos HDD o SSD.

En el Algoritmo 2 se explica el funcionamiento de Delta-CC. En el momento que se inicializa el emisor se crean dos hebras. La hebra0 envía paquetes UDP (thread0 en la línea 31) de acuerdo al tamaño de CWND y espera un retardo T_{CWND} para enviar los próximos datos. La hebra1 calcula: el número de emisores en cada receptor y el ancho de banda agregado que en el tiempo recibe el receptor (líneas 4-5); valor óptimo de $CWND$ y T_{CWND} (líneas 7-23), si el número de emisor es 1 el algoritmo asigna el primer valor óptimo de CWND y para calcular T_{CWND} si $buffer < 80\,\%$ se calcula T_1, en caso opuesto se calcula T_{1w}. T_1 y T_{1w} es el retardo a usar entre cada $CWND$ en el caso de un emisor. Cuando en el sistema envía más de un emisor de forma simultánea aumenta la probabilidad de congestión, por lo tanto, el algoritmo asigna un CWND menor al caso anterior. Además, para este caso, si $buffer < 80\,\%$ se calcula t_1 y en caso opuesto se calcula t_{1w}. T_1 y t_1 se utilizan para una tasa máxima de envío, y, T_{1w} y t_{1w} para una tasa mínima de envío. Con la finalidad de detectar congestión (líneas 25-28) Delta-CC compara el ancho de banda de los receptores y la diferencia $Delta\_Tx\_Rx$ con el ancho de banda crítico $critial\_BW$, como la tasa máxima de recepción de los receptores, y con el $umbral\_Tx\_Rx$ para detectar congestión en el conmutador, respectivamente. Delta-CC actualiza los parámetros $CWND$ y T_{CWND} si el ancho de banda de los receptores o $Delta\_Tx\_Rx$ es mayor que el ancho de banda crítico o $umbral\_Tx\_Rx$, respectivamente. El módulo *handler_broadcast()* es el encargado de recibir la información de control que se envía desde los emisores (línea 2).

Algoritmo 2 Control de Congestión Delta-CC

Require: estadística de receptores

Ensure: $CWND$ y T_{CWND}

```
1: init() {crea thread0 y thread1}
2: handler_broadcast()
3: {thread1:} calcula
4: n_senders = ∑ i
5: BW_r = ∑ BW_i
6: {calcula CWND}
7: if n_senders = 1 then
8:    CWND = o_CWND
9:    {calcula T_CWND}
10:   if buffer < 80 % then
11:      calculate_T_1()
12:   else
13:      calculate_T_1w()
14:   end if
15: else
16:   CWND = o_CWND
17:   {calcula T_CWND}
18:   if buffer < 80 % then
19:      calculate_t_1()
20:   else
21:      calculate_t_1w()
22:   end if
23: end if
24: {detección congestión}
25: if BW_r > critial_BW or Delta_Tx_Rx > umbral_Tx_Rx then
26:   {ralentiza velocidad}
27:   CWND = optimal_CWND
28:   T_CWND = optimal_T_CWND
29: end if
30: {thread0:} tarea
31: send_UDP_packets()
```

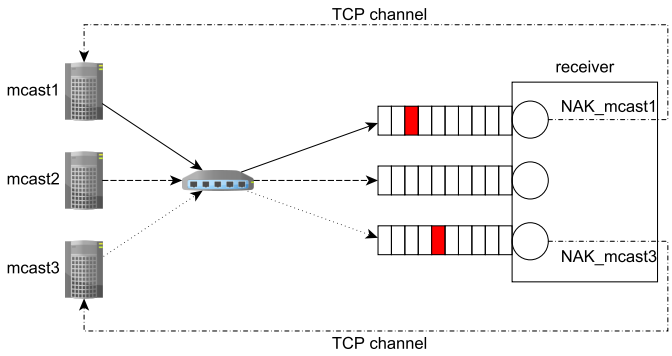

Figura 6.7: Control de flujo basado en NAK con múltiples envíos multicast.

6.3.5. Mecanismo de fiabilidad

La mejora del rendimiento de la red se puede obtener con las comunicaciones multicast. En comunicaciones TCP es necesario enviar un ACK por cada paquete recibido. Los esquemas basados en NAK pueden reducir el tráfico de la red. Además, como se ha estudiado en [106] los mecanismos fiables basados en NAK tienen mejor escalabilidad y mejor rendimiento de red que los basados en ACK.

En nuestra propuesta es posible generar múltiples paquetes NAK (si es necesario) desde cualquier receptor. Cuando un cliente detecta la pérdida de la secuencia de un paquete, el receptor envía un paquete NAK al emisor para solicitar la retransmisión del paquete. Todas las retransmisiones se realizan por un canal TCP. Por ejemplo, si un receptor se ha asociado a tres direcciones multicast (es decir, recibe datos desde tres nodos físicos) este debería establecer tres canales TCP, Figura 6.7.

Si Delta-CC puede evitar pérdidas excesivas de datos, el protocolo utilizará un menor ancho de banda.

6.4. Evaluación de rendimiento

En esta sección se usan dos configuraciones o casos de estudio para evaluar el protocolo Delta-RMP: (A) en esta configuración cada receptor se asocia a sólo una dirección multicast, es decir, sólo recibe datos desde una dirección; (B) en esta otra configuración cada receptor se asocia a más de una dirección multicast, es decir, cada receptor recibe datos desde diferentes direcciones.

6.4.1. Configuración del cluster

Debido a que un sistema de almacenamiento normalmente se implementa en un cluster de computadores, por lo tanto se han realizado evualuaciones para conocer el comportamiento del protocolo en un entorno real de arquitectura cluster. En particular, se ha utilizado la misma configuración de la subsección 5.3.4, pero incluyendo discos SSD. Los discos añadidos son discos SSD Kingston con capacidad de 120GB Firmware 521ABBF0 y una tasa de escritura de hasta 450MB/s.

