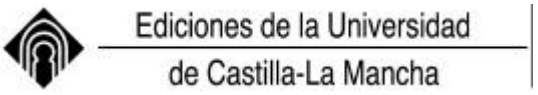
Aurelio Bermúdez Marín

DISEÑO DE MECANISMOS EFICIENTES PARA LA GESTIÓN DE SUBREDES INFINIBAND

I.S.B.N. Ediciones de la UCLM
84-8427-370-9



Cuenca, 2005

Diseño de Mecanismos Eficientes para la Gestión de Subredes InfiniBand

TESIS DOCTORAL
PRESENTADA AL DEPARTAMENTO DE INFORMÁTICA
DE LA UNIVERSIDAD DE CASTILLA - LA MANCHA
PARA LA OBTENCIÓN DEL TÍTULO DE
DOCTOR EN INFORMÁTICA

POR

Aurelio Bermúdez Marín

DIRIGIDA POR

Dr. Rafael Casado González

Albacete, 13 de septiembre de 2004

Agradecimientos

Hace algunos años inicié este viaje que ahora llega a su destino. Afortunadamente, nunca me he sentido solo. Al contrario, son muchos los que me han acompañado, unos durante algunos tramos, y otros a lo largo de todo el recorrido. En cierto modo, esta Tesis también es suya.

Rafa es, sin duda alguna, la persona que más ha influido en el desarrollo de este trabajo. Para empezar, gracias a su insistencia me "enrolé" en esta aventura. Trabajar codo con codo con él después ha sido toda una experiencia. Pero, si en su faceta de investigador y docente es (como poco) brillante, en la de amigo es sobresaliente. Sinceramente, me ilusiona poder seguir trabajando contigo.

También ha sido un privilegio contar en todo momento con la colaboración de Paco Quiles y José Duato. Además de brindarme la oportunidad de comenzar a investigar, cosa que siempre le agradeceré, Paco me ha ayudado a valorar el lado humano de nuestro trabajo. Por otra parte, la experiencia de José nos ha orientado siempre en la dirección acertada. Sus ideas y consejos han sido especialmente valiosos. Aunque sé que nunca podré estar a la altura de ninguno de los dos, espero, al menos, no haberles decepcionado.

Por supuesto, no me olvido del resto de miembros del Grupo de Arquitecturas y Redes de Altas Prestaciones, cuya enumeración empieza a ser imposible. Sin embargo, sí quisiera destacar mi agradecimiento a Antonio Garrido y a José Luis, dos compañeros a los que siempre he podido recurrir.

Por último, y ya en el terreno personal, quisiera agradecer a mi mujer, a mi familia y a mis amigos los esfuerzos realizados y la compresión y paciencia infinitas que han demostrado conmigo. Aunque no siempre es fácil de transmitir, todos ellos han intuido, de una u otra forma, la importancia de lo que estaba haciendo.

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

En los últimos años hemos experimentado un impresionante incremento en las prestaciones que los computadores y las redes de interconexión son capaces de proporcionar. De hecho, las prestaciones de las estaciones de trabajo se duplican cada 18–24 meses. Este hecho ha propiciado el procesamiento distribuido de información empleando clusters de estaciones de trabajo (COW, *Clusters Of Workstation*) [16]. En la actualidad, los clusters suponen una plataforma idónea para la ejecución de aplicaciones distribuidas y paralelas a un coste razonable. La red de interconexión es un elemento crítico en estos sistemas. Algunos ejemplos de subsistemas de interconexión para clusters son Gigabit Ethernet [32], Autonet [72, 80], Myrinet [14], ServerNet [31, 34] y, más recientemente, QsNet [43, 60] e InfiniBand [36, 82]. El principal objetivo de la red de interconexión en estos entornos es proporcionar altas prestaciones (en términos de ancho de banda y latencia) manteniendo, al mismo tiempo, una alta disponibilidad, fiabilidad y escalabilidad.

Un cluster de estaciones de trabajo está sujeto a frecuentes modificaciones, relacionadas con la distribución de sus componentes y/o la congestión de sus recursos. Unas veces, estas modificaciones responden a las necesidades propias de los usuarios, como la conexión y desconexión de dispositivos "en caliente", o la redistribución de la red de interconexión. Otras veces los cambios son, en realidad, fallos en los componentes del sistema. Para garantizar la disponibilidad del cluster, éste requiere la existencia de un mecanismo de gestión que actúe ante condiciones cambiantes. Por supuesto, sería deseable que la intervención de dicho mecanismo perjudicara lo menos posible a las aplicaciones que se ejecutan sobre el cluster.

InfiniBand es una arquitectura de comunicaciones idónea, entre otras cosas, para construir clusters para la ejecución de aplicaciones distribuidas [51, 70]. De hecho, en el mes de noviembre de 2003, una máquina basada en InfiniBand ocupaba la segunda posición en la categoría de clusters del *Top500* [87], y la tercera posición en el ranking absoluto[1]. La especificación de InfiniBand define un entorno SAN en el que múltiples nodos de procesamiento y unidades de E/S son interconectados a través de una red regular o irregular compuesta de conmutadores y enlaces punto a punto.

[1] Entre las diez primeras posiciones nos encontramos, además, con varios sistemas basados en QsNet y Myrinet. Como veremos, estas tecnologías tienen mucho en común con InfiniBand.

Lejos de ser una norma estricta, la especificación deja un amplio espacio de trabajo a fabricantes e investigadores. Uno de los aspectos más importantes y menos detallados de dicha especificación es la gestión de las subredes. InfiniBand define una infraestructura básica de gestión, responsable de configurar y activar la subred, y de detectar y asimilar cualquier cambio topológico, sin intervención del exterior. Asimilar el cambio supone adaptar a la nueva situación topológica las rutas que siguen los paquetes de aplicación para alcanzar su destino. La forma concreta en que las entidades de gestión definidas en la especificación llevan a cabo todas estas tareas está por determinar. El objetivo principal de esta Tesis es contribuir al desarrollo de los primeros mecanismos para la gestión de la subred.

1.1 Motivación

En principio, podrían proponerse multitud de implementaciones para cada una de las tareas relacionadas con la gestión de cambios, todas ellas compatibles con la especificación de InfiniBand. Sin embargo, nosotros estamos especialmente interesados en aquellos diseños que no interfieran en la ejecución normal de las aplicaciones. Consideraremos una implementación concreta más eficiente que otra si el proceso de asimilación del cambio es más transparente a las aplicaciones. Nuestro objetivo es que la red pueda mantener los niveles de calidad de servicio pactados en todo momento, incluso ante la ocurrencia de un cambio en su configuración topológica.

Lo anterior supone que el mecanismo de gestión debe ser capaz de detectar y asimilar el cambio con rapidez. Además, su ejecución no debe dejar sin servicio a las aplicaciones, ni obligar a que parte de los paquetes de aplicación tengan que ser descartados durante el proceso. En una situación ideal, sólo se descartarían aquellos paquetes que viajan por aquellas rutas que han desaparecido. Por supuesto, también es deseable que el tráfico de control necesario para detectar y asimilar el cambio no afecte al tráfico de las aplicaciones, y que el tiempo necesitado por el mecanismo de gestión sea independiente de la carga presente en la red en el momento del cambio.

Además de minimizar el impacto sobre las aplicaciones, nuestros mecanismos intentarán respetar escrupulosamente las reglas dictadas por la especificación de InfiniBand. En este sentido, las implementaciones no deben requerir modificación alguna en el estándar. Tampoco deben requerir la incorporación de nuevos elementos, ni hacer uso de los ya existentes para fines distintos a los que están destinados. Esta filosofía podría obligarnos en algún caso a sacrificar las prestaciones de nuestras propuestas, pero nos asegura la compatibilidad con cualquier producto comercial InfiniBand.

Por último, sería deseable que nuestros mecanismos de gestión fueran fácilmente extrapolables a otras tecnologías de red. Evidentemente, esto será posible en la medida en que dichas tecnologías se asemejen a las características de la arquitectura InfiniBand.

1.2 Objetivos y metodología

Como se ha indicado, el objetivo principal de esta Tesis es proponer implementaciones eficientes para las principales tareas de gestión de la subred en InfiniBand, de cara a obtener un protocolo de asimilación de cambios optimizado.

Este objetivo global puede descomponerse en una serie de objetivos parciales. A continuación describimos dichos subobjetivos y la metodología aplicada para su consecución.

1. **Estudiar la arquitectura InfiniBand**, a través de su especificación, la bibliografía existente y otros materiales como publicaciones científicas o los documentos disponibles en la página web "oficial" de InfiniBand.

2. **Desarrollar y validar un modelo de los componentes propios de una subred InfiniBand**, incluyendo las entidades de gestión de la subred. Para conseguir este objetivo, emplearemos la herramienta de modelado y simulación *OPNET Modeler*.

3. **Diseñar, desarrollar y evaluar un prototipo de mecanismo de gestión de la subred**. Definiremos cada una de las tareas de gestión de forma que sea compatible con la especificación de InfiniBand. El objetivo es disponer de un mecanismo completamente funcional a corto plazo, aunque su eficacia sea baja. Su evaluación nos permitirá detectar los principales cuellos de botella en el proceso de asimilación de cambios.

4. **Ajustar los mecanismos de detección de cambios topológicos definidos en la especificación**, con el objetivo de reducir el tiempo de detección sin introducir una gran sobrecarga en la red.

5. **Proponer y evaluar un mecanismo para la adquisición de la topología de la subred** que minimice el tiempo invertido en la exploración y la sobrecarga producida por los paquetes de control empleados.

6. **Diseñar y analizar un algoritmo para la obtención de rutas para encaminamiento dentro de la subred** que reduzca el tiempo de cómputo total. Demostrar su validez formalmente.

7. **Mejorar el proceso de actualización de las tablas de encaminamiento de la subred**, disminuyendo el perjuicio ejercido sobre el tráfico que soporta. Modelar y evaluar la mejora.

8. **Evaluar mediante simulación el mecanismo de gestión mejorado completo**.

1.3 Desarrollo de la Tesis

El resto de esta memoria cubre los objetivos propuestos, tal y como se describe a continuación.

En el Capítulo 2 y en el Capítulo 3 se describe InfiniBand, centrando nuestro interés fundamentalmente en dos focos. El primero de ellos es el nivel de enlace de la arquitectura, en el que se definen funcionalidades como los canales virtuales, el control de flujo, o el formato de los paquetes de datos. El otro punto de interés es el nivel de gestión de la subred, en el que se definen las entidades participantes y su implicación.

En el Capítulo 4 presentamos el modelo de InfiniBand sobre el que hemos desarrollado y evaluado nuestros mecanismos de gestión. Una vez descrito el entorno de trabajo, se detallan las consideraciones de diseño más relevantes y el modelado de los enlaces, conmutadores, nodos terminales y entidades de gestión.

El Capítulo 5 aborda las primeras tareas relacionadas con la gestión de un cambio en la subred; su detección y la obtención de la nueva configuración topológica. Las prestaciones de los distintos mecanismos de detección se evalúan en función del tiempo de respuesta y del impacto sobre el tráfico de las aplicaciones. En cuanto a los mecanismos de adquisición de la topología, se proponen y evalúan comparativamente dos propuestas basadas en la exploración completa y parcial de la misma.

En el Capítulo 6 se describe y analiza un algoritmo optimizado para la obtención de rutas en InfiniBand. La mejora consiste básicamente en reducir el número de entradas en tablas a computar. Se demuestra formalmente que la propuesta es libre de bloqueo. El nuevo algoritmo de cálculo de rutas se usa como base para proponer un mecanismo de gestión mejorado.

En el Capítulo 7 se describen y evalúan una serie de propuestas para distribuir las tablas de encaminamiento a los conmutadores, alternativas al mecanismo de distribución estática tradicional. Cada propuesta es un refinamiento de la anterior, en el que se reduce la cantidad de recursos afectados por el proceso de actualización, sin incrementar su complejidad.

El Capítulo 8 presenta una evaluación de prestaciones detallada de un mecanismo de gestión para InfiniBand que reúne simultáneamente todas las propuestas formuladas en esta Tesis.

Finalmente, el Capítulo 9 recoge las conclusiones y aportaciones de esta Tesis Doctoral y esboza las posibles líneas de trabajo futuras.

Capítulo 2
La arquitectura InfiniBand

La arquitectura InfiniBand [29, 62, 82] (IBA, *InfiniBand Architecture*) es un estándar comercial de comunicaciones desarrollado por la IBTA (*InfiniBand Trade Association*). Esta asociación, fundada en 1999, engloba a más de 180 compañías entre las cuales se encuentran IBM, Intel, Microsoft, Compaq, HP, Sun, Dell, 3Com, Cisco, NEC, etc. Aproximadamente 100 personas trabajaron durante 14 meses en el desarrollo de la especificación 1.0 de InfiniBand (octubre de 2000). La versión 1.0.a fue presentada en junio de 2001. La versión actual de la especificación (1.1) data de noviembre de 2002 [36]. Durante los años 2001 y 2002 se anunciaron los primeros productos comerciales para InfiniBand.

En este capítulo se da una visión general de esta arquitectura de red, deteniéndonos en los elementos del nivel físico y de enlace, puesto que son los que más nos interesan desde el punto de vista de la gestión de la subred.

2.1 Generalidades

El estándar InfiniBand define un entorno de red de área de sistema (SAN, *System Area Network*) en el que múltiples nodos terminales son interconectados a través de una red conmutada. Dichos nodos se clasifican en unidades de procesamiento y unidades de E/S, y se conectan a la red a través de tarjetas de interfaz (CA, *Channel Adapter*).

La red de interconexión (*fabric*) que comunica los elementos anteriores está jerarquizada en subredes gestionadas de forma autónoma. Cada subred se compone de un conjunto de conmutadores conectados entre sí mediante enlaces punto a punto. La distribución de la topología resultante es totalmente flexible. Incluso pueden utilizarse varios enlaces en paralelo para aumentar el ancho de banda, o proporcionar rutas redundantes para comunicaciones tolerantes a fallos. Diferentes subredes pueden conectarse entre sí mediante encaminadores. La Figura 1 muestra un ejemplo de subred InfiniBand.

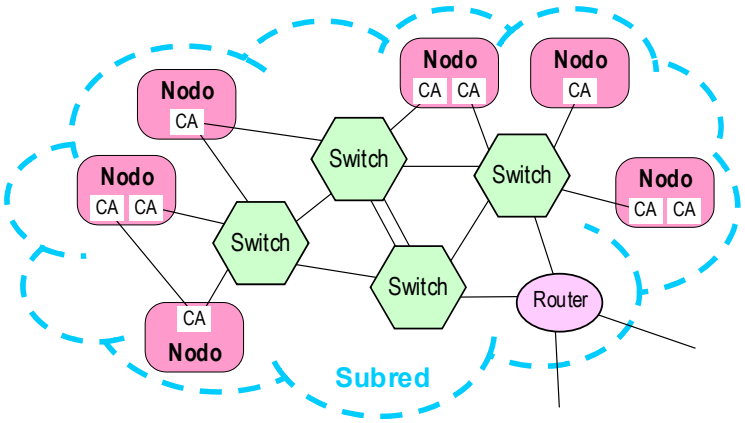


Figura 1. Subred InfiniBand.

Cada subred es gestionada de forma activa, lo que significa que al menos debe existir en la misma un nodo capaz de ejecutar el papel de gestor de la subred (SM, *Subnet Manager*). Su tarea consiste en configurar y administrar la subred. Por ejemplo, el SM es la entidad encargada de computar y enviar las tablas de encaminamiento a los conmutadores, estableciendo de esta forma las rutas a través de la subred.

InfiniBand es una red de altas prestaciones. Por tanto, su objetivo es proporcionar alta productividad y baja latencia, sin que ello suponga una carga adicional para las CPUs dedicadas a la ejecución de aplicaciones de alto nivel. Para ello emplea una serie de protocolos de transporte confiables implementados en hardware, junto con mecanismos sofisticados como *kernel bypass* o *zero-copy*.

2.1.1 Arquitectura de capas

InfiniBand presenta una arquitectura de capas o niveles equivalentes a la arquitectura OSI [85], mostrada en la Figura 2. A continuación se describe brevemente la funcionalidad de cada nivel:

- Nivel físico. Define aspectos como las técnicas de señalización (InfiniBand emplea codificación 8B/10B), las tasas de transferencia (*bit rates*), los tipos de cables y conectores, etc. Los principios del nivel físico se encuentran, junto con las especificaciones mecánicas, en el volumen 2 de la especificación de InfiniBand [36].

- Nivel de enlace. Este nivel define el mecanismo de control de flujo entre enlaces y el encaminamiento de paquetes de datos dentro de la subred, en base a los identificadores locales almacenados en la cabecera de nivel de enlace incluida en cada paquete.

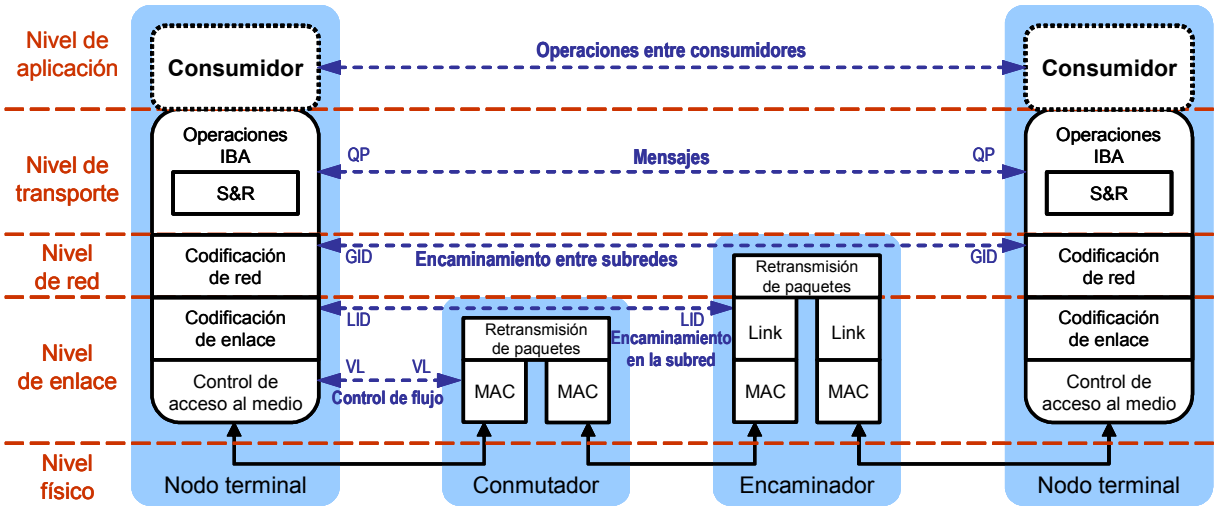


Figura 2. Modelo de comunicaciones InfiniBand.

- Nivel de red. Define cómo son encaminados los paquetes entre subredes. Para ello los dispositivos tienen asignados identificadores globales, que incluyen en la cabecera de nivel de red de los paquetes.

- Nivel de transporte. Proporciona la comunicación entre los nodos terminales, estableciendo conexiones (*channels*) o relaciones conceptuales entre ellos. InfiniBand proporciona cinco tipos de servicio para las conexiones, accedidos a través de pares de colas (QP, *Queue Pair*). Cada par de colas se configura para una determinada clase de servicio. La capa de transporte realiza la segmentación en paquetes de los mensajes de nivel superior, y su posterior reensamblaje.

- Nivel de aplicación. En esta capa se encuentran los consumidores (*consumers*). Un consumidor es un proceso (o hilo de ejecución) que hace uso del hardware de InfiniBand, forme o no parte del sistema operativo. La especificación de InfiniBand denomina consumidor a todo lo que está por encima de la tarjeta de interfaz. Cada consumidor tiene asociado uno o más pares de colas, que emplea para la comunicación con otro u otros consumidores.

2.1.2 Aplicaciones de la arquitectura InfiniBand

Al contrario que otras tecnologías de red de altas prestaciones, que fueron concebidas para aplicaciones específicas, como la creación de redes para almacenamiento (Fibre Channel [28, 90]) o la comunicación en clusters (Myrinet [14]), la arquitectura InfiniBand es adecuada para una amplia variedad de aplicaciones. De hecho, uno de sus objetivos es proporcionar al computador un único tipo de conexión de red para todos los tipos de tráfico, eliminando así la necesidad de emplear múltiples adaptadores (tarjetas Ethernet, interfaces para acceder a clusters de altas prestaciones, adaptadores para redes de almacenamiento, etc.).

La arquitectura InfiniBand está dirigida fundamentalmente hacia tres áreas de aplicación [29]:

- Interconexión de dispositivos de E/S, tales como unidades de disco o puertos hacia otras redes (SAN, *System Area Network*).
- Interconexión de estaciones de trabajo, empleando la subred InfiniBand como una red de área local (LAN, *Local Area Network*).
- Comunicación entre procesos (IPC, *Inter-process Communication*). InfiniBand permite la comunicación directa entre procesos alojados en distintos terminales sin necesidad de emplear el sistema operativo, accediendo de esta forma directamente al hardware, como solución para realizar computación de altas prestaciones (HPC, *High Performance Computing*) en clusters de estaciones de trabajo. Se trata de una idea heredada de la arquitectura VIA [16] (*Virtual Interface Architecture*) y mejorada con nuevas características.

En un primer momento, InfiniBand fue presentada como un firme candidato para sustituir al bus PCI. Sin embargo, parece ser que opciones como Rapid I/O [68], HyperTransport [35], PCI-X [58], PCI Express [15], o PCI Express Advanced Switching [50] son mucho más adecuadas para la conexión de dispositivos de E/S a nivel local. Por su parte, InfiniBand se está dirigiendo fundamentalmente hacia la interconexión de sistemas externos [57].

2.2 Direccionamiento de dispositivos y formato del paquete

Para posibilitar el encaminamiento dentro de la subred, y entre diferentes subredes, existen tres tipos de identificadores.

Cada CA, conmutador o encaminador tiene asignado de fábrica un GUID (*Global Unique ID*) de 64 bits con formato IEEE EUI-64. Este identificador es análogo a la dirección MAC presente en las tarjetas Ethernet.

Cada puerto de un CA o encaminador, y el puerto 0 de cada conmutador (destinado a tareas de gestión), tiene asociado al menos un identificador global (GID, *Global IDentifier*). Un GID es una dirección válida IPv6 de 128 bits. Los paquetes cuyo emisor y cuyo destinatario se encuentran en subredes distintas deben incluir en su cabecera sus respectivos SGID (*Source GID*) y DGID (*Destination GID*). Esta información es utilizada por los encaminadores intermedios al encaminar el paquete, y se omite si el paquete no cambia de subred. La Figura 3 muestra el formato básico de un GID. Los 64 bits de mayor peso identifican la subred donde reside el puerto referenciado. Los 64 bits inferiores se corresponden con el GUID del puerto.

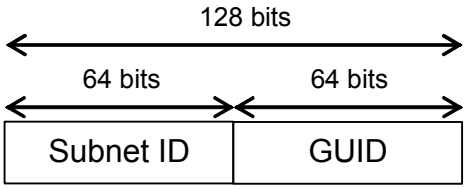


Figura 3. Formato de un identificador global (GID).

Cada puerto de un CA o encaminador y el puerto 0 de cada conmutador tiene asociado un identificador local (LID, *Local IDentifier*) exclusivo de 16 bits, asignado por el SM. Existe un mecanismo que permite asociar múltiples LIDs a un mismo puerto. Este mecanismo no es aplicable al puerto 0 de los conmutadores. El conjunto de LIDs asignados a un puerto se define a partir de un LID base y un LMC (*LID Mask Control*) de 3 bits. El rango de LIDs consecutivos definidos por el par LID/LMC es [LID, LID+2^{LMC}-1]. De esta forma, cada puerto puede ser referenciado por un máximo de 128 LIDs. Dentro de la misma subred, no está permitido el solapamiento entre los LIDs asignados a dos puertos distintos.

Todo paquete transmitido incluye en su cabecera el SLID (*Source LID*) del emisor y el DLID (*Destination LID*) del destinatario. Esta información, y no el GUID, es la utilizada por los conmutadores para encaminar los paquetes dentro de la subred. Para soportar encaminamiento multidestino (*multicast routing*), un LID puede hacer referencia a un grupo de dispositivos. De hecho, de los 65536 posibles LIDs, los primeros 49152 son empleados para direccionamiento unidestino (*unicast*), y los 16384 restantes son destinados a direccionamiento multidestino.

Por último, existe un LID especial, denominado *permissive address*, que permite hacer referencia a un puerto cuyo LID no ha sido asignado todavía. Esta dirección se emplea, por ejemplo, para los campos SLID y DLID de los paquetes de exploración de topología intercambiados por las entidades de gestión tras la activación de la subred. Como veremos en el Capítulo 3, estos paquetes emplean un mecanismo de encaminamiento especial, en el que los campos SLID y DLID no son tenidos en cuenta.

La Figura 4 muestra los campos requeridos para llevar a cabo el encaminamiento de un paquete de datos dentro de los límites de la subred (a nivel de enlace). La cabecera del paquete (LRH, *Local Route Header*) incluye el LID del emisor (SLID) y del destinatario (DLID), el nivel de servicio (SL) y el canal virtual (VL). Los tres primeros parámetros son fijos. En cambio, el canal virtual se actualiza a medida que el paquete viaja por la subred. El campo LVer identifica la versión del protocolo de nivel de enlace. Los campos Rsv2 y Rsv5 están reservados (viajan a 0 y son ignorados en el receptor). El campo LNH (*Link Next Header*) identifica el tipo de información de cabecera que sigue al LRH (por ejemplo la cabecera del nivel de transporte). La longitud del paquete (PktLen) viene expresada en palabras de 4 bytes, contando desde el primer byte del LRH hasta el último byte anterior al VCRC.

Tras la cabecera del nivel de transporte (*IBA Transport Header*), el campo Packet Payload contiene los datos de aplicación que deben ser transmitidos de extremo a extremo.

El tamaño de este campo no puede exceder la MTU (*Maximum Transfer Unit*) definida para la ruta, que puede ser de 256 bytes, 512 bytes, 1 Kbyte, 2 Kbytes o 4 Kbytes.

Finalmente, los campos ICRC (*Invariant CRC*) y VCRC (*Variant CRC*) se emplean para detección de errores en la transmisión. El ICRC cubre todos los campos cuyo contenido no varía de origen a destino. Es calculado para todos los paquetes del mensaje, y no cambia. El VCRC cubre todos los campos del paquete y debe ser recalculado en cada enlace. El tamaño máximo de todas las cabeceras más los CRCs es 126 bytes.

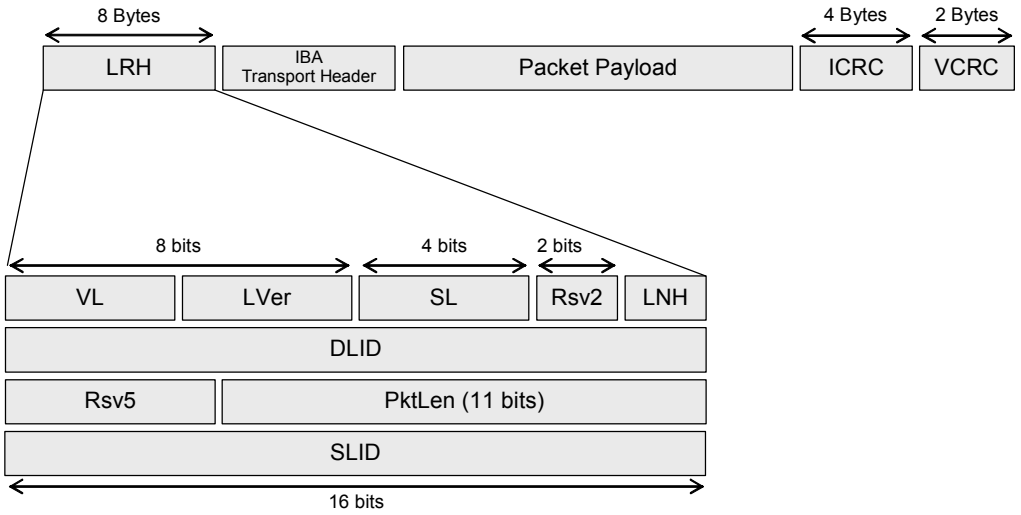


Figura 4. Formato del paquete de datos para encaminamiento local.

Cuando la comunicación se lleva a cabo entre distintas subredes el paquete debe incluir un campo GRH (*Global Route Header*) tras el LRH. Este campo incluye información para el nivel de red de los encaminadores (SGID, DGID, etc.).

2.3 Nodos terminales y tarjetas de interfaz

Una red InfiniBand da servicio a dos tipos de nodos terminales (*end nodes*): nodos de procesamiento (*processor node*) y unidades de E/S (*I/O unit*). La Figura 5 muestra un ejemplo de ambos tipos de nodos terminales.

Los nodos de procesamiento pueden estar constituidos por múltiples CPUs y módulos de memoria, y se conectan a la red mediante uno o varios HCAs (*Host Channel Adapter*).

Por su parte, las unidades de E/S contienen uno o más controladores (*I/O controller*) que proporcionan acceso a dispositivos de E/S de cualquier naturaleza, desde una simple consola hasta un sistema RAID (*Redundant Array of Inexpensive Disks*). Estas unidades se conectan a la red mediante TCAs (*Target Channel Adapter*).

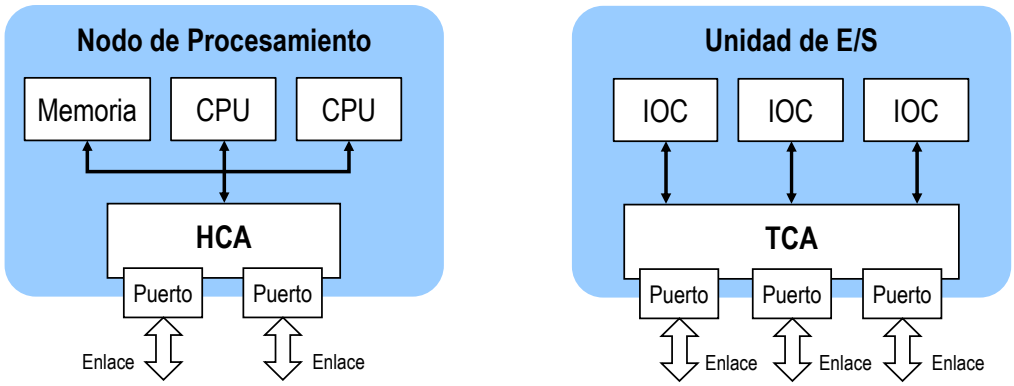


Figura 5. Nodo de procesamiento y unidad de E/S.

Las tarjetas de interfaz (CA, *Channel Adapter*) dan acceso a la red a los nodos terminales. Dicho de otra forma, son la fuente y el destino de los paquetes que atraviesan la red. Un CA emisor fragmenta un mensaje en paquetes, que serán posteriormente reensamblados por el CA destinatario. Como acabamos de ver, el CA tiene una denominación distinta (HCA o TCA) en función del tipo de nodo terminal al que ofrece sus servicios. La diferencia fundamental entre ambos tipos de CAs es que el HCA debe implementar los verbos (Sección 2.7.3).

Cada CA puede tener uno o varios puertos (hasta 254 puertos, numerados a partir de 1), como se muestra en la Figura 6. Cada uno de ellos se conecta generalmente a un puerto de un conmutador, si bien también es posible una conexión directa con alguno de los puertos de otro CA. Un controlador de DMA (*Direct Memory Access*) accede a la memoria del nodo al que está conectado el CA.

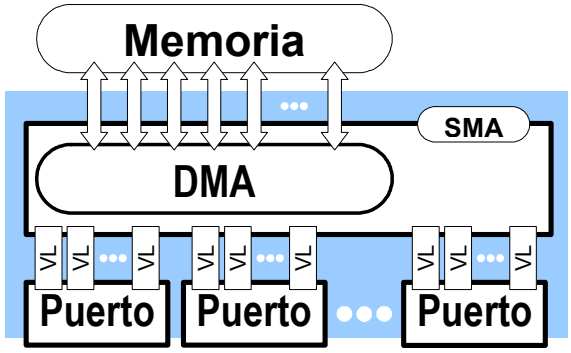


Figura 6. Estructura interna de un CA multipuerto.

2.4 Enlaces y puertos

Los enlaces físicos (*links*) son canales bidireccionales para la comunicación punto a punto entre los nodos terminales de la red. Con una frecuencia de la señal de 2.5 Gbaudios se obtiene un ancho de banda bidireccional de 500 Mbytes/s. No obstante, se puede incrementar

esta magnitud mediante la utilización en paralelo de múltiples líneas. La especificación contempla la utilización de 1, 4 (2 Gbytes/s) o 12 (6 Gbytes/s) enlaces en paralelo, denominados respectivamente 1X, 4X y 12X.

Los medios de transmisión especificados son el cobre y la fibra óptica. El enlace de cobre básico (1X) requiere 4 cables, un par trenzado en cada sentido. La longitud del cable 1X puede alcanzar los 250 metros. La longitud de los cables 4X y 12X puede alcanzar los 125 metros. Por su parte, el enlace óptico básico consta de dos fibras ópticas, una en cada sentido. Los enlaces ópticos 1X pueden alcanzar los 10 Kms de longitud. En la especificación 1.0 no está definida la longitud máxima de los enlaces ópticos 4X y 12X. A su vez, estos parámetros pueden variar en futuras versiones de la especificación.

Es posible emplear repetidores (*repeaters*) para compensar la pérdida de señal en enlaces de larga distancia. Otra de las funciones de los repetidores es la de actuar como interfaz entre la fibra óptica y el cable de cobre.

2.4.1 Máquina de estados del puerto

La especificación de InfiniBand utiliza una máquina de estados para definir el comportamiento de los enlaces. Posteriormente, cada implementación puede adoptar las medidas que considere oportunas para reflejar dicho comportamiento de la forma más eficiente posible. Un puerto puede encontrarse en cualquiera de los estados descritos a continuación. La Figura 7 muestra las posibles transiciones entre estados.

- *LinkDown*. En este estado el enlace físico está desconectado, o conectado a un dispositivo inactivo. Por tanto, la transmisión está físicamente deshabilitada. Todo paquete suministrado por los niveles superiores para su transmisión, es automáticamente descartado.

- *LinkInitialize*. En este estado hay actividad en el enlace. A diferencia del estado anterior, la transmisión está físicamente habilitada. No obstante, sólo se permite la transmisión de paquetes de gestión de la subred (SMP, *Subnet Management Packet*), y de paquetes de control de flujo. Cualquier otro tipo de paquete suministrado por los niveles superiores para su transmisión, o recibido en el puerto, es automáticamente descartado.

- *LinkArm*. En este estado, el puerto puede transmitir y recibir SMPs y paquetes de control de flujo. De nuevo, cualquier paquete presentado por los niveles superiores para su transmisión será descartado. Sin embargo, el puerto puede recibir cualquier otro tipo de paquete.

- *LinkActive*. En este estado, el puerto puede transmitir y recibir todo tipo de paquetes.

- *LinkActDefer*. El puerto entra en este estado cuando, encontrándose en estado *LinkActive*, el nivel físico informa de un fallo en el enlace. Si el error se mantiene durante un cierto tiempo (*LinkDownTimeOut*), el puerto pasa a estado *LinkDown*. Si por el contrario el nivel físico consigue subsanar el error a tiempo, el puerto regresa al estado *LinkActive*. En este estado, el puerto no acepta la recepción de nuevos paquetes, aunque continúa procesando los paquetes que ya estaba recibiendo. Todos los paquetes presentados por los niveles superiores para su transmisión son descartados.

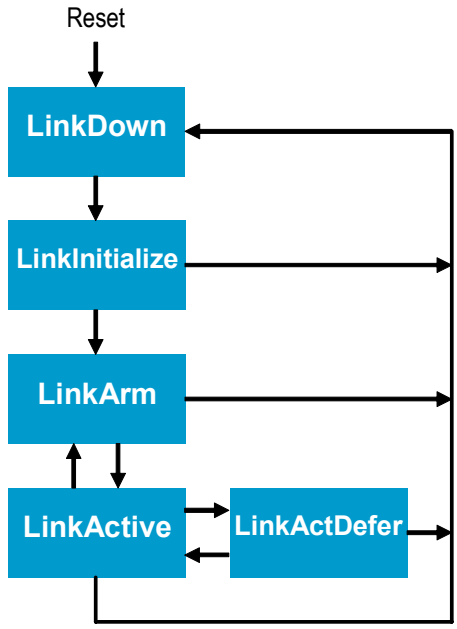


Figura 7. Máquina de estados del puerto.

Por simplicidad, las condiciones ante las que se producen las transiciones no han sido representadas en la Figura 7. En general, una transición entre dos estados puede deberse a un cambio en la actividad del enlace físico y/o a la recepción de una orden del SM para activar o desactivar el puerto.

En primer lugar, para notificar un cambio en la actividad del enlace, el nivel físico emplea un mensaje especial, denominado *Phy_Link* (*physical link status*). Este mensaje es enviado por el nivel físico del puerto a la lógica del enlace cada vez que el estado del enlace cambia, y puede tomar los valores *Down* o *Up*. Como ejemplo, la transición del estado *LinkDown* al estado *LinkInitialize* se produce automáticamente cuando *Phy_Link* = *Up*.

En cuanto a la otra causa de cambio de estado, el SM puede ordenar explícitamente una transición al estado *LinkDown*, *LinkArm* o *LinkActive*. Para ello, envía un mensaje al puerto especificando el valor *Down*, *Armed* o *Active*, respectivamente. El formato de este mensaje y el mecanismo para su envío serán detallados en el Capítulo 3 de esta memoria. Como ejemplo, el SM puede reiniciar el nivel de enlace de un puerto activo mediante una orden *Down*. Ante su recepción, la máquina de estados del puerto saltará al estado *LinkDown*, y de ahí, automáticamente (dado que hay actividad en el enlace), al estado *LinkInitialize*. Una

vez en estado *LinkInitialize*, el SM puede activar el puerto, especificando el valor *Armed* en un nuevo mensaje de cambio de estado. Esa orden provocará una transición al estado *LinkArm*. Finalmente, la transición al estado *LinkActive* se producirá bien mediante una nueva orden del SM, bien de forma automática, tras la recepción en el puerto del primer paquete de datos.

2.4.2 Canales virtuales

En la Figura 8 se representan dos puertos conectados a través de un enlace. Cada puerto puede incluir hasta 16 canales virtuales (VL, *Virtual Lane*), denominados VL0-VL15, y cuyo objeto es proporcionar flujos de datos independientes sobre el mismo enlace físico. Cada canal virtual tiene asociado un conjunto dedicado de *buffers* (no necesariamente implementados a nivel físico), con capacidad para albergar al menos un paquete. InfiniBand emplea conmutación *virtual cut-through* [26, 40].

Los canales VL0-VL14 se utilizan para transmitir paquetes de datos entre aplicaciones. Cada puerto puede implementar 1 (VL0), 2 (VL0, VL1), 4 (VL0-VL3), 8 (VL0-VL7) o 15 (VL0-VL14) canales virtuales de datos. Por tanto, la presencia del canal VL0 es obligatoria, siendo opcionales el resto de canales de datos. Por su parte, el canal VL15 está reservado para distribuir los paquetes de gestión de la subred (SMPs), y su implementación es obligatoria.

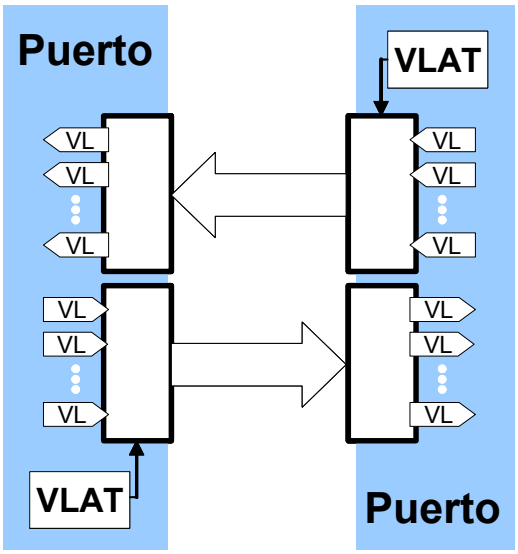


Figura 8. Dos puertos InfiniBand conectados con un enlace físico.

Es posible que los puertos conectados a los extremos de un enlace físico implementen un número distinto de canales virtuales de datos. En estos casos, sólo están operativos los canales virtuales implementados en ambos extremos. Para realizar esta operación, cada puerto contiene el número de canales de datos implementados en un parámetro interno denominado *PortInfo.VLCap*. A su vez, cada puerto contiene el número de canales de datos operativos en

otro parámetro interno denominado *PortInfo.OperationalVLs*. Durante la inicialización de la subred, este segundo atributo es configurado por el SM con un valor menor o igual que el del atributo *PortInfo.VLCap*. Cada paquete de datos transmitido por el enlace incluye el número de canal virtual utilizado. Cuando el paquete alcanza el otro extremo, la lógica del enlace recupera dicha información para alojar el paquete en el canal virtual correspondiente.

2.4.3 Arbitraje de canales

En cada puerto es necesaria la existencia de una unidad de arbitraje del canal, que debe seleccionar el canal virtual que depositará el siguiente paquete en el enlace físico. La especificación de InfiniBand define que el canal virtual de gestión (VL15) tiene la máxima prioridad. Los paquetes de control de flujo (descritos más adelante) son los siguientes en prioridad. Por último, los canales de datos (VL0-VL14) tienen la prioridad más baja.

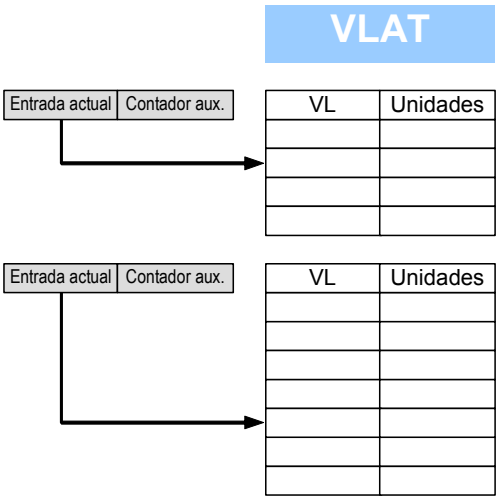


Figura 9. Estructura de la tabla VLAT.

La unidad de arbitraje utiliza una tabla (VLAT, *VL Arbitration Table*) configurada por el SM para establecer el orden (y el tiempo) de ocupación del enlace físico por parte de los diferentes canales de datos. Como se muestra en la Figura 9, cada entrada en la tabla de arbitraje contiene dos campos. El primero de ellos es un valor entero comprendido entre 0 y 14 que indica el canal de datos al que hace referencia dicha entrada. Si este campo hace referencia a un canal virtual no implementado, o al VL15, la entrada es ignorada. El segundo campo de la entrada es un valor entero, comprendido entre 0 y 255, que representa la cantidad de información que el canal virtual referenciado está autorizado a transmitir antes de proceder a interpretar otra entrada de la tabla. Este valor representa unidades de transmisión de 64 bytes. Un valor nulo en este campo indica que la entrada debe ser ignorada.

Las entradas en la tabla VLAT se agrupan en dos zonas: un conjunto de alta prioridad (HP, *High-Priority list*), y un conjunto de baja prioridad (LP, *Low-Priority list*). El primer

conjunto debe contener al menos una entrada. El segundo conjunto debe contener al menos tantas entradas como canales virtuales estén implementados en el puerto. Ambos conjuntos pueden tener un tamaño diferente, pero nunca superior a 64 entradas cada uno. En todo momento, sólo uno de ambos conjuntos estará activo. A continuación se describe la forma en la que la unidad de arbitraje conmuta entre dichos conjuntos de alta y baja prioridad, así como la forma en la que las entradas del área activa son inspeccionadas. En lo sucesivo, para cada entrada consideraremos que existe un paquete "disponible" para su transmisión, si se cumple que el canal virtual al que la entrada hace referencia contiene un paquete aguardando y, además, existe crédito suficiente para trasmitirlo.

Un atributo entero denominado *Limit of HighPriority* (comprendido entre 0 y 255) representa el número de bytes (módulo 4096) que la zona de alta prioridad puede transmitir antes de ofrecer dicha posibilidad a la zona de baja prioridad. Un contador auxiliar, denominado *HighPriCounter*, es inicializado con este valor. La longitud de cada paquete de alta prioridad transmitido es sustraída de dicho contador y, cuando alcanza un valor negativo, es reiniciado a su valor original. Al completar la transmisión de un paquete se realiza la siguiente comprobación para determinar el conjunto de entradas activo en la transmisión del siguiente paquete:

1. Si el área de alta prioridad contiene un paquete disponible para su transmisión y, además, el contador *HighPriCounter* no ha expirado, entonces el área de alta prioridad es activada.

2. En otro caso, el contador *HighPriCounter* es reiniciado y el área de baja prioridad activada.

Existen dos casos particulares, ambos relacionados con el valor del atributo *Limit of HighPriority*. Si este atributo tiene un valor nulo, entonces se considera que el área de alta prioridad puede emitir solamente un paquete antes de activar el área de baja prioridad. Por el contrario, el valor 255 representa el infinito. En este caso, es posible que el área de baja prioridad nunca pase a estado activo (solamente lo hará cuando ninguno de los canales virtuales referenciados en las entradas del área de alta prioridad tenga un paquete disponible).

Dentro de cada área, las entradas son inspeccionadas de forma secuencial y cíclica. Para ello, ambas incorporan un puntero y un contador auxiliar de unidades. Cuando una tabla se activa, se consulta la entrada actualmente direccionada por el puntero. Si existe un paquete disponible para su transmisión en el canal virtual referenciado, y el valor del contador auxiliar es positivo, entonces el paquete es transmitido y su longitud sustraída del valor del contador. En caso contrario, el puntero pasa a referenciar la siguiente entrada de la tabla, el valor del contador auxiliar es inicializado con el número de unidades asociado a esta entrada, y se repite la comprobación anterior. El proceso continúa hasta encontrar un paquete que transmitir o hasta inspeccionar todas las entradas del conjunto. En este último caso se activa la otra tabla.

Nótese que, aunque se utilizan unidades de 64 bytes para determinar la cantidad de información transmitida por un canal virtual antes de conmutar al siguiente, en realidad se procede a la transmisión de paquetes completos.

2.4.4 Control de flujo

El control de flujo descrito en esta sección se establece a nivel de enlace, y no entre extremos. En lo sucesivo, consideraremos que los dos puertos de un enlace se descomponen en un "transmisor" y un "receptor". El control de flujo garantiza que el transmisor no desbordará los recursos disponibles en el receptor.

El canal VL15 no está sujeto a control de flujo de ningún tipo (en cada enlace o entre extremos). La ventaja inmediata que se desprende de esta decisión es que los SMPs pueden ser transmitidos en cualquier momento. Como inconveniente, un SMP puede ser descartado por el receptor situado al otro extremo del enlace, sin emitir notificación alguna, en caso de no disponer de recursos suficientes para alojarlo. Como veremos en el próximo capítulo, existe un mecanismo de *timeouts* y retransmisiones para manejar estas situaciones.

Por su parte, los canales de datos (VL0-VL14) utilizan control de flujo basado en créditos (*credit-based flow control*) [26]. La información de control de flujo se transmite mediante paquetes de control específicos (*flow control packets*), cuya estructura se muestra en la Tabla 1. Estos paquetes deben ser transmitidos con la frecuencia suficiente para garantizar la utilización eficiente del enlace. En cualquier caso, un nuevo paquete debe ser transmitido antes de que pasen 65536 unidades de tiempo simbólico (*symbol time*). Se define esta unidad como el tiempo requerido para la transmisión de 8 bits a través del enlace.

Tabla 1. Paquete de control de flujo.

Bits	31-24		23-16	15-8		7-0
Bytes						
0-3	Op		FCTBS	VL		FCCL
4-5	LPCRC					

Mediante estos mensajes, el receptor informa al transmisor del límite de crédito máximo que está autorizado a transmitir desde la inicialización del enlace. A su vez, el transmisor informa al receptor de la cantidad total de información que ha enviado desde que el enlace fue inicializado, con el objeto de subsanar posibles inconsistencias entre emisor y receptor debidas a errores en la transmisión (ya afecten a los paquetes de datos o de control de flujo). Como cada puerto actúa como emisor y receptor simultáneamente, ambos valores se incluyen en el mismo paquete de control de flujo.

Un valor 0x0 en el campo Op (*operand*) indica que éste es un paquete de control de flujo normal. Un valor 0x1 en dicho campo indica que éste es un paquete de inicialización del flujo. Todo paquete recibido con un valor diferente debe ser descartado.

El campo VL (*virtual lane*) indica el canal virtual al que la información de control de flujo contenida en el paquete hace referencia.

El campo FCTBS (*flow control total blocks sent*) indica la cantidad total de información emitida por el transmisor. Dicha cantidad se expresa en bloques de control de flujo de 64 bytes, módulo 4096. Teniendo en cuenta su tamaño (12 bits), el módulo utilizado, y el tamaño de cada bloque, este campo puede indicar hasta 1 Gbyte de información transmitida antes de desbordarse. Para calcular el número de bloques que mide un paquete de datos se divide su longitud (descontada la cabecera y el CRC) entre 64, y se redondea al siguiente entero.

El campo FCCL (*flow control credit limit*) indica el límite de crédito de control de flujo, expresado en bloques de 64 bytes, módulo 4096. Para calcular el crédito disponible, la parte receptora de cada canal utiliza un contador de 12 bits denominado ABR (*adjusted blocks received*), que contiene el valor del campo FCTBS del último paquete de control de flujo recibido. En cada paquete de control de flujo transmitido, El campo FCCL es asignado con el valor ABR más el número de bloques para los que existen recursos disponibles, hasta un máximo de 2048 bloques (128 Kbytes).

Finalmente, el campo LPCRC (*link packet cyclic redundancy check*) contiene un código de redundancia cíclica aplicado sobre los 4 primeros bytes del paquete.

2.4.5 Niveles de servicio

Es posible que cada puerto dentro de un mismo conmutador disponga de un número distinto de canales virtuales, bien porque estén implementados de esta forma, bien porque algunos de ellos hayan sido deshabilitados por contener el puerto situado al otro extremo del enlace un número menor de canales virtuales. Este hecho hace imposible transmitir un paquete en base a un valor de VL especificado en su cabecera. En su lugar se aplica un criterio más abstracto, basado en el concepto de niveles de servicio (SL, *Service Level*). Cada paquete lleva en su cabecera un valor de SL que no varía a lo largo de su ruta; sin embargo, en cada conmutador intermedio puede que sea necesario actualizar el VL del paquete.

Para establecer una correspondencia entre los 16 posibles SLs (de 0 a 15) y los canales virtuales implementados por cada puerto, se emplea una tabla de mapeo (SLtoVLMT, *SL to VL Mapping Table*) configurada por el SM. En los conmutadores, esta correspondencia se establece para cada pareja puerto de entrada – puerto de salida. Para cada CA y encaminador la correspondencia se establece a nivel de puerto. En la Sección 2.5.2 se detalla el uso de esta tabla.

2.4.6 Control de inyección

Para permitir la coexistencia de dispositivos con distinta velocidad en la red, InfiniBand incorpora un mecanismo de control de inyección (*static rate control*). Este mecanismo impide que un puerto rápido pueda llegar a saturar a otro más lento.

Consideremos el escenario de la Figura 10. El puerto en el CA *X* es capaz de transmitir paquetes a 6 Gbytes/s hacia los CAs *Y* y *Z*. A esa velocidad, los buffers en los puertos de salida del conmutador intermedio se desbordarán rápidamente, ya que la velocidad a la que pueden vaciarse está limitada por la velocidad de los enlaces 1X y 4X. Entonces, el buffer en el puerto de entrada del conmutador también se llenará, y el control de flujo provocará que el CA *X* deje de inyectar paquetes. Finalmente, el buffer de salida en el CA *X* se llenará y dejará de aceptar los paquetes que reciba para transmisión, sea cual sea su destino.

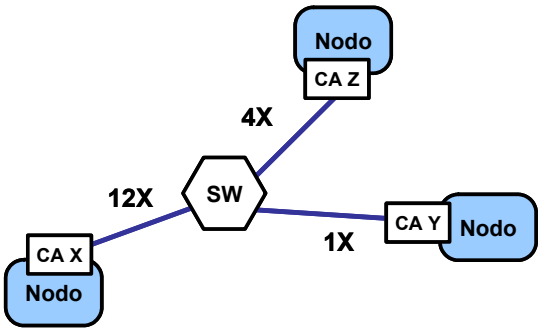


Figura 10. Ejemplo de aplicación del mecanismo de control de inyección

El mecanismo de control de inyección actúa en todos los CAs conectados a enlaces 4X o 12X. Básicamente, se basa en introducir un retardo (IPD, *inter-packet delay*) entre la emisión de dos paquetes dirigidos hacia el mismo destinatario. El IPD es un valor entero de 8 bits que se establece en el momento de crear una conexión (más adelante trataremos aspectos del nivel de transporte).

La Tabla 2 muestra algunos de los 256 valores posibles para el IPD, empleados para la conversión entre distintas velocidades.

Tabla 2. Algunos valores para el IPD.

IPD	Tasa	Comentario
0	100%	Puertos con la misma velocidad
2	33%	Adecuado para conversión de 12X a 4X
3	25%	Adecuado para conversión de 4X a 1X
11	8%	Adecuado para conversión de 12X a 1X

2.5 Conmutadores

La Figura 11 muestra la estructura interna de un conmutador (*switch*) InfiniBand. Un conmutador puede incluir hasta 255 puertos. Uno de ellos (no mostrado en la figura), etiquetado como 0, es un puerto lógico reservado para los paquetes de gestión, que tienen como

emisor o destinatario al propio conmutador (lo veremos más tarde). El resto (numerados desde 1 hasta 254) son puertos físicos que permiten la conexión con otros dispositivos (conmutadores, encaminadores o CAs). Un elemento de conmutación interno (por ejemplo una red *crossbar*) establece las conexiones físicas entre los canales de entrada y los canales de salida.

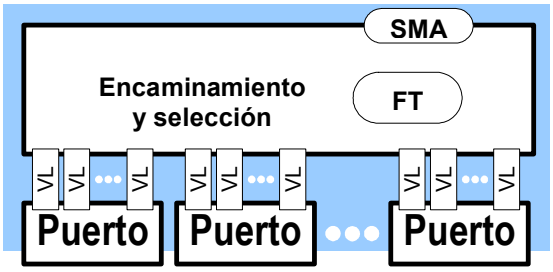


Figura 11. Estructura interna de un conmutador InfiniBand.

2.5.1 Tablas de encaminamiento

Para establecer la conexión entre un puerto de entrada y un puerto de salida, el conmutador consulta una tabla de encaminamiento (FT, *Forwarding Table*). Esta tabla puede estructurarse de dos formas alternativas denominadas, respectivamente, RFT (*Random Forwarding Table*) y LFT (*Linear Forwarding Table*). Como veremos más adelante, el contenido de las tablas es establecido por parte del SM durante el proceso de inicialización de la subred y modificado ante la ocurrencia de cualquier cambio topológico.

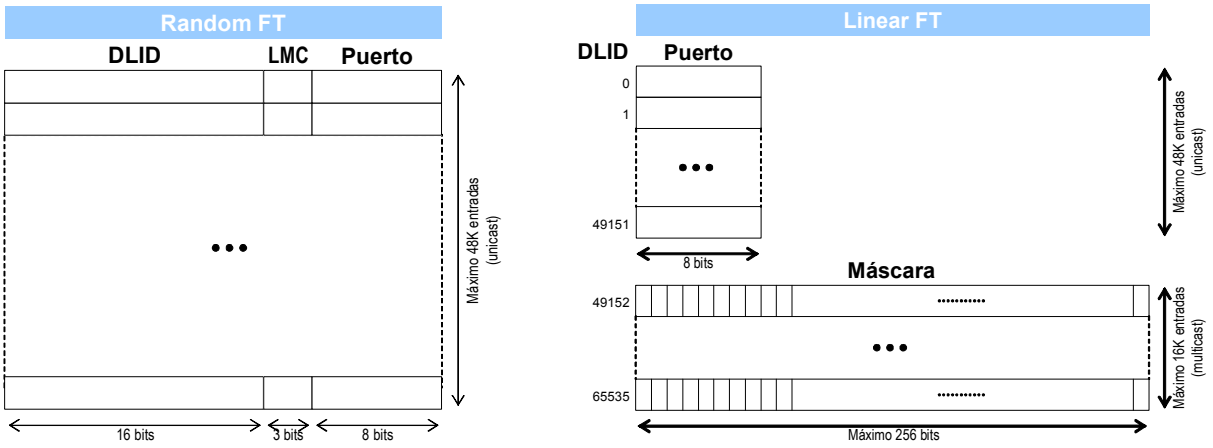


Figura 12. Estructura de las tablas RFT y LFT.

El formato de la tabla RFT se muestra en la Figura 12 (izquierda). Cada entrada en esta tabla proporciona un puerto de salida para una pareja DLID/LMC. El número máximo de entradas en esta tabla está limitado a 48K por la especificación.

La Figura 12 (derecha) muestra el formato de la tabla LFT. La principal diferencia entre esta tabla de encaminamiento y la anterior es que el campo DLID está implícito en la posi-

ción que ocupa cada entrada en la tabla. Las primeras 48K entradas de la tabla LFT son empleadas para encaminamiento unidestino. El único campo existente en estas líneas es el número de puerto de salida a utilizar.

Las 16K entradas restantes son usadas para encaminamiento multidestino, cuya implementación es opcional. Cuando un conmutador detecta que el DLID de un paquete corresponde a una dirección multidestino, lo replica y reenvía por un conjunto preestablecido de puertos. Para ello, cada entrada multidestino contiene una máscara que define dicho conjunto. El tamaño de esta máscara dependerá del número de puertos implementados, pudiendo alcanzar un máximo de 32 bytes (256 bits).

Además del mecanismo de encaminamiento en base a destino (*destination/LID routing*), los conmutadores deben soportar encaminamiento fuente o dirigido (*directed routing*). En este caso, el propio paquete contiene la secuencia de puertos de salida a emplear para alcanzar su destino. Este tipo de encaminamiento es empleado durante el proceso de descubrimiento de la subred, cuando los LIDs de los distintos dispositivos todavía no han sido asignados, ni las tablas de encaminamiento distribuidas.

2.5.2 Encaminamiento y selección de paquetes

La función principal de un conmutador es la retransmisión de paquetes (*packet relay*). Cada paquete es retransmitido desde el canal de entrada de uno de sus puertos hasta el canal de salida de otro puerto. Nótese que no se tiene en cuenta el estado del puerto de salida, lo cual puede provocar, en algunas situaciones, el descarte del paquete.

Si el paquete emplea encaminamiento en base a destino, el conmutador debe consultar en primer lugar la tabla de encaminamiento (FT) para obtener el puerto de salida correspondiente al DLID del paquete (Figura 13). Se contempla la posibilidad de realizar esta consulta (no la transmisión) antes de recibir el paquete completo en el canal de entrada. Si como resultado de la consulta se obtiene como puerto de salida el mismo puerto de entrada, el paquete es descartado (esta situación se permite al aplicar encaminamiento dirigido). Si el conmutador implementa una tabla RFT y la consulta a la misma no devuelve ningún puerto de salida, el paquete se envía al puerto por defecto del conmutador, en caso de que éste haya sido configurado por el SM con un valor válido. En caso contrario, el paquete es descartado.

Si el canal de entrada es el VL15, el paquete es transferido al VL15 del puerto de salida indicado en la tabla de encaminamiento. Si dicho canal no dispone de espacio suficiente, el paquete es descartado. Por el contrario, si el canal de entrada es un canal de datos (VL0-VL14), y el conmutador soporta varios de ellos, entonces debe obtener el canal de salida sobre el que depositar el paquete mediante una consulta a la tabla SLtoVLMT. Tal y como se muestra en la Figura 13, esta consulta requiere los puertos de entrada y salida del paquete y su SL. Si el VL de salida obtenido no está implementado en el puerto de salida (o bien es el VL15), el paquete será automáticamente descartado. En caso contrario, el paquete será retransmitido

cuando el canal virtual de salida disponga de espacio suficiente para almacenarlo completamente. El paquete permanecerá en el canal virtual de entrada hasta ese momento, o hasta que sea descartado por otra razón. Los paquetes deben ser transmitidos en un puerto y SL dados en el mismo orden en el que fueron recibidos. No obstante, es posible (aunque no imprescindible) proceder a la transmisión de paquetes detenidos detrás de otros paquetes bloqueados en un canal virtual de entrada debido a la falta de espacio en el puerto de salida.

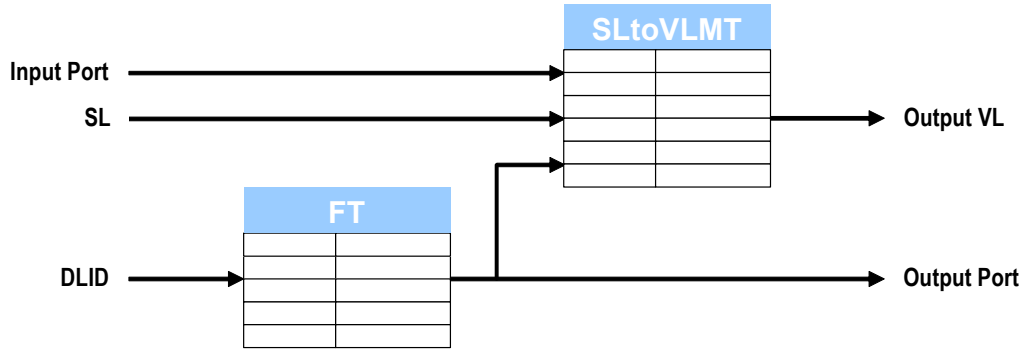


Figura 13. Uso de las tablas FT y SLtoVLMT en un conmutador.

Finalmente, el método de arbitraje cuando múltiples canales virtuales de entrada tienen paquetes destinados al mismo canal virtual de salida está abierto al implementador, pero este arbitraje debería servir a todos los puertos de entrada equitativamente.

2.6 Encaminadores

La Figura 14 muestra la estructura interna de un encaminador (*router*) InfiniBand. Los encaminadores son los componentes responsables del encaminamiento entre subredes. Su función es retransmitir los paquetes de datos desde uno de sus puertos de entrada a uno de sus puertos de salida. Los encaminadores no son totalmente transparentes a los CAs, pues éstos deben conocer, y especificar en el paquete, el LID del encaminador y el GID del destinatario.

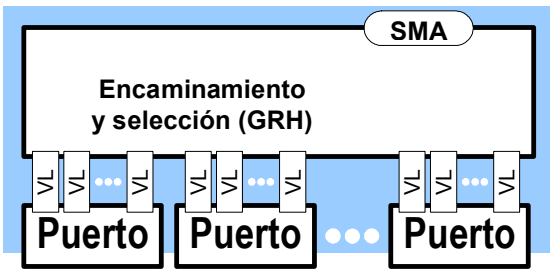


Figura 14. Estructura de un encaminador

La Figura 15 muestra tres subredes InfiniBand unidas por medio de dos encaminadores. En cuanto a su implementación física, los encaminadores pueden estar incorporados junto a conmutadores o tarjetas de interfaz, en un mismo dispositivo.

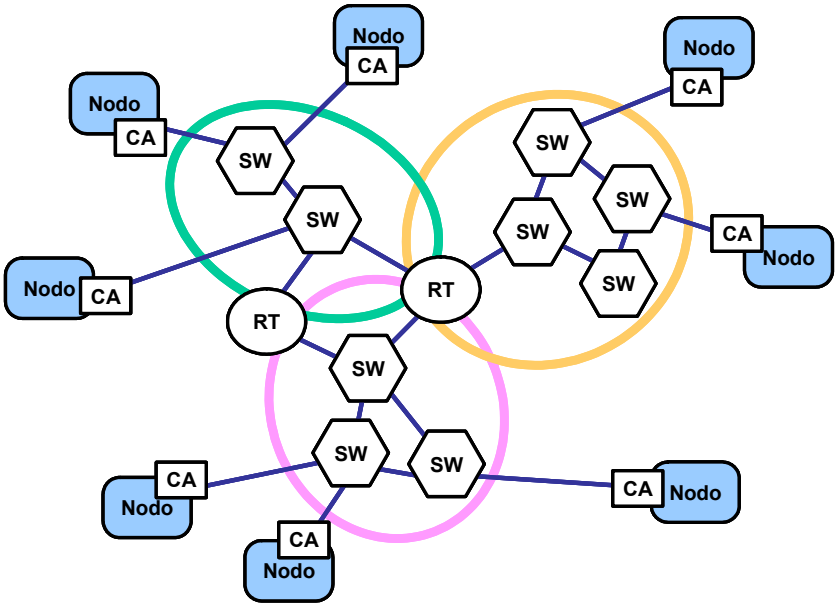


Figura 15. Subredes conectadas por medio de encaminadores.

2.6.1 Encaminamiento de paquetes

El encaminador realiza su función consultando una tabla de encaminamiento (*routing table*) establecida por el software de configuración durante el proceso de inicialización de la red. La estructura concreta de esta tabla no está definida en la especificación de InfiniBand.

Cada vez que recibe un nuevo paquete, el encaminador usa el campo DGID contenido en su cabecera GRH para determinar la subred en la que se encuentra el destinatario del paquete. En particular, emplea el campo *Subnet ID* dentro del GID (ver Figura 3, en página 9) para realizar una búsqueda en su tabla de encaminamiento y retransmite el paquete a través del puerto indicado en la correspondiente entrada.

La cabecera LRH del paquete debe ser modificada cuando éste cambia de subred. Así, el nuevo SLID del paquete coincide con el LID del encaminador, y el nuevo DLID es asignado al LID del siguiente encaminador si el destino no se encuentra en la nueva subred o al LID del destino en la nueva subred. Los valores SGID y DGID de la cabecera GRH son preservados durante toda la ruta del paquete.

2.7 Nivel de transporte

La especificación del nivel de transporte es muy extensa. En esta sección se describe cómo las aplicaciones se comunican con el CA, por medio de colas de trabajo y verbos. También se mencionan los diferentes tipos de servicio ofrecidos por la red.

2.7.1 Acceso al CA: channel interface

Los consumidores (procesos del sistema operativo, aplicaciones, o controladores de dispositivo) acceden al enlace físico a través de uno o varios CAs. Para ello, envían peticiones de trabajo (WR, *work request*) al CA, y reciben notificaciones de finalización de los trabajos (WC, *work completion*). Los WRs son instrucciones que deben ser ejecutadas por el hardware de la red. Los tipos de WR contemplados son:

- El envío y recepción de un mensaje (*send/receive*). Estas operaciones operan sobre bloques de memoria dentro del espacio de direcciones del consumidor.
- La ejecución de una operación RDMA (*remote direct memory access*). Empleadas para transferir directamente datos hacia o desde direcciones de memoria en nodos remotos.
- El enlace de una ventana de memoria (*bind a memory window*).
- La ejecución de una operación atómica (*atomic operation*) que permita a un consumidor leer y modificar datos en una dirección de memoria remota.

El acceso desde el consumidor al hardware del HCA no es directo, sino que se realiza a través de un interfaz denominado CI (*Channel Interface*). La funcionalidad de este interfaz está accesible a través de un conjunto de operaciones denominado "verbos" (*verbs*), descrito posteriormente. La Figura 16 muestra el interfaz entre el consumidor y el HCA.

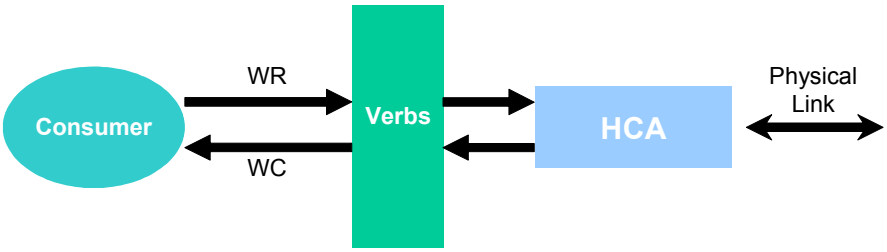


Figura 16. Interfaz entre el consumidor y el HCA.

2.7.2 Colas de trabajo

El CI proporciona al consumidor un conjunto de colas de trabajo (WQ, *Work Queue*) para acceder a los servicios de transporte y la capacidad de programar el CA para transferir datos y mensajes. Las WQs se crean en parejas, dando lugar a pares de colas (QP, *Queue Pair*) independientes entre sí. Una de las colas (*send queue*) se emplea para operaciones de envío entre consumidores y la otra (*receive queue*) para operaciones de recepción.

Una QP está ligada a un determinado puerto dentro del HCA, hacia el que envía paquetes y desde el que los recibe. Además, desde su creación tiene asociado un determinado tipo de servicio. Los servicios orientados a la conexión asocian una QP local con una QP remota. En servicios tipo datagrama, la QP de destino puede variar; en este caso, el destino debe ser definido en cada elemento de cola.

Las WRs enviadas por el consumidor al HCA son "traducidas" por el CI a elementos de cola (WQE, *WQ Element*) e irán destinados a la cola de envío o a la cola de recepción dentro de un par de colas. Finalmente, el hardware se encargará de interpretar cada WQE para ejecutar la operación indicada. Por ejemplo, para una operación de envío de datos (en la cola de envío) el HCA toma el WQE, crea un mensaje, lo segmenta en paquetes, añade las correspondientes cabeceras y envía los paquetes al puerto apropiado.

Cada WQ tiene asociada (desde su creación) una cola de finalización (CQ, *Completion Queue*). Las CQs representan el mecanismo de notificación de la finalización de WRs. Cuando el HCA completa un WQE, un CQE (*CQ Element*) se añade en la CQ correspondiente. Cada CQE incluye toda la información necesaria para un WC. No es necesario que las dos colas de un QP compartan la misma CQ. La Figura 17 muestra el modelo completo.

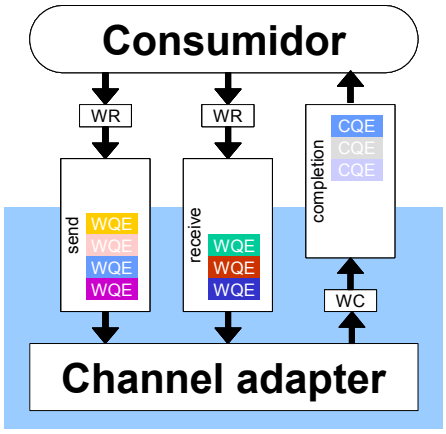


Figura 17. Modelo de colas para la ejecución de las operaciones del consumidor.

Existen tres tipos de operaciones para una cola de envío: SEND, RDMA y MEMORY BINDING. En el caso del SEND, el WQE debe especificar la ubicación en la memoria del consumidor del bloque de datos que el hardware debe enviar. Cuando el tipo de servicio de la QP es *datagram*, la QP de destino para la operación de envío viene indicada como parte del

WR emitido por el consumidor. Una operación de acceso remoto a DMA (RDMA) debe especificar la ubicación de los datos a enviar/recibir en el espacio de memoria del consumidor destino. Por otro lado, sólo existe un tipo de operación para una cola de recepción, y sirve para especificar los *buffers* en la memoria del consumidor que el hardware va empleando para almacenar los datos a medida que los va recibiendo. Este modelo facilita el esquema de *zero-copy networking*.

La especificación define dos mecanismos para la notificación de la terminación de un WR emitido por el consumidor. Por un lado, el consumidor puede consultar (*polling*) la CQ correspondiente en busca de un WC. Por otro lado, es posible programar la notificación automática de la llegada de un nuevo elemento a una CQ.

Las QPs tienen asignado un número (QPN) único dentro del CA. Existen dos pares de colas especiales (QP0 y QP1) reservadas para la comunicación con los dos interfaces de gestión definidos en InfiniBand, el SMI (*Subnet Management Interface*) para gestión de la subred, y el GSI (*General Services Interface*) para el resto de clases de gestión. Podemos considerar al SMI y al GSI como dos casos especiales de consumidores que acceden al enlace a través de QP0 y QP1, respectivamente. Es obligatoria la existencia en cada puerto de las colas QP0 y QP1 y de al menos una QP de datos.

2.7.3 Verbos

El consumidor accede al HCA y a los pares de colas que éste implementa través de verbos. Normalmente, el fabricante proporciona un API (*Application Programming Interface*) que incorpora la semántica de los verbos. A través de este interfaz, el consumidor abre un HCA, crea una o varias QPs, sitúa sus peticiones de trabajo (WRs) en las colas correspondientes (*send* o *receive*), o consulta una CQ para comprobar si la ejecución de un trabajo ha sido completada.

Existen dos clases de verbos: *privileged* y *user-level*. Esta clasificación determina el tipo de consumidores que pueden hacer uso de los verbos. La mayoría de las funciones clave especificadas por los verbos (como el envío de un WR o la comprobación de una CQ) pueden ser empleadas por consumidores no privilegiados, eliminando así la sobrecarga debida a la invocación del núcleo del sistema operativo. Este mecanismo se conoce como *kernel bypass*. Sin embargo, otras funciones como la apertura de un HCA o la creación de una QP sólo pueden ser empleadas por consumidores privilegiados.

2.7.4 Tipos de servicio ofrecidos por la arquitectura

La flexibilidad para escoger entre distintos tipos de servicio permite a la arquitectura InfiniBand dar servicio a un amplio rango de aplicaciones. Cada QP es configurada para una clase de servicio particular, que determina la forma en que dicha QP es asociada a otras QPs.

El tipo de servicio de transporte también determina exactamente qué tipo de operaciones son posibles sobre una QP. A continuación se describen los tipos de servicio contemplados.

Reliable connection (RC)

Cada QP está asociada con una QP remota y juntas ejecutan un protocolo confiable para el intercambio de mensajes y datos. De esta forma, todos los datos son entregados a la otra QP en orden y sin errores. Este tipo de servicio es útil para entrada/salida y otros tipos de aplicaciones donde existe una relación uno a uno entre dos entidades y todos los datos deben ser transferidos de forma confiable.

Unreliable connection (UC)

En este caso la QP también está asociada con una QP remota. Sin embargo, las QPs no ejecutan un protocolo confiable, y los mensajes pueden perderse. Este tipo de servicio es útil donde existe una relación uno a uno entre dos entidades pero la pérdida de datos no es un factor crítico, como ocurre en la transmisión de flujos de audio o vídeo.

Unreliable datagram (UD)

La QP no está asociada con una QP remota particular y no ejecuta un protocolo confiable. Por lo tanto, los mensajes pueden perderse. Sin embargo, la QP puede enviar y recibir mensajes hacia/desde otras QPs configuradas con este servicio. Esta clase de servicio es idéntico al proporcionado por la mayoría de las redes LAN, donde las capas de nivel superior ejecutan protocolos de transporte y fiables.

Reliable datagram (RD)

De nuevo, la QP no está asociada con una QP remota particular, lo que significa que puede enviar y recibir mensajes hacia/desde otras QPs configuradas con este servicio. A diferencia del servicio UD, todos los datos son entregados fiablemente entre las QPs. Esta clase de servicio es ideal para aplicaciones paralelas donde existe una relación muchos a muchos.

Raw datagram service

Se trata de un servicio similar a UD, pero técnicamente no se trata de otro servicio del nivel de transporte. Se trata de un servicio de enlace de datos que permite a otros protocolos de transporte ejecutarse por encima de la arquitectura InfiniBand. Existen dos tipos de servicio *raw datagram*: *EtherType* e *IPv6*.

La Figura 18 muestra algunos ejemplos de comunicación entre pares de colas, a través de servicios orientados y no orientados a la conexión.