6.4.2. Detección de pérdida de paquetes

La primer parte de la evaluación consiste en analizar la pérdida de paquetes en el cluster para las dos configuraciones, (A) y (B). En este caso no se ha utilizado el control de congestión Delta-CC con la intención de identificar el máximo ancho de banda que el protocolo podría permitir sin pérdida de paquetes. Con este experimento es posible determinar a partir de qué momento no existe pérdida de paquetes de acuerdo al tamaño de CWND y retardo entre cada CWND usados. Con estos parámetros y con un tamaño óptimo de paquete es posible evitar la saturación en el búfer del conmutador. El tamaño óptimo de paquete de acuerdo a los experimentos realizados es de 32KB para el primer caso y 8KB para el segundo caso. Consecuentemente, si se reduce la pérdida de paquetes tanto como sea

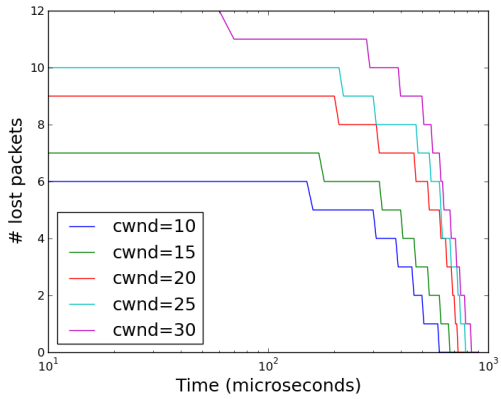

Figura 6.8: Pérdida de paquetes para los discos HDD.

posible el control de congestión Delta-CC puede actuar de una forma más eficiente. Para demostrar una mayor eficiencia del protocolo en cuanto a las escrituras en disco, se ha estudiado en qué medida las escrituras continuas en disco afectan el rendimiento global a causa de la pérdida de paquetes. De tal manera que el protocolo pueda hacer uso del máximo ancho de banda para obtener el mejor rendimiento de las comunicaciones multicast. Para las dos configuraciones de estudio se han utilizado discos HDD y SDD de manera separada para determinar la eficiencia del protocolo de acuerdo a la tecnología de almacenamiento utilizada. Aunque, de antemano se puede deducir que los discos SSD pueden aportar mejores beneficios, es necesario ajustar parámetros de escritura para tomar mayor ventaja de la tecnología.

Dado que existe la posibilidad de que sea un solo emisor el que envía datos a R receptores. El receptor, cada vez que recibe un paquete, lo escribe directamente en disco, por ejemplo, de 32KB para ambos discos en el primer caso (A). De manera particular, se ha comprobado que las escrituras en disco no afectan el rendimiento global del sistema. En un sistema de almacenamiento es necesario asegurar la escritura de los datos en el disco para responder las demandas de disponibilidad. Por lo tanto,

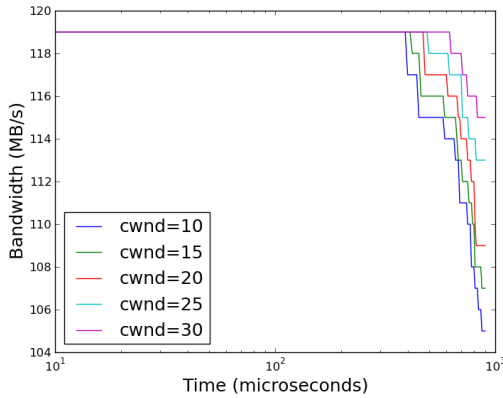

Figura 6.9: Ancho de banda obtenido según la pérdida de paquetes para los discos HDD.

se ha realizado un análisis en detalle para conocer el valor óptimo de sincronización (*sync_size*). En este caso es posible evaluar con tamaños de CWND de 10, 15, 20, 25 y 30 paquetes. El algoritmo inserta un retardo en microsegundos entre cada CWND para reducir la tasa de envío hasta obtener un pérdida de paquetes de 0. En las Figuras 6.8 y 6.9 se muestran los mejores resultados obtenidos. El mejor tamaño de ventana CWND a utilizar es de 30 paquetes, que corresponde a un ancho de banda por encima de los 115MB/s. En cuanto al rendimiento del disco HDD, el valor óptimo de *sync_size* es de 64 paquetes, que corresponde a un ancho de banda de escritura ligeramente por encima de los 22MB/s.

Con esta misma configuración, pero con el uso de la tecnología SSD, es posible mejorar ligeramente el rendimiento de la red. Como consecuencia, el retardo de tiempo usado entre cada CWND se reduce considerablemente. En la Figura 6.10 se muestra la reducción de la pérdida de paquetes de acuerdo al retardo utilizado en cada CWND. El protocolo con los discos SSD permite experimentar menos pérdida de paquetes que los discos HDD con un ancho de banda de red alrededor de 119MB/s para todos los tamaños de

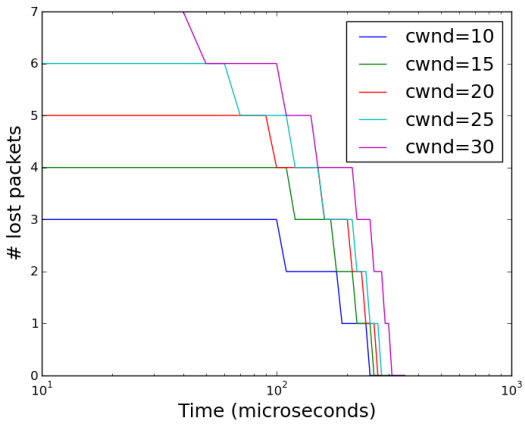

Figura 6.10: Pérdida de paquetes para los discos SSD.

CWND. Además, como se puede observar en la Figura 6.10, es posible llegar a una pérdida de paquetes 0 con un tiempo de retardo menor entre cada CWND. El ancho de banda de escritura de los discos SSD está por encima de 140MB/s con un tamaño de *sync_size* de 128 paquetes para obtener el mejor rendimiento del disco.

La segunda configuración estudiada (B) que consiste en que un receptor puede recibir tráfico desde más de un emisor. Debido a la poca capacidad del búfer del conmutador, este caso puede ser un evento crítico para el sistema, además, la probabilidad de la pérdida de paquetes se incrementa considerablemente. Por lo tanto, se ha estudiado en detalle el comportamiento del protocolo bajo estas condiciones. Para realizar este análisis se han realizado una serie de experimentos y se han encontrado dos valores óptimos de CWND, 2 y 3 paquetes.

En cuanto al análisis con los discos SSD, la Figura 6.11 con CWND 2 y la Figura 6.12 con CWND 3, muestran el ancho de banda comparativo de emisor y receptor, así como la pérdida de paquetes que se obtiene. Los resultados muestran, para el caso de los discos SSD, que el valor óptimo de

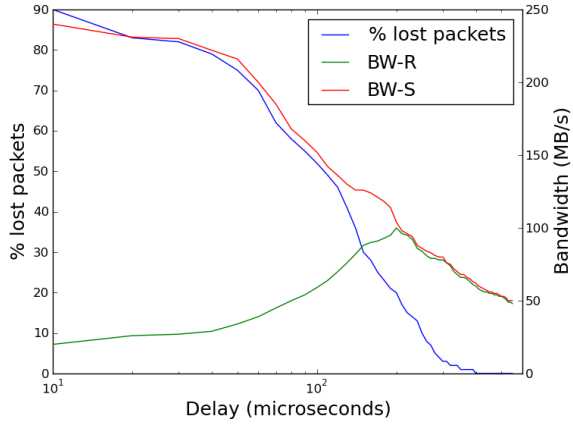

Figura 6.11: Ancho de banda de emisor y receptor, y pérdida de paquetes para los discos SSD con un CWND=2

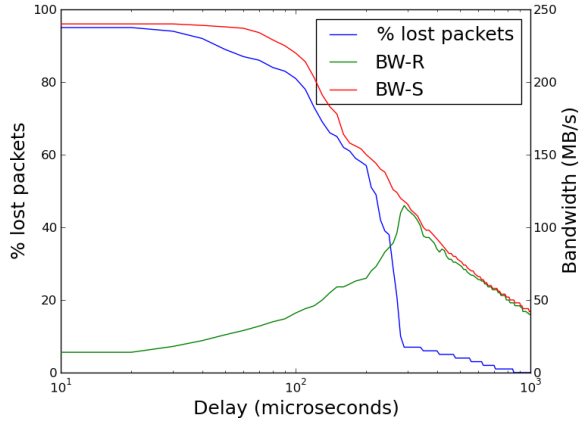

Figura 6.12: Ancho de banda de emisor y receptor, y pérdida de paquetes para los discos SSD con un CWND=3

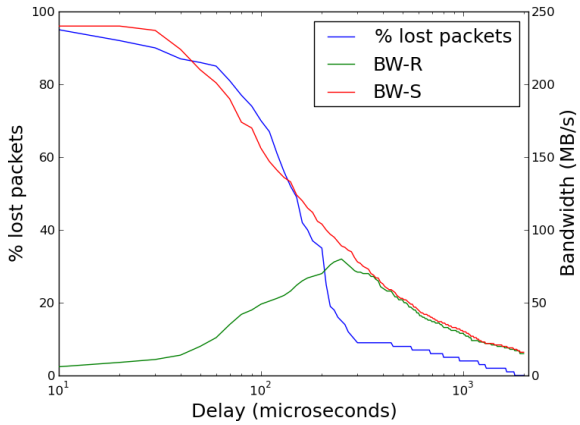

Figura 6.13: Ancho de banda de emisor y receptor, y pérdida de paquetes para los discos HDD con un CWND=2.

CWND es de 2 paquetes donde se alcanza un ancho de banda de red por encima de 50MB/s con una pérdida de paquetes 0, Figura 6.11. En cuando al tamaño de ventana CWND 3, en la Figura 6.12 se puede ver que el ancho de banda disminuye por debajo de 50MB/s para obtener una pérdida de paquetes de 0. De acuerdo al análisis, se plantea que no es necesario realizar un estudio con un CWND mayor de 3 puesto que las prestaciones globales se ven afectadas de una manera considerable, tanto en el rendimiento de la red como del disco duro.

Acerca del uso de los discos HDD, en la Figura 6.13 y la Figura 6.14 se muestra cómo el ancho de banda de la red decrementa debido al bajo ancho de banda de escritura de los discos, además de la latencia agregada por cada escritura en disco. En este caso, para obtener una pérdida de paquetes 0, el ancho de banda de la red se reduce hasta aproximadamente 18MB/s para un tamaño de ventana CWND 2 y 15MB/s para CWND 3.

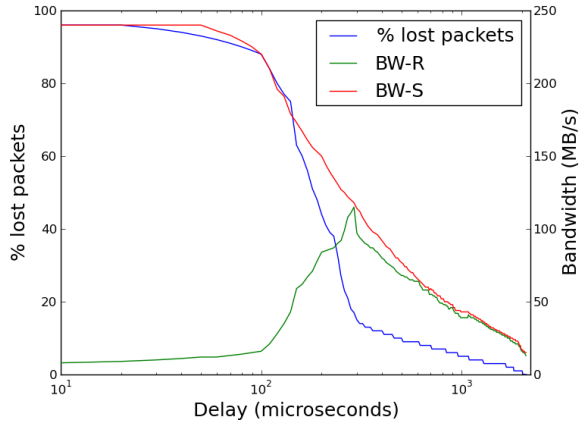

Figura 6.14: Ancho de banda de emisor y receptor, y pérdida de paquetes para los discos HDD con un CWND=3.

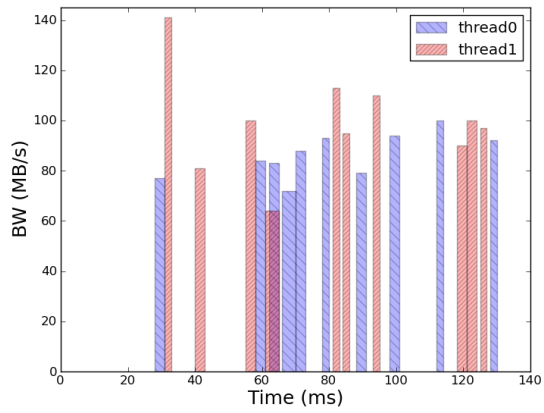

Figura 6.15: Comportamiento del disco SSD cuando dos hebras escriben.

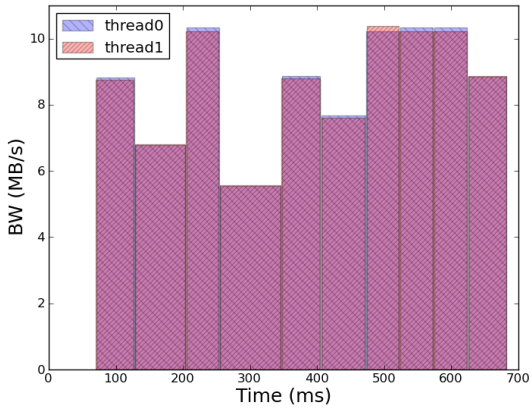

Figura 6.16: Comportamiento del disco HDD cuando dos hebras escriben.