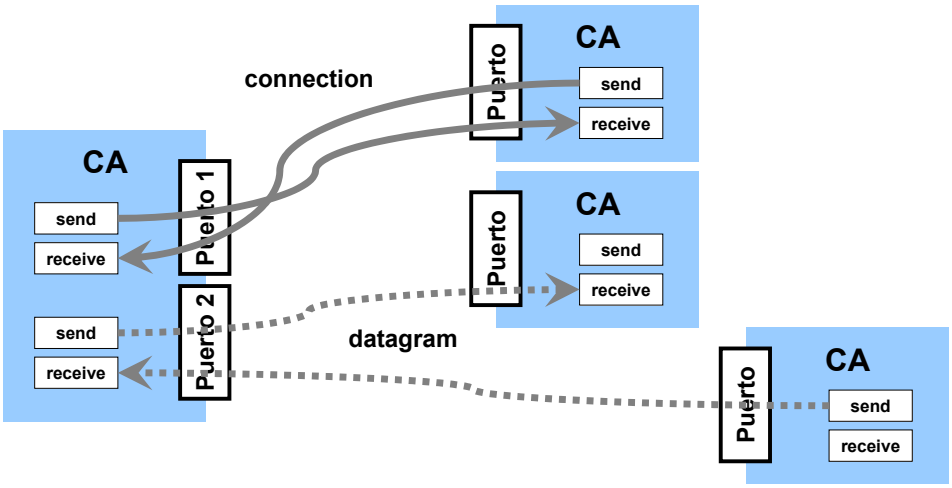


Figura 18. Servicios orientados y no orientados a la conexión.

2.8 Productos InfiniBand

2.8.1 Productos comerciales

En la actualidad, existe un amplio abanico de empresas dedicadas a la fabricación y/o comercialización de productos compatibles con la arquitectura InfiniBand, si bien muchas de las compañías que surgieron inmediatamente después de la publicación de las primeras versiones de la especificación han acabado desapareciendo o siendo absorbidas por otras. Aparte de los "buques insignia" de la InfiniBand Trade Association (IBM, Sun, Dell, etc.), algunas de las empresas "supervivientes" son *Mellanox* [52], *RedSwitch* [71], *QLogic* [67], *Voltaire* [89], *Banderacom* [4], o *InfiniCon* [38]. También hay que mencionar a empresas como *Vieo* [88] o *Lane15* (en la actualidad fusionada con *InfiniSwitch* [39]), dedicadas al desarrollo de herramientas para la gestión de redes InfiniBand.

Mellanox Technologies fue una de las empresas pioneras en el lanzamiento al mercado de dispositivos InfiniBand. Su primer producto "estrella", el chip *InfiniBridge* [27], incorporaba toda la funcionalidad de un CA y un conmutador con enlaces a 2.5 Gbps (1X) y 10 Gbps (4X). La tercera generación de chips de Mellanox incluye el *InfiniHost* (un HCA) y conmutadores de hasta 144 puertos (*InfiniScale*).

Mellanox comercializa diversas tarjetas y conmutadores basados en estos chips. Entre sus productos, ofrece el cluster *Server Blade*, mostrado en la Figura 19. En (a) se muestra la tarjeta multifunción *Nitro II Server Blade*, que incluye un *InfiniHost HCA MT23108*, un procesador *Intel Xeon* a 2.2Ghz, 256 Mbytes de memoria para el HCA y 1 Gbyte de memoria de sistema. En (b) se muestra el conmutador *Nitrex II Switch Blade*, que incluye 16 puertos 4X.

En (c) se muestra el chasis de sistema, con soporte para almacenar hasta 12 servidores y dos conmutadores. Finalmente, en la Figura 20 se muestra la topología del sistema.

(a) Nitro II Server Blade

(b) Nitrex II Switch

(c) Chasis del sistema

Figura 19. Gama de productos *Server Blade* de Mellanox.

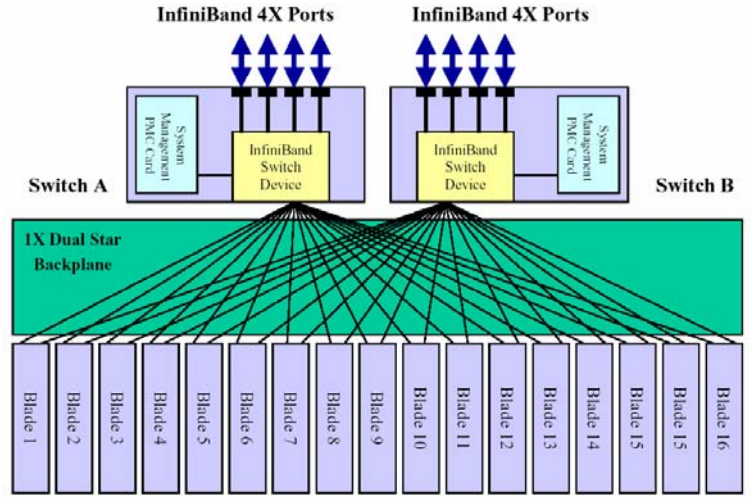


Figura 20. Topología del sistema *Server Blade* de Mellanox.

2.8.2 Código abierto

Como consecuencia de la proliferación de productos comerciales para InfiniBand, ha surgido una gran demanda de código abierto (*open source*) para este entorno. El objetivo actual es la inclusión de soporte para InfiniBand en las distribuciones estándar del sistema operativo Linux.

En este sentido, el proyecto *Linux InfiniBand Project* de *SourceForge* [83] abarca varios subproyectos que tienen como objetivo común proveer una serie de componentes Infini-

Band para Linux. Por su parte, el proyecto *Open InfiniBand Stack* de *SourceForge* [84] está dedicado a la obtención de un paquete completo de software abierto para InfiniBand, con soporte, por ejemplo, para computación de altas prestaciones o aplicaciones de almacenamiento.

Por otro lado, Mellanox [52] ha preparado su propio paquete de código abierto para InfiniBand. Dicho paquete está disponible bajo licencias GPL y BSD, e incluye los siguientes componentes:

- VAPI (*InfiniBand Verbs Application Programming Interface device driver*)
- CM (*Communications Management*)
- OpenSM (*Subnet Management*)
- IPOverIB. Aplicaciones basadas en TCP/IP.
- SDP (*Socket Direct Protocol*)
- SRP (*SCSI RDMA Protocol*)
- DAPL (*Direct Access Programming Library*)

2.9 Otras redes de altas prestaciones

En la literatura pueden encontrarse numerosas tecnologías de red con características muy similares a InfiniBand. Se trata también de redes conmutadas, muy escalables, que ofrecen altas prestaciones (gran ancho de banda y baja latencia), bajo coste y gran fiabilidad, y que están orientadas en mayor o menor medida a cubrir los mismos ámbitos de aplicación que InfiniBand. Es decir, la interconexión de dispositivos de E/S y nodos de procesamiento en una SAN o la construcción de redes de estaciones de trabajo para computación de altas prestaciones. Repasaremos aquí muy brevemente las características principales de algunas de las tecnologías más representativas.

2.9.1 ServerNet

ServerNet (y ServerNet II) [31, 34] es una tecnología de red de área de sistema (SAN) desarrollada por Tandem Labs, que también proporciona soporte para comunicación entre procesos (IPC) en clusters. La velocidad de sus enlaces va desde 1.25 a 5.0 Gbps. La red admite cualquier topología. Por motivos de fiabilidad, cada tarjeta dispone de dos puertos que le permiten conectarse a dos redes distintas.

Como técnica de conmutación, ServerNet usa *wormhole* [26]. Al contrario que InfiniBand, no emplea canales virtuales. ServerNet emplea encaminamiento distribuido. Cada conmutador almacena internamente su propia tabla de encaminamiento. Una consulta a esta tabla devolverá el puerto de salida por el que cada paquete será retransmitido. Para evitar bloqueos, impide dependencias circulares entre paquetes [22].

2.9.2 Autonet

Autonet [80] es una red desarrollada a principio de los años 90 en el *System Research Center* (SRC) de Digital. Básicamente, se trata de una red de área local autoreconfigurable, que emplea enlaces punto a punto bidireccionales de 100 Mbps. Los conmutadores pueden conectarse entre sí y con las estaciones de trabajo de forma completamente arbitraria. Para mayor fiabilidad, cada estación de trabajo suele acceder a la red a través de dos conmutadores. La Figura 21 muestra un ejemplo.

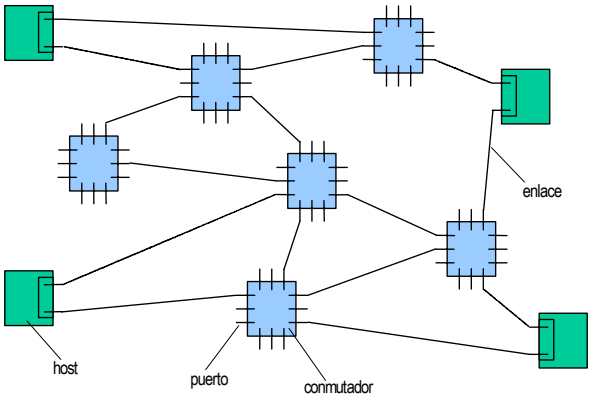


Figura 21. Una topología típica en Autonet.

La técnica de conmutación empleada es *wormhole*. El control de flujo está basado en el empleo de órdenes *START-STOP*, generándose un comando *STOP* hacia el emisor cuando el buffer del receptor supera la mitad de su capacidad. Como en ServerNet, el encaminamiento de paquetes es distribuido. Además, el encaminamiento es libre de bloqueos y se basa en *up*/down**.

2.9.3 Myrinet

Myrinet [14] es una red de área local y de área de sistema desarrollada por Myricom y muy extendida en los últimos años en el ámbito de la investigación. Se trata de una red altamente escalable y con muy baja tasa de errores, que admite cualquier configuración topológica. El ancho de banda bidireccional de sus enlaces es de 1.28 Gbps.

Al igual que Autonet y ServerNet, Myrinet emplea conmutación *wormhole*. El control de flujo es *STOP and GO*, muy similar al de Autonet. Myrinet emplea encaminamiento fuente. A partir de la información almacenada en el paquete, establecida en el nodo origen, cada conmutador determina el puerto de salida a emplear, hasta que el paquete alcanza el nodo destino. Las tablas de encaminamiento, que están alojadas en las tarjetas de interfaz de los nodos terminales, se obtienen aplicando la regla *up*/down**, como sucede en Autonet.

2.9.4 QsNet

Al igual que las tecnologías anteriores, QsNet (*Quadrics Network*) [60] es una red propietaria. Proporciona 400 Mbps en cada dirección. Dos elementos clave en QsNet son sus interfaces de red programables (*Elan*) y los conmutadores de altas prestaciones (*Elite*).

QsNet admite topologías *fat-tree* cuaternarias, un subconjunto de la clase *k-ary n-tree* [61]. Una topología *k-ary n-tree* tiene k^n nodos terminales y $n \times k^{n-1}$ conmutadores conectados como una red delta. En QsNet, k es 4, y n es el número de etapas de la red. Como ejemplo, la Figura 22 muestra un árbol cuaternario de dimensión 2, que emplea $2x4^{2-1}=8$ conmutadores para conectar $4^2=16$ nodos terminales.

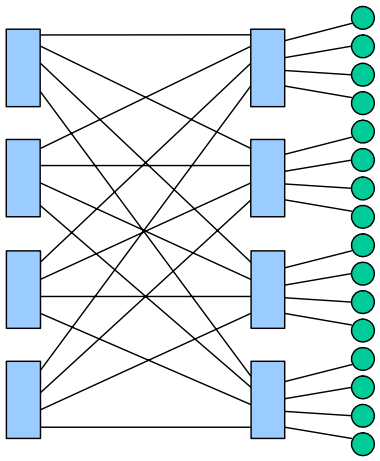


Figura 22. Una topología típica en QsNet.

De nuevo, la técnica de conmutación elegida es *wormhole*. QsNet emplea canales virtuales. El encaminamiento es adaptativo y fuente. *Elan* incluye en cada paquete su ruta a través de la red. El paquete se encamina en los *Elite* intermedios de forma muy similar a como se hace en Myrinet. Finalmente, cada paquete individual debe ser reconocido por el destinatario.

Capítulo 3
Gestión de la subred

Uno de los aspectos más importantes de la arquitectura InfiniBand es la gestión interna de la subred (*subnet management*). Dicha gestión engloba tareas como la exploración de la topología, la determinación del conjunto de rutas a emplear dentro de la subred, y la actualización de las tablas de encaminamiento en los conmutadores durante el proceso de inicialización de la subred o tras la ocurrencia de un cambio topológico.

Este capítulo describe detalladamente la infraestructura para la gestión de la subred prevista en la especificación de InfiniBand. Además, se aporta una descripción general de las principales tareas de gestión desempeñadas. En capítulos posteriores de esta memoria se profundizará más en cada una de estas tareas, proponiendo y evaluando diversas implementaciones para cada una de ellas.

El concepto de gestión en InfiniBand es muy general y abarca numerosos aspectos, además de la gestión de la subred: diagnóstico de prestaciones, gestión de dispositivos, soporte para SNMP, establecimiento y mantenimiento de las comunicaciones, etc. La infraestructura de gestión en InfiniBand abarca las siguientes clases (*management classes*):

- *Subnet Management (Subn)*. Especifica la forma en la que un gestor de la subred (SM) descubre y configura la subred, incluyendo la carga de las tablas de encaminamiento de los conmutadores y el establecimiento de parámetros operacionales (LIDs, SLtoVLMTs, etc.) en conmutadores y CAs.

- *Subnet Administration (SubAdm)*. Especifica la forma en la que las entidades de gestión o las aplicaciones acceden a la información relativa a la configuración actual. Una entidad denominada *SubnAdm agent* (SA) proporciona toda esta información.

- *Communication Management (CommMgt)*. Define la forma de establecer y terminar una conexión entre las QPs situadas en ambos extremos de la comunicación, y el medio para establecer rutas alternativas para dicha conexión (*automatic path migration*). Cada CA debe incluir un *CommMgt agent*, denominado *communication manager* (CM).

- *Performance Management (Perf)*. Especifica la forma en la que una aplicación de gestión recupera estadísticas relativas a prestaciones e información de errores desde los nodos.

- *Device Management (DevMgt)*. Define la forma de obtener atributos correspondientes a unidades y controladores de entrada/salida. Un ejemplo es la determinación del tipo de controlador y los parámetros del protocolo. La existencia de un *DevMgt agent* es opcional.

- *Baseboard Management (BM)*. Especifica la forma de enviar mensajes a elementos de gestión detrás de CAs y conmutadores. Estos elementos de gestión pueden estar distribuidos a lo largo del chasis. Un ejemplo es la monitorización de la temperatura del propio chasis. Todos los nodos deben proporcionar un *BM agent*.

- *SNMP Tunneling (SNMP)*. Especifica la forma y el formato de los datos para proporcionar el protocolo SNMP, encapsulando un paquete SNMP dentro de un paquete de gestión (MAD, *Management Datagram*). La existencia de un *SNMP agent* es opcional.

- *Vendor Specific (Vendor)*. Esta clase de gestión proporciona un marco de trabajo básico para que un fabricante defina operaciones de gestión no contempladas en la especificación de InfiniBand.

- *Application Specific (Application)*. Proporciona un marco de trabajo básico para la implementación de operaciones de gestión propias de las aplicaciones.

3.1 Elementos para la gestión de la subred

A continuación describiremos las entidades que intervienen en el proceso de gestión de la subred, así como los paquetes de control y el interfaz necesarios para implementar los protocolos de gestión en InfiniBand.

3.1.1 Gestor de la subred (SM)

En toda subred existe, al menos, un gestor de la subred (en adelante SM, *Subnet Manager*) que, por simplicidad, suele considerarse un proceso centralizado que reside en uno de los puertos de un conmutador, CA o encaminador. Un SM puede estar implementado en software, en hardware o en una mezcla de ambos.

Las principales funciones del SM son las siguientes:

- Descubre la topología de la subred que gestiona.
- Asigna una dirección (un LID) a cada puerto de la subred.
- Establece los caminos entre todos los dispositivos de la subred.
- Rastrea la subred periódicamente en busca de cambios topológicos.

Si existen múltiples SMs en la subred simultáneamente, se produce una contienda entre ellos, de la que finalmente uno se considera activo: aquel con mayor prioridad ejecutado en el puerto con menor GUID. Para ello, el protocolo de contienda (*handover*) se implementa por medio de la máquina de estados mostrada en la Figura 23.

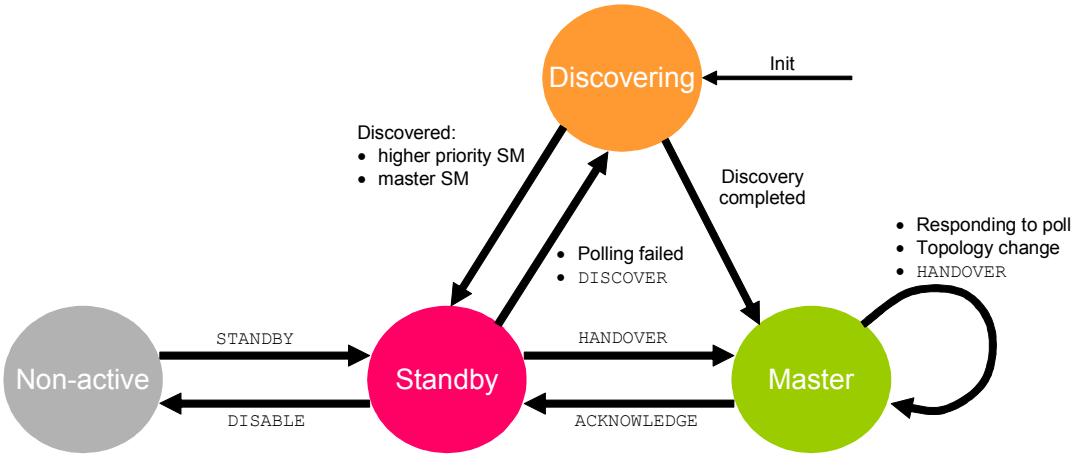


Figura 23. Máquina de estados para un gestor de la subred.

Una vez activado, un SM pasa al estado *Discovering* y comienza a explorar la topología de la subred. Si durante dicha exploración encuentra un SM de mayor prioridad, con igual prioridad y menor GUID, o en el estado *Master*, pasa al estado *Standby*. Si, por el contrario, consigue completar la exploración de la topología, pasa al estado *Master*.

Un *standby* SM sondea periódicamente la actividad del *master* SM. Si concluye que éste ha caído, vuelve al estado *Discovering*. Si el *master* considera que hay demasiados sondeos en la red, puede enviar una orden *DISABLE* a un *standby* SM, que pasaría al estado *Non-active*. De la misma forma, el *master* puede forzar a un *standby* SM a pasar a estado *Discovering* (mediante una orden *DISCOVER*), o cederle directamente el control (mediante una orden *HANDOVER*). En este último caso, el *standby* SM debe adquirir la información topológica (posiblemente a través del SA asociado al *master* SM actual), informar del cambio a toda la subred, y suplantar, mediante una orden *ACKNOWLEDGE*, al *master* SM actual.

En el estado *Non-active* el SM no realiza ninguna función, pero puede ser "despertado" por el *master* SM (mediante una orden *STANDBY*), pasando al estado *Standby*.

Un SM que alcanza el estado *Master* procede a inicializar la subred, asignando LIDs a todos sus elementos, determinando la MTU de las rutas y el número de canales virtuales ope-

rativos en cada puerto, y cargando las tablas de encaminamiento de los conmutadores. Posteriormente, rastrea la red con regularidad para detectar la ocurrencia de cambios topológicos o la activación de otros SMs. Si encuentra un *standby* SM con mayor prioridad, le cederá el control (mediante el protocolo descrito anteriormente) y pasará al estado *Standby*. En lo sucesivo, cuando hablemos del SM nos estaremos refiriendo al *master* SM.

Por último, los gestores del resto de las clases de gestión contempladas en InfiniBand se denominan colectivamente gestores de servicios generales (GSM, *General Services Manager*). Algunos ejemplos son el gestor de dispositivos (DM, *Device Manager*) o el gestor de comunicaciones (CM, *Communication Manager*).

3.1.2 Agentes de gestión de la subred (SMAs)

Para el desarrollo de sus funciones, el SM interacciona con ciertas entidades pasivas, denominadas agentes de gestión de la subred (SMA, *Subnet Management Agent*), presentes en todos los dispositivos de la subred (conmutadores, CAs y encaminadores). Los agentes se limitan a procesar los paquetes de gestión enviados por el SM, generando las respuestas oportunas para proporcionar información sobre el dispositivo. Algunos ejemplos son el tipo de dispositivo, el número de puertos implementados, si soporta funciones de gestión, etc. El SMA se encarga también de recibir información para la configuración del nodo local, como por ejemplo el LID asignado a un puerto o la ubicación del *master* SM. La única excepción a la pasividad de los agentes es la capacidad de emitir una notificación (*trap*) al SM ante la ocurrencia de determinados eventos.

La Figura 24 muestra un ejemplo de subred InfiniBand compuesta de 7 nodos terminales, 5 conmutadores y dos encaminadores, en la que se han representado los SMAs y el SM. Por simplicidad, no se han representado los CAs que dan acceso a la red a los nodos terminales.

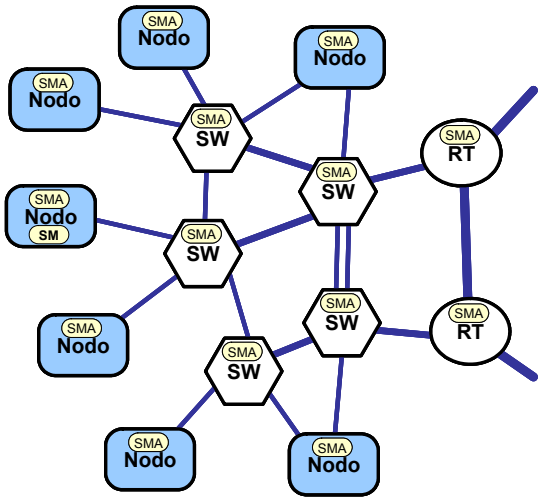


Figura 24. Entidades de gestión en una subred InfiniBand.

De manera análoga a los GSMs, también existen agentes para las clases de gestión distintas a la clase de gestión de la subred, denominados agentes de gestión de servicios generales (GSA, *General Services Management Agent*). Como excepción, en el caso de la clase de gestión de la comunicación (*communication management*), el gestor (CM) y el agente (CMA) son en realidad la misma entidad.

También es interesante comentar el papel del administrador de la subred (SA, *Subnet Administration agent*), es decir, el agente para la clase de administración de la subred (*subnet administration*). Se trata de una entidad pasiva que típicamente forma parte del propio SM, y cuya misión es responder a las consultas que otras entidades hacen sobre la información recogida en la base de datos del SM. Por ejemplo, el SA podría ser interrogado por parte una aplicación software sobre las rutas alternativas para alcanzar un determinado puerto y los DLIDs o DGIDs asociados a estas rutas.

3.1.3 Paquetes de gestión

MADs

Todo el proceso de gestión en InfiniBand (incluyendo la gestión de la subred) es llevado a cabo mediante el intercambio de paquetes MAD (*Management Datagram*) entre gestores y agentes. Estos paquetes emplean el tipo de servicio UD (*unreliable datagram*), es decir, se envían sin establecer una conexión previa y su recepción no está garantizada. La Figura 25 muestra la estructura de un paquete UD conteniendo un MAD.

La cabecera del datagrama (campo *UD headers* en la figura) incluye el LRH (cuyo formato se ha descrito en la Sección 2.2), la cabecera base del nivel de transporte (BTH, *Base Transport Header*) y la cabecera de transporte para paquetes de tipo datagrama (DETH, *Datagram Extended Transport Header*).

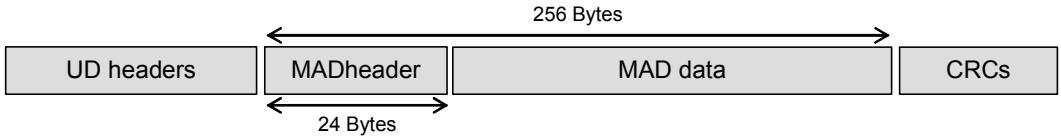


Figura 25. Un MAD encapsulado en un paquete InfiniBand.

Los MADs se clasifican según la clase de gestión, existiendo para cada uno de ellos un formato, un uso y un comportamiento asociado específicos. En la mayoría de los casos, el envío de un MAD (*request*) debe ir seguido de la correspondiente respuesta (*response*). En determinadas ocasiones, es necesario el envío consecutivo de una secuencia de MADs. La Tabla 3 muestra el formato base de los MAD. Los 24 bytes de cabecera (MADHeader) son comunes. Los 232 bytes siguientes varían en función del tipo de MAD. Los valores entre paréntesis junto al nombre de cada campo especifican un valor fijo.

Tabla 3. Formato base de un MAD.

Bits / Bytes	31-24	23-16	15-8	7-0
0	BaseVersion (1)	MgmtClass	ClassVersion (1)	R / Method
4	Status		ClassSpecific	
8	TransactionID			
12				
16	AttributeID		Reserved	
20	AttributeModifier			
24	MAD Data			
…				
252				

El campo MgmtClass (*management class*) identifica la clase de gestión a la que corresponde el MAD. En particular, los valores 0x01 y 0x81 se corresponden con las dos clases dedicadas a la gestión de la subred (*subnet management class*), denominadas *LID routed* y *directed route* respectivamente.

Cada clase de gestión soporta un conjunto de métodos que indican el tipo de operación a realizar en el destino, tras la recepción del MAD. Esta información se encuentra en el campo Method (*management class method*). El bit más significativo en este campo (bit R) indica si el MAD corresponde a una petición o a una respuesta, o si se trata de un mensaje que no requiere respuesta. Los métodos posibles son:

- *Get* (valor 0x01). Solicita un atributo a un dispositivo.
- *Set* (valor 0x02). Solicita el establecimiento de un atributo en un dispositivo.
- *GetResp* (valor 0x81). Responde a una petición *Get* o *Set*.
- *Send* (valor 0x03). Envía un datagrama que no requiere respuesta.
- *Trap* (valor 0x05). Se trata de un datagrama no solicitado enviado por un dispositivo para informar de un evento de interés.
- *Report* (valor 0x06). Reenvío (*forward*) de una notificación.
- *ReportResp* (valor 0x86). Respuesta a un *Report*.
- *TrapRepress* (valor 0x07). Indica a un dispositivo emisor de notificaciones que deje de enviarlas.

El campo Status aporta información sobre el estado de la operación para la que se está empleando el MAD. Los 8 bits de menor peso se emplean para indicaciones comunes a todas las clases de gestión, del tipo "MAD descartado", "campo inválido", etc. El uso de los demás bits es específico de cada clase; en concreto, para las clases de gestión de la subred sólo se emplea el bit de mayor peso (se tratará posteriormente).

El campo ClassSpecific sólo se emplea para las clases de gestión de la subred, estando reservado para el resto de clases.

El campo TransactionID es el identificador de la transacción. El contenido de este campo es específico de cada implementación, y sirve para identificar la operación que está siendo ejecutada por el emisor del MAD. Cuando una operación requiere el envío de una secuencia de MADs, en el campo TransactionID de todos los paquetes de la secuencia viaja el mismo valor. Además, el receptor de un MAD debe emplear el mismo valor en el paquete (o secuencia de paquetes) de respuesta.

El campo AttributeID identifica el atributo de interés para el MAD. Los atributos son estructuras cuyos componentes normalmente representan registros hardware en conmutadores, CAs, o encaminadores. Cada clase de gestión tiene su propio conjunto de atributos. Cada atributo tiene asignado un identificador único.

El campo AttributeModifier proporciona información adicional sobre los atributos para algunas combinaciones de atributo y método. Por ejemplo, cuando se solicita información sobre un puerto, este campo identifica el número de puerto de interés en el dispositivo destinatario.

Uso de timeouts

La especificación de InfiniBand prevé un mecanismo de *timeouts* para determinar el tiempo máximo que el emisor de un MAD debe esperar una respuesta al mismo y el tiempo máximo que puede transcurrir entre la recepción de dos MADs consecutivos dentro de una secuencia.

El cálculo del tiempo de espera se realiza en base al valor de dos parámetros relevantes. Por un lado, cada puerto almacena una estimación del retardo máximo de propagación para alcanzar cualquier otro puerto dentro de la subred (*subnettimeout*). Este valor es establecido por el SM una vez concluida la exploración de la topología de la subred. Cada puerto debe almacenar además una estimación del tiempo entre la recepción de un MAD y la emisión de la correspondiente respuesta o entre la emisión de dos MADs consecutivos dentro de una secuencia (*resptimevalue*). Este segundo parámetro viene dado por la tecnología, siendo su valor por defecto de 1 ms en las versiones 1.0 y 1.0.a de la especificación de InfiniBand. Por su parte, la versión 1.1 de la especificación [36] fija un valor máximo de 4.3 s para este parámetro.

La expresión *2*subnettimeout+resptimevalue* permite calcular el tiempo máximo de espera de una respuesta. Por otro lado, para estimar el tiempo máximo entre la recepción de dos MADs consecutivos dentro de una secuencia se emplea la expresión: *subnettimeout+resptimevalue*.

La especificación de InfiniBand no detalla qué hacer si una vez transcurrido el tiempo de espera no se ha recibido la respuesta al MAD. El emisor puede optar por repetir el envío

del paquete de gestión. Tras varios reintentos sin respuesta, el emisor deberá liberar todos los recursos asociados a la operación que estaba ejecutando cuando envió (por primera vez) el MAD, dando por supuesto que el destinatario no es alcanzable en estos momentos. Si el destinatario del paquete de gestión es alcanzable, pero la respuesta llega con demora debido (por ejemplo) a que parte de la ruta de retorno está congestionada, es posible que el emisor reciba varias respuestas casi al mismo tiempo, correspondientes a cada uno de los reintentos. Como es lógico, en estos casos el emisor sólo deberá considerar la primera de las respuestas, descartando el resto.

SMPs

Los MADs asociados a la gestión de la subred se denominan SMPs (*Subnet Management Packet*). Los MADs empleados por el resto de clases se denominan GMPs (*General Management Packet*). Los SMPs sólo utilizan el VL15, siendo además los únicos paquetes permitidos en dicho canal virtual. Nunca sobrepasan los límites de la subred y pueden emplear encaminamiento en base a destino (*destination/LID routing*) o encaminamiento dirigido (*directed routing*). El formato del SMP para el primer tipo de encaminamiento se muestra en la Tabla 4.

Tabla 4. SMP LID routed.

Bits / Bytes	31-24	23-16	15-8	7-0
0 ... 20	Common MAD Header			
24 28	M_Key			
32 ... 60	Reserved (32 bytes)			
64 ...	SMP Data (64 bytes)			
128 ... 252	Reserved (128 bytes)			

Los SMPs que emplean encaminamiento basado en destino son encaminados de la misma manera que los paquetes de datos. Esto quiere decir que el SMP incluye el DLID del destinatario en su cabecera, y una consulta a la tabla de encaminamiento en cada conmutador intermedio proporciona el puerto de salida. La Figura 26 muestra el procesamiento de este tipo de SMPs en cada salto.

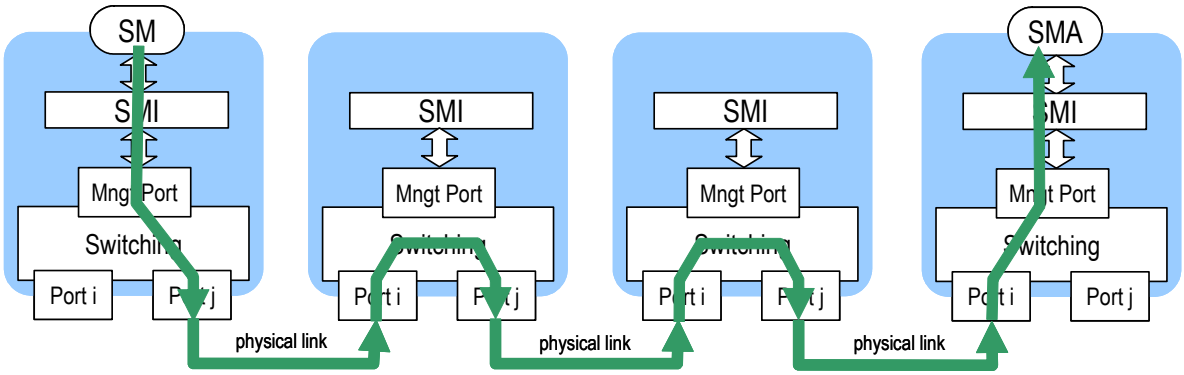


Figura 26. Ejemplo de SMP encaminado en base a destino.

La Tabla 5 muestra el formato de los SMPs que emplean encaminamiento dirigido. En este caso, el propio paquete incluye la secuencia de puertos de salida a emplear en cada uno de los conmutadores intermedios de la ruta. Un SMP de este tipo sólo puede partir del SM. El campo D (*upstream/downstream direction*) indica la dirección del SMP, es decir, si éste viaja desde o hacia el SM. Initial Path indica la secuencia de puertos de salida a emplear por el SMP; cada byte en este campo representa un número de puerto de conmutador. El campo Return Path es análogo al anterior, y se establece durante el "viaje de ida" del SMP. Hop Count indica el número de puertos en los campos de ruta, mientras que Hop Pointer apunta al puerto de salida para el SMP en el conmutador actual. Estos dos últimos campos componen el campo ClassSpecific del MAD.

Tabla 5. SMP directed route.

Bits / Bytes	31-24	23-16	15-8	7-0
0	Common MAD Header1			
4	D	Status	Hop Pointer	Hop Count
… 20	Common MAD Header2			
24 28	M_Key			
32	DrSLID		DrDLID	
36 … 60	Reserved (28 bytes)			
64	SMP Data (64 bytes)			
128	Initial Path (64 bytes)			
192 … 252	Return Path (64 bytes)			

El encaminamiento dirigido contempla la posibilidad de que una ruta se complete con dos segmentos optativos de encaminamiento basado en destino, situados con anterioridad y posterioridad al segmento de encaminamiento dirigido. Para ello, el paquete incorpora los

campos DrSLID y DrDLID. Si no existen dichos segmentos, ambos campos estarán asignados a la *permissive address*. Por el contrario, si existe el segmento inicial, el campo DrSLID contendrá el LID del emisor y, si existe el segmento final, el campo DrDLID contendrá el LID del destinatario. Por su parte, los campos SLID y DLID del paquete cambiarán de valor en función del segmento en el que se encuentre el paquete. Así, dentro de un segmento de encaminamiento dirigido estarán asignados a la *permissive address*, y dentro de un segmento de encaminamiento basado en destino estarán asignados a los respectivos emisor y destinatario del segmento.

Los SMPs con encaminamiento dirigido son más lentos que los encaminados en base a destino. La razón es que deben ser procesados en cada uno de los conmutadores intermedios de la ruta por parte del SMI (*Subnet Management Interface*). Esta entidad se encarga de extraer el puerto de salida para el SMP en el conmutador actual (campos Initial Path y Return Path), de actualizar el valor del campo Hop Pointer en cada salto, de construir la ruta de regreso (campo Return Path) durante el viaje de ida del SMP, y de modificar el valor de los campos SLID y DLID en la cabecera del SMP cuando éste incorpora algún segmento con encaminamiento en base a destino. La Figura 27 muestra un ejemplo de este procesamiento.

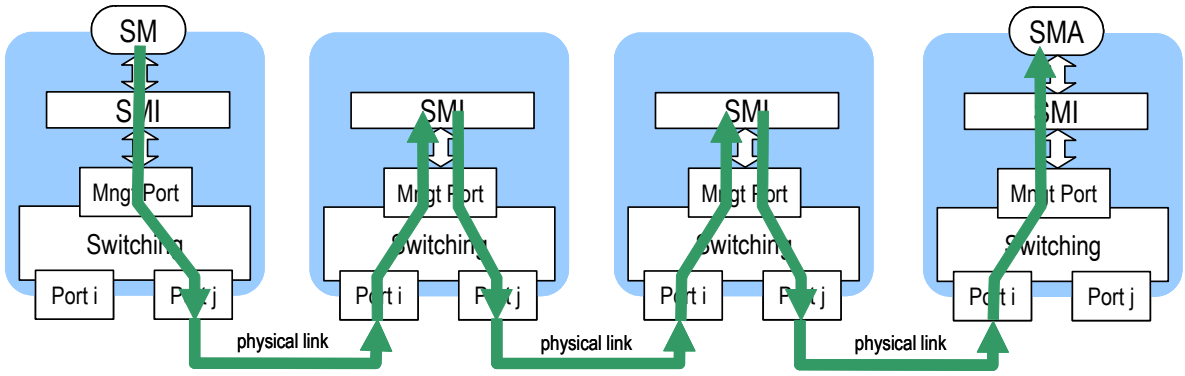


Figura 27. Ejemplo de SMP con encaminamiento dirigido.

En ambos tipos de SMPs, el campo M_Key (*management key*) contiene una clave de 64 bits utilizada en el proceso de autentificación del SM emisor, requerido para autorizar las operaciones de inicialización y configuración incluidas en el paquete. El receptor del SMP ignorará su contenido si el valor de este campo no coincide con un parámetro local previamente configurado por el SM.

El campo SMP Data, de 64 bytes de longitud, contiene el valor del atributo indicado en la cabecera del SMP. Los distintos tipos de atributos se describen a continuación. Finalmente, los campos reservados se asignan a cero.

Métodos y atributos de gestión de la subred

Las clases de gestión de la subred emplean un subconjunto de los métodos antes descritos: *SubnGet*, *SubnSet*, *SubnGetResp*, *SubnTrap* y *SubnTrapRepress*.

Los SMPs más comunes son los de tipo *SubnGet* y *SubnSet*. El SM sondea la subred y aprende su topología mediante el envío de paquetes *SubnGet*. Cada destino responde enviando un *SubnGetResp*, que incluye los datos solicitados. Posteriormente, el SM configura los dispositivos de la subred enviando paquetes *SubnSet*. Cada destino responde enviando un paquete *SubnGetResp*, que incluye los valores establecidos para los atributos indicados en el paquete *SubnSet*. Entonces, el SM inspecciona el paquete de respuesta para comprobar la correcta asignación de atributos en el destino.

La Tabla 6 muestra algunos de los atributos empleados para la gestión de la subred, señalando en cada caso los tipos de SMPs que los emplean y una breve descripción de su contenido.

Tabla 6. Algunos atributos empleados por la clase de gestión de la subred de InfiniBand.

Nombre	Métodos	Descripción
Notice	*Get, Set, Trap, TrapRepress*	Información referente a una notificación: tipo de evento notificado (puerto en servicio, puerto fuera de servicio, cambio en el estado de un puerto, etc.), número de puerto afectado, LID, etc.
NodeInfo	*Get*	Datos genéricos sobre un dispositivo: tipo (conmutador, CA o encaminador), número de puertos, GUID, etc.
SwitchInfo	*Get, Set*	Información sobre un conmutador: tamaño de la FT, puerto por defecto, cambio en el estado de un puerto, etc.
PortInfo	*Get, Set*	Información sobre un puerto: LID/LMC, estado, MTU (soportada/operativa), VLs (soportados/operativos), tamaño de la VLAT, *management key*, etc. El valor contenido en el campo AttributeModifier indica el número del puerto
SLtoVLMappingTable	*Get, Set*	SLtoVLMT para una pareja puerto de entrada – puerto de salida. AttributeModifier especifica ambos puertos
VLArbitrationTable	*Get, Set*	VLAT. 32 entradas (pares VL – peso) en cada SMP. AttributeModifier indica el puerto al que corresponde la VLAT y qué parte de la tabla (alta o baja prioridad) se está enviando
LinearForwardingTable	*Get, Set*	LFT. Un bloque de 64 entradas (puertos de salida) en cada SMP. AttributeModifier indica el bloque que se está enviando
RandomForwardingTable	*Get, Set*	RFT. 16 entradas (pares LID – puerto de salida) en cada SMP. AttributeModifier indica el bloque que se está enviando
MulticastForwardingTable	*Get, Set*	Tabla para encaminamiento multidestino. 32 entradas por SMP. AttributeModifier indica el bloque que se está enviando
SMInfo	*Get, Set*	Información sobre un SM: estado (*Master*, *Standby*,...), prioridad, etc. AttributeModifier permite especificar una orden (DISCOVER, DISABLE, etc.)

3.1.4 Interfaz de gestión de la subred (SMI)

Las entidades productoras y consumidoras de SMPs son los SMs y SMAs. Como se requiere la presencia de un SM en la subred (al menos) y un SMA en cada dispositivo, podemos concluir que algunos dispositivos incluirán ambos simultáneamente. Para que ambos tipos de entidades puedan coexistir en el mismo dispositivo es necesaria la presencia de un interfaz de gestión de la subred (SMI, *subnet management interface*).

La relación entre estas tres entidades se muestra en la Figura 28. El SMI se implementa sobre un puerto, y proporciona servicios al SM y al SMA. Como se desprende de la figura, la comunicación con el SMA o el SM implementados en un dispositivo es siempre a través del SMI (incluso la comunicación entre ambos). En los conmutadores, solamente el puerto 0 soporta al SMI, estando destinado a la gestión, teniendo para ello asignado un LID propio. En el caso de los CAs y encaminadores, todos los puertos soportan un SMI, disponiendo de LID propio.

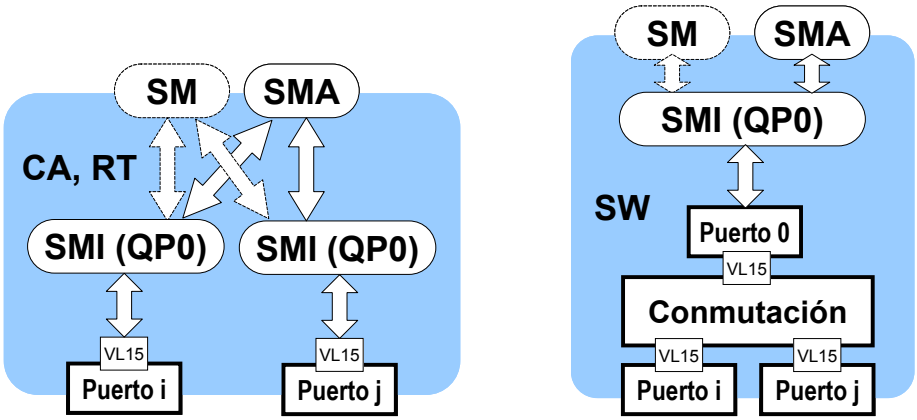


Figura 28. Disposición del SMI, SMA y SM en los dispositivos InfiniBand.

Como ya se adelantó en el Capítulo 2, existe una dirección especial, denominada *permissive address*, que permite al SM comunicarse con un SMI sin necesidad de conocer el LID asignado al puerto. En un conmutador, todos los paquetes recibidos en puertos distintos del 0 con un DLID igual a la dirección *permissive address* son enviados directamente al puerto de gestión.

Los SMPs recibidos en el SMI son validados y descartados en caso de no cumplir algunos de los requisitos exigidos, como contener el valor 15 en el campo VL, especificar una de las clases de gestión de la subred (campo MgmtClass) o contener alguno de los atributos (campo AttributeID) soportados por la clase especificada.

Cuando un conmutador, CA o encaminador soporta un SM, algunos SMPs recibidos en el SMI irán destinados a éste, mientras que otros irán destinados al SMA. La clase, método y atributo del SMP determinará cuál es la entidad destinataria del mismo. En concreto, los métodos *SubnGet*, *SubnSet* y *SubnTrapRepress* tienen como destinatario el SMA. Los métodos *SubnGetResp* y *SubnTrap* tienen como destinatario el SM. Por último, los SMPs *SubnGet, SubnSet* y *SubnGetResp* con atributo *SMInfo* tienen como destinatario el SM.

El SMI está asociado en cada puerto al par de colas QP0. La especificación de InfiniBand define otro interfaz de gestión, denominado GSI (*General Services Interface*). Éste es el interfaz empleado por todas las clases de gestión distintas de la clase de gestión de la subred, y está asociado al par de colas QP1 en cada puerto. La Figura 29 resume las principales características de ambos interfaces de gestión.

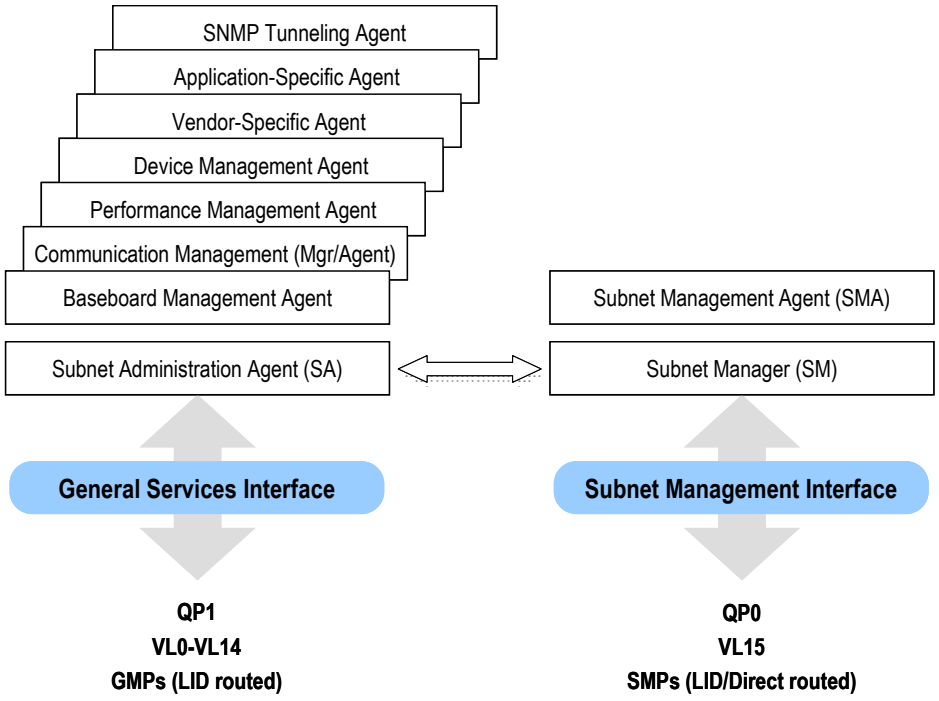


Figura 29. Interfaces de gestión.

3.2 Mecanismo de gestión de la subred

En esta sección se describen las funciones principales del mecanismo de gestión de la subred, tal y como han sido descritas en la especificación de InfiniBand. En la mayoría de los casos, la especificación deja abierta la forma en que los procesos de gestión son llevados a cabo, estableciendo clara y únicamente su finalidad. En algunas ocasiones, sin embargo, se sugieren posibles implementaciones. En cualquier caso, el funcionamiento interno del SM y los SMAs debe ser definido por cada fabricante, sin que necesariamente se deba mantener la compatibilidad entre los productos desarrollados por distintos fabricantes. La responsabilidad de mantener dicha compatibilidad es trasladada al usuario.

Las tareas que comprende el mecanismo de gestión de la subred son las siguientes:

- Adquisición de topología.
- Detección de cambios.
- Obtención de rutas.
- Configuración de dispositivos.
- Activación de la subred.

3.2.1 Adquisición de la topología de la subred

Como hemos visto anteriormente, una vez activado, un SM pasa al estado *Discovering* e inicia un proceso de descubrimiento de la subred (*subnet discovery*). En este proceso se identifican todos los dispositivos existentes, incluyendo otros SMs. Además, el proceso de descubrimiento permite al SM recopilar una serie de datos sobre los dispositivos de la subred, como son, por ejemplo, el número de canales virtuales operativos, la MTU y velocidad de cada puerto, o el tipo y tamaño de la tabla de encaminamiento de cada conmutador. Toda esta información será empleada posteriormente durante la fase de configuración de la subred.

La implementación concreta del algoritmo para la exploración de la topología no está detallada en la especificación de InfiniBand. Ésta únicamente establece que el SM debe enviar una serie de SMPs para encontrar todos los dispositivos (y SMs) presentes en la subred[1]. No obstante, se sugiere la implementación de un SM centralizado. El trabajo del *Management Working Group* de InfiniBand apunta en la misma dirección [37].

El SM ejecuta una búsqueda sobre el grafo que modela la subred. Cada vez que descubre un nuevo conmutador, procede a explorar los dispositivos conectados al otro extremo de cada uno de sus puertos activos. Para averiguar la naturaleza de cada dispositivo, el SM emplea SMPs *SubnGet(NodeInfo)*[2]. El estado de los puertos es recopilado mediante SMPs *SubnGet(PortInfo)*. Los SMAs responden a estos paquetes con SMPs con el método *SubnGetResp*. La búsqueda por una rama finaliza cuando un conmutador no tiene más puertos activos, o la exploración alcanza un CA o un encaminador. La Figura 30 muestra el momento en que el SM comienza a explorar uno de los puertos del conmutador al que está conectado el nodo terminal en el que reside.

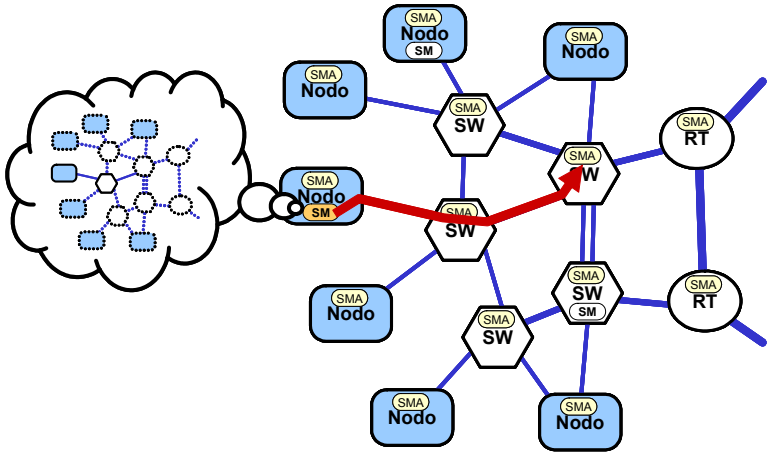


Figura 30. Ejemplo de exploración de la subred.

[1] La especificación también contempla la posibilidad de que la topología de la subred sea obtenida por el SM directamente desde un fichero de disco. Por motivos de eficiencia del mecanismo de gestión, hemos desechado esta opción.

[2] A menudo nos referiremos a un tipo de SMP concreto mediante el nombre del método empleado y, a continuación, entre paréntesis, el nombre del atributo contenido en él.

Cuando la exploración alcanza a un nuevo dispositivo, el SM le asigna un LID, que le comunica junto con el suyo propio. De esta forma, el SMA presente en dicho dispositivo podrá, en el futuro, remitir información al SM. Para ello, el SM envía un SMP *SubnSet(PortInfo)*[1] para cada puerto de la subred.

En cuanto al mecanismo de encaminamiento de los SMPs empleados por el proceso de descubrimiento, hay que considerar la posibilidad de que ciertos puertos no tengan asignado un LID. Esto sucede, por ejemplo, tras la activación inicial de la subred. También es posible que los conmutadores no dispongan todavía de tablas de encaminamiento. Por todo ello, los paquetes de exploración deben emplear encaminamiento dirigido.

En el Capítulo 5 de esta memoria se describen y evalúan dos implementaciones concretas para el proceso de exploración de la topología de la subred en InfiniBand.

3.2.2 Detección de cambios topológicos

El SM (en estado *Master*) es la entidad encargada de detectar la ocurrencia de cambios topológicos. Más concretamente, el SM debe ser capaz de descubrir una variación en el estado de alguno (o algunos) de los puertos de la subred que gestiona. Un cambio de estado en un puerto puede deberse a la activación o desactivación del dispositivo que hay al otro lado del enlace, o bien a la adición o supresión de enlaces entre dispositivos. La desactivación de dispositivos incluye el caso de fallo. Por otro lado, nótese que si el cambio consiste en la reasignación de alguno de los enlaces de la subred, en realidad el SM detectará la ocurrencia de dos cambios topológicos consecutivos.

La especificación de InfiniBand contempla dos mecanismos complementarios para la detección de cambios. Por un lado, el SM realiza barridos periódicos de la subred (*subnet sweeping*). La frecuencia de estos barridos puede ajustarse en función de parámetros tales como el tamaño de la subred o el tiempo deseado de respuesta ante fallos. El SM se comunica con el SMA asociado a cada conmutador, y examina todos sus puertos buscando posibles cambios en su estado. Para ello, emplea SMPs *SubnGet(PortInfo)*. En concreto, el SM examina el valor del componente *PortState*[2] en el atributo *PortInfo* devuelto por el SMA. El orden concreto de barrido no ha sido especificado, sin embargo, es evidente que la velocidad de detección del cambio es más dependiente de la frecuencia de barrido que del orden en que se realicen las consultas.

Para agilizar el proceso, y dado que los cambios son poco frecuentes, puede utilizarse un único SMP *SubnGet(SwitchInfo)* para detectar cambios globales en el conmutador. Uno de

[1] En concreto, el identificador asignado al puerto y la ubicación del SM vienen dados por los componentes *LID*, *LMC*, *MasterSMLID* y *MasterSMSL* del atributo *PortInfo*.

[2] En los 4 bits de este componente se codifica el estado actual del puerto (ver sección 2.4.1).

los componentes del atributo *SwitchInfo*, denominado *PortStateChange*[1], es el que indica si se ha producido un cambio en el estado de alguno de los puertos del conmutador. En caso de existir tal cambio, otros SMPs realizarán un análisis individualizado de cada puerto.

Además de los barridos periódicos ejecutados por el SM, existe la posibilidad de que el SMA alojado en un conmutador informe acerca de la ocurrencia de un cambio en el estado de alguno de sus puertos. Para ello, envía al SM un SMP *SubnTrap(Notice)*[2]. Obviamente, para poder realizar este envío, el SMA local debe conocer el LID del puerto al que está asociado el SM. Opcionalmente, el SMA puede continuar enviando notificaciones hasta que recibe un SMP *SubnTrapRepress(Notice)* por parte del SM. Aunque los SMPs no emplean reconocimientos (emplean el tipo de servicio UD), una notificación "exitosa" puede reducir notablemente el tiempo de detección del cambio.

Una vez que el SM tiene conocimiento de la ocurrencia de un cambio en la estructura de la subred, deberá actualizar inmediatamente su información topológica. La especificación de InfiniBand no detalla la forma concreta en la que esta tarea debe ser llevada a cabo. Simplemente se establece que si el cambio consiste en la desactivación de un puerto, el SM deberá actualizar el conjunto de rutas dentro de la subred. Si, por el contrario, el cambio consiste en la activación de un puerto, el SM deberá descubrir el dispositivo o dispositivos que han aparecido al otro lado del enlace, e igualmente actualizar las rutas.

Finalmente, una cuestión importante relativa a la detección (y posterior asimilación) de cambios es la diferenciación entre los mismos. Es posible que un mismo cambio topológico sea detectado por múltiples dispositivos. También puede ocurrir que múltiples cambios consecutivos se solapen en el tiempo, confundiendo al mecanismo de gestión de la subred. En cuanto a la detección múltiple, todos los cambios detectados por el SM durante el mismo proceso de barrido serán asimilados simultáneamente. Por otro lado, cualquier cambio adicional en un dispositivo que suceda después de que todos sus vecinos hayan sido sondeados por el proceso de barrido actual, será detectado durante la ejecución del siguiente barrido.

3.2.3 Obtención de rutas para encaminamiento dentro de la subred

Con la información topológica recopilada, el SM calcula las rutas que emplearán los paquetes de aplicación dentro de la subred. Estas rutas se encuentran distribuidas en forma de tablas de encaminamiento situadas en los conmutadores. Como se muestra en la Figura 12 (Sección 2.5.1, en página 20), las tablas proporcionan un puerto de salida para cada DLID.

[1] *PortStateChange* es un bit que se activa cada vez que el estado de alguno de los puertos del conmutador pasa de *Down* a *Initialize*, de *Initialize* a *Down*, de *Armed* a *Down*, o de *Active* a *Down*.

[2] El atributo *Notice* contiene un componente (*TrapNumber*) que indica el tipo de evento que está siendo notificado por medio del *trap*. En concreto, nos interesa el evento *"link state changed"* (*TrapNumber*=128). El componente *DataDetails* del atributo *Notice* aporta información adicional sobre el *trap*. En el caso del *trap* número 128, *DataDetails* indica el LID del puerto cuyo estado ha cambiado.

Por consiguiente, el encaminamiento en InfiniBand es determinista, es decir, sólo se ofrece una opción de encaminamiento para cada destino. Además, la información sobre el puerto de entrada no es tenida en cuenta a la hora de encaminar los paquetes dentro de la subred.

Como es obvio, la información conjunta de las tablas debe ser coherente para garantizar la conectividad entre todos los puertos de la subred. En otras palabras, debe existir al menos un camino a través de la subred entre cada par de puertos. Además, la especificación de InfiniBand deja muy claro que, independientemente del algoritmo empleado para obtener las entradas en las tablas de encaminamiento, el conjunto de rutas resultantes debe ser libre de bloqueo [26]. En la actualidad, existen numerosas propuestas para la obtención de rutas en InfiniBand [23, 44, 48, 49, 78, 79]. En el Capítulo 6 de esta memoria se propone un algoritmo de cálculo de rutas alternativo, que persigue reducir al máximo el tiempo de cómputo, manteniendo en todo momento la libertad de bloqueos.

3.2.4 Configuración de dispositivos

La actividad del SM tiene como fin último la configuración de los dispositivos presentes en la subred, y su posterior activación. Mediante SMPs de tipo *SubnSet*, el SM establece en puertos y conmutadores el valor de parámetros tales como el número de canales virtuales operativos, o las tablas de mapeo (SLtoVLMT). Los SMAs deben confirmar la configuración con SMPs de tipo *SubnGetResp*, que permitirán al SM comprobar la correcta asignación de los valores enviados. La Tabla 7 muestra algunos parámetros de funcionamiento representativos que el SM debe inicializar (la lista completa es mucho mayor).

En esta Tesis, hemos centrado nuestra atención en la configuración de las tablas para encaminamiento unidestino (LFT y RFT). El SM distribuye estas tablas a los conmutadores de la subred por medio de los SMPs *SubnSet(LinearForwardingTable)* y *SubnSet(RandomForwardingTable)*. Estos SMPs también deben emplear encaminamiento dirigido, pues no pueden hacer uso de las tablas que pretenden actualizar.

La especificación de InfiniBand no define el orden en que las tablas de encaminamiento deben ser configuradas. Sin embargo, una distribución descontrolada de las mismas puede conducir a situaciones de bloqueo. La razón es que, aunque las rutas antes y después del envío de las nuevas tablas sean libres de bloqueo, la coexistencia de dos esquemas de encaminamiento durante el periodo transitorio no es necesariamente libre de bloqueo. En el Capítulo 7 se comparan varias implementaciones para el proceso de distribución de tablas de encaminamiento.

En cuanto a la obtención y distribución de otros parámetros de funcionamiento, como las tablas de arbitraje (VLAT) o las de mapeo (SLtoVLMT), dicho estudio sobrepasa los objetivos de este trabajo. A este respecto, en [1] se propone una metodología para computar las tablas de arbitraje para proporcionar calidad de servicio (QoS, *Quality of Service*) en InfiniBand.

Tabla 7. Parámetros establecidos por el SM.

Atributo	Componente	Descripción
PortInfo	LID	LID base asignado al puerto
PortInfo	LMC	LMC asignado al puerto
PortInfo	OperationalVL	Número de VLs de datos operativos en el puerto. Establecido en función del número de VLs implementados en ambos extremos
PortInfo	NeighborMTU	MTU operativa en el puerto. Establecido en función de la MTU soportada en ambos extremos
PortInfo	SubnetTimeout	Retardo máximo para alcanzar cualquier otro puerto dentro de la subred
PortInfo	MasterSMLID	LID asignado al puerto donde reside el *master* SM
PortInfo	MasterSMSL	SL requerido para enviar un mensaje al *master* SM
PortInfo	VLHighLimit	Límite de alta prioridad para el arbitraje de canales
PortInfo	M_Key	*management key* (sección 3.1.3)
SwitchInfo	DefaultPort	Puerto por defecto en conmutadores que implementan la RFT
VLArbitrationTable	-	Tabla de arbitraje
SLtoVLMappingTable	-	Tabla de mapeo
LinearForwardingTable	-	Tabla de encaminamiento unidestino LFT
RandomForwardingTable	-	Tabla de encaminamiento unidestino RFT
MulticastForwardingTable	-	Tabla de encaminamiento multidestino

3.2.5 Activación de la subred

Una vez configurados todos los dispositivos de la subred, el SM debe proceder a la activación de sus puertos, con el objetivo de permitir el intercambio de tráfico de aplicación entre ellos. En el caso concreto del encendido inicial, la activación afectará a todos los puertos de la subred. Por el contrario, en el caso de la aparición de nuevos dispositivos o enlaces en la topología, es posible que la activación sólo afecte a un subconjunto de los puertos, si bien, como veremos más tarde, algunos mecanismos de distribución de tablas podrían requerir la desactivación previa de todos los puertos de la subred.

Para activar la subred, en primer lugar, el SM envía un SMP *SubnSet(PortInfo)* a todos los puertos, especificando el valor *Armed* para el componente *PortState* del atributo *PortInfo*. Como se detalla en la Sección 2.4.1, la transición desde el estado *Armed* al estado *Active* (en el que la actividad de los puertos es total) se producirá bien mediante un nuevo SMP *SubnSet(PortInfo)* con el valor *Active* para el componente *PortState*, bien de forma automática, tras la recepción en el puerto del primer paquete de datos. Una vez que todos los puertos han pasado al estado *Active*, la subred es totalmente operativa.

En este caso, los SMPs empleados para la activación de los puertos pueden hacer uso de encaminamiento en base a destino, dado que las nuevas tablas de encaminamiento ya han sido cargadas en los conmutadores de la subred.

3.3 Gestión de las comunicaciones y migración automática de rutas

A continuación se describe, de forma muy general, la forma de establecer y mantener las conexiones entre pares de colas (QPs) en InfiniBand. Aunque trataremos asuntos del nivel de transporte, algunos de estos mecanismos son susceptibles de ser empleados ante la ocurrencia de un cambio topológico en la subred [41, 54].

3.3.1 Gestión de las comunicaciones

La clase de gestión de la comunicación (*communication management*) de InfiniBand especifica los protocolos y mecanismos para establecer, mantener y liberar conexiones para los servicios de nivel de transporte RC, UC y RD. También especifica un protocolo de resolución de identificadores de servicio (*Service ID*) que permite a los clientes del servicio UD localizar la QP que proporciona un determinado servicio. La naturaleza del servicio de transporte *raw datagram* no requiere ningún tipo de gestión de la comunicación.

Las conexiones entre QPs son manejadas por los gestores de la comunicación (CM, *Communication Manager*) en ambos extremos. En cada CA debe residir un CM. Los CMs se comunican entre sí empleando los MADs específicos de la clase de gestión de la comunicación, y el interfaz de gestión GSI (en la cola QP1). El CM iniciador es el responsable de recopilar o calcular la mayor parte de la información necesaria para establecer la conexión. Gran parte de esta información está disponible a través del SA proporcionado por el SM[1].

3.3.2 Migración automática de rutas

La migración automática de rutas (APM, *Automatic Path Migration*) es una función opcional del nivel de transporte de InfiniBand que hace posible la migración de una conexión de una ruta primaria hacia una ruta alternativa, proporcionando de esta forma tolerancia a fallos. Esta función está disponible para los servicios de transporte RC, UC y RD.

Los mensajes que intercambian los CMs para el establecimiento de la conexión especifican opcionalmente información sobre un par de puntos terminales alternativos (LIDs/GIDs), para soportar la migración automática de rutas. De esta forma, si la ruta principal falla, la conexión conmuta a la ruta alternativa, que se convierte en primaria. La ruta alternativa hacia el destino puede ser especificada en el momento de establecer la conexión, o bien en cualquier otro momento.

[1] El acceso al SA es posible gracias a que el SM programó en su momento el puerto en el que se encuentra el CM con el valor *MasterSMLID*.

Como se ha dicho, el CM aprende todas las rutas entre el nodo local y el remoto haciendo consultas al SA. La ruta alternativa puede o no emplear el mismo puerto de salida del CA. El SM posibilita el empleo del mismo puerto para la ruta alternativa mediante la asignación de múltiples LIDs/GIDs a los puertos físicos.

3.4 Gestión de cambios en otras redes

Para concluir este capítulo, en esta sección describiremos, someramente, la forma en la que se llevan a cabo las tareas de gestión descritas para InfiniBand en las diferentes tecnologías de red presentadas en la Sección 2.9 (página 30).

En la literatura, los mecanismos de gestión (o reconfiguración) de Autonet y Myrinet están descritos en mayor profundidad. En capítulos posteriores volveremos sobre algunos aspectos concretos del mecanismo de gestión en estas redes.

3.4.1 ServerNet

En ServerNet, los fallos son detectados mediante una lógica de auto comprobación (*self-checking logic*) [33]. Cada enlace es comprobado periódicamente mediante el intercambio de símbolos de control.

En caso de fallo en enlaces o elementos de encaminamiento, los paquetes son reenviados o reencaminados por caminos alternativos a través de la red. Los paquetes perdidos en enlaces caídos son detectados con un protocolo de reconocimiento.

Este aislamiento de fallos mediante hardware permite al software de gestión su reparación. El software de gestión emplea mensajes de control, intercalados con los mensajes de datos, para acceder a los registros internos de los dispositivos o actualizar las tablas de encaminamiento en los conmutadores.

3.4.2 Autonet

Cada conmutador en Autonet incorpora un microprocesador (un *Motorola 68000*), que ejecuta el programa *Autopilot*. Este programa monitoriza el estado de los puertos del conmutador e implementa el protocolo de reconfiguración de la red [72].

El estado de los puertos es supervisado periódicamente. En caso de cambio, se dispara automáticamente el protocolo distribuido de reconfiguración, que obtiene la nueva topología y las nuevas tablas de encaminamiento. El protocolo tiene cinco pasos. En primer lugar, los

conmutadores intercambian una serie de paquetes de control para obtener un árbol de expansión (*spanning tree*) [59] de la topología. A continuación, la información topológica y el árbol de expansión se propagan hacia la raíz del mismo (el conmutador detector del cambio e iniciador del proceso). Después, se procede a la asignación (de forma centralizada) de direcciones a los nodos terminales y conmutadores. Tras esta etapa, toda la información recopilada en la raíz es de nuevo propagada a todos los conmutadores de la red, empleando el propio árbol de expansión. Finalmente, y ya de forma distribuida, los conmutadores calculan y cargan sus propias tablas de encaminamiento.

3.4.3 Myrinet

Myrinet es una red robusta a fallos hardware, debido a un sistema de supervisión continua. A diferencia de Autonet, el mecanismo de reconfiguración es ejecutado por la tarjeta de interfaz en los nodos terminales. Dicha tarjeta incorpora un chip (*LANai*) que incluye un microprocesador RISC de 32 bits, donde se ejecuta el programa de control de Myrinet (MCP, *Myrinet Control Program*). Entre las tareas realizadas por este software, se encuentra el proceso de reconfiguración[1].

Myrinet incorpora dos mecanismos para la detección de cambios. En primer lugar, los conmutadores y las tarjetas de interfaz envían periódicamente símbolos de control. Por otro lado, el *mapper* (uno de los terminales donde se ejecuta el MCP) lleva a cabo periódicamente una exploración de la topología, con el objetivo de compararla con la anterior y determinar si ha habido algún cambio. Cuando el nuevo mapa de la red no coincide con el anterior, el nodo *mapper* distribuye el nuevo mapa a los MCPs en el resto de terminales de la red, para que éstos construyan finalmente sus tablas.

3.4.4 QsNet

QsNet implementa tolerancia a fallos mediante protocolos de detección de fallos y retransmisión automática de paquetes.

La detección de fallos en QsNet está implementada en hardware. En primer lugar, cada paquete individual debe ser reconocido por el interfaz de red de destino, mediante un paquete de reconocimiento que éste envía al interfaz de origen. Si un interfaz de red (*Elan*) detecta un error durante la transmisión, envía inmediatamente una notificación del mismo (sin esperar al reconocimiento). Si un conmutador (*Elite*) detecta un error, envía un mensaje de error hacia la fuente y hacia el destino.

[1] La versión actual del software de control de Myrinet se denomina *Myricom GM*.

En caso de cambio, los interfaces de red en cada extremo y los conmutadores intermedios aíslan el enlace y/o el conmutador que ha fallado. En concreto, el emisor recibe una notificación sobre el componente que ha fallado, tras lo cual puede reintentar la retransmisión del paquete un determinado número de veces. Si esto no funciona, puede reconfigurar sus tablas para evitar el componente que ha fallado.

Capítulo 4
Modelado de la arquitectura InfiniBand

Los resultados presentados en esta memoria han sido obtenidos empleando técnicas de simulación. Esto ha supuesto una tarea previa de obtención de un modelo de la arquitectura de InfiniBand. Para desarrollar este modelo, hemos empleado la herramienta de modelado y simulación OPNET *Modeler*. Como resultado de la tarea de modelado, hemos obtenido una plataforma sobre la que programar las distintas estrategias de gestión de la subred propuestas y analizar detalladamente sus prestaciones.

En este capítulo describimos, de forma muy general, las características del programa OPNET *Modeler* y el flujo de trabajo con esta herramienta. A continuación se detalla el modelo de InfiniBand desarrollado. Nos centraremos en primer lugar en el modelado de los elementos de los niveles físico y de enlace. Después describiremos el modelo de gestión de la subred. Finalmente, mostraremos la metodología de simulación utilizada en este trabajo.

4.1 Entorno de modelado: OPNET *Modeler*

OPNET (*Optimum Network Performance*) es un entorno de desarrollo de redes de comunicaciones y sistemas distribuidos, incluyendo redes inalámbricas. Existen diferentes paquetes del mismo producto, concebidos para solucionar necesidades muy variadas. Por ejemplo, OPNET *Guru* está orientado al análisis de rendimiento de redes. Por su parte, OPNET *Modeler* [55] (Figura 31) está orientado a la investigación y desarrollo de nuevas arquitecturas y protocolos de comunicaciones, incorporando herramientas para las fases de modelado, simulación, recogida y análisis de resultados. En esta sección nos centraremos (sin pretender ser exhaustivos) en este último paquete[1].

[1] El modelo actual de InfiniBand ha sido desarrollado sobre diversas revisiones de las versiones 8 y 9 de OPNET *Modeler*. La versión actual es la 10.0. Por otra parte, en la descripción que haremos aquí de la herramienta, no cubriremos los numerosos módulos adicionales orientados a aplicaciones específicas que ésta ofrece (*ACE*, *NetDoctor*, *Terrain Modeling*, *Wireless*,…).

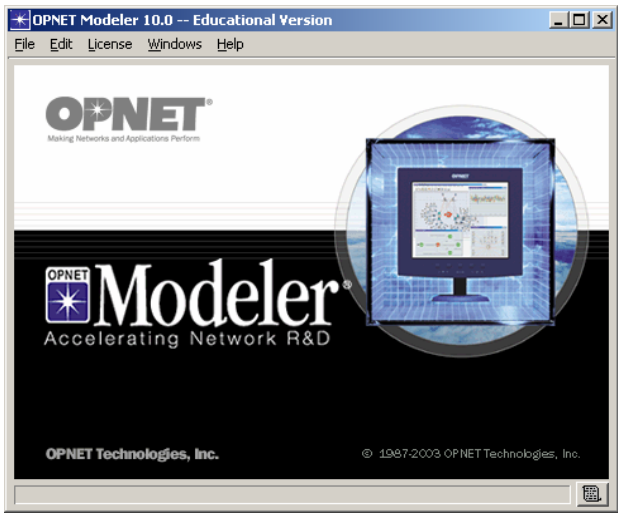


Figura 31. Ventana inicial de OPNET Modeler (versión 10.0).

Una de las principales características de OPNET es que se trata de un entorno orientado a objetos. Los modelos OPNET están basados en objetos pertenecientes a clases, con sus características y comportamiento definidos en forma de un conjunto de atributos configurables. El usuario puede crear nuevas clases durante el modelado del sistema, las cuales pueden derivarse de otras o especializarse para aportar aspectos más específicos para aplicaciones particulares.

A partir de la estructura de clases podemos descomponer el modelo jerárquicamente en los siguientes niveles: aplicaciones, subredes, los dispositivos que componen dichas subredes y la arquitectura interna de cada uno de ellos. Básicamente OPNET trabaja con tres niveles de abstracción, denominados respectivamente dominios de red, de nodo y de proceso. Cada dominio dispone de un conjunto de editores sobre los que realizar el modelo.

4.1.1 Dominio de red

Los modelos de red en OPNET están formados por nodos, enlaces y subredes. Los nodos representan dispositivos de red. Los enlaces pueden ser de tipo punto a punto o bus. Las subredes agrupan un conjunto de componentes de red. El editor de proyectos permite especificar la topología de la red y configurar determinados atributos en sus nodos y enlaces. La Figura 32 muestra una subred InfiniBand con topología irregular en el editor de proyectos.

La topología se construye a partir de los elementos de la paleta de objetos. Como ejemplo, en la Figura 32 puede verse la paleta con algunos de los objetos que componen una subred InfiniBand: un conmutador, un nodo terminal con tarjeta de interfaz (CA) y un enlace punto a punto. OPNET proporciona una amplia biblioteca de dispositivos y enlaces (la *Standard Model Library*). Todos ellos pueden ser adaptados, editando sus parámetros o cambiando la lógica de sus modelos subyacentes. Por supuesto también es posible modelar nuevos dispositivos (como ha sido nuestro caso). La Figura 33 muestra algunos de los atributos del

nodo etiquetado como *switch 7* en la topología de la Figura 32 y los atributos del enlace que conecta los nodos *switch 4* y *switch 7*. Aunque aquí no lo trataremos, además del mecanismo "manual" de construcción de la topología de la red, desde el editor de proyectos podemos importar topologías generadas con otras herramientas.

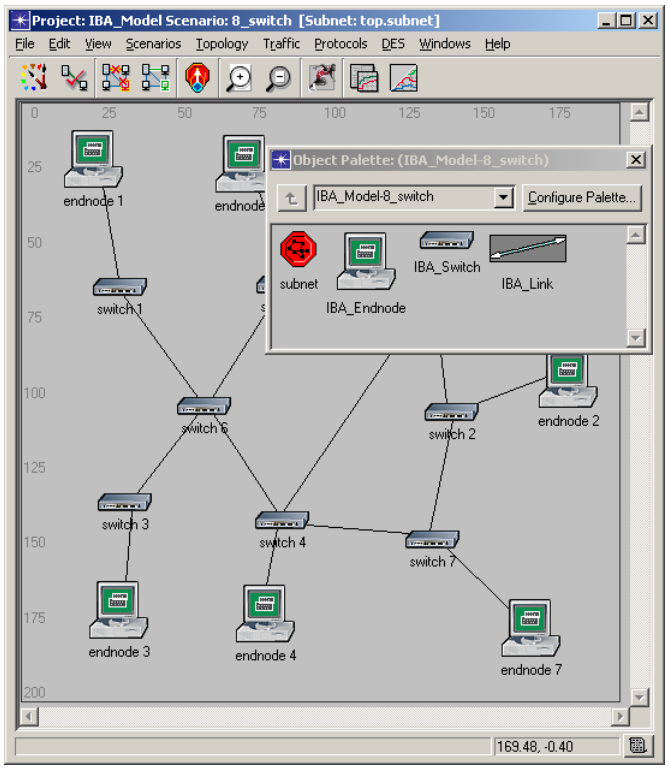


Figura 32. Editor de proyectos y paleta de objetos.