6.4.3. Análisis del comportamiento de los discos.

Para obtener los resultados previos, se ha analizado el comportamiento de los discos HDD y SSD para demostrar la manera que el protocolo puede tomar una mayor ventaja de los discos SSD. En la Tabla 6.1 se muestran resultados con diferentes tamaños de paquetes y de sincronización para discos SSD. De tal manera que, el ancho de banda promedio del disco puede llegar hasta 86.3MB/s. En la Tabla 6.1, con la relación de *packet_size* $>$ *8* y *sync_size* $>$ *32* es posible obtener el mejor rendimiento del disco pero con pérdida de paquetes porque el tiempo de cada sincronización se incrementa. Por otro lado, para *packet_size* $<$ *8* y *sync_size* $<$ *32* el protocolo no experimenta pérdida de paquetes, pero el rendimiento del disco queda por debajo del 50 %. Respecto a los discos HDD, en la Tabla 6.2 se hace un resumen de los resultados obtenidos en el análisis. Los resultados obtenidos con los discos HDD muestran en qué medida el bajo rendimiento de estos discos puede penalizar el rendimiento global del protocolo.

Para ambos tipos de disco, los mejores valores de *packet_size* y *sync_size* son 8KB y 256KB respectivamente. En la Figura 6.15 y la Figura 6.16 se

Tabla 6.1: Análisis de rendimiento de los discos SSD con el protocolo.

packet_size \ sync_size	4	8	16	32	64	128
4	9.07	15.56	27.9	58.4	84.6	92.5
8	14.8	29.6	53.4	**86.3**	101.5	114.1
16	37.3	46.4	89.5	117.8	129	124.3
32	64.9	97.8	123.4	113	123.3	102.9

Tabla 6.2: Análisis de rendimiento de los discos HDD con el protocolo.

packet_size \ sync_size	4	8	16	32	64	128
4	1.1	1.5	2.0	4.2	5.8	6.5
8	1.2	1.8	3.1	**6.3**	8.8	10.3
16	3.1	3.8	6.7	7.9	11.2	12.6
32	6.9	7.5	9.1	9.9	13.4	14.2

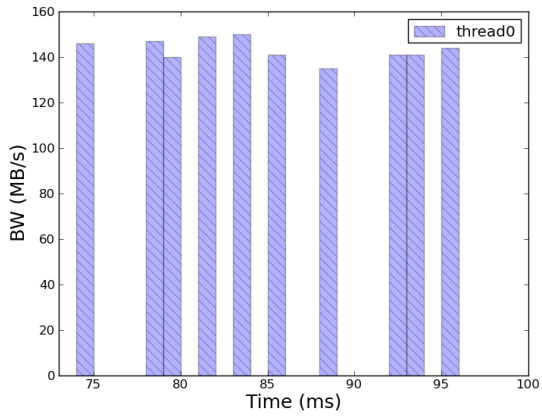

Figura 6.17: Comportamiento del disco SSD cuando escribe una hebra.

muestra el rendimiento global de los discos SSD y HDD respectivamente cuando dos hebras escriben en el disco de manera simultánea, es decir, cuando reciben datos desde dos direcciones multicast.

Con objeto de que se pueda usar para comparar, la Figura 6.17 presentan resultados para discos SSD para el caso en el que escribe sólo una hebra. En este caso se ha utilizado un tamaño de paquete de 8KB y un tamaño de sincronización de 512KB. Se ha elegido este tamaño de sincronización para aprovechar las prestaciones de estos discos. Según las pruebas realizadas, si se utiliza un tamaño menor de sincronización el rendimiento del disco no supera los 110MB/s. El ancho de banda máximo que puede alcanzar el disco SSD con los parámetros de evaluación está por encima de los 140MB/s (ver Figura 6.17). La Figura 6.18 presenta los resultados del comportamiento del disco HDD cuando escribe una hebra. En cuanto a los discos HDD se han establecido los valores de tamaño de paquete y sincronización a 8KB y 1024KB respectivamente. Usar un tamaño de sincronización menor a 1024KB implica un peor rendimiento del disco HDD.

También, se ha realizado una evaluación de los discos para un caso en el

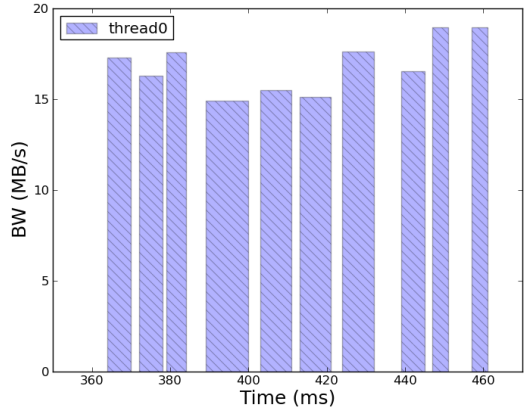

Figura 6.18: Comportamiento del disco HDD cuando escribe una hebra.

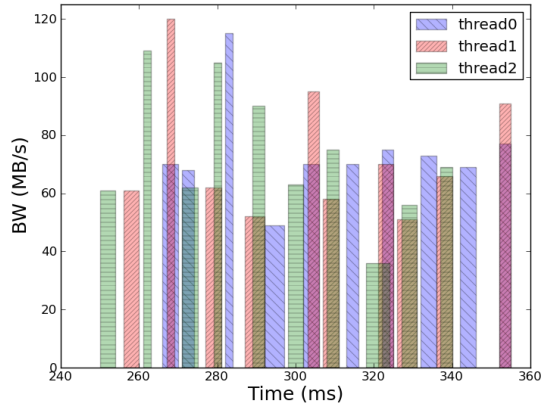

Figura 6.19: Comportamiento del disco SSD cuando escriben tres hebras.

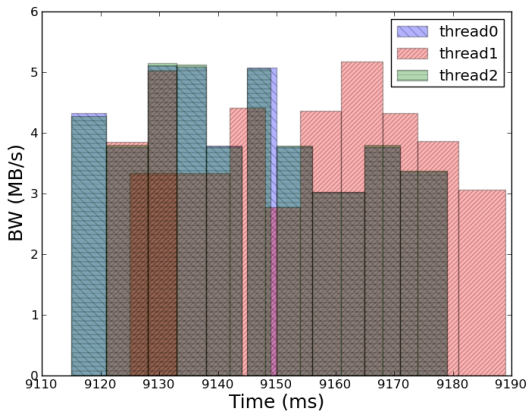

Figura 6.20: Comportamiento del disco HDD cuando escriben tres hebras.

que tres hebras escriben simultáneamente. En cuanto al tamaño de paquete y sincronización se han establecido a 8KB y 256KB respectivamente para ambos discos. Usar valores menores o mayores empeoran el rendimiento del disco. La Figura 6.19 muestra los resultados del disco SSD. De tal manera que el ancho de banda máximo está alrededor de 120MB/s. En el caso de los discos HDD, en la Figura 6.20 se presentan los resultados. En estos resultados se puede apreciar en que medida la latencia incluida por el movimiento de las cabezas del disco penaliza seriamente el rendimiento del disco. El máximo ancho de banda del disco supera ligeramente los 5MB/s. Mientras que la sumatoria del ancho de banda en una marca de tiempo está por debajo de los 15MB/s.