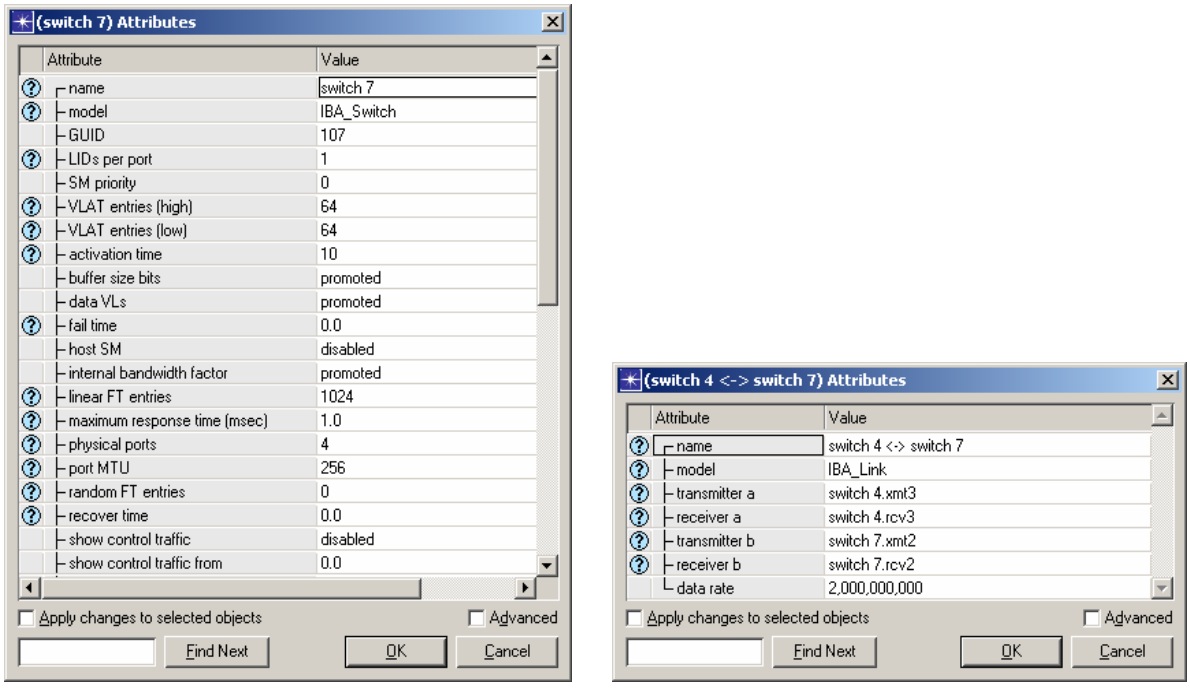


Figura 33. Atributos de uno de los nodos y de uno de los enlaces de la red.

Una vez definida la topología de la red, el siguiente paso es definir el modelo de tráfico. El tráfico puede ser importado desde un archivo o un programa externos, pero también puede ser generado durante la propia simulación. Existen dos formas de generar tráfico durante la simulación. Una posibilidad consiste en establecer el valor de determinados atributos en uno o varios objetos de la red, de forma que actúen como generadores de flujos de paquetes. Otra opción es emplear alguno de los modelos de generación de tráfico basados en aplicaciones estándar (FTP, HTTP, voz, y videoconferencia) incluidos en OPNET. En ambos casos, es posible añadir tráfico de fondo a la simulación, coexistiendo en la red con el tráfico analizado.

Haciendo uso de un compilador externo[1], el modelo completo es transformado en un motor de simulación ejecutable. Las simulaciones en OPNET están dirigidas por eventos. El núcleo de simulación maneja una lista global de eventos y un reloj compartido que marca el tiempo de simulación. Los eventos son eliminados de la lista y entregados al módulo apropiado en orden temporal. El proceso asociado al módulo destino es interrumpido, pasando a ejecutar las acciones programadas para el evento en cuestión. Estas acciones pueden implicar (directa o indirectamente) la generación de nuevos eventos. En estos casos, el núcleo de simulación recibe peticiones desde los procesos para insertar nuevos eventos en la lista global de eventos.

El editor de proyectos permite seleccionar los parámetros (o estadísticas) a evaluar, realizar un conjunto de simulaciones y visualizar los resultados obtenidos. También es posible observar gráficamente la evolución de la simulación. En concreto, OPNET puede visualizar el flujo de paquetes entre dispositivos dentro de una red o entre módulos dentro del mismo dispositivo.

Además, OPNET incorpora un potente depurador interactivo en modo línea de comando (ODB, *OPNET Debugger*), con el que analizar los errores surgidos durante la ejecución de las simulaciones. Desde el inductor del ODB podemos ejecutar acciones tan variadas como establecer un punto de ruptura en la simulación, avanzar el reloj de la simulación hasta el siguiente evento de la lista, seguir la pista de un paquete desde que es generado, o visualizar el estado de la memoria dinámica en cualquier momento de la simulación. La Figura 34 muestra un ejemplo.

Finalmente, OPNET incluye opciones de *profiling*. Esto permite determinar el tiempo real de CPU necesario para ejecutar determinadas funciones programadas en el modelo. Un ejemplo de aplicación podría ser la obtención del tiempo de ejecución de un algoritmo de cómputo de tablas de encaminamiento. El *profiling* también suele ser empleado para detectar fragmentos de código ineficiente en el modelo.

[1] Como *Microsoft Visual C++* en entornos Windows, o *gcc* en Solaris.

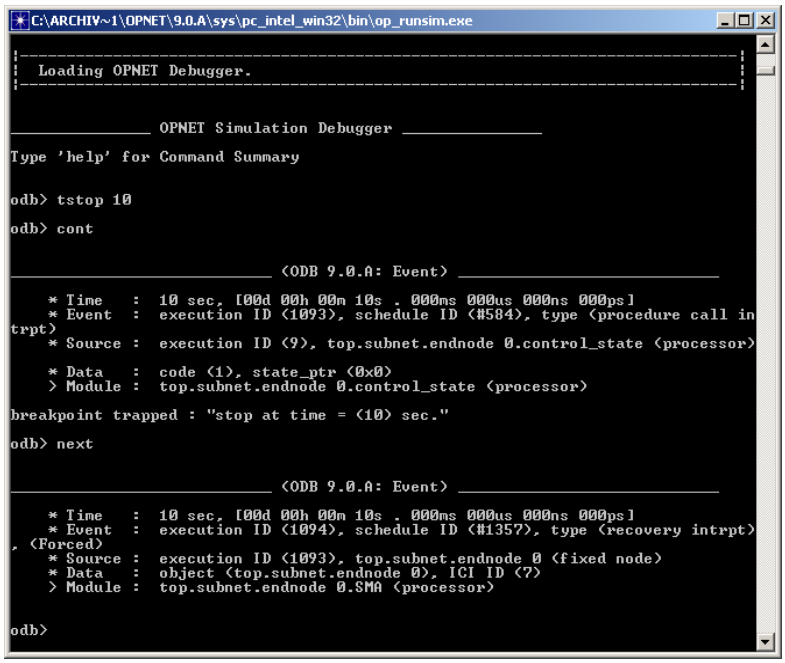


Figura 34. Depuración del modelo desde el ODB.

4.1.2 Dominio de nodo

OPNET denomina nodo a cada dispositivo que compone la topología (servidores, terminales, conmutadores,...). Un modelo de nodo se construye empleando módulos básicos de OPNET. Cada módulo puede generar, enviar o recibir paquetes hacia/desde otros módulos. Los módulos representan normalmente aplicaciones, capas de protocolo, o recursos físicos como *buffers* o puertos. En el editor de nodos se modela la forma en la que dichos módulos se interconectan. En la Figura 35 aparece el editor de nodos conteniendo un modelo de nodo terminal para InfiniBand. A continuación se describen brevemente los objetos de este editor.

- **Processor** (procesador). Es un objeto de uso general. Su comportamiento se especifica por medio de una máquina de estados (descrita posteriormente).
- **Queue** (cola). Se trata de un módulo muy similar al procesador. Incluye facilidades internas para la gestión de paquetes. También contempla la gestión de subcolas atendiendo a diversas prioridades.
- **Packet Stream** (flujo de paquetes). Conecta dos módulos (procesador o cola), permitiendo a los paquetes fluir de uno a otro.
- **Statistic Wire** (cable estadístico). Conecta dos módulos, permitiéndoles intercambiar información de control.

- **Receiver** (receptor). Permite que los paquetes accedan al nodo desde otro nodo por medio de un enlace conectado al mismo. Hay dos tipos de receptores: punto a punto y bus.

- **Transmitter** (transmisor). Permite que los paquetes sean enviados fuera del nodo a través de un enlace conectado al mismo. Al igual que antes, tenemos transmisores punto a punto y bus.

- **Logical Association** (asociación lógica). Indica que dos módulos están relacionados. Se emplea, por ejemplo, en enlaces punto a punto bidireccionales para indicar que un transmisor y un receptor están conectados al mismo enlace (formando un puerto en el nodo).

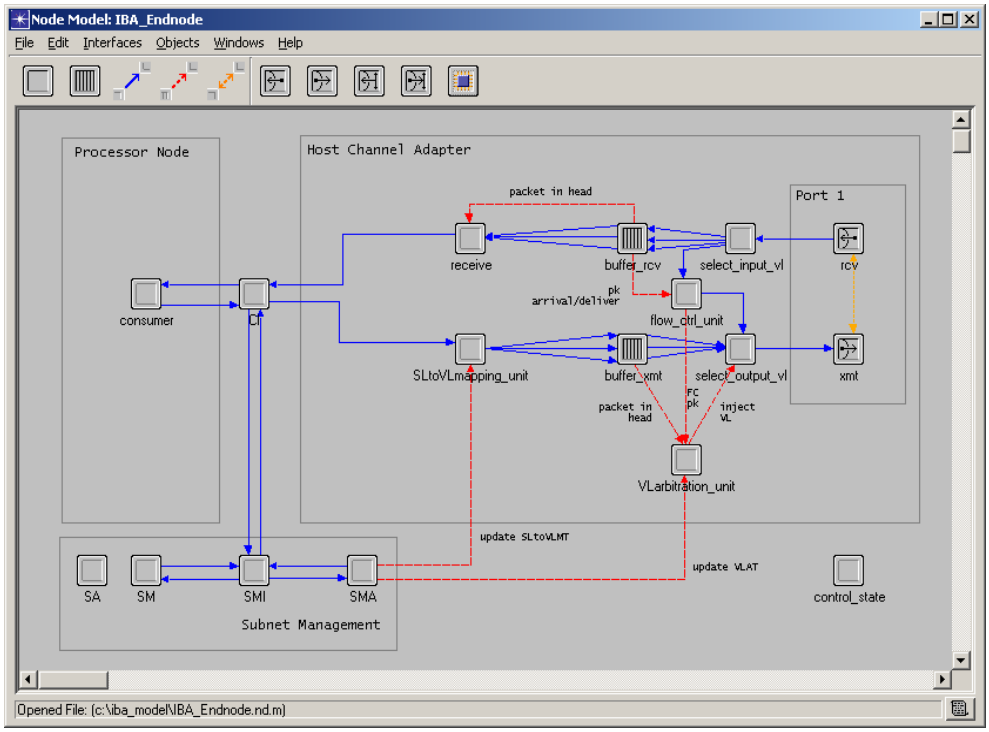


Figura 35. Editor de nodos.

4.1.3 Dominio de proceso

Dentro de un nodo, algunos módulos tienen un comportamiento predefinido. Tal es el caso de los receptores y transmisores que se conectan directamente a los enlaces. Por el contrario, los procesadores y las colas deben programarse. En OPNET, el comportamiento de un módulo se programa en forma de máquina de estados (FSM, *Finite State Machine*) desde el editor de procesos. La Figura 36 muestra la máquina de estados asociada al módulo etiquetado como *flow_ctrl_unit* en el modelo de nodo de la Figura 35. Los objetos propios del editor de procesos se describen a continuación:

- **State** (estado). Representa un posible estado de la máquina finita. Existen dos tipos de estados: no forzados y forzados. En el primer caso, la máquina se detiene al alcanzar el estado. En el segundo caso, siempre se produce una transición al siguiente estado.

- **Transition** (transición). Indica una transición de un estado inicial a otro final. Cada transición puede contener una condición con los requisitos para su ejecución.

- **Initial State** (estado inicial). Especifica cuál es el estado inicial de la máquina de estados.

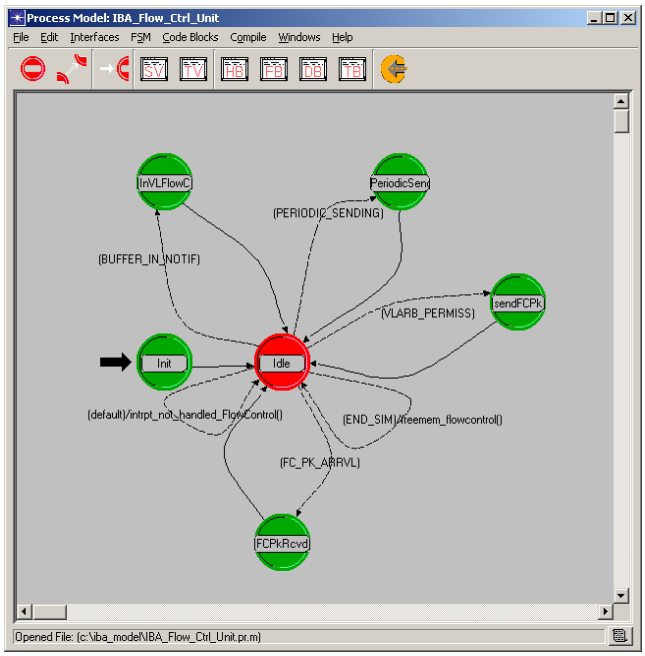


Figura 36. Editor de procesos.

Cada estado o transición puede tener asociado un bloque de código expresado en lenguaje *Proto-C*. Este lenguaje es una variación de C/C++, enriquecido con las funciones propias del núcleo de simulación de OPNET. Estas funciones (KPs, *Kernel Procedure*) están agrupadas en diversas categorías. Existen KPs para el tratamiento de eventos, la transmisión de paquetes, la recogida de estadísticas, la creación y destrucción dinámica de procesos, etc.

Además del código asociado a los elementos de la máquina de estados, existen bloques de código global utilizados para declarar tipos de datos, constantes, variables y funciones que afectan a toda la máquina de estados. Finalmente, dentro de un modelo de proceso está permitido incluir archivos externos con código auxiliar escrito en C/C++.

4.1.4 Otros editores

El trabajo en OPNET se desarrolla fundamentalmente desde los tres editores descritos. Para completar nuestra visión del flujo de trabajo con esta herramienta, a continuación se repasan otros editores de interés. La lista completa es mucho mayor.

Editor de enlaces

Como se ha mencionado, OPNET tiene dos modelos de enlace básico: punto a punto (*simplex/duplex*) y bus. Empleando el editor de enlaces, el usuario puede modelar una amplia gama de tecnologías (fibra óptica, coaxial, par trenzado,...), especificando parámetros tales como el ancho de banda, la tasa de errores, el retardo de propagación, los tipos de paquetes soportados por el enlace, etc. La Figura 37 muestra algunos de los parámetros de un enlace InfiniBand.

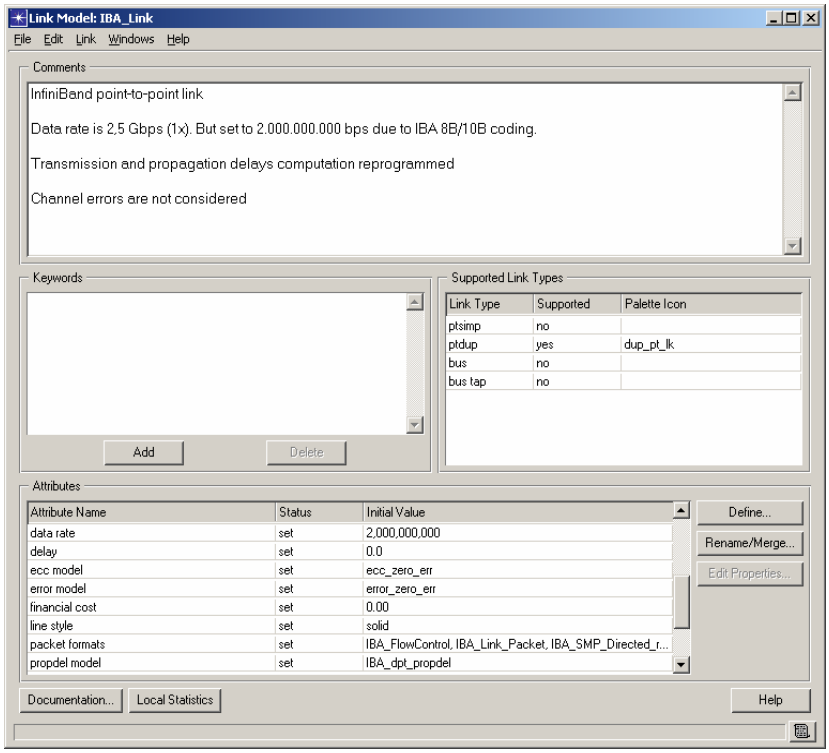


Figura 37. Editor de enlaces.

Editor de paquetes

Este editor ofrece la posibilidad de especificar gráficamente la colección de campos contenidos en cada tipo de paquete. Los atributos propios de un campo son su nombre, tipo, tamaño y valor por defecto. El tipo puede ser indefinido (datos de relleno), numérico (entero o real), estructurado (con formato definido por el usuario), o encapsulado (incluyendo otros paquetes de protocolos de nivel superior). La Figura 38 muestra el formato de un paquete de control de flujo en InfiniBand.

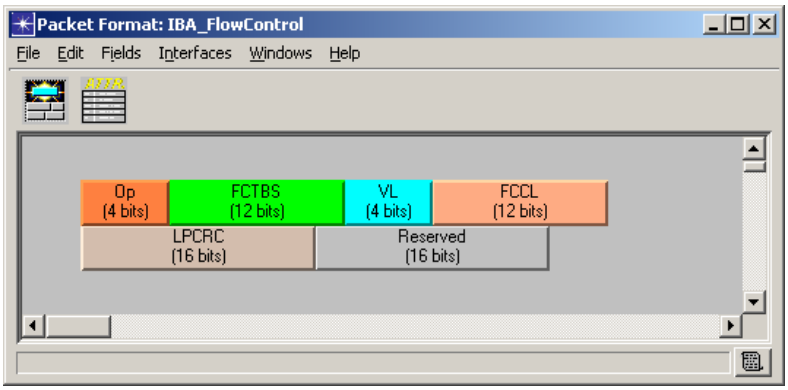


Figura 38. Editor de paquetes.

ICI editor

Este editor sirve para definir la estructura interna de un ICI (*Interface Control Information*). Los ICIs aportan información extra a un evento de simulación. Se emplean para la transferencia de información de control entre los procesos. Un proceso emisor puede programar un evento que más tarde será manipulado por otro proceso receptor. El emisor puede asociar un conjunto de datos (un ICI) al evento, de forma que el proceso receptor pueda tener acceso a estos datos cuando sea interrumpido para atender el evento. La Figura 39 muestra el ICI empleado por la unidad de arbitraje de un conmutador para configurar el *crossbar* interno.

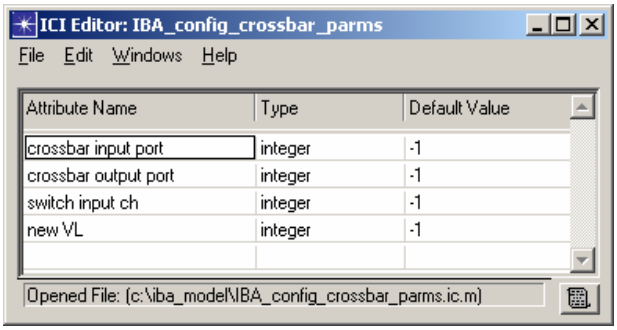


Figura 39. Editor de ICIs.

Probe editor

Sirve para especificar modelos de prueba (*probe model*). Un modelo de prueba incluye las estadísticas que serán recogidas (almacenadas en disco) durante la simulación. Existen distintos tipos de estadísticas (globales, asociadas a nodos o enlaces particulares, etc.), unas predefinidas y otras programadas por el usuario. Además, para cada estadística se puede indicar la manera en que es almacenada en disco (todos los valores, sólo uno, medias por intervalos, etc.). Figura 40 muestra un ejemplo de modelo de pruebas.

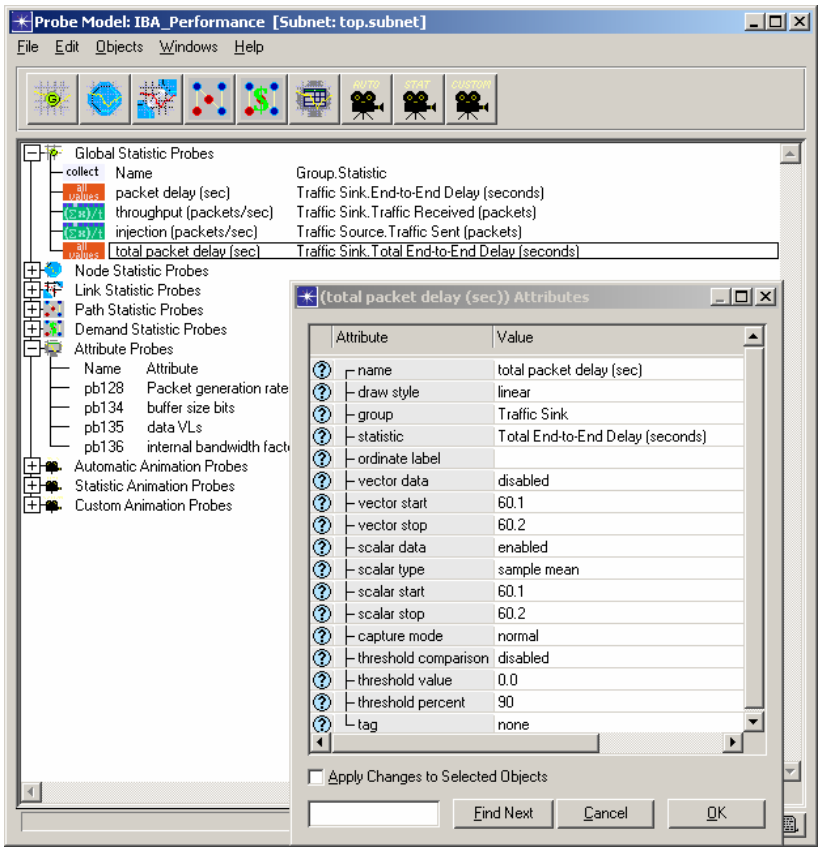


Figura 40. Editor de modelos de prueba.

Simulation tool

Permite definir secuencias de simulación (*simulation sequence*). Cada secuencia de simulación incluye uno o varios conjuntos de simulación (*simulation set*). Cada conjunto de simulación está basado en un modelo de red y un modelo de prueba concretos, y permite la variación del valor de uno o varios de los atributos de simulación que han quedado por definir en el modelo de red. La Figura 41 muestra una secuencia de simulación y algunos detalles del conjunto etiquetado como *16_switch* en la secuencia.

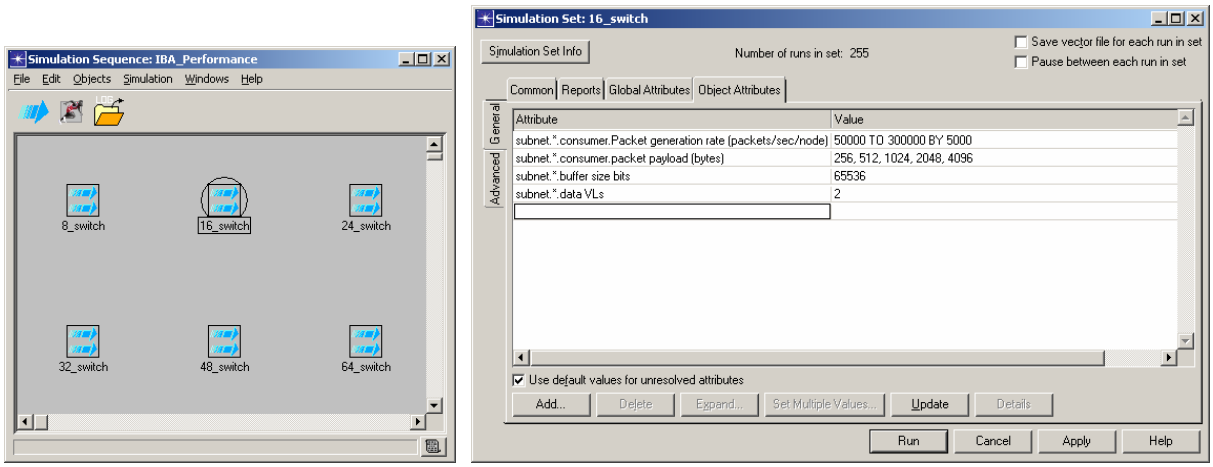


Figura 41. Editor de secuencias de simulación.

Analysis tool

Sirve para crear gráficas a partir de los ficheros de resultados obtenidos durante las simulaciones. Se trata de una de las herramientas menos desarrolladas del entorno. De hecho, la mayoría de los usuarios de OPNET exporta los resultados de simulación para representarlos gráficamente más tarde empleando herramientas como *MATLAB* o *gnuplot*. La Figura 42 muestra un ejemplo de este editor.

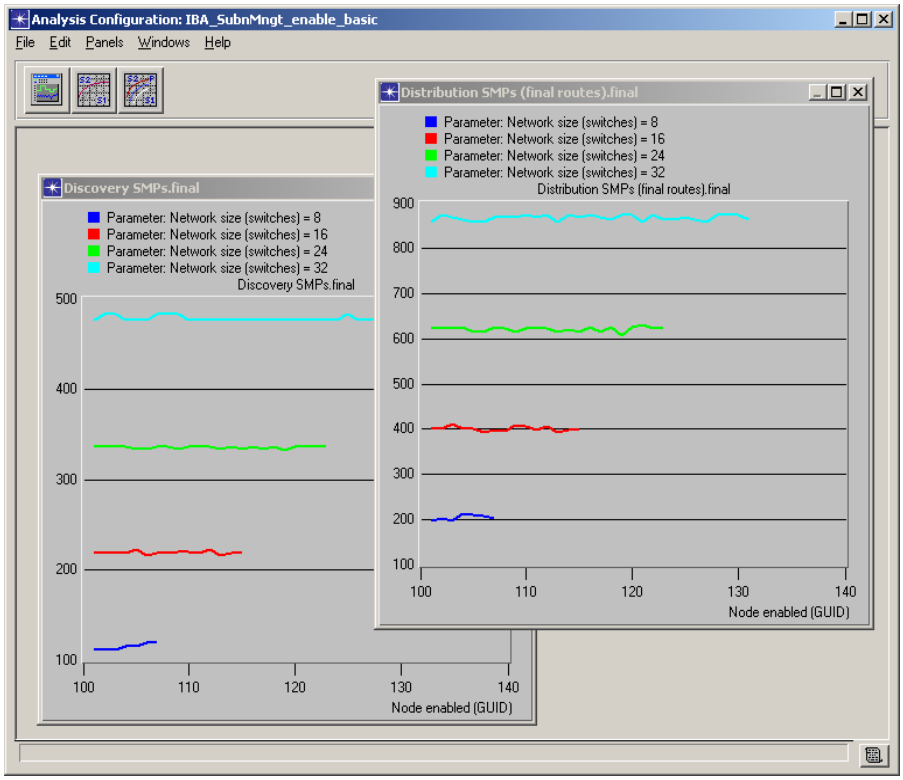


Figura 42. Herramienta de análisis.

4.1.5 Herramientas auxiliares

Aparte del entorno de modelado, dos herramientas prácticamente imprescindibles durante el proceso de modelado son un depurador de código C/C++ (no incluido en OPNET *Modeler*) y una herramienta para el análisis del uso de la memoria dinámica.

En particular, nosotros hemos empleado el entorno de *Microsoft Visual C++* [53] para la depuración del código incluido en los modelos de proceso desarrollados. Esta herramienta permite hacer una ejecución paso a paso del código, incluyendo las opciones típicas de colocación de puntos de ruptura en sentencias concretas o la inspección de variables durante la ejecución. Por otro lado, hemos usado el programa *Rational Purify* [69] para el seguimiento de la memoria dinámica durante las simulaciones. Esta herramienta nos ha ayudado a encontrar errores típicos como accesos incorrectos a variables creadas dinámicamente o detectar aquellos bloques en memoria que han quedado sin liberar.

4.2 Modelado de los niveles físico y de enlace

Una vez descrito el entorno de modelado, llega el momento de describir nuestro modelo de InfiniBand en OPNET. Este modelo ha sido presentado en [6, 11]. Además, en [30, 73, 74] pueden encontrarse descripciones más detalladas sobre aspectos concretos del mismo.

Hay que matizar aquí que la especificación de InfiniBand deja abiertas muchas cuestiones relativas a la implementación de la arquitectura. Por tanto, en ocasiones hemos tenido que tomar algunas decisiones de diseño. Algunos ejemplos son la arquitectura interna del conmutador, o la forma de implementar los buffers asociados a los canales virtuales.

Comenzamos la descripción del modelo con un repaso de los elementos correspondientes a los niveles inferiores de la arquitectura. En concreto, se han modelado los niveles inferiores de los dispositivos existentes dentro de una subred; enlaces punto a punto, conmutadores y nodos terminales con tarjeta de interfaz (CA).

4.2.1 Enlaces

Hemos modelado el enlace de cobre de InfiniBand a partir de un enlace básico punto a punto dúplex (Figura 37). En principio, el enlace tiene un ancho de banda de 2.5 Gbps, es decir, se trata de un enlace 1X. Sin embargo, con muy pocos cambios es posible modelar enlaces 4X o 12X. Además, es necesario aclarar que debido a la codificación 8B/10B [36] empleada por InfiniBand, en realidad un enlace 1X tiene un ancho de banda efectivo de 2 Gbps[1]. Por otro lado, y aunque OPNET lo permite, no hemos contemplado la ocurrencia de errores en la transmisión de paquetes a través del enlace.

En OPNET, ante la emisión de un paquete, el enlace genera un evento de recepción en el dispositivo destinatario. Por defecto, este evento se produce cuando el paquete ha sido completamente recibido, modelando una conmutación *store-and-forward* [26]. Para modelar una transmisión segmentada (tipo *virtual cut-through*) hemos reprogramado el modelo de cálculo del retardo de transmisión (*transmission delay model*), de forma que dicha notificación se produce tras la recepción de la cabecera del paquete. Por supuesto, también se ha modelado la transmisión del resto de campos del paquete. En el caso concreto de un paquete de control de flujo, la notificación se produce tras la recepción del paquete completo.

También hemos reprogramado el modelo de cálculo del retardo de propagación para el enlace (*propagation delay model*). Así, hemos considerado un retardo de propagación fijo de

[1] El motivo de esta reducción en el ancho de banda efectivo se debe a que la codificación 8B/10B expande cada carácter de 8 bits en uno de 10 bits antes de su transmisión. Por ello, la cantidad real de datos transmitidos por el enlace se degrada en un 20%.

100 ns para todos los enlaces. Dicho valor se obtiene al considerar una longitud fija de 20 metros por enlace y un tiempo de propagación de la señal en el medio de 5 ns/m [76].

4.2.2 Paquetes

Los enlaces modelados soportan tres tipos de paquetes: de datos, de control de flujo y de gestión de la subred (SMPs). Los paquetes de datos responden al formato mostrado en la Sección 2.2 (página 8). La Figura 43 muestra su estructura en campos, tal y como ha sido definida en el editor de paquetes de OPNET.

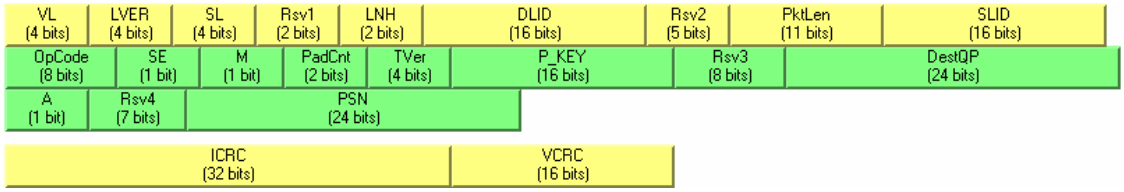


Figura 43. Definición en OPNET del paquete de datos para encaminamiento local.

En la figura puede observarse en primer lugar la cabecera LRH. Debido a que el paquete sólo será encaminado dentro de los límites de la subred, no se incluye una cabecera GRH. El LRH va seguido de la cabecera base del nivel de transporte (BTH). Finalmente, aparecen los CRCs. Nótese que el *payload* del paquete no aparece aquí de manera explícita. Este campo es creado dinámicamente por programa, tras la generación de un nuevo paquete, con un tamaño que es un atributo de la simulación.

La Figura 38 muestra un paquete de control de flujo en el editor de paquetes. El significado de cada uno de sus campos fue detallado en la Sección 2.4.4. Por último, la Figura 44 muestra el modelado de los dos tipos de SMPs descritos en la Sección 3.1.3. En ambos casos, tras las cabeceras del datagrama (LRH, BTH y DETH), aparecen los 24 bytes de cabecera del MAD, seguidos de los 232 bytes de datos del MAD. Finalmente, los CRCs.

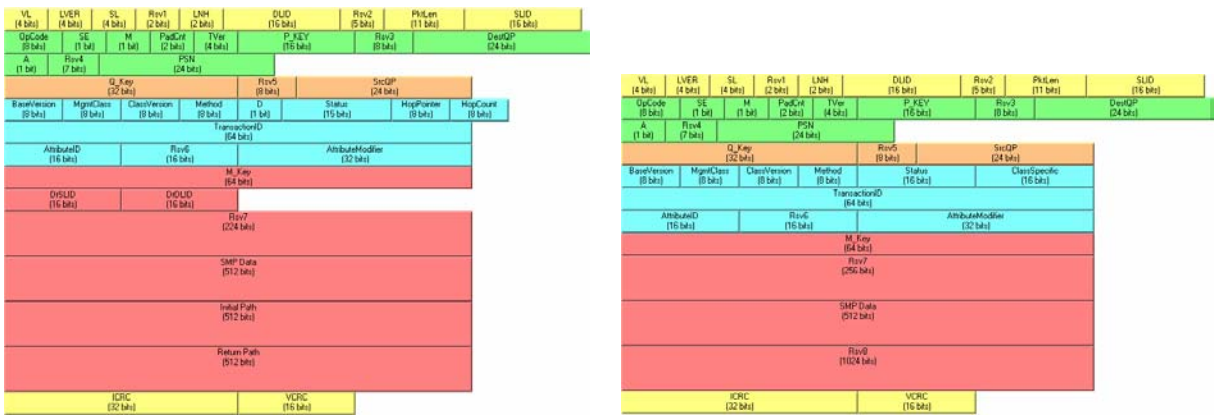


Figura 44. Definición en OPNET de los SMPs con encaminamiento dirigido y en base a destino.

4.2.3 Conmutadores

La Figura 45 muestra el modelo en OPNET para un conmutador InfiniBand de 4 puertos. La arquitectura corresponde a un conmutador *virtual cut-through* con un *crossbar* demultiplexado [25], es decir, con un puerto separado para cada canal virtual. Como alternativa, podría haberse empleado un *crossbar* multiplexado, más simple (requiere menos puertos, lo cual reduce el área de silicio), pero con la desventaja de la posible contención en las entradas.

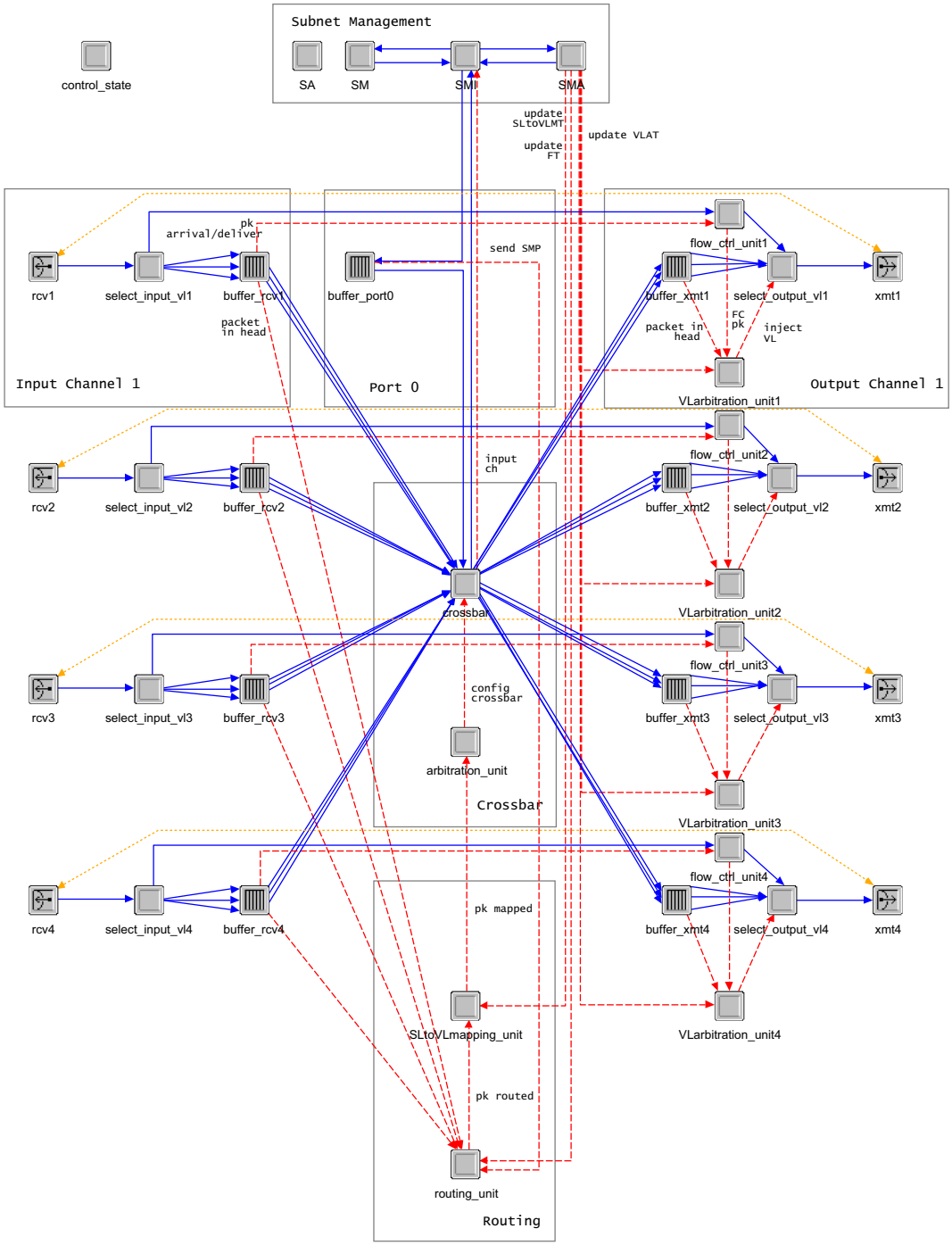


Figura 45. Modelo de conmutador.