Con el fin de conocer el comportamiento real de los discos HDD y SSD se ha realizado una evaluación del comando *dd*. La orden *dd* es ampliamente conocida y utilizada para evaluar el rendimiento de los discos de acuerdo a parámetros establecidos. Los fines de interés de la prueba ha permitido utilizar los parámetros *bs* (tamaño de bloque), *count* (número de bloques) y *fdatasync* (que permite asegurar la escritura física en disco). En la Tabla

Tabla 6.3: Evaluación del comportamiento de los discos con la orden *dd* de acuerdo al número de hebras de escritura según los parámetros establecidos.

No. hebras	BW SSD	BW HDD
1	130	23.1
2	146.4	21.8
3	156.6	21.2

6.3, se presentan los resultados de la evaluación según el número de hebras de escritura que se utiliza. En todos los casos el tamaño del bloque total de escritura es de 10MB (es decir, bs = 8KB y count = 1280).

6.4.4. Evaluación de envío/recepción simultánea

Los resultados de esta subsección se han obtenido con los siguientes parámetros: *packet_size* = 8KB, *CWND* = 2 paquetes, *sync_size* = 32 paquetes, y para aprovechar la capacidad de almacenamiento se han utilizado discos *SSD*. El análisis se ha realizado para conocer el comportamiento de los nodos cuando de forma simultánea envían datos a una dirección multicast específica y a su vez reciben datos desde dos o tres nodos diferentes, según la configuración de las Figuras 6.21 y 6.23. Además, según la velocidad de envío de cada emisor en las configuraciones se asegura que no hay pérdida de paquetes. El tamaño total de envío comprende desde 1MB hasta 256MB.

En la Figura 6.21 se muestra un ejemplo de configuración con tres nodos. En este caso, los nodos node-0, node-1 y node-2 simultáneamente envían datos a diferentes direcciones multicast. Al mismo tiempo, los tres nodos también son receptores de datos de direcciones multicast diferentes a las que estos envían. Por ejemplo, de manera particular el node-0, envía datos a la dirección multicast 239.6.6.0 y al mismo tiempo recibe datos de las direcciones multicast 239.7.7.0 y 239.8.8.0. Es decir, previamente

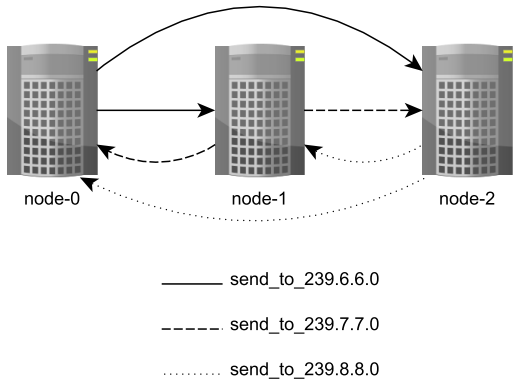

Figura 6.21: Envíos simultáneos desde 3 emisores a 2 receptores.

node-0 se ha asociado a ambas direcciones. Los resultados obtenidos para la configuración (en este caso del nodo-0) se muestran en la Figura 6.22. Como se puede ver en la figura, el ancho de banda del emisor no supera los 25MB/s, mientras que el ancho de banda agregado de recepción que se obtiene en todas las hebras está por debajo de 50MB/s. El consumo de ancho de banda total para el node-0 está sobre el 60 %.

La Figura 6.23 muestra una situación más compleja. Es decir, cada uno de los nodos ahora recibe tráfico desde tres direcciones multicast. Por lo tanto, es posible una saturación más rápida en los receptores. De igual forma, las escrituras en disco aumentan considerablemente. Si se usa como ejemplo la figura, el nodo-0 envía datos a la dirección multicast 239.6.6.0, y al mismo tiempo recibe datos desde los nodos que envían a las direcciones 239.7.7.0, 239.8.8.0 y 239.9.9.0. La Figura 6.24 muestra los resultados de la situación más compleja que se puede presentar. Considerando el nodo-0, en esta configuración, dada la posible saturación del nodo debido a las continuas recepciones y a su vez el envío de datos, es necesario disminuir la velocidad de envío para evitar la pérdida de paquetes. Por lo tanto, se puede lograr un ancho de banda de envío alrededor de 19MB/s, mientras

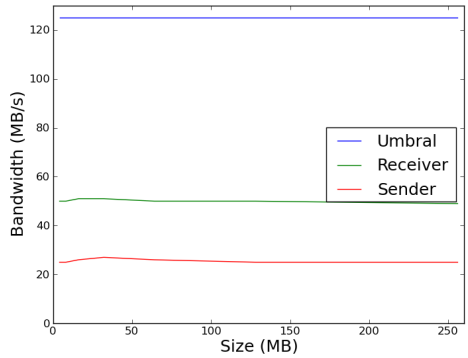

Figura 6.22: Envíos simultáneos desde 3 emisores a 2 receptores.

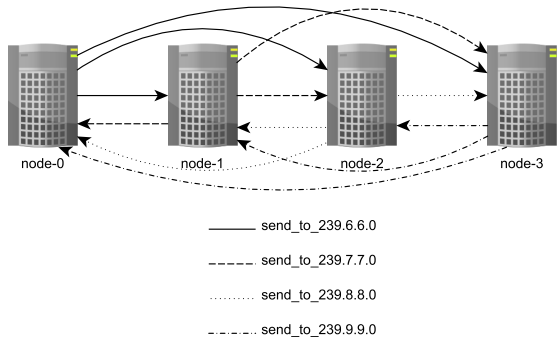

Figura 6.23: Envíos simultáneos desde 4 emisores a 3 receptores.

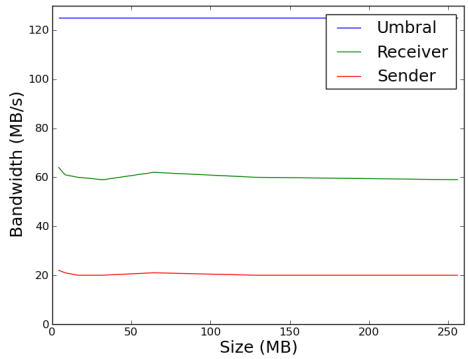

Figura 6.24: Envíos simultáneos desde 3 emisores a 2 receptores.

que el ancho de banda agregado de recepción para el mismo nodo-0 está ligeramente por encima de los 60MB/s. De tal manera, que el nodo-0 consume aproximadamente el 63 % de ancho de banda total.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

Para lograr la mejora de prestaciones de la interfaz de comunicación de los sistemas de ficheros en red se han realizado diferentes tareas que han permitido seguir una línea específica para tal fin.