Podemos agrupar los módulos que lo componen en canales de entrada, lógica de encaminamiento, lógica de conmutación, canales de salida y puerto de gestión. El puerto 0 está asociado a las entidades de gestión de la subred, cuyo modelado será descrito en la Sección 4.3. A continuación se describe la función de cada uno de los módulos restantes, y la forma en que interactúan entre sí.

Canales de entrada

En el margen izquierdo de la figura se encuentran los canales de entrada de cada uno de los 4 puertos del conmutador. En cada uno de ellos podemos observar un módulo receptor (rcv_i) conectando la lógica interna al enlace. Este receptor está unido a un demultiplexor.

El módulo demultiplexor ($select\_input\_vl_i$) analiza la cabecera de cada paquete recibido y determina si el destinatario es un canal virtual o la unidad de control de flujo del puerto ($flow\_ctrl\_unit_i$). Hemos modelado una "versión reducida" de la máquina de estados del puerto (Sección 2.4.1). En particular, un puerto puede estar en estado *Down*, *Initialize* o *Active*. En estado *Down*, el módulo $select\_input\_vl_i$ descarta todos los paquetes recibidos. En estado *Initialize* descartará únicamente paquetes de datos. En estado *Active* el módulo admite todo tipo de paquetes.

Los paquetes de datos son depositados en los buffers asociados a los canales virtuales. Aunque en la Figura 45 sólo aparecen representados tres canales virtuales, el número de canales virtuales soportados por cada conmutador y nodo terminal de la subred es un atributo del nodo, y puede llegar a ser de 16 (15 de datos y uno de gestión). Los buffers asociados a los canales virtuales a la entrada se modelan mediante una cola FIFO múltiple ($buffer\_rcv_i$) conectada al *crossbar*. El tamaño de los buffers es también un atributo configurable en cada nodo. Cuando el espacio disponible en un buffer varía (debido a la recepción de un nuevo paquete o a que un paquete acaba de abandonar el buffer), éste envía una notificación a la unidad de control de flujo. Por otra parte, cuando un paquete alcanza la cabeza de un buffer, se envía una notificación a la unidad de encaminamiento. Finalmente, si un buffer (de datos o de gestión) no dispone de suficiente espacio como para alojar completamente el paquete entrante, éste es descartado. En el caso de los buffers de datos, esta situación es poco probable, debido al efecto del control de flujo. No obstante, la especificación contempla esta posibilidad.

Lógica de encaminamiento

En la parte inferior de la Figura 45 se encuentra la lógica de encaminamiento de paquetes, formada por la unidad de encaminamiento y la unidad de mapeo.

Cuando un paquete alcanza la cabecera de un canal de entrada al *crossbar*, la unidad de encaminamiento (*routing_unit*) le atribuye un canal de salida del mismo. Esta unidad atiende las peticiones de los canales de entrada de forma secuencial y priorizada. En primer lugar, se atienden las peticiones de los canales virtuales de gestión (VL15), priorizando el

puerto interno de gestión. Para los canales virtuales de datos (VL0-VL14) se aplica una política *round-robin*. En [76] se asignan 100 ns al proceso completo de encaminamiento, mapeo y arbitraje del *crossbar*. En este trabajo se ha considerado un tiempo de 40 ns para la primera tarea.

Para realizar su función, la unidad de encaminamiento incorpora una tabla de encaminamiento, de tipo LFT o RFT (Sección 2.5.1). El número de entradas implementadas en dicha tabla es un atributo del conmutador. El mecanismo de encaminamiento en base a destino y las posibles causas de descarte de paquetes se detallan en la Sección 2.5.2.

Si el paquete emplea encaminamiento dirigido, si éste ha sido recibido a través de un puerto físico (distinto del puerto de gestión), deberá ser enviado al SMI del conmutador a través del puerto 0. Entonces, en caso de que el paquete no haya alcanzado su destino, el SMI volverá a inyectarlo en la subred (de nuevo a través del puerto 0). Los paquetes con encaminamiento dirigido recibidos a través del puerto 0 deben tener asociado un puerto de salida. El tratamiento en el SMI de un paquete con encaminamiento dirigido se detalló en la Sección 3.1.3.

Cada vez que un paquete es encaminado con éxito, el resultado es comunicado a la unidad de mapeo (*SLtoVLmapping_unit*), que debe determinar el canal virtual para el paquete, en el puerto de salida indicado por la unidad de encaminamiento. Si se trata de un paquete de gestión (SMP), el paquete es mapeado directamente a VL15. En otro caso, se realiza una consulta a la tabla SLtoVLMT (Sección 2.5.2). Hemos modelado un tiempo de 20 ns para la operación de mapeo. Finalmente, la unidad de mapeo proporciona la información generada a la unidad de arbitraje del *crossbar* (*arbitration_unit*).

Lógica de conmutación

En el centro de la Figura 45 podemos observar el *crossbar* y la unidad de arbitraje. El *crossbar* es una entidad pasiva que recibe peticiones desde la unidad de arbitraje para realizar el cruce de los paquetes desde los buffers asociados a los canales virtuales en la entrada (*buffer_rcv*$_i$) a los canales virtuales en la salida (*buffer_xmt*$_i$). Es posible que múltiples paquetes estén cruzando simultáneamente, lógicamente, siempre y cuando no compartan el mismo puerto de salida en el *crossbar*.

Cuando el paquete es enviado al SMI del conmutador a través del puerto interno, el *crossbar* debe notificar además el puerto de entrada empleado por el mismo. Esta información es empleada por el SMI para construir la ruta de retorno cuando el paquete de gestión emplea encaminamiento dirigido.

Se ha modelado un tiempo de configuración del *crossbar* de 2 ns, desde que la unidad de arbitraje ordena el cruce hasta que el paquete comienza a cruzar. Por otro lado, se ha considerado un tiempo de cruce de 4 ns [76]. Este valor representa el tiempo para inyectar un nuevo byte en un enlace 1X (a 2 Gbps). Para enlaces 4X y 12X, el tiempo de cruce debe reducirse a 1 ns o 0.33 ns, respectivamente. El tiempo que tarda en cruzar el paquete completo

depende de su tamaño y del ancho de banda dentro del conmutador. Por defecto, este parámetro coincide con el ancho del enlace, pero es fácilmente modificable mediante un atributo del conmutador.

La unidad de arbitraje (*arbitration_unit*) configura el *crossbar*, indicándole que debe establecer una conexión entre uno de sus puertos de entrada y uno de sus puertos de salida. La política aplicada atiende al siguiente esquema: 1) prioridad del canal virtual de gestión (VL15) sobre los canales de datos (VL0-VL14); 2) equidad entre los VL0-14. Hemos considerado un tiempo para la tarea de arbitraje de 40 ns.

Este módulo recibe peticiones de arbitraje desde la unidad de mapeo. Después, solicita el control sobre los recursos implicados en la conmutación, es decir, el puerto de salida del *crossbar* y el buffer asociado al canal virtual en la salida (*buffer_xmt*$_i$). En concreto, debe asegurarse de que existe espacio suficiente como para alojar el paquete completo en el canal de salida. Para ello, hace uso de la información recibida desde el buffer. En caso de que algún recurso no esté disponible, la solicitud se inserta en una cola. Posteriormente, cuando un recurso es liberado, se asigna a la siguiente petición en espera en la cola.

Canales de salida

Continuando con la descripción del modelo de conmutador de la Figura 45, en el margen derecho se encuentra la lógica de los puertos asociada a los canales de salida.

Las salidas del *crossbar* están conectadas directamente a los buffers asociados a los canales virtuales a la salida (*buffer_xmt*$_i$). Al igual que los buffers de entrada, los buffers de salida también han sido modelados mediante colas múltiples. Cuando un paquete alcanza la cabecera de un buffer, se envía una notificación a la unidad de arbitraje del canal. Por otra parte, cuando un paquete abandona completamente uno de los buffers, la unidad de arbitraje del *crossbar* es informada. Si un buffer no dispone de espacio para alojar un paquete, éste es descartado si se trata de un buffer de gestión (VL15). Para buffers de datos, esta situación no es posible, dado que la unidad de arbitraje ya ha realizado esta comprobación.

Los buffers de salida están conectados a un multiplexor (*select_output_vl*$_i$) conectado al transmisor unido al enlace (*xmt*$_i$). Este selector toma un paquete de uno de los buffers de salida o un paquete de control de flujo y lo envía al enlace físico (a través del transmisor). Debe ser configurado por la unidad de arbitraje del canal, de forma análoga a la configuración del *crossbar* por parte de la unidad de arbitraje del conmutador, si bien en este caso no es posible solapar cruces. Nótese que, en función del estado actual del puerto (*Down*, *Initialize* o *Active*) y del número de canales virtuales y la MTU operativos en ambos extremos del enlace, el módulo *select_output_vl*$_i$ puede descartar paquetes, en lugar de transmitirlos.

Cada puerto incorpora una unidad de control de flujo (*flow_ctrl_unit*$_i$). Se ha modelado el mecanismo de control de flujo basado en créditos previsto en la especificación de InfiniBand y descrito en la Sección 2.4.4. Más en detalle, la unidad de control de flujo recibe notificaciones desde los buffers de entrada (*buffer_rcv*$_i$) siempre que se produce alguna varia-

ción en el espacio disponible en éstos. Esta información es enviada periódicamente al otro extremo del enlace mediante paquetes de control de flujo. Para poder efectuar estos envíos, debe realizarse previamente una petición a la unidad de arbitraje del canal. Por último, ante la llegada de un paquete de control de flujo, la unidad de control de flujo debe informar del límite de crédito a la unidad de arbitraje del canal, dado que ésta es la encargada en última instancia de decidir si un paquete puede o no hacer uso del enlace físico.

La lógica del canal de salida se completa con la unidad de arbitraje del canal (*VLarbitration_unit*$_i$). Esta unidad recibe peticiones de servicio desde los buffers de salida y desde la unidad de control de flujo y determina cual será el siguiente paquete transmitido por el enlace. Como se detalla en la Sección 2.4.3, se transmiten prioritariamente los paquetes que viajan por VL15. Posteriormente se atienden las peticiones procedentes de la unidad de control de flujo. Finalmente se atienden los canales virtuales de datos (VL0-14), en función del contenido de la tabla de arbitraje (VLAT) y el límite de alta prioridad. El número de entradas implementadas en cada parte de esta tabla (alta y baja prioridad) es un atributo del conmutador. La duración del proceso de arbitraje ha sido fijada en 40 ns.

Puerto de gestión

En la parte superior de la Figura 45 (justo debajo de los elementos de gestión) se encuentra el puerto interno del conmutador, numerado como puerto 0. Contiene únicamente un buffer (*buffer_port0*), implementado mediante una cola FIFO, para la recepción de paquetes de gestión desde el SMI. Cada vez que un nuevo SMP llega a la cabeza del buffer, éste informa a la unidad de encaminamiento del conmutador. Como ya se ha dicho, si el SMP emplea encaminamiento dirigido, además debe notificar el puerto de salida a utilizar. En caso contrario (encaminamiento en base a destino), la unidad de encaminamiento enviará el paquete por el puerto de salida indicado en su tabla de encaminamiento.

Atributos del conmutador

La Figura 46 muestra una ventana de OPNET con la lista de atributos[1] disponibles desde el editor de proyectos para cada conmutador de la subred. Los atributos que aparecen en primer lugar en la lista han sido declarados a nivel de nodo. A continuación, aparecen otra serie de atributos correspondientes a algunos de los módulos incluidos en nuestro modelo de conmutador (en concreto, *SM*, *SMA*, *control_state* y *routing_unit*). En realidad, estos atributos han sido declarados en los modelos de proceso (máquinas de estado) asociados a estos módulos.

[1] Existen otros atributos predefinidos en OPNET para cada nodo, que hacen referencia, por ejemplo, a su ubicación física o a su operatividad.

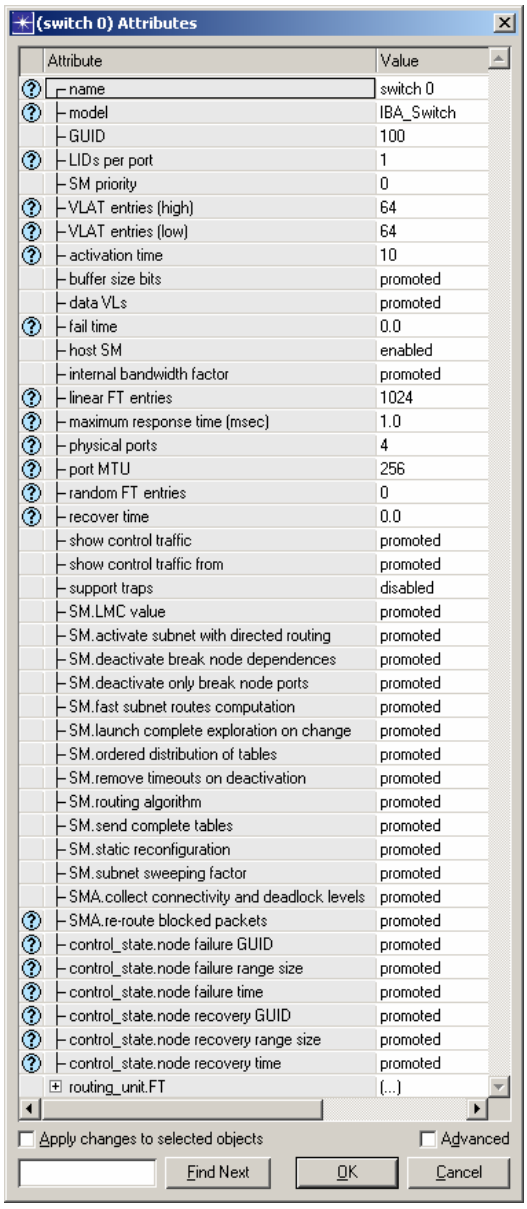


Figura 46. Atributos de un conmutador.

Algunos atributos han sido promocionados (*promoted*). En la terminología de OPNET, eso quiere decir que su valor no se establece en el nivel actual (en este caso a nivel de red). Por el contrario, su valor es fijado en un nivel superior (en este caso a la hora de configurar una o varias simulaciones para esta red). Esto permite, por ejemplo, preparar un conjunto de simulaciones con distintos valores para el atributo *data_VLs* (cuyo valor aparece promocionado en la Figura 46). Esta es también la razón por la que algunos atributos de los módulos del conmutador estén disponibles a nivel de red (fueron promocionados a nivel de nodo).

4.2.4 Nodos terminales

La Figura 47 muestra nuestro modelo de nodo terminal, compuesto por un nodo de procesamiento y una tarjeta de interfaz (HCA). Los elementos de gestión de la subred (en la parte inferior de la figura) serán descritos en la Sección 4.3.

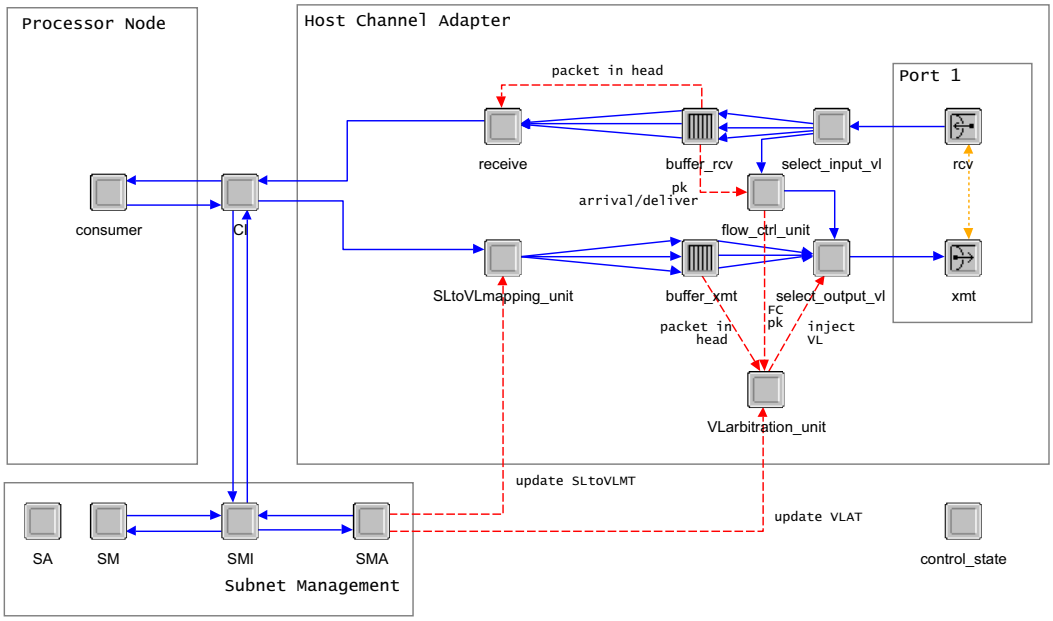


Figura 47. Modelo de nodo terminal.

Nodo de procesamiento

El modelo de nodo de procesamiento es muy simple. Únicamente contiene un módulo (*consumer*) que modela el comportamiento de una aplicación de nivel superior que accede a la tarjeta de interfaz a través del interfaz del canal (módulo *CI*). El *consumer* es el encargado de generar los paquetes de datos y también es el consumidor final del tráfico de aplicación recibido en el nodo terminal.

El modelo de generación de tráfico implementado actualmente es muy sencillo. El módulo *consumer* dispone de una serie de atributos para definir el intervalo de tiempo durante el cual genera paquetes de datos, la tasa de generación (en paquetes por segundo) y el tamaño (en términos de MTU) de los mismos. Para cada paquete, el nodo terminal de destino y el nivel de servicio aplicado se obtienen de forma aleatoria. En la parte de recepción, el *consumer* básicamente se encarga de actualizar ciertas estadísticas y destruir el paquete recibido.

El módulo *CI* modela el acceso a la tarjeta de red. En otras palabras, hace de interfaz entre los dos módulos consumidores contemplados (el *consumer* y el *SMI*) y el hardware de la red. El *CI* envía todos los paquetes que recibe desde los consumidores a la unidad de mapeo de la interfaz de red. Por otro lado, analiza los paquetes recibidos desde el módulo de recepción de la interfaz de red y los reenvía al consumidor correspondiente en función de su tipo.

Tarjeta de interfaz

Aunque InfiniBand permite CAs multipuerto, por simplicidad, la tarjeta de interfaz modelada dispone de un único puerto de comunicaciones.

Los módulos del puerto se comportan de la misma forma que los módulos equivalentes en el conmutador. De hecho, en la mayoría de los casos, el modelo de proceso es exactamente el mismo, no distinguiéndose entre si el puerto se encuentra en un conmutador o en un CA. Éste es el caso de los selectores de entrada y de salida (*select_input_vl* y *select_output_vl*) y de las unidades de control de flujo y de arbitraje del canal (*flow_ctrl_unit* y *VLarbitration_unit*). En cuanto a los buffers de entrada y de salida y a la unidad de mapeo (*buffer_rcv*, *buffer_xmt*, y *SLtoVLmapping_unit*), tienen asignados modelos de proceso distintos, pero con un comportamiento prácticamente idéntico. A continuación resaltamos algunas diferencias entre el nodo terminal y el conmutador.

Los paquetes a transmitir no deben ser encaminados, pero sí mapeados. La ausencia de *crossbar* permite que la unidad de mapeo (*SLtoVLmapping_unit*) esté directamente conectada a los buffers asociados a los canales virtuales a la salida (*buffer_xmt*). En el canal de entrada del puerto, la llegada de un paquete a la cabeza de un buffer de entrada es comunicada al módulo *receive*, que simplemente se encarga de acceder a la cola correspondiente, extraer el paquete y reenviarlo hacia el módulo *CI*.

Atributos del nodo terminal

La Figura 48 muestra una ventana de OPNET con los atributos disponibles desde el editor de proyectos para cada nodo terminal de la subred. Son prácticamente los mismos atributos que hemos visto para un conmutador. Lógicamente, hay atributos específicos de los conmutadores (*LIDs per port*, *linear FT entries*, *random FT entries*, y *support traps*) que ahora no aparecen. Por el contrario, otros atributos son específicos de nodos terminales (*LID multicast* y *port SM*). Respecto a los atributos promocionados desde módulos internos, obviamente en este caso no aparece el módulo *routing_unit*. En su lugar, aparecen los atributos del módulo *consumer*.

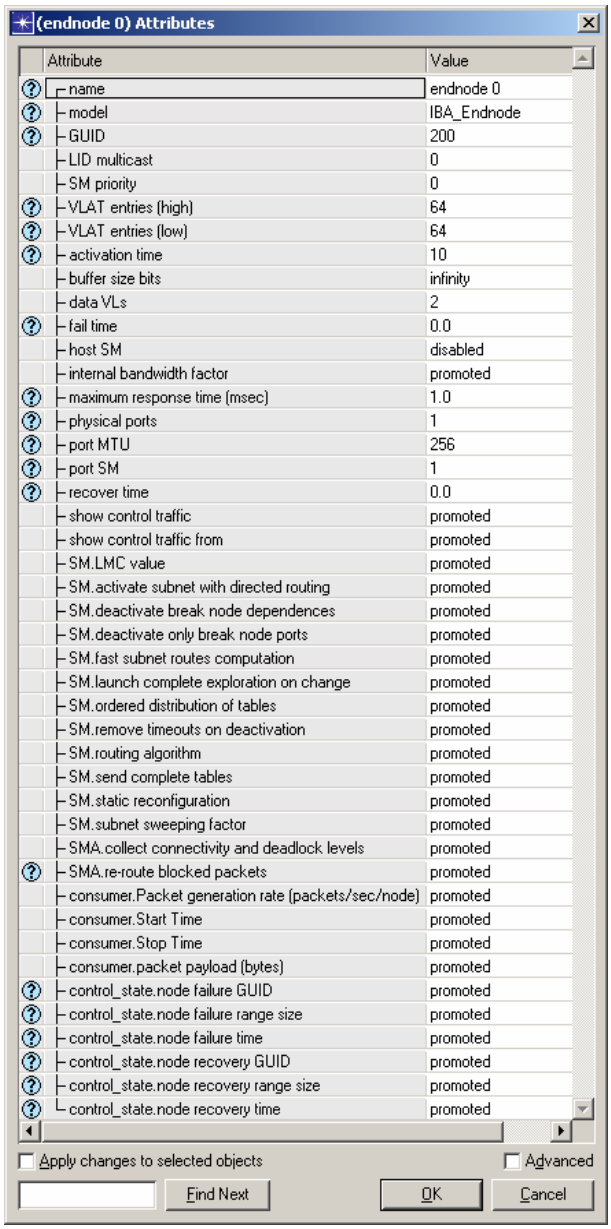


Figura 48. Atributos de un nodo terminal.

4.2.5 Validación del modelo

La validación del modelo se ha llevado a cabo, fundamentalmente, a través del análisis de los resultados de simulación obtenidos con la propia herramienta. Estos resultados nos han permitido corregir algunas anomalías en su funcionamiento. Además, en algunas ocasiones, los resultados de simulación han sido comparados cualitativamente con los publicados para otros simuladores o emuladores de InfiniBand [76, 91].

Como ejemplo del proceso de validación, la Figura 49(a) muestra una topología compuesta por 7 nodos terminales y 3 conmutadores, en la que 6 los terminales de la izquierda están programados para enviar paquetes de datos hacia el terminal de la derecha. La Figura

49(b) muestra la productividad de la subred (el tráfico recibido) a medida que crece el tráfico emitido. Puede observarse que el tráfico recibido no llega a superar los 2 Gbps, el ancho de banda efectivo de los enlaces 1X.

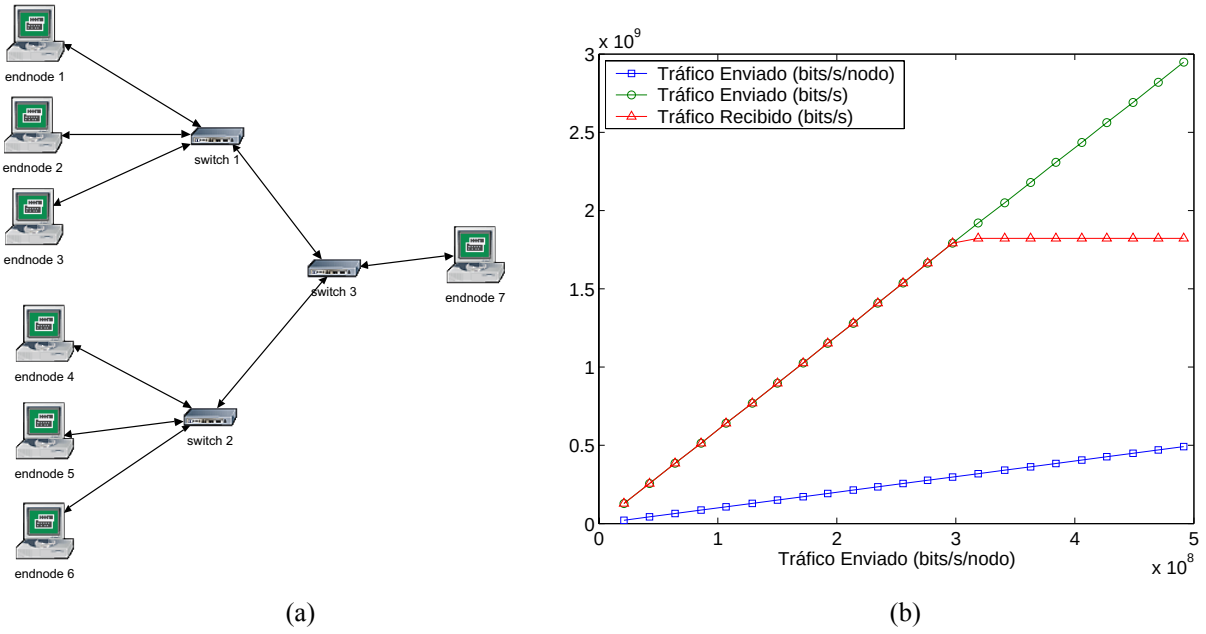


Figura 49. Ejemplo de validación del modelo.

4.3 Gestión de la subred

En relación a la infraestructura de gestión de la subred, se han modelado las entidades de gestión definidas en la especificación de InfiniBand. También se ha modelado el mecanismo de transmisión de paquetes de gestión de la subred (SMPs), incluyendo el empleo de encaminamiento dirigido y el protocolo de reconocimiento y retransmisión. A continuación se detallan todos estos aspectos.

4.3.1 Módulos de gestión en modelos de nodo

Los modelos de conmutador (Figura 45) y nodo terminal (Figura 47) incluyen tres módulos, denominados *SMI*, *SMA* y *SM*, que modelan la funcionalidad de las entidades de gestión descritas en la Sección 3.1.

Los dos primeros módulos están activos en todos los nodos terminales y conmutadores de la subred. El módulo *SM* sólo lo estará en aquellos nodos que alojan un SM. Esta circunstancia se indica a través de uno de sus atributos. Para estos nodos hay que indicar, además, la

prioridad del SM (necesaria para implementar el protocolo de contienda) y el puerto al que está asociado el SM (si se trata de un nodo terminal).

Los modelos de proceso asociados a los módulos *SM* y *SMA* (que serán descritos en secciones posteriores) son los mismos para los dos tipos de nodos modelados. En el caso del módulo *SMI*, el modelo de proceso es ligeramente distinto para el caso del conmutador. Esto se debe, fundamentalmente, a que el interfaz de gestión del conmutador debe implementar el encaminamiento dirigido de SMPs.

En cuanto a la relación con el resto de módulos del nodo, el *SM* únicamente intercambia SMPs con el *SMI*. El módulo *SMI* en conmutadores envía SMPs al buffer del puerto de gestión (*buffer_port0*) y los recibe desde el *crossbar*. En nodos terminales, el *SMI* está conectado al interfaz del canal (*CI*), con el que intercambia paquetes. Por su parte, el *SMA* distribuye a los distintos módulos del nodo local la información de configuración (tablas de encaminamiento, límites de alta prioridad, canales virtuales operativos, etc.) recibida desde el SM.

4.3.2 Interfaz de gestión (SMI) y encaminamiento dirigido

Tal y como se describe en la Sección 3.1.4, el módulo *SMI* inyecta en la red los SMPs generados por el SM y el SMA. Además, valida los SMPs recibidos en el nodo actual, los analiza y los envía a la entidad de gestión correspondiente si han llegado a su destino, o los devuelve a la red en caso de tratarse de un conmutador intermedio para un SMP con encaminamiento dirigido.

Para implementar el encaminamiento dirigido, el SMI del conmutador lleva a cabo las acciones descritas en la Sección 3.1.3. Se ha modelado tanto el encaminamiento dirigido "puro" como el encaminamiento dirigido con encaminamiento basado en destino en el segmento inicial de la ruta.

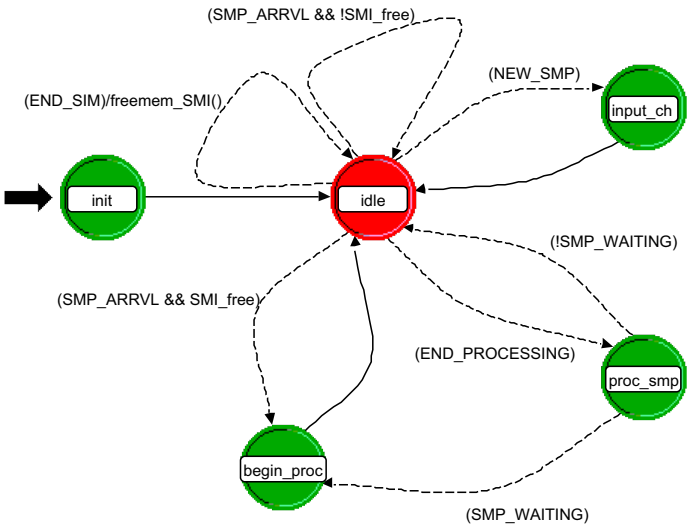


Figura 50. Modelo de proceso del interfaz de gestión.

La Figura 50 muestra la máquina de estados en el modelo de proceso asociado a este módulo. En principio, el SMI está en un estado de inactividad (*idle*) hasta que recibe un SMP. En ese momento pasa al estado *begin_proc*, donde añade un evento futuro para modelar el tiempo que dura el procesamiento del SMP y regresa al estado de reposo. Para programar este evento, emplea uno de los atributos del nodo ("*maximum response time (msec)*"). Cuando ese tiempo ha transcurrido, la máquina de estados pasa al estado *proc_smp*, donde el paquete de gestión es procesado. Los SMPs recibidos son tratados secuencialmente, es decir, un nuevo SMP no comienza a ser procesado hasta que no haya finalizado el procesamiento del SMP actual.

4.3.3 Modelado del agente de gestión de la subred

El módulo *SMA* modela el comportamiento del agente de gestión de la subred. Tal y como se describe en la Sección 3.1.2, esta entidad debe recibir, procesar y responder a los SMPs enviados por el SM.

Si es el caso, el SMA debe configurar ciertos parámetros de funcionamiento en algunos de los módulos del nodo con la información de gestión recibida desde el SM. La Tabla 8 muestra la lista completa de los elementos que son configurados por nuestro modelo de SMA, incluyendo los módulos que reciben la información. Una descripción de estos parámetros puede encontrarse en la Tabla 7 (en la página 50).

Tabla 8. Información distribuida por el modelo de SMA a los módulos locales.

Parámetro	Módulo(s) de destino
OperationalVL	select_input_vl, select_output_vl
NeighborMTU	select_input_vl, select_output_vl
PortState	select_input_vl, select_output_vl
VLHighLimit	VLarbitration_unit
VLArbitrationTable	VLarbitration_unit
DefaultPort	routing_unit
RandomForwardingTable	routing_unit
LinearForwardingTable	routing_unit
MulticastForwardingTable	routing_unit
SLtoVLMappingTable	SLtoVLmapping_unit

El resto de información de configuración recibida desde el SM es almacenada en el propio SMA, de cara a su uso posterior. En concreto, se trata de la dirección asignada a cada puerto (el par de valores *LID/LMC*), la ubicación del *master* SM (par *MasterSM-LID/MasterSMSL*) y el valor del parámetro *SubnetTimeout*.

También se ha modelado el mecanismo de envío de notificaciones tras la detección de un cambio en el estado de alguno de los puertos del nodo local (Sección 3.2.2). Esto incluye

el envío repetido de notificaciones en caso de no recibir respuesta desde el SM. Se emplea el parámetro *SubnetTimeout* para calcular el tiempo entre dos notificaciones consecutivas. Un atributo del nodo especifica si el SMA local soporta estos mecanismos.

La Figura 51 muestra la máquina de estados en el modelo de proceso asociado al módulo *SMA*. Al igual que sucede en el modelo del SMI, el SMA trata de forma secuencial los SMPs recibidos desde el interfaz de gestión. El tiempo de procesamiento de cada SMP se computa en base a los datos obtenidos al ejecutar la función de tratamiento sobre una máquina real. En el modelo de proceso también se incluyen dos estados relacionados con la detección de cambios en el estado de los nodos (*change*) y el envío repetido de notificaciones (*next_trap*).

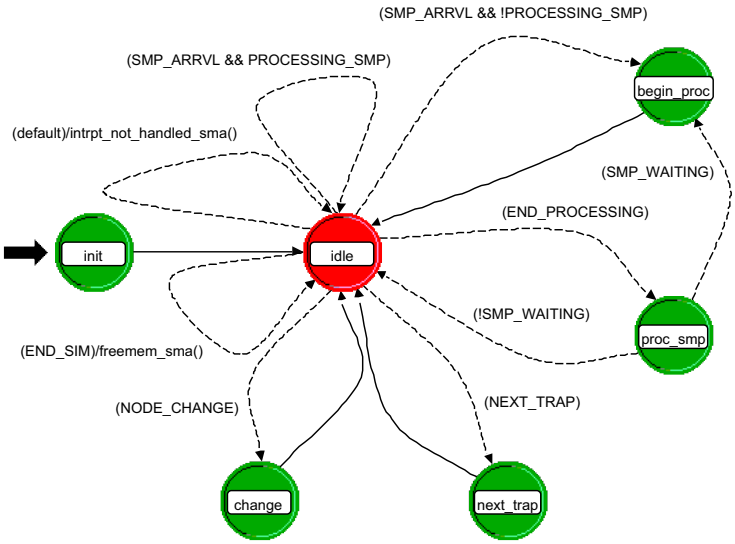


Figura 51. Modelo de proceso del agente de gestión de la subred.

Los diagramas de flujo de la Figura 52 resumen el comportamiento del SMA modelado. Se representan las tareas asociadas al procesamiento de un SMP en el SMA (izquierda) y su comportamiento tras la detección de un cambio en el estado de alguno de los puertos del nodo local (derecha).

Es importante destacar aquí que, independientemente de las características concretas del mecanismo de gestión de la subred, el comportamiento del SMA en nuestro modelo es siempre el que acabamos de detallar. Precisamente, una de las ventajas de nuestras propuestas es que no requieren modificar la funcionalidad del SMA más allá de lo recogido en la especificación de InfiniBand. Únicamente la implementación del SM debe ser modificada en cada caso. Ésta es la razón de que en este capítulo todavía no podamos describir el modelo del gestor de la subred con el nivel de detalle que hemos empleado para el agente de gestión.

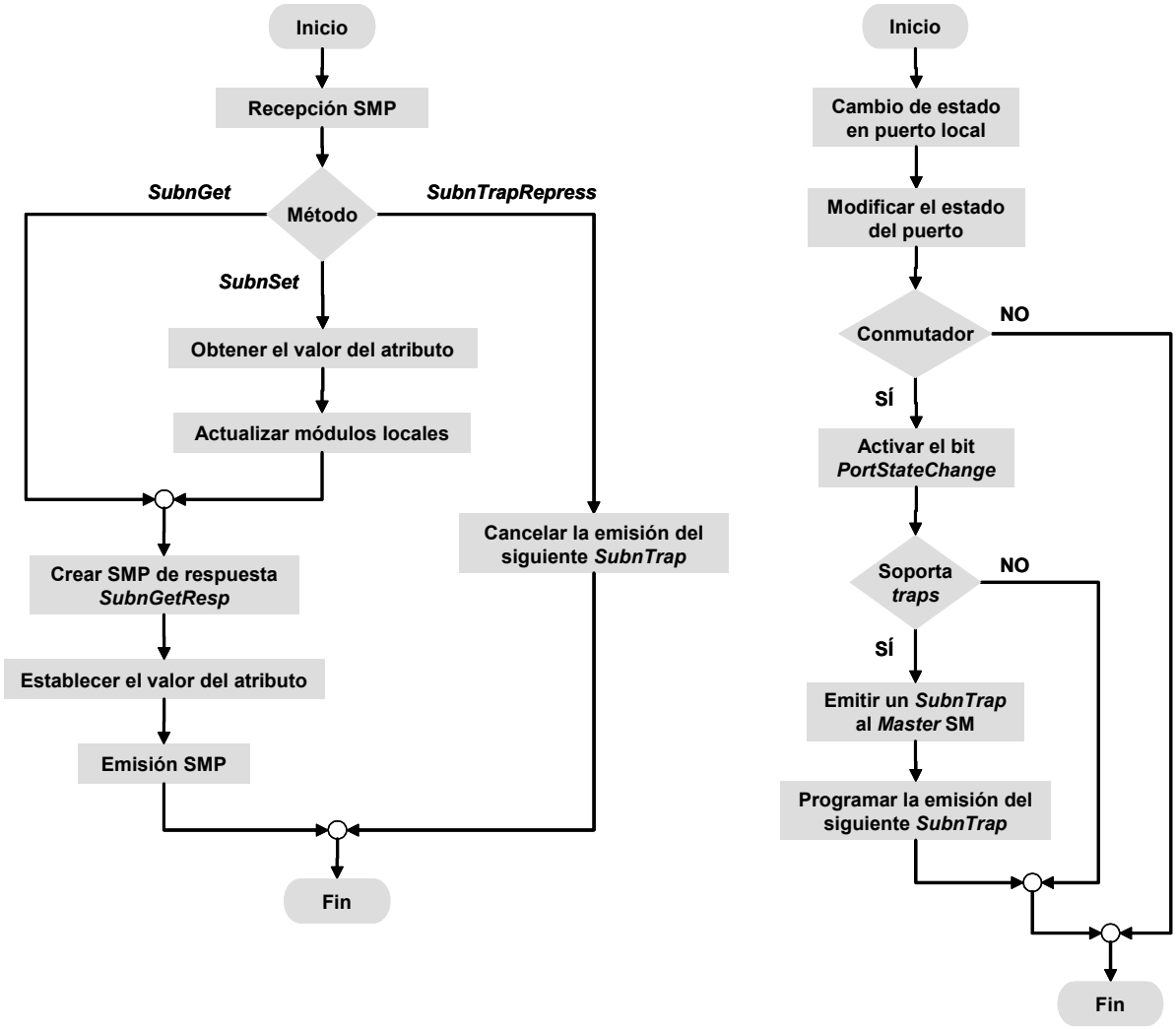


Figura 52. Comportamiento del SMA.

4.3.4 Modelado del gestor de la subred

El gestor de la subred es, con diferencia, el módulo más complejo en nuestro modelo de InfiniBand. Para tratar mejor toda esta complejidad, su funcionalidad ha sido descompuesta en cinco subprocesos, a los que nos referiremos como *dispatcher*, *handover*, *discoverer*, *builder* y *distributor*. La Figura 53 muestra esta descomposición funcional.

Esta división de las tareas de gestión es posible gracias a que OPNET permite la creación, destrucción e invocación dinámica de procesos. Cada uno de los subprocesos del gestor de la subred está definido mediante un modelo de proceso diferente. En principio, el módulo *SM* (presenten en nodos terminales y conmutadores) tiene asociado el modelo de proceso del subproceso *dispatcher*. Éste es el padre del resto de subprocesos. Se encarga de crearlos (al comienzo de la simulación) y de despertarlos (invocarlos) para que realicen sus tareas. También es posible la invocación entre los propios subprocesos.

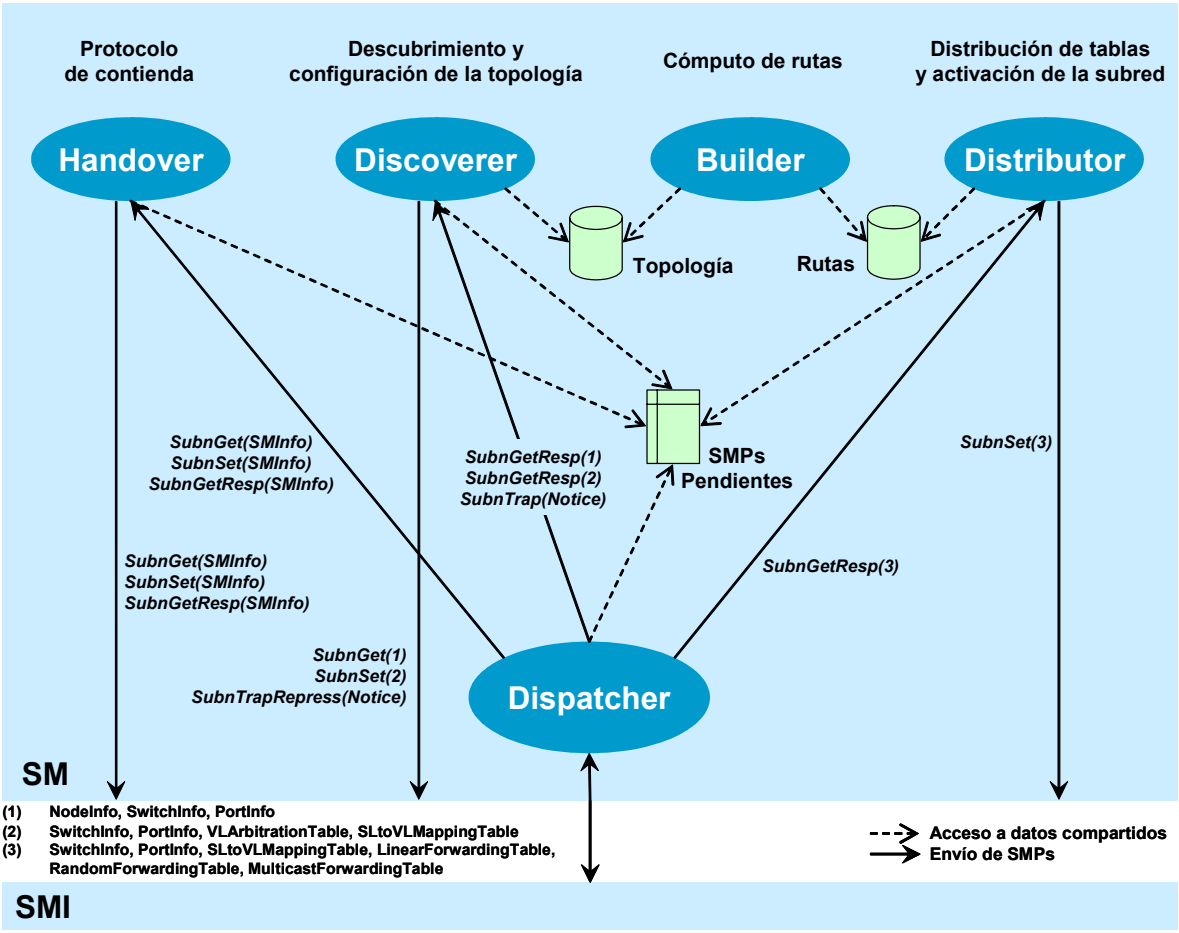


Figura 53. Diseño funcional del gestor de la subred.

A continuación describimos, a grandes rasgos, el papel que cada subproceso desempeña dentro del gestor de la subred y la relación entre ellos. En capítulos posteriores discutiremos con mayor detalle la implementación de los subprocesos *discoverer*, *builder* y *distributor*.

Dispatcher

Además de crear (y destruir) el resto de subprocesos del SM, el *dispatcher* es el receptor de los paquetes de gestión enviados desde el módulo *SMI*. En función del método y atributo de gestión del SMP y de la información en una lista de SMPs cuyas respuestas están aún pendientes, debe decidir qué subproceso del SM es el que debe procesar el paquete de gestión recibido. El subproceso en cuestión es invocado por el *dispatcher*, proporcionándole al mismo tiempo el SMP.

La Figura 54 muestra la máquina de estados en el modelo de proceso del *dispatcher*. Cuando el nodo que aloja el SM se activa, el *dispatcher* pasa al estado *sm_active* e invoca al subproceso *handover*. Cada vez que se recibe un nuevo SMP en el módulo *SM*, o queda algún SMP pendiente de ser procesado a su entrada, se produce una transición al estado *begin_proc*. Si se trata de una respuesta a un SMP previo, el *dispatcher* actualiza la lista de SMPs pendien-

tes. Después, programa un evento futuro que modela el tiempo necesario para procesar el SMP en el subproceso de destino. El tiempo de procesamiento depende del subproceso receptor del SMP, y es computado en base al tiempo obtenido al ejecutar las distintas funciones de tratamiento de SMPs en una máquina real. Transcurrido este tiempo, el paquete es entregado al subproceso correspondiente (estado *deliver_smp*)

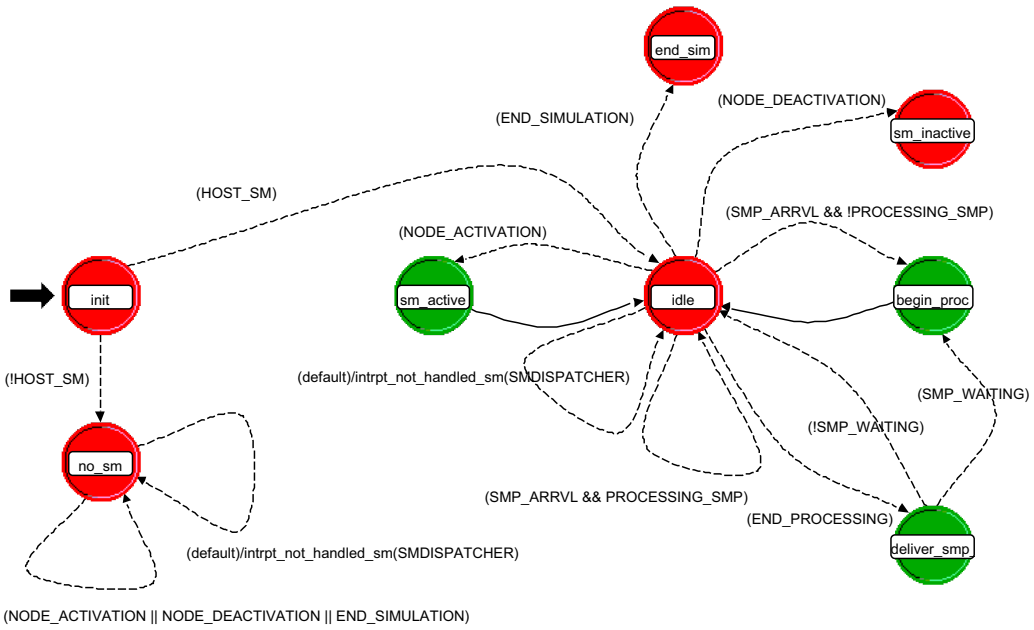


Figura 54. Modelo de proceso del *dispatcher*.

Handover

Este subproceso debe garantizar que, en caso de existir varios módulos *SM* activos en la subred, sólo uno de ellos gestionará la misma (el *master*). Para ello, debe modelar el protocolo de contienda descrito en la Sección 3.1.1. En el modelo actual, el protocolo de contienda no está implementado, permitiéndose un solo SM por subred. El *handover* actual se limita a invocar al subproceso *discoverer*, que deberá iniciar la tarea de descubrimiento de la topología de la subred.

Discoverer

Éste es el subproceso del SM encargado de detectar la ocurrencia de cambios en la subred, descubrir su topología y configurar algunos parámetros de gestión en los SMAs remotos. Para la detección de cambios, se ha modelado tanto el proceso de barrido periódico de la topología como el tratamiento de notificaciones (Sección 3.2.2). Los diagramas de flujo de la Figura 55 describen las acciones para iniciar el proceso de barrido (izquierda), procesar una respuesta a un SMP de barrido (centro) y procesar una notificación de cambio recibida desde un SMA remoto (derecha).

En el Capítulo 5 profundizaremos en distintas implementaciones para el proceso de exploración de la topología, ejecutado por el *discoverer* tras la activación inicial de la subred y ante la detección de un cambio. No obstante, mencionaremos aquí que se ha modelado el comportamiento básico, ya descrito en la Sección 3.2.1. Como detallaremos después, en nuestras implementaciones el *discoverer* descubre nuevos dispositivos de forma centralizada y según el orden de propagación de los paquetes de gestión. La diferencia principal entre nuestras propuestas está en si el *discoverer* hace o no uso de la información topológica correspondiente a la configuración anterior.

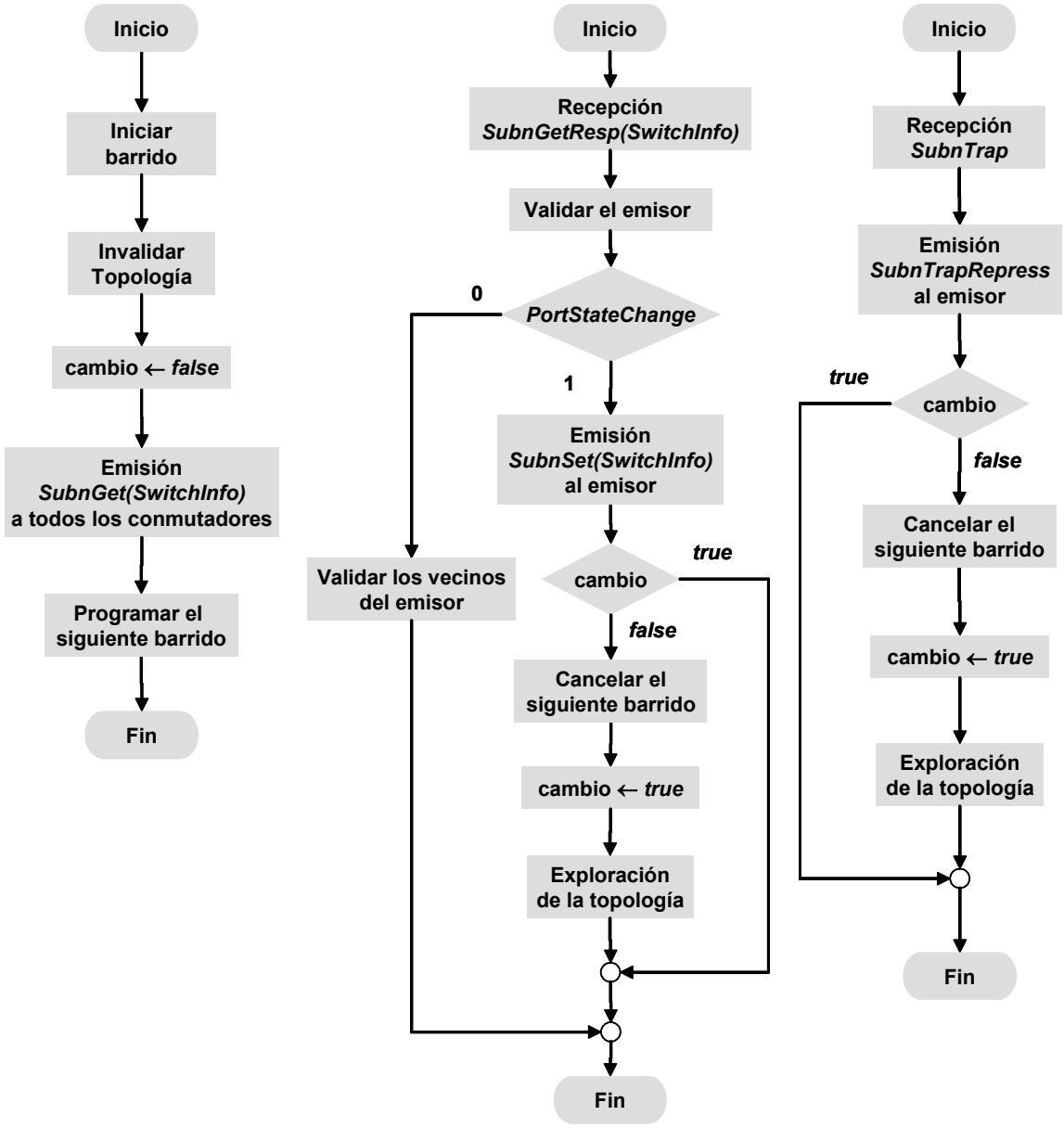


Figura 55. Tareas relacionadas con la detección de cambios.

Durante el proceso de exploración, el *discoverer* recopila gran cantidad de información. En unos casos, se trata de información topológica necesaria para construir el mapa de la subred. En otros, son datos que serán utilizados para configurar posteriormente los dispositi-

vos de la subred. La Tabla 9 muestra toda esta la información. Nótese que estos datos se corresponden con algunos de los atributos en nuestros modelos de conmutador y nodo terminal (mostrados en la Figura 46 y la Figura 48).

A partir de toda la información recopilada, y una vez descubierta la topología completa, el subproceso *discoverer* distribuye a los dispositivos de la subred los parámetros recogidos en la Tabla 7, exceptuando las tablas de encaminamiento para los conmutadores, que serán distribuidas más tarde por el subproceso *distributor*.

Tabla 9. Información recogida por el *discoverer* durante el proceso de exploración.

Atributo	Componente	Descripción
NodeInfo	NodeType	Tipo de nodo (CA, conmutador o encaminador)
NodeInfo	NumPorts	Número de puertos físicos en el nodo
NodeInfo	NodeGUID	GUID del nodo
SwitchInfo	LinearFDBCap	Número de entradas en la tabla LFT
SwitchInfo	RandomFDBCap	Número de entradas en la tabla RFT
SwitchInfo	MulticastFDBCap	Número de entradas en la tabla de encaminamiento multidestino
SwitchInfo	LIDsPerPort	Número de combinaciones LID/LMC que pueden ser asignadas a un mismo puerto
PortInfo	VLCap	VLs de datos implementados en un puerto
PortInfo	MTUCap	MTU soportada en un puerto
PortInfo	CapabilityMask	Capacidades del puerto (IsSM, IsTrapSupported)
PortInfo	VLArbitrationHighCap	Número de entradas en la parte de alta prioridad de la VLAT
PortInfo	VLArbitrationLowCap	Número de entradas en la parte de baja prioridad de la VLAT
PortInfo	RespTimeValue	Tiempo máximo entre la recepción de un SMP en el puerto y la emisión de la correspondiente respuesta

La Figura 56 muestra la máquina de estados asociada al *discoverer*. El proceso permanece en un estado de inactividad (*idle*) hasta que llega el momento de iniciar el siguiente barrido en busca de cambios. En ese momento, pasa al estado *init_sweeping*, donde realiza las acciones recogidas en la Figura 55 (izquierda). El *discoverer* es invocado por el *dispatcher* cada vez que éste recibe un nuevo SMP. Entonces, el *discoverer* pasa al estado *invocation*, donde procesa el paquete, generando, si es el caso, nuevos SMPs que son inyectados en la red a través del módulo *SMI*. Cada vez que se agota el tiempo de espera para la respuesta a un SMP emitido por el *discoverer*, se produce una transición al estado *proc_timeout*, donde se accede a la lista de SMPs pendientes. Si se ha superado el número de reinyecciones prefijado, el SMP es descartado. En caso contrario, el SMP es reinyectado nuevamente en la red. Cada vez que el *discoverer* termina de procesar un SMP, comprueba si quedan SMPs pendientes. Si no es así, y se detectó la ocurrencia de algún cambio en la topología, el *discoverer* recoge estadísticas, invoca al subproceso *builder* y regresa al estado de inactividad.

Builder

El subproceso *builder* es el encargado de obtener un conjunto de rutas para el encaminamiento dentro de la subred a partir de la información topológica recogida por el *discoverer*.

En nuestro modelo, hemos empleado *up*/down** [80] como algoritmo de encaminamiento libre de bloqueos, proponiendo dos implementaciones para el cálculo de las rutas. Como veremos en el Capítulo 6, una de ellas calcula un puerto de salida en cada conmutador para cada posible destinatario (DLID), mientras que, en la otra, el cálculo del puerto de salida en muchos casos puede obviarse, reduciendo así el tiempo global de cómputo de las rutas.

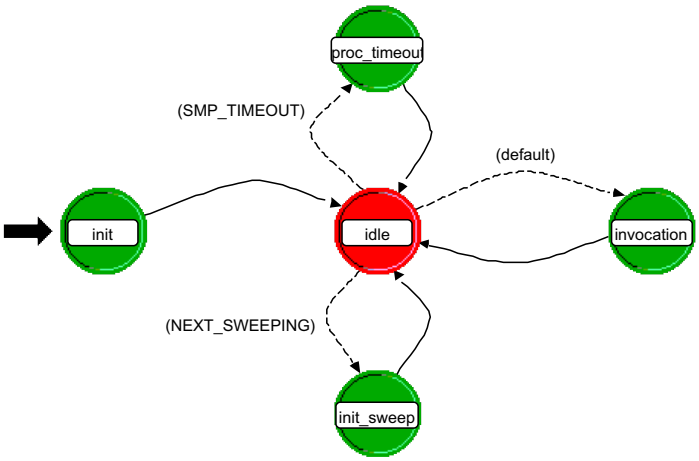


Figura 56. Modelo de proceso del *discoverer*.

La Figura 57 muestra la máquina de estados para el modelo de proceso del *builder*. En el estado *build*, el subproceso ha sido invocado y debe programar un evento futuro para modelar el tiempo de cómputo de las rutas. En el estado *end_computation* las rutas ya han sido computadas. Entonces, las estadísticas correspondientes son recogidas, el subproceso *distributor* es invocado y el subproceso *builder* regresa al estado de reposo (*idle*). El estado *send_entry* se emplea si el mecanismo de gestión requiere distribuir las entradas de las tablas de encaminamiento una a una, a medida que van siendo obtenidas. Este estado hace una invocación puntual al subproceso *distributor* para que distribuya una entrada individual. El estado *begin_final* se describe posteriormente.

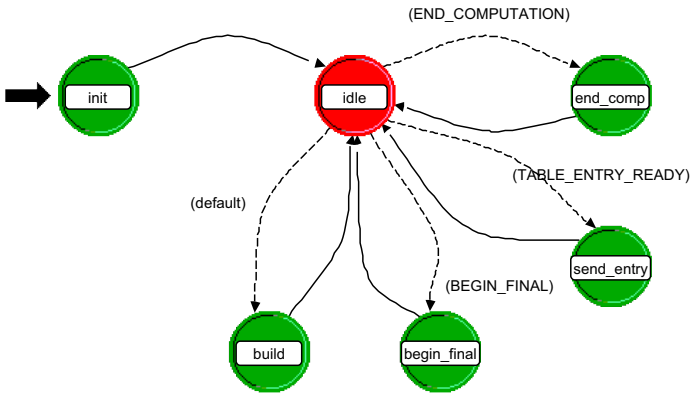


Figura 57. Modelo de proceso del *builder*.

Distributor

El *distributor* realiza la última tarea del SM tras la detección de un cambio. En concreto, distribuye las rutas obtenidas por el subproceso *builder*, en forma de tablas de encaminamiento para los conmutadores de la subred. Como se indicó en 3.2.4, la actualización de las tablas de encaminamiento debe realizarse de forma controlada, para evitar la aparición de bloqueos en la subred.

En el Capítulo 7 veremos distintas implementaciones para el mecanismo de distribución de tablas. Todas ellas se basan en impedir el empleo de cierta cantidad de rutas durante la ejecución del proceso de actualización. Obviamente, finalizada la carga de las tablas, todas las rutas vuelven a estar operativas. Algunas de las propuestas requieren configurar el estado de los puertos de la subred durante las distintas fases del proceso, mientras que otras suponen la modificación de algunas tablas de mapeo (SLtoVLMT) en los conmutadores.

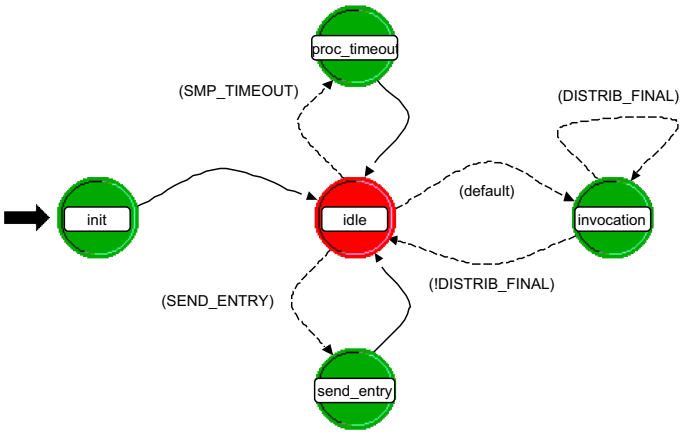


Figura 58. Modelo de proceso del *distributor*.

La Figura 58 muestra la máquina de estados del subproceso *distributor*. Como hemos dicho, el proceso es invocado por el *builder*, en unos casos para enviar una sola entrada (estado *send_entry*) y en otros para distribuir todas las tablas de encaminamiento de la subred (estado *invocation*). También es invocado por el subproceso *dispatcher* cada vez que éste recibe un nuevo SMP relativo a la tarea de envío de tablas. Cuando no quedan paquetes pendientes, el *distributor* recoge estadísticas, programa el siguiente barrido de la topología y regresa al estado de inactividad. El estado *proc_timeout* tiene exactamente la misma función que en el caso del subproceso *discoverer*.

Determinación del comportamiento del gestor

En nuestro modelo existen implementaciones alternativas para las principales tareas del gestor de la subred. Para poder seleccionar el comportamiento exacto del SM en cada simulación, el modelo de proceso del *dispatcher* (asociado al módulo *SM* en el modelo de nodo) ofrece una serie de atributos. El valor de estos atributos puede ser especificado desde el

propio editor de nodos, o promocionado para ser definido más tarde. La lista completa aparece en la Figura 46 y en la Figura 48, tras los atributos propios del modelo de nodo.

Confirmación de SMPs

El gestor de la subred debe recibir una confirmación para cada SMP (o secuencia de SMPs) emitido. Como se describe en la Sección 3.1.3, el SM debe esperar dicha confirmación durante un periodo de tiempo definido en la especificación. En nuestro modelo, para computar el tiempo de espera, hemos empleado las mismas expresiones descritas en aquel momento, basadas en los parámetros *subnettimeout* y *resptimevalue*.

Como *resptimevalue* se emplea el valor máximo del parámetro *RespTimeValue* de todos los puertos de la subred. El parámetro *RespTimeValue* se establece a nivel de nodo, a través del atributo "*maximum response time (msec)*" en el modelo de nodo.

Por otro lado, el valor de *subnettimeout* es computado en función de la distancia máxima en la subred, el tiempo necesario para transmitir cada SMP a través el enlace y el valor de *resptimevalue*. La expresión completa es:

subnettimeout = *max_distance* * (*LinkDelayPerSMP* + *max_resptimevalue*)

4.3.5 Modelado del tiempo para las tareas de gestión

En algunos de los modelos de proceso descritos ha sido necesario establecer el tiempo necesario para llevar a cabo determinadas tareas de gestión. Éste el caso del procesamiento de los paquetes de gestión en el SMA y el SM, o del cómputo de las rutas para la subred. En los sistemas reales, lo usual es que las tareas de gestión sean llevadas a cabo por aplicaciones software que corren sobre un microprocesador dedicado.

Para modelar los tiempos con mayor precisión, y asumiendo una implementación software de las entidades de gestión, se ha medido en primer lugar el tiempo necesario para llevar a cabo estas tareas en una máquina real. Para ello, se han lanzado numerosas ejecuciones del modelo, empleando un conjunto de topologías con distintos tamaños, y recogiendo los tiempos de CPU por medio de la función de *profiling* de OPNET. Obviamente, los resultados dependen de la arquitectura sobre la que se realicen estas pruebas[1].

Los valores obtenidos han sido ajustados empleado las expresiones que mejor los aproximaban en cada caso. Estas expresiones son, casi siempre, función del tamaño de la topología, y han sido finalmente incluidas en el modelo. De esta forma, los tiempos para las

[1] Nosotros hemos empleado una máquina basada en *Intel Pentium III Mobile* a 1.06 GHz, con un chipset *Intel 830 PCI* y con 512 Mbytes de SDRAM y 512 Kbytes de cache L2.

tareas de gestión se obtienen dinámicamente de acuerdo con el tamaño de la subred que estamos simulando.

Como ejemplo, la Figura 59 muestra los resultados experimentales relativos al tiempo de CPU para el procesamiento de un SMP, tanto en el SM (subproceso *discoverer*) como en el SMA. La gráfica incluye las expresiones empleadas para ajustar los puntos. Los coeficientes de determinación (R^2) indican una fuerte relación (lineal) entre el tamaño de la subred y el tiempo de procesamiento del SMP en el SM, cosa que no sucede en el caso del SMA, como era de esperar.

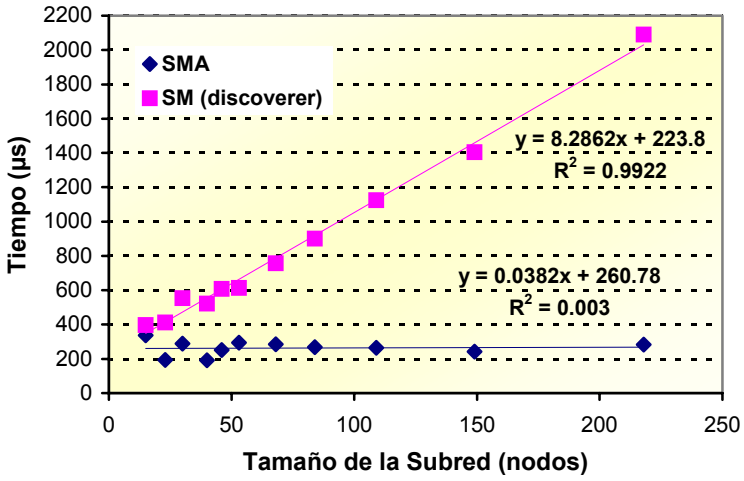


Figura 59. Tiempo consumido por el SM y el SMA para procesar un SMP, en función del tamaño de la subred.

La Figura 60 muestra el tiempo necesario para computar las tablas de encaminamiento para todos los conmutadores de la subred (subproceso *builder* del SM). De nuevo se muestra la expresión empleada para el ajuste. En este caso, la relación es de tipo potencial.

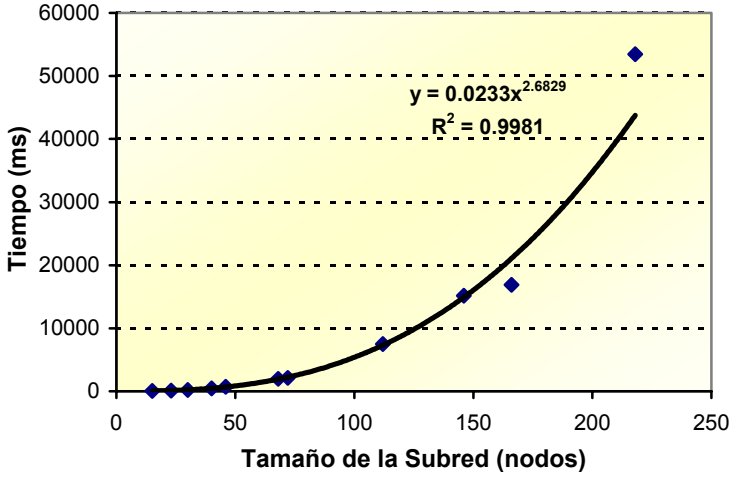


Figura 60. Tiempo consumido por el SM para computar todas las rutas de la subred, en función del tamaño de la misma.

Los resultados mostrados en las gráficas anteriores corresponden a los mecanismos básicos de descubrimiento de la topología y cómputo de tablas. En capítulos posteriores se mostrarán los resultados correspondientes a las distintas mejoras propuestas.

4.3.6 Modelado de cambios

OPNET proporciona la capacidad de activar y desactivar individualmente los nodos y enlaces de la red. Cada dispositivo tiene un atributo predefinido (*condition*) que indica si el nodo o enlace está operativo. Además, cuando el valor de este atributo cambia, el núcleo de simulación puede informar de este hecho a los módulos que lo hayan solicitado, a través de una interrupción.

Para modelar la ocurrencia de cambios durante la simulación, hemos incluido en cada modelo de nodo un módulo específico (*control_state*), un atributo para programar el instante en el que el nodo debe activarse por primera vez y dos atributos más para indicar dos instantes posteriores en los que el nodo podría fallar y eventualmente recuperarse. El modelo de proceso asociado al módulo *control_state* lee estos valores y, llegados esos momentos, se activa y modifica el valor del atributo *condition* del nodo.

La detección del cambio de estado de un nodo por parte de cada uno de sus nodos vecinos se realiza en el módulo *SMA* en cada uno de éstos. Los SMAs están programados para ser notificados por el núcleo de simulación ante cualquier cambio. Como se detalla en la Figura 52, tras la detección, el SMA debe actualizar el estado del puerto conectado al nodo que acaba de activarse o desactivarse. En el primer caso, el nuevo estado será *Initialize* y en el segundo *Down*. Si se trata de un SMA alojado en un conmutador que soporta el envío de notificaciones, el SMA emitirá la correspondiente notificación al SM. En otro caso, el SM tendrá conocimiento del cambio de estado en uno de los puertos del nodo actual durante el proceso normal de barrido.

4.3.7 Estadísticas de gestión

La Tabla 10 muestra las estadísticas empleadas para la evaluación detallada de los mecanismos de gestión de la subred propuestos en esta Tesis. Todas ellas están definidas como estadísticas globales, es decir, no están asociadas a ningún modelo de proceso en concreto.

Básicamente, las estadísticas nos ayudan a evaluar la duración de las distintas tareas de gestión, la cantidad de información de gestión necesaria (paquetes de gestión) y la forma en la que el mecanismo de gestión afecta a las prestaciones ofrecidas por la red (productividad, latencia, paquetes descartados, etc.).

Tabla 10. Estadísticas relativas a la gestión de la subred.

Nombre	Descripción
Received SMPs	SMPs recibidos por todas las entidades de gestión de la subred
Received SubnMngt bytes	Información de gestión recibida en las entidades de gestión
Average Detection Time	Tiempo medio de detección de cambios
Sweeping SMPs Sent	Número total de SMPs enviados por el proceso de barrido
Management Time (from change to provisional routes)	Tiempo de asimilación del cambio (desde su ocurrencia hasta la distribución de rutas provisionales)
Management Time (from detection to provisional routes)	Tiempo de asimilación del cambio (desde su detección hasta la distribución de rutas provisionales)
Management Time (from change to final routes)	Tiempo de asimilación del cambio (desde su ocurrencia hasta la distribución de rutas definitivas)
Management Time (from detection to final routes)	Tiempo de asimilación del cambio (desde su detección hasta la distribución de rutas definitivas)
Change Detection Time	Tiempo desde la ocurrencia del cambio hasta su detección
Topology Discovery Time	Tiempo de descubrimiento de la topología de la subred
Provisional Tables Computation Time	Tiempo de cómputo de rutas provisionales
Final Tables Computation Time	Tiempo de cómputo de rutas definitivas
Tables Change Percentage	Porcentaje de entradas en tablas de encaminamiento que cambian con respecto a la configuración anterior
Tables Distribution Time (provisional routes)	Tiempo de distribución de tablas provisionales
Tables Distribution Time (final routes)	Tiempo de distribución de tablas definitivas
Discovery SMPs	SMPs de descubrimiento de la topología
Distribution SMPs (provisional routes)	SMPs de envío de tablas provisionales
Distribution SMPs (final routes)	SMPs de envío de tablas definitivas
Subnet Deactivation SMPs	SMPs para desactivar la subred
Tables Sending SMPs	SMPs para enviar las tablas de encaminamiento
Tables Updating Percentage	Porcentaje de entradas en tablas de encaminamiento distribuidas a los conmutadores de la subred
Subnet Activation SMPs	SMPs para activar la subred
Discarded Packets	Número total de paquetes descartados durante la simulación
Discarded Data Packets (inactive ports)	Paquetes de datos descartados en puertos en estado *Down*
Discarded Data Packets (deactivated ports)	Paquetes de datos descartados en puertos en estado *Initialize*
Discarded Data Packets (output buffer overflow)	Paquetes de datos descartados a causa del desbordamiento de los buffers de salida
Discarded Data Packets (Pout=Pin)	Paquetes de datos descartados debido a la coincidencia del puerto de salida con el puerto de entrada
Discarded Data Packets (unroutable packets)	Paquetes de datos descartados al no encontrarse un puerto de salida en la tabla de encaminamiento
Discarded Data Packets (clearing buffers)	Paquetes de datos descartados al vaciar los buffers de un nodo desconectado
Discarded Data Packets (unmappeable packets)	Paquetes de datos descartados al no poder ser mapeados
Connectivity Level	Nivel de conectividad entre los nodos terminales de la subred, según la configuración actual de los conmutadores
Deadlock Level	Porcentaje de los enlaces de la subred actualmente involucrados en bloqueos potenciales
Traffic Sent (packets)	Tráfico emitido durante la simulación
Traffic Received (packets)	Tráfico recibido durante la simulación
End-to-End Delay (seconds)	Latencia de cabecera (desde generación)
Total End-to-End Delay (seconds)	Latencia del paquete completo (desde generación)

En el modelo de prueba correspondiente, para algunas de estas estadísticas sólo se recoge un valor por simulación, casi siempre su última variación o su valor medio. En otros casos, nos interesa examinar la evolución temporal de la estadística, forzando la recogida de cualquier variación en su valor a lo largo de un intervalo de tiempo dentro de la simulación.

4.4 Metodología de simulación

El modelo de InfiniBand descrito en este capítulo es una herramienta que puede emplearse para analizar diversos aspectos clave de esta arquitectura. En este trabajo, nosotros lo hemos utilizado para evaluar las prestaciones de los mecanismos de gestión de la subred propuestos. En esta sección describimos la metodología general seguida a la hora de preparar las simulaciones.

4.4.1 Topologías

Empleando los modelos de enlace, conmutador y nodo terminal descritos, es posible construir y evaluar cualquier configuración topológica para la subred. Para las simulaciones presentadas en esta memoria hemos empleado un conjunto de topologías irregulares con tamaños de 8, 16, 24, 32, 48 y 64 conmutadores. En ellas, cada conmutador está conectado a un nodo terminal (siempre que disponga de algún puerto libre). Además, no todos los puertos de los conmutadores están necesariamente conectados a otros dispositivos. La Figura 61 muestra algunos ejemplos.

Además de estas topologías arbitrarias, también hemos analizado algunas configuraciones más regulares, como es el caso de las redes Clos [21], ampliamente empleadas en entornos SAN y para comunicación entre procesos (IPC) [81]. La Figura 62 muestra un ejemplo. Estas redes tienen excelentes propiedades que las hacen ideales para la construcción de clusters (bisección completa, escalabilidad, modularidad, redundancia a través de múltiples caminos, facilidades para comunicación colectiva, etc.).