En primer lugar, se han realizado una serie de estudios con el fin de conocer las principales elementos para la comunicación que utilizan los sistemas de ficheros en red:

- Como una de las primeras aproximaciones se han estudiado técnicas de *channel bonding* y SDP en Infiniband. En el caso de *channel bonding* es posible incrementar el ancho de banda pero al mismo tiempo existe la posibilidad de incremento de tráfico en la red. SDP sobre Infiniband permite obtener ancho de banda elevado y disminuir la sobrecarga de la CPU. En este trabajo se ha considerado no seguir la línea con SDP debido a que ha dejado de formar parte del desarrollo principal de Infiniband para Linux (OFED) y que no existe soporte directo para

redes Gigabit.

- Otra de las posibilidades que puede apoyar la mejora de prestaciones de la interfaz de comunicaciones ha sido el estudio de técnicas de compresión de datos. En este sentido, se ha identificado el aporte de la compresión en la reducción de tráfico en la red y cómo afecta en los tiempos globales de comunicación.

- Se han estudiado los beneficios de las comunicaciones multicast en entornos de redes Gigabit. Además, se han estudiado las propuestas de protocolos existentes para identificar las características principales de tal manera que nos permitiera definir una solución viable para hacer frente al objetivo de la tesis.

- Se ha diseñado e implementado un protocolo multicast fiable que permite el envío simultáneo de datos a destinos específicos de la configuración de red. De tal manera que disminuye el tráfico de la red en gran medida.

- Se ha desarrollado un modelo que permite el uso eficiente del soporte hardware del conmutador. Además, el modelo garantiza baja tasa de pérdida de paquetes con un elevado ancho de banda en los receptores. Para obtener los mejores resultados se ha realizado un análisis en detalle de los posibles factores que influyen en la pérdida de paquetes en redes multicast.

- Se ha definido un protocolo multicast fiable, que permite el envío de datos desde muchos emisores simultáneamente a nodos interesados en recibir el tráfico, asegura baja tasa de pérdida de paquetes y que además se ajusta a la tecnología de almacenamiento disponible en el sistema. El protocolo permite dar soporte de redundancia de datos entre los nodos de almacenamiento. Los resultados obtenidos pueden satisfacer las demandas de los sistemas de almacenamiento en red.

7.2. Trabajo futuro

En las posibles actividades para continuar con la línea de mejoras Delta-RMP y conocer las beneficios que puede ofrecer a un sistema de almacenamiento real se destacan:

- La realización de un estudio detallado del modelo de replicación que usa HDFS para intentar incluir el protocolo desarrollado en esta tesis. Aunque nuestra implementación se ha realizado usando el lenguaje C, se intentará integrar el código fuente independientemente de que HDFS ya esté implementado en Java. Para conseguir esto se puede utilizar *The Java^{TM}Native Interface* [63]. La funcionalidad del protocolo en HDFS puede potenciar la mejora de la distribución de los datos.

- Adaptación del protocolo al sistema de ficheros AbFS para satisfacer las demandas de replicación de datos entre los servidores. Además, incluir técnicas de compresión de datos para disminuir más la carga de la red.

- La realización de un estudio sobre la configuración y ejecución del protocolo de una forma automática independientemente de la tecnología del conmutador o del almacenamiento presente en la configuración del sistema, usando algoritmos adaptativos que evalúen en tiempo real el estado del sistema.

Bibliografía

[1] http://people.sc.fsu.edu/~jburkardt/datasets/hbsmc/hbsmc.html.

[2] Channel bonding. http://www.linuxfoundation.org/collaborate/workgroups/networking/bonding.

[3] Dlspi: Database for large-scale peptide identification. http://compbio.eecs.wsu.edu/MR-MSPolygraph/.

[4] Ganglia monitoring system. http://ganglia.info/.

[5] Heartbeat. http://linux-ha.org/wiki/Heartbeat.

[6] Human genome. http://hgdownload.cse.ucsc.edu/downloads.html#human.

[7] The matrix market web site. http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/.

[8] Noaa. ftp://ftp.ncdc.noaa.gov/pub/data/noaa/.

[9] DRBD: Distributed replicated block device. http://drbd.linbit.com/.

[10] *The* DRBD *Users Guide 8.4*.

[11] Xml benchmark. http://www.xml-benchmark.org/.

[12] Cloud data management interface. Technical report, Apr. 2010.

[13] D. A. R. P. Agency. TCP: Transmission control protocol. Technical report, Sep, 1981. RFC 793.

[14] M. Anguita and A. F. Díaz. Gestión de metadatos en sistemas de ficheros distribuidos v.0.1.informe interno. departamento de arquitectura y tecnología de computadores. grupo de investigación casip. universidad de granada. 2011.

[15] I. T. Association et al. *InfiniBand Architecture Specification: Release 1.0*. InfiniBand Trade Association, 2000.

[16] M. P. Atkinson, M. Carpenè, E. Casarotti, S. Claus, R. Filgueira, A. Frank, M. Galea, T. Garth, A. Gemünd, H. Igel, I. A. Klampanos, A. Krause, L. Krischer, S. H. Leong, F. Magnoni, J. Matser, A. Michelini, A. Rietbrock, H. Schwichtenberg, A. Spinuso, and J. Vilotte. VERCE delivers a productive e-science environment for seismology research. *CoRR*, abs/1510.01989, 2015.

[17] J. W. Atwood. A classification of reliable multicast protocols. *I E E E Network*, 18(3):24–34, 2004.

[18] A. B. and J. M. C. Bormann, M. Handley. NACK-oriented reliable multicast (NORM) transport protocol. Technical report, Nov, 2009. RFC 5740.

[19] P. S. B. Callaghan, B. Pawlowski. NFS version 3 protocol specification. Technical report, 1995.

[20] P. Balaji, S. Narravula, K. Vaidyanathan, S. Krishnamoorthy, J. Wu, and D. K. Panda. Sockets direct protocol over infiniband in clusters: is it beneficial? In *Performance Analysis of Systems and Software, 2004 IEEE International Symposium on - ISPASS*, pages 28–35, 2004.

[21] A. D. Birrell and B. J. Nelson. Remote procedure call. Technical report, 1981.

[22] A. D. Birrell and B. J. Nelson. Implementing remote procedure calls. *ACM Trans. Comput. Syst.*, 2(1):39–59, Feb. 1984.

[23] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su. Myrinet: A gigabit-per-second local area network. *IEEE micro*, (1):29–36, 1995.

[24] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Zomaya. Energy-efficient data replication in cloud computing datacenters. *Cluster Computing*, 18(1):385–402, 2015.

[25] P. J. Braam. The Lustre storage architecture. 2002.

[26] P. J. Braam et al. The lustre storage architecture. 2004.

[27] H. E. Camacho Cruz. *Incremento de la localidad de datos en Sistemas de Ficheros*. Universidad de Granada, 2013.

[28] P. Carns, W. Ligon, R. Ross, and P. Wyckoff. BMI: a network abstraction layer for parallel i/o. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 8 pp.–, April 2005.

[29] P. H. Carns, W. B. Ligon, III, R. B. Ross, and R. Thakur. PVFS: A parallel file system for linux clusters. In *In Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327. USENIX Association, 2000.

[30] N. Chaturvedi and D. Chandra Jain. Analysis of replication and replication algorithms in distributed system. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(5):261–266, may 2012.

[31] H.-C. Cheng and J.-P. Sheu. Design and implementation of a distributed file system. *Software: Practice and Experience*, 21(7):657–675, 1991.

[32] S. Chowdhury and K. Fatema. Analysing TCP performance when link experiencing packet loss. Master's thesis, University of Gothenburg, Göteborg, Sweden, 2013.

[33] S. Corporation. Portals network programming interface. http://www.cs.sandia.gov/Portals/.

[34] S. Deering. Rfc-1112: Host extension for ip multicasting. *Request For Comments*, 1989.

[35] B. Depardon, G. Le Mahec, and C. Séguin. Analysis of six distributed file systems. 2013.

[36] A. F. Díaz, M. Anguita, H. E. Camacho, E. Nieto, and J. Ortega. Two-level hash/table approach for metadata management in distributed file systems. *The Journal of Supercomputing*, 64(1):144–155, 2013.

[37] A. F. Díaz, J. Ortega, A. Ortiz, G. Garay, and A. Prieto. Improving data sharing on infiniband networks. In *Proceedings of the International Conference on Computational and Mathematical Methods in Science and Engineering,* CMMSE14 *2014 3-7*, Jul 2014.

[38] P. M. Dickens. A lightweight, high performance communication protocol for grid computing. *Cluster Computing*, 13(1):47–66, 2010.

[39] K. R. Fall and W. R. Stevens. *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley, 2011.

[40] W. C. Fenner. Internet group management protocol, version 2. 1997.

[41] R. Filgueira, D. E. Singh, J. Carretero, A. Calderón, and F. García. Adaptive-compi: Enhancing mpi-based applications - performance and scalability by using adaptive compression. *Int. J. High Perform. Comput. Appl.*, 25(1):93–114, Feb. 2011.

[42] T. K. FOUNDATION. Mit kavli institute (MKI) for astrophysics and space research. *URL http://space.mit.edu/*.

[43] A. F. D. García. *Comunicación eficiente en redes de área local para procesamiento paralelo en Clusters*. PhD thesis, Universidad de Granada, 2001.

[44] J. Gemmell, T. Montgomery, T. Speakman, and J. Crowcroft. The PGM reliable multicast protocol. *Network, IEEE*, 17(1):16–22, Jan 2003.

[45] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI (2Nd Ed.): Portable Parallel Programming with the Message-passing Interface*. MIT Press, Cambridge, MA, USA, 1999.

[46] Y. Gu and R. Grossman. Toward efficient and simplified distributed data intensive computing. *Parallel and Distributed Systems,* IEEE *Transactions on*, 22(6):974–984, June 2011.

[47] Y. Gu and R. L. Grossman. UDT: UDP-based data transfer for high-speed wide area networks. *Computer Networks*, 51(7):1777 – 1799, 2007. Protocols for Fast, Long-Distance Networks.

[48] S. Ha, I. Rhee, and L. Xu. Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, July 2008.

[49] I. Haddad. *The* HAS *architecture : a highly available and scalable cluster architecture for Web servers*. PhD thesis, Concordia University, Concordia University Libraries, 2006.

[50] A. Hadoop. Apache hadoop. *URL http://hadoop. apache. org*, 2011.

[51] J. L. Hennessy and D. A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.

[52] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. *SIGCOMM Comput. Commun. Rev.*, 25(4):328–341, Oct. 1995.

[53] M. Kadhum and H. Hassanein. Congestion-aware TCP-friendly system for multimedia transmission based on UDP. *Cluster Computing*, 18(2):693–705, 2015.

[54] S. Kasera, G. Hjalmtusson, D. Towsley, and J. Kurose. Scalable reliable multicast using multiple multicast channels. *Networking, IEEE/ACM Transactions on*, 8(3):294–310, Jun 2000.

[55] M. Ko. iSCSI Extensions for RDMA Specification. Internet-Draft draft-ko-iwarp-iser-02, Internet Engineering Task Force, Dec. 2004. Work in Progress.

[56] I. Kouvelas, B. Cain, B. Fenner, S. Deering, and A. Thyagarajan. Internet group management protocol, version 3. 2002.

[57] A. Kulkarni and R. Bhagwat. Cache coherency in distributed file system. *International Journal of Computer Science and Security (IJCSS)*, 4(1):1, 2010.

[58] R. G. Lane, S. Daniels, and X. Yuan. An empirical study of reliable multicast protocols over ethernet-connected networks. *Performance Evaluation*, 64(3):210 – 228, 2007.

[59] D. J. Leith, R. Shorten, and G. McCullagh. Experimental evaluation of cubic-tcp. In *6th International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2008)*, March 2008.

[60] E. Levy and A. Silberschatz. Distributed file systems: Concepts and examples. *ACM Computing Surveys (CSUR)*, 22(4):321–374, 1990.

[61] D. Li, M. Xu, Y. Liu, X. Xie, Y. Cui, J. Wang, and G. Chen. Reliable multicast in data center networks. *Computers, IEEE Transactions on*, 63(8):2011–2024, Aug 2014.

[62] J. Li and M. Veeraraghavan. May.

[63] S. Liang. *Java Native Interface: Programmer's Guide and Reference*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1999.

[64] F. S. M. Christensen, K. Kimball. Rfc 4541: Considerations for internet group management protocol (IGMP) and multicast listener discovery (MLD) snooping switches, May, 2006.

[65] A. Mankin, V. Paxson, A. Romanow, and S. Bradner. Ietf criteria for evaluating reliable multicast transport and application protocols. 1998.

[66] M. McBride and H. Lui. Multicast in the data center overview, 2012. IETF, Internet Draft.

[67] K. Meth, M. Chadalapaka, J. Satran, and D. L. Black. Internet Small Computer System Interface (iSCSI) Protocol (Consolidated). RFC 7143, Oct. 2015.

[68] W. G. S. Microsystems. Sun grid engine: Towards creating a compute power grid. In *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, CCGRID '01, pages 35–, Washington, DC, USA, 2001. IEEE Computer Society.

[69] V. Miliotis, L. Alonso, and C. Verikoukis. Coopnc: A cooperative multicast protocol exploiting physical layer network coding. *Ad Hoc Networks*, 14:35–50, mar 2014.

[70] C. K. Miller. *Multicast networking and applications*. Addison-Wesley Reading, 1999.

[71] M. N. Nelson, B. B. Welch, and J. K. Ousterhout. Caching in the sprite network file system. *ACM Transactions on Computer Systems (TOCS)*, 6(1):134–154, 1988.

[72] E. Nieto, H. E. Camacho, M. Anguita, A. F. Díaz, and J. Ortega. Fault tolerant PVFS2 based on data replication. In *Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on*, pages 107–112. IEEE, 2010.

[73] B. Nowicki. NFS: Network file system protocol specification. Technical report, 1989.

[74] K. Obraczka. Multicast transport protocols: a survey and taxonomy. *Communications Magazine, IEEE*, 36(1):94–102, Jan 1998.

[75] A. Ortiz, J. Ortega, A. F. Diaz, and A. Prieto. Analyzing the benefits of protocol offload by full-system simulation. In *Parallel, Distributed and Network-Based Processing, 2007. PDP '07. 15th EUROMICRO International Conference on*, pages 229–237, Feb 2007.

[76] H. Osterloh. *IP Routing Primer Plus*. Sams Publishing, 2001.

[77] C. Partridge. *Gigabit networking*. Addison-Wesley Professional, 1994.

[78] S. Paul, K. Sabnani, J.-H. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). *Selected Areas in Communications, IEEE Journal on*, 15(3):407–421, Apr 1997.

[79] G. Pfister. *In Search of Clusters*. Parallel programming computer architecture. Prentice Hall PTR, 1998.

[80] D. Plummer. Ethernet address resolution protocol: Or converting network protocol addresses to 48. bit ethernet address for transmission on ethernet hardware. 1982.

[81] J. Postel. UDP: User datagram protocol. Technical report, Aug, 1980. RFC 768.

[82] B. Quinn and K. Almeroth. Ip multicast applications: Challenges and solutions. 2001.

[83] S. Ramany and V. Yakubov. System and method for real-time balancing of user workload across multiple storage systems with shared back end storage, Apr. 21 2009. US Patent 7,523,286.

[84] L. S. Rani, K. Sudhakar, and S. V. Kumar. Distributed file systems: A survey. *International Journal of Computer Science and Information Technologies*, 5(3):3716–3721.

[85] D. Rosenberg. Open-source lustre gets supercomputing nod. http://www.cnet.com/news/open-source-lustre-gets-supercomputing-nod/.

[86] R. B. Ross, R. Thakur, et al. PVFS: A parallel file system for linux clusters. In *Proceedings of the 4th annual Linux showcase and conference*, pages 391–430, 2000.

[87] P. Schwan. Lustre: Building a file system for 1,000-node clusters. In *PROCEEDINGS OF THE LINUX SYMPOSIUM*, page 9, 2003.

[88] S. Shepler, M. Eisler, D. Robinson, B. Callaghan, R. Thurlow, D. Noveck, and C. Beame. Network file system (NFS) version 4 protocol. Technical report, 2003.

[89] T.-L. Sheu and S.-T. Lin. A multicast retransmission scheme using negative ack in wireless networks. In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 1144–1149, March 2013.

[90] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[91] T. Simpy. Simpy. http://simpy.readthedocs.org/en/latest/.

[92] G. Staples. Torque resource manager. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[93] E. Strohmaier, J. Dongarra, H. Simon, M. Meuer, and H. Meuer†. The TOP500 list. http://www.top500.org.

[94] A. Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 4th edition, 2002.

[95] R. Tang, J. Xiong, J. Ma, and D. Meng. A new way to high performance nfs for clusters. In *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)*, pages 51–55, Dec 2005.

[96] P. D. Team. Parallel virtual file system, version 2. http://www.pvfs.org, 2003.

[97] T. D. Thanh, S. Mohan, E. Choi, S. Kim, and P. Kim. A taxonomy and survey on distributed file systems. In *Networked Computing and Advanced Information Management, 2008. NCM '08. Fourth International Conference on*, volume 1, pages 144–149, Sept 2008.

[98] D. Towsley, J. Kurose, and S. Pingali. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *Selected Areas in Communications, IEEE Journal on*, 15(3):398–406, 1997.

[99] J. Wang and Z. Xu. Cluster file systems: a case study. *Future Generation Computer Systems*, 18(3):373 – 387, 2002. Cluster Computing.

[100] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 307–320. USENIX Association, 2006.

[101] R. Wittmann and M. Zitterbart. *Multicast Communication: Protocols, Programming, & Applications*. Morgan Kaufmann, 2000.

[102] C.-H. J. Wu and J. D. Irwin. *Introduction to Computer Networks and Cybersecurity*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2013.

[103] J. Wu. Improving the throughput of novel cluster computing systems. 2015.

[104] R. Xiong, J. Luo, and F. Dong. Optimizing data placement in heterogeneous hadoop clusters. *Cluster Computing*, pages 1–16, 2015.

[105] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control (BIC) for fast long-distance networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2514–2524 vol.4, March 2004.

[106] K. Yamamoto, Y. Sawa, M. Yamamoto, and H. Ikeda. Performance evaluation of ack-based and nak-based flow control schemes for reliable

multicast. In *TENCON 2000. Proceedings*, volume 1, pages 341–345 vol.1, 2000.

[107] R. Yavatkar, J. Griffoen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proceedings of the Third ACM International Conference on Multimedia*, MULTIMEDIA '95, pages 333–344, New York, NY, USA, 1995. ACM.

[108] X. Zhang, Y. Xu, and S. Jiang. Youchoose: Choosing your storage device as a performance interface to consolidated i/o service. *ACM Transactions on Storage (TOS)*, 7(3):9, 2011.

[109] C. Zhou and S.-N. Yu. Using cluster computers in bioinformatics research. *Journal of Shanghai University (English Edition)*, 7(4):370–374.

[110] W. Zhou. Fault management in distributed systems. 2010.