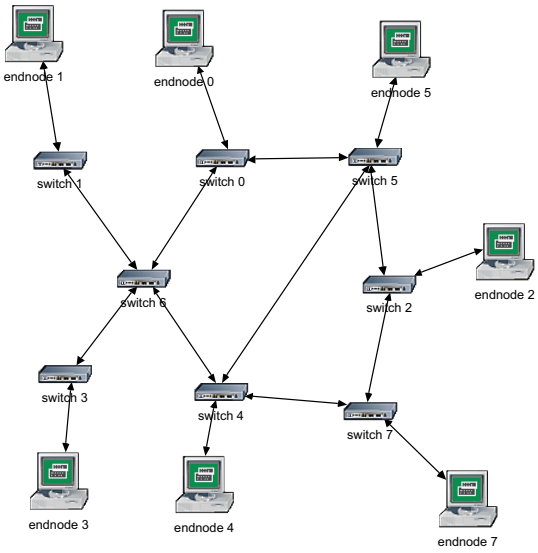
En todos los casos, hemos considerado que el SM está alojado en el conmutador etiquetado como *switch 0*.

4.4.2 Dispositivos de la subred

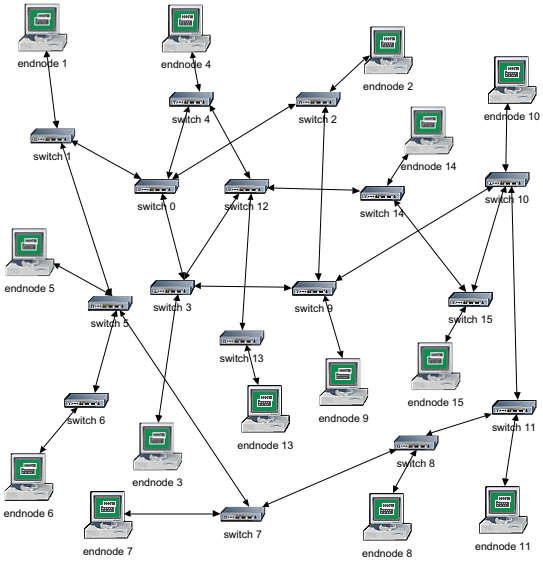
Si no se especifica otra cosa, el ancho de banda de los enlaces es 1X. En los puertos de HCAs y conmutadores, el número de canales virtuales de datos es 2 (VL0, VL1), y las tablas de arbitraje (VLAT) de alta y baja prioridad tienen un tamaño de 64 entradas.

En los conmutadores, el ancho de banda interno coincide con el ancho de banda del enlace, el tamaño de los buffers es de 32768 bits (4 Kbytes), y las tablas de encaminamiento lineales (LFTs) tienen capacidad para almacenar 1024 entradas

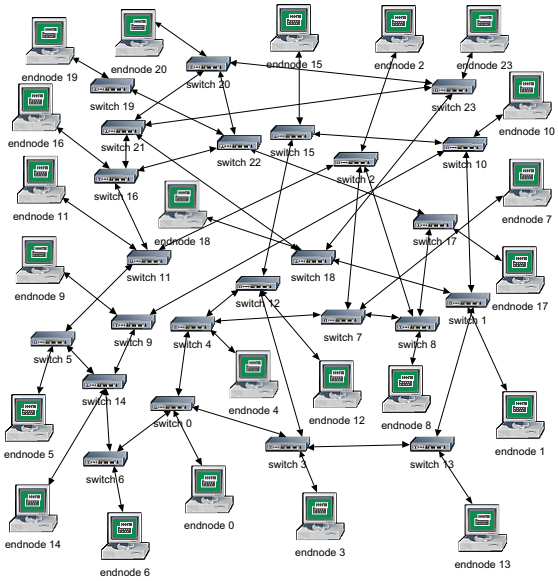
Las estrategias utilizadas para el rellenado de las tablas VLAT y SLtoVLMT no son relevantes para esta Tesis. En todos los casos, los enlaces físicos son asignados a los distintos canales virtuales de datos en la salida siguiendo un esquema *round-robin*. Por otro lado, para llevar a cabo el proceso de mapeo de SL a VL, se ha considerado siempre una asignación cíclica de los canales virtuales disponibles. La implementación de estrategias diferentes supone muy pocas modificaciones en el modelo.



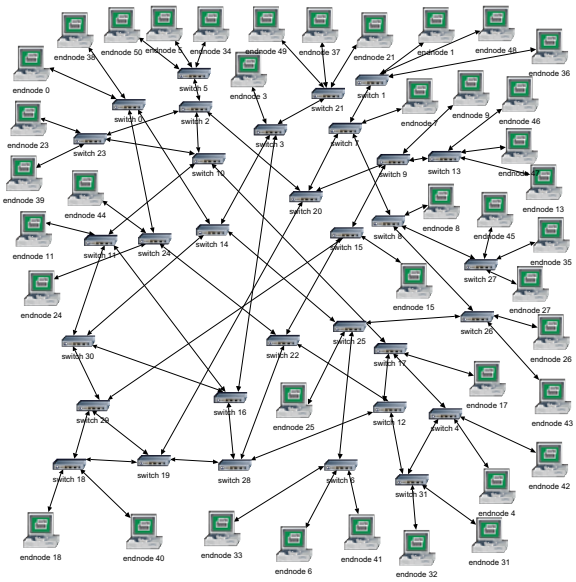
(a) 8 conmutadores



(b) 16 conmutadores



(c) 24 conmutadores



(d) 32 conmutadores

Figura 61. Cuatro topologías irregulares.

4.4.3 Fuentes de tráfico

El modelo de tráfico empleado se ha descrito en la Sección 4.2.4. La tasa de generación de paquetes es un parámetro de cada simulación cuando evaluamos las prestaciones de una topología concreta. Por otro lado, para la evaluación de los mecanismos de gestión se han empleado tasas de generación equivalentes al 25% y al 50% de la tasa de saturación de la topología (si no se indica otra cosa, se trata del 25%). Hemos escogido cargas bajas con el objetivo de impedir la saturación la red durante el proceso de asimilación del cambio. Por defecto, la MTU de cada paquete se establece en 256 bytes (la mínima MTU permitida en InfiniBand).

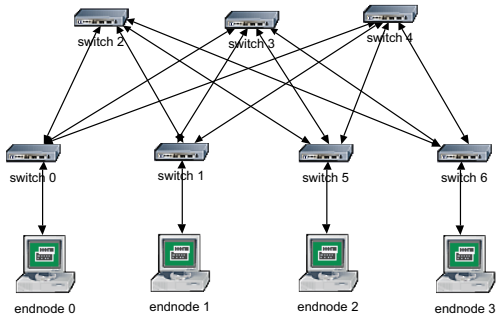


Figura 62. Red clos (3, 1, 4).

4.4.4 Eventos durante la simulación

En todos los casos, hemos programado la activación de la subred en tiempo 10 s. Los consumidores comienzan a generar paquetes de datos en tiempo 60 s. Para evaluar prestaciones consideramos un periodo transitorio de 0.1 s y, tras dicho periodo, un periodo útil de 0.1 s durante el cual se recogen las estadísticas. De esta forma, la simulación tiene una duración total de 60.2 s.

Por otro lado, para evaluar los mecanismos de gestión, simulamos un cambio topológico en tiempo 60.1 s. Dicho cambio consiste en la activación o desactivación de uno o varios de los conmutadores de la subred. La simulación finaliza una vez que el mecanismo de gestión bajo estudio ha asimilado el cambio completamente.

Capítulo 5
Detección de cambios y descubrimiento de la topología

5.1 Introducción

Tras la detección de un cambio, el gestor de la subred debe actualizar su base de datos topológica. Esta base de datos incluye información acerca de los dispositivos activos en la subred y la forma en la que están conectados entre sí. Esta información es fundamental, entre otras cosas, para la obtención de las rutas que emplearán los paquetes dentro de la subred.

En este capítulo, en primer lugar, analizaremos las prestaciones de los dos mecanismos de detección de cambios recogidos en la especificación de InfiniBand. Después, presentaremos una implementación básica para el proceso de descubrimiento de la topología, en la que el gestor de la subred descarta toda la información relativa a la configuración previa y obtiene la nueva configuración partiendo de cero. A continuación, propondremos una implementación mejorada para el proceso de descubrimiento. La mejora consistirá en explorar únicamente la región que ha sido afectada por el cambio, reduciendo así el número de paquetes de gestión y por tanto el tiempo necesario para asimilar la nueva topología. Finalmente, mostraremos una evaluación comparativa de ambos mecanismos de exploración.

La especificación de InfiniBand contempla la posibilidad de que las tareas propias del SM sean llevadas a cabo por varios nodos a lo largo de la subred. Sin embargo, por simplicidad, las implementaciones que nosotros propondremos en esta Tesis para el proceso de descubrimiento de la topología (y para el resto de tareas de gestión de la subred) se basan en un SM centralizado en uno de los nodos de la subred.

5.2 Elección y ajuste del mecanismo de detección de cambios

Cualquier cambio en la topología de la subred se traduce en un cambio en el estado de al menos uno de sus puertos. El gestor de la subred debe ser capaz de detectar dicho cambio de estado. Obviamente, es preferible que la detección sea rápida. Los mecanismos de detección contemplados en la especificación de InfiniBand, barrido periódico y notificaciones, han sido descritos en la Sección 3.2.2. Por otra parte, en la Sección 4.3.4 se han detallado las tareas relacionadas con la detección de cambios en nuestro modelo de gestor de la subred (subproceso *discoverer*).

La implementación del mecanismo opcional de notificaciones está totalmente detallada en la especificación. En cuanto al mecanismo obligatorio de barrido, una de las pocas cosas que quedan por definir es la frecuencia con la que deben realizarse los sondeos. Como es lógico, cuanto mayor sea esta frecuencia, menor será el tiempo de detección del cambio. Sin embargo, sería necesario evaluar en qué medida el aumento de la frecuencia de barrido afecta a las prestaciones ofrecidas por la subred.

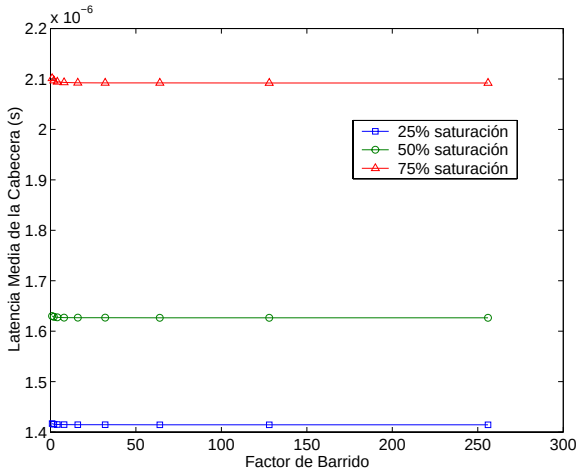
Para comenzar, la Tabla 11 recoge el tiempo que dura el proceso completo de barrido para algunas subredes con topología irregular (las cuatro primeras aparecen en la Figura 61). Estos tiempos incluyen el envío de paquetes de sondeo a todos los conmutadores de la subred y la recepción de las correspondientes respuestas. Obviamente, a medida que aumenta el tamaño de la subred, el tiempo necesario para sondear todos sus conmutadores se incrementa.

Tabla 11. Tiempos de barrido para algunas topologías irregulares.

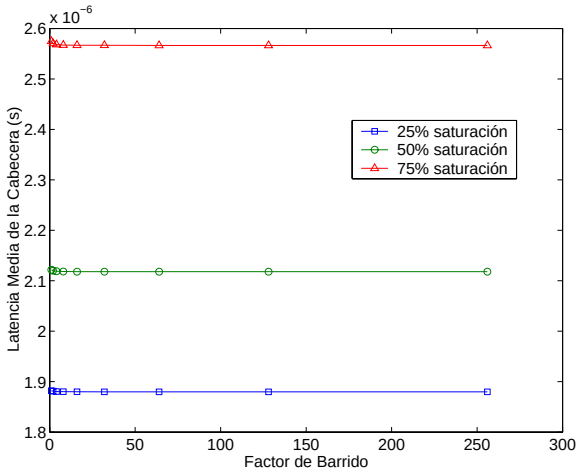
Tamaño de la subred	Tiempo de barrido (s)
8 conmutadores (15 nodos)	0.001202668
16 conmutadores (30 nodos)	0.002732652
24 conmutadores (46 nodos)	0.003858949
32 conmutadores (53 nodos)	0.004984007
48 conmutadores (112 nodos)	0.007236010
64 conmutadores (146 nodos)	0.009603447

La Figura 63 muestra la sobrecarga introducida por el proceso de barrido sobre el tráfico de aplicación, en función de la frecuencia con que se repite dicho proceso. Los resultados corresponden a las subredes de 8, 16 y 24 conmutadores de la Figura 61. En cada gráfica se muestra la latencia media (desde generación) de la cabecera de los paquetes de datos. Las series representan la tasa de generación de paquetes. En concreto, para cada subred se han empleado tasas de generación correspondientes al 25%, al 50% y al 75% de la tasa que satura la topología.

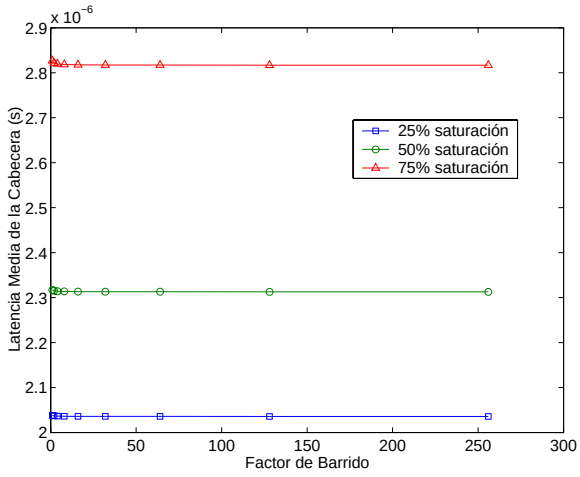
Como frecuencia de barrido máxima hemos considerado el caso en el que el SM inicia el siguiente proceso de barrido inmediatamente después de la finalización del proceso anterior. En este caso, el SM estaría barriendo la subred constantemente. En el eje horizontal de las gráficas de la Figura 63 aparece el valor de un factor de barrido (desde 1 hasta 256) que dará lugar a las distintas frecuencias de barrido empleadas. Un factor de 1 significa que el tiempo entre las ejecuciones del proceso de barrido coincide con el tiempo de barrido de la subred (Tabla 11), es decir, la máxima frecuencia de barrido. Un factor de barrido de 4 significa que el SM lleva a cabo uno de cada cuatro posibles barridos.



(a) 8 conmutadores



(b) 16 conmutadores



(c) 24 conmutadores

Figura 63. Sobrecarga introducida por el proceso de barrido.

Los resultados nos permiten concluir que el aumento de la frecuencia de barrido (de derecha a izquierda) no introduce apenas sobrecarga en el tráfico de aplicación. El incremento en la latencia es apenas perceptible. Esto se debe a que una frecuencia muy alta no se traduce

necesariamente en una gran ocupación del canal físico por parte de los SMPs de barrido emitidos por el SM.

En este sentido, la Figura 64 muestra el tiempo durante el cual los canales de salida en los puertos del conmutador que aloja el SM están siendo empleados para enviar SMPs de sondeo. De nuevo hemos considerado las topologías de 8, 16 y 24 conmutadores de la Figura 61, asumiendo que el SM está alojado en el conmutador etiquetado como *switch 0*.

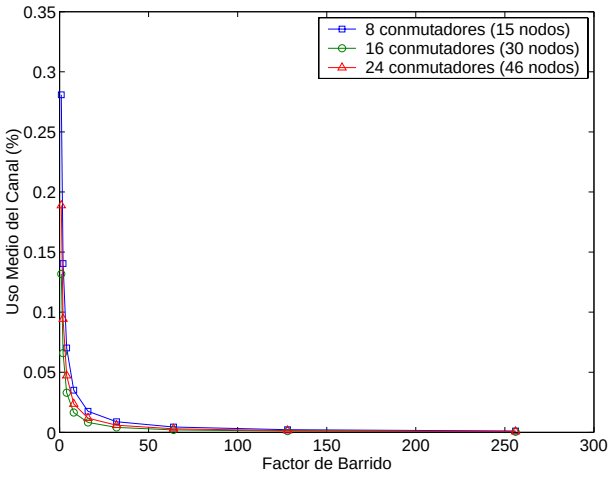


Figura 64. Porcentaje medio de ocupación del canal por parte de los SMPs de barrido.

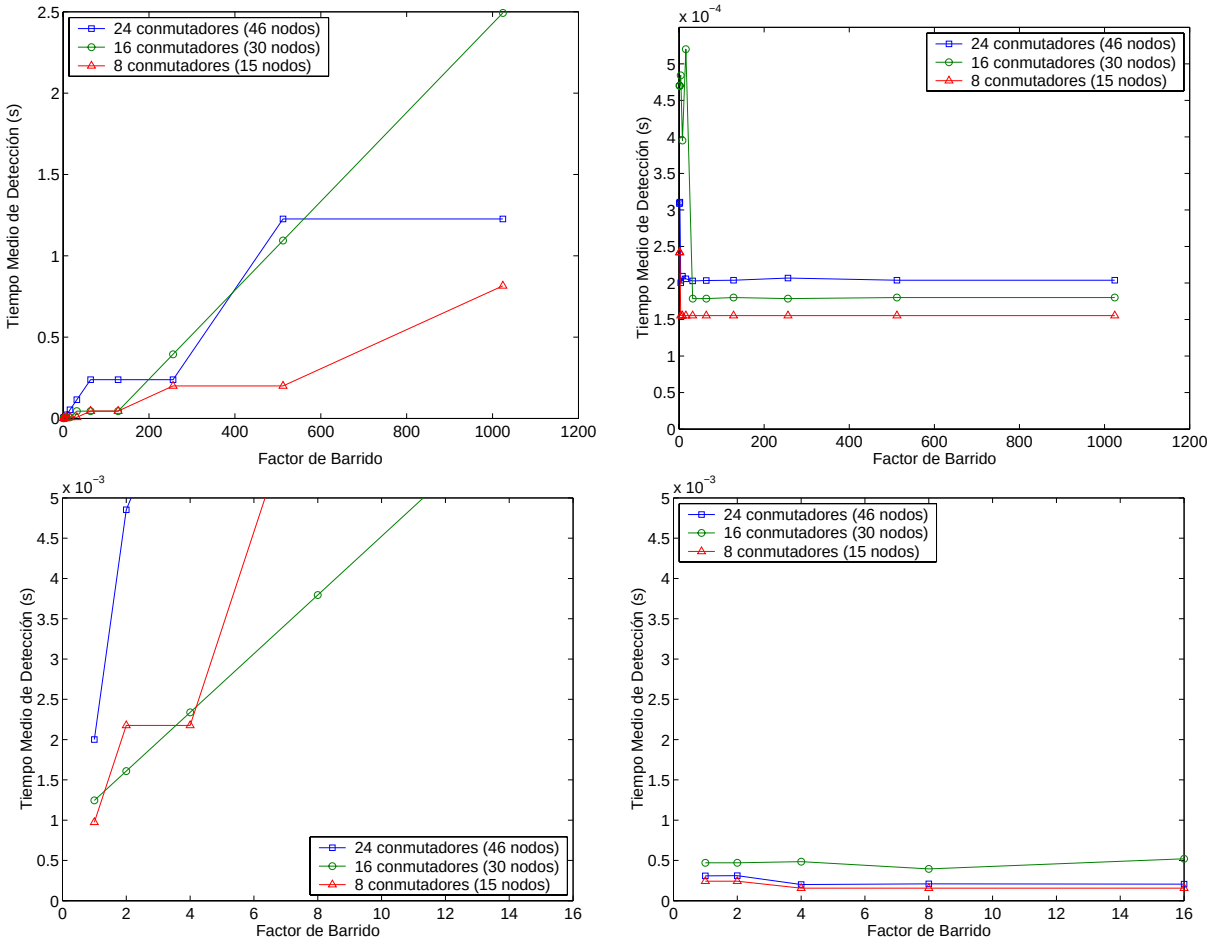
Estos porcentajes de ocupación del canal tan bajos se deben a que, aunque el SM esté generando constantemente SMPs de barrido, cada uno de esos SMPs tiene que ser procesado por el SMI antes de ser entregado a la red, lo cual introduce una secuenciación entre ellos.

Una vez comprobado que la elección de la frecuencia con la que se sondea la topología no supone un perjuicio importante para el tráfico de las aplicaciones, veamos con qué rapidez reaccionan al cambio los mecanismos de detección contemplados en InfiniBand.

La Figura 65 muestra el tiempo que transcurre desde que se produce el cambio en la topología hasta que el SM es consciente de ello. Las gráficas de la izquierda han sido obtenidas empleando únicamente el mecanismo de barrido. Las de la derecha han sido obtenidas activando en todos los conmutadores de la subred la capacidad de emitir notificaciones. Obsérvese que las gráficas inferiores muestran un detalle de las gráficas superiores. Cada serie corresponde a una topología diferente. Para cada factor de barrido (desde 1 hasta 1024), se han ejecutado varias simulaciones, desactivando en cada caso un conmutador distinto, y después se ha obtenido la media aritmética de los tiempos de detección.

Como era de esperar, en la Figura 65(a) puede observarse una tendencia ascendente en el tiempo de detección, a medida que aumenta el factor de barrido. Por otro lado, en la Figura 65(b) se observa que el tiempo de detección mediante notificaciones no depende del factor de barrido, salvo para valores muy pequeños de este parámetro. En estos casos, los SMPs de barrido retrasan la llegada de las notificaciones. Además, en las gráficas inferiores puede verse

que el tiempo de detección es siempre menor cuando las notificaciones están habilitadas. A la vista de estos resultados, parece obvio que este mecanismo es preferible al de barrido.



(a) Mecanismo de Barrido

(b) Barrido y notificaciones

Figura 65. Tiempo de detección del cambio en función del mecanismo empleado.

5.3 Descubrimiento completo de la topología

El primer mecanismo de descubrimiento que hemos considerado es rudimentario. Consiste en la obtención de la topología completa de la subred, descartando cualquier información de configuración previamente recopilada, tal y como se hace en Myrinet [14] y Autonet [80].

En la Sección 3.2.1 se describen los tipos de SMPs que el SM debe emplear para explorar los dispositivos activos en la subred, de acuerdo con la especificación de InfiniBand. Como se comentó, todos los paquetes de exploración deben emplear encaminamiento dirigido, dado que las nuevas tablas de encaminamiento todavía no han sido cargadas. La Figura 66

muestra las tareas relativas al proceso de exploración de dispositivos en nuestro modelo de gestor de la subred, asumiendo el mecanismo de descubrimiento completo.

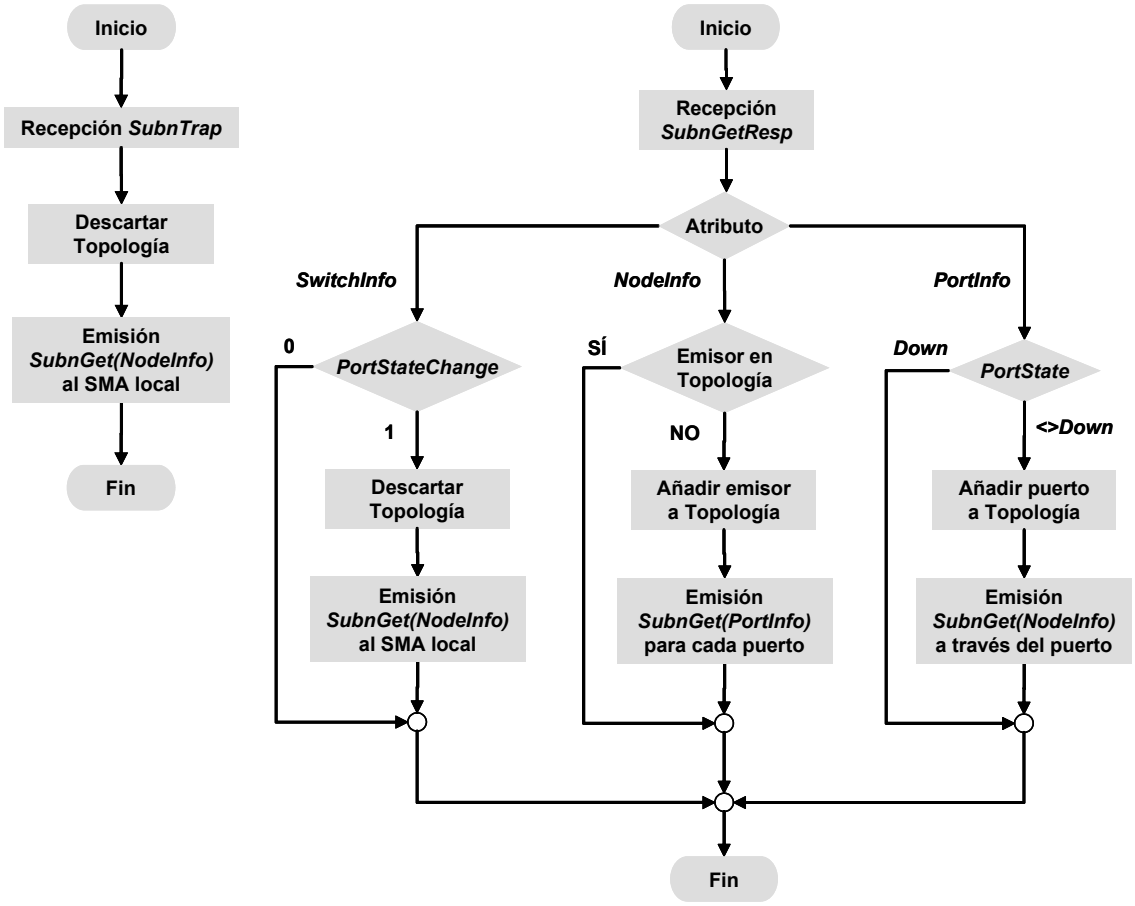


Figura 66. Comportamiento del SM tras la recepción de una notificación y de una respuesta a un SMP de tipo *SubnGet*.

Una vez detectado el cambio, ya sea mediante la recepción de una notificación, o durante el proceso de barrido periódico, el proceso de descubrimiento completo comienza con la emisión de un primer SMP *SubnGet(NodeInfo)* al SMA en el nodo local. Para ello, se emplea una ruta dirigida vacía. Después, cada vez que el SM recibe una respuesta a un SMP previo (un SMP de tipo *SubnGetResp*), si se trata de información sobre un nuevo dispositivo, éste se añade a la base de datos topológica. Después, se inyectan en la red nuevos SMPs *SubnGet(PortInfo)*, con el objetivo de recoger información sobre todos sus puertos. Si el SMP recibido contiene información sobre un puerto, si dicho puerto está activo, éste se añade a la topología. Además, se genera un nuevo SMP *SubnGet(NodeInfo)*, que deberá ser respondido por el SMA del dispositivo que hay al otro lado del enlace.

Para construir los *SubnGet(PortInfo)*, el SM reutiliza la ruta de ida del paquete recibido. En el caso de los *SubnGet(NodeInfo)*, a la ruta de ida del paquete recibido se le concatena el puerto que conduce al dispositivo de interés, añadiendo así un salto. De esta forma, todos los paquetes emplean encaminamiento dirigido "puro", sin segmentos con encaminamiento basado en destino.

El proceso de exploración finaliza cuando el SM haya recibido respuesta a todos los SMPs emitidos, o bien los correspondientes tiempos de espera hayan expirado. Una vez obtenido el mapa completo de la subred, el SM podrá comenzar a configurar los dispositivos activos en la misma, mediante el envío de SMPs de tipo *SubnSet* (Sección 3.2.4).

En nuestra implementación, los SMPs de exploración se propagan por la red de forma no determinista, es decir, no hay ninguna ordenación especial entre ellos. El SM crea nuevos paquetes de exploración a medida que va recibiendo respuestas a SMPs previos desde los SMAs en los dispositivos de la subred. Se trata por tanto de un recorrido de la topología en orden de propagación. Ésta es la filosofía empleada por el mecanismo de obtención de topología de Autonet [72]. Otras posibles implementaciones podrían ejecutar un recorrido en anchura (BFS, *Breadth First Search*), tal y como hace Myrinet [47], o bien un recorrido en profundidad (DFS, *Depth First Search*), tal y como se propone en [77].

La Figura 67 muestra una posible secuencia de exploración para una subred irregular compuesta por 8 conmutadores y 7 nodos terminales. La figura incluye los identificadores locales (LIDs) asignados por el SM a los puertos de la subred durante la exploración, asumiendo que todos los dispositivos se encuentran activos. El SM está alojado en el nodo terminal marcado con un círculo. Los valores en los extremos de los enlaces representan números de puerto en conmutadores y CAs. Por simplicidad, no representamos los CAs en los nodos terminales.

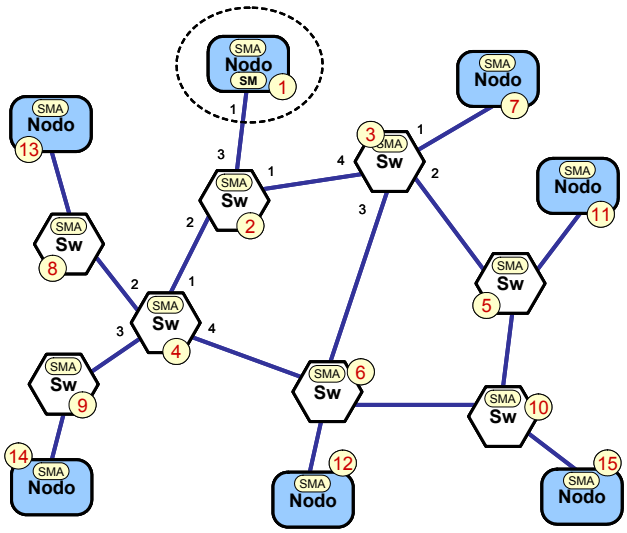


Figura 67. Ejemplo de subred con topología irregular.

A modo de ejemplo ilustrativo, la Tabla 12 muestra los SMPs generados por el proceso de exploración para descubrir (únicamente) los nodos *1*, *2*, *3* y *4* de la Figura 67. Para explorar la topología completa (los 15 dispositivos), el SM emplea exactamente 97 SMPs.

Tabla 12. SMPs generados por el mecanismo de descubrimiento completo.

	Peticiones (*SubnGet*, *SubnSet*)					Respuestas (*SubnGetResp*)	
Transacción	**Método**	**Atributo**	**Modificador del atributo**	**Componentes de interés**	**Ruta de ida**	**Valor de los componentes**	**Ruta de retorno**
1	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	( )	NodeGUID=200, NodeType=CA, NumPorts=1,…	( )
2	SubnGet	PortInfo	1	PortState, VLCap,…	( )	PortState<>DOWN,…	( )
3	SubnSet	PortInfo	1	LID, MasterSMLID,…	( )	**LID=1**, MasterSMLID=1,…	( )
4	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1)	NodeGUID=100, NodeType=Switch, NumPorts=4,…	(3)
5	SubnGet	SwitchInfo	n/a	LinearFDBCap, LIDsPerPort,…	(1)	…	(3)
6	SubnGet	PortInfo	0	CapabilityMask, RespTimeValue,…	(1)	…	(3)
7	SubnGet	PortInfo	1	PortState, VLCap,…	(1)	PortState<>DOWN,…	(3)
8	SubnGet	PortInfo	2	PortState, VLCap,…	(1)	PortState<>DOWN,…	(3)
9	SubnGet	PortInfo	3	PortState, VLCap,…	(1)	PortState<>DOWN,…	(3)
10	SubnGet	PortInfo	4	PortState, VLCap,…	(1)	PortState=DOWN,…	(3)
11	SubnSet	PortInfo	0	LID, MasterSMLID,…	(1)	**LID=2**, MasterSMLID=1,…	(3)
12	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,1)	NodeGUID=105, NodeType=Switch, NumPorts=4,…	(4,3)
13	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,2)	NodeGUID=106, NodeType=Switch, NumPorts=4,…	(1,3)
14	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,3)	NodeGUID=200, NodeType=CA, NumPorts=1,…	(1,3)
15	SubnGet	SwitchInfo	n/a	LinearFDBCap, LIDsPerPort,…	(1,1)	…	(4,3)
16	SubnGet	PortInfo	0	CapabilityMask, RespTimeValue,…	(1,1)	…	(4,3)
17	SubnGet	PortInfo	1	PortState, VLCap,…	(1,1)	PortState<>DOWN,…	(4,3)
18	SubnGet	PortInfo	2	PortState, VLCap,…	(1,1)	PortState<>DOWN,…	(4,3)
19	SubnGet	PortInfo	3	PortState, VLCap,…	(1,1)	PortState<>DOWN,…	(4,3)
20	SubnGet	PortInfo	4	PortState, VLCap,…	(1,1)	PortState<>DOWN,…	(4,3)
21	SubnGet	SwitchInfo	n/a	LinearFDBCap, LIDsPerPort,…	(1,2)	…	(1,3)
22	SubnGet	PortInfo	0	CapabilityMask, RespTimeValue,…	(1,2)	…	(1,3)
23	SubnGet	PortInfo	1	PortState, VLCap,…	(1,2)	PortState<>DOWN,…	(1,3)
24	SubnGet	PortInfo	2	PortState, VLCap,…	(1,2)	PortState<>DOWN,…	(1,3)
25	SubnGet	PortInfo	3	PortState, VLCap,…	(1,2)	PortState<>DOWN,…	(1,3)
26	SubnGet	PortInfo	4	PortState, VLCap,…	(1,2)	PortState<>DOWN,…	(1,3)
27	SubnSet	PortInfo	0	LID, MasterSMLID,…	(1,1)	**LID=3**, MasterSMLID=1,…	(4,3)
28	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,1,1)	…	…
29	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,1,2)	…	…
30	SubnSet	PortInfo	0	LID, MasterSMLID,…	(1,2)	**LID=4**, MasterSMLID=1,…	(1,3)
31	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,1,3)	…	…
32	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,2,1)	…	…
33	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,1,4)	…	…
34	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,2,2)	…	…

	Peticiones (*SubnGet, SubnSet*)					Respuestas (*SubnGetResp*)	
Transacción	**Método**	**Atributo**	**Modificador del atributo**	**Componentes de interés**	**Ruta de ida**	**Valor de los componentes**	**Ruta de retorno**
35	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,2,3)	…	…
36	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	(1,2,4)	…	…
...							

En la tabla, para cada SMP emitido por el SM, identificado por un identificador de transacción, se muestra en primer lugar el contenido del paquete de ida (petición) y, a continuación, el contenido de la correspondiente respuesta. En los SMPs de respuesta, el método, atributo y modificador de atributo no cambian de valor.

Las rutas dirigidas se expresan mediante listas de puertos de conmutador entre paréntesis. Estas listas están vacías para aquellos SMPs destinados al nodo en el que se encuentra alojado el SM. El modificador del atributo especifica un número de puerto cuando se emplea el atributo *PortInfo*. Para el resto de atributos de gestión mostrados en la tabla no es aplicable. El puerto número *0* hace referencia al puerto de gestión en el conmutador.

Los SMPs *3*, *11*, *27* y *30* son empleados por el SM para enviar los LIDs a los dispositivos recién descubiertos. Los SMPs *5*, *15* y *21* sirven para obtener ciertos valores internos cuando un nuevo conmutador es descubierto. Por simplicidad, ambos comportamientos no han sido incluidos en la Figura 66.

5.4 Descubrimiento parcial

En general, los cambios afectan a una región muy reducida de la topología [45]. Algunas excepciones son la activación inicial de la subred o la fusión de dos subredes inicialmente disjuntas. En la mayoría de las ocasiones, el cambio consiste únicamente en la activación o desactivación (o fallo) de un dispositivo individual, o bien puede tratarse de la adición, supresión o reasignación de alguno de los enlaces de la subred.

Por tanto, parece lógico que una implementación mejorada del proceso de descubrimiento debería aprovechar la información topológica previamente recopilada, explorando únicamente la región de la subred que se ha visto directamente afectada por el cambio. De esta forma, el número de SMPs de exploración y el tiempo total de descubrimiento podrían reducirse notablemente.

Además, este mecanismo de descubrimiento parcial puede combinarse con el empleo de SMPs con encaminamiento basado en destino en el segmento inicial de la ruta dirigida, reduciendo así la sobrecarga debida al procesamiento del SMP en cada SMI intermedio (Sección 3.1.3).

Comenzaremos esta sección con una descripción informal del mecanismo de descubrimiento parcial, analizando por separado la manera de actuar en los casos de activación y desactivación de dispositivos. Después, describiremos el protocolo propuesto y veremos algunos ejemplos de funcionamiento.

5.4.1 Activación de dispositivos

Cuando el SM detecta la activación de alguno de los puertos de la subred, éste debe comenzar inmediatamente la exploración de los nuevos dispositivos que han aparecido al otro lado del enlace conectado a dicho puerto. Para ello, puede proceder exactamente de la misma manera que en el caso del descubrimiento completo. La única diferencia es que el proceso de exploración sólo debe ser aplicado sobre la nueva región. Es decir, la exploración debe detenerse cuando los SMPs alcanzan dispositivos ya conocidos.

En este caso, y aprovechando la información de las tablas de encaminamiento previas, los paquetes de exploración pueden emplear un segmento inicial con encaminamiento basado en destino, hasta alcanzar el conmutador donde se ha detectado el cambio. A partir de ese punto, el camino hasta el nuevo dispositivo (o dispositivos) debe ser especificado mediante una ruta dirigida. Nótese que la activación inicial de la subred es un caso particular, en el cual no hay una configuración previa para actualizar, y cada paquete de exploración debe emplear encaminamiento dirigido "puro" (sin segmentos iniciales).

La Figura 68 muestra la misma subred de la Figura 67, en la que han aparecido un nuevo conmutador y dos nodos terminales. El SM detectará la activación del puerto número *3* en los conmutadores *8* y *9*. Después, deberá descubrir los nuevos dispositivos y asignarles los LIDs *16*, *17* y *18*. Los SMPs de exploración pueden alcanzar los conmutadores *8* y *9* empleando encaminamiento basado en destino. Desde esos conmutadores, las rutas serán dirigidas. En particular, será necesario un solo salto para llegar al nuevo conmutador y dos saltos para alcanzar los nuevos nodos terminales.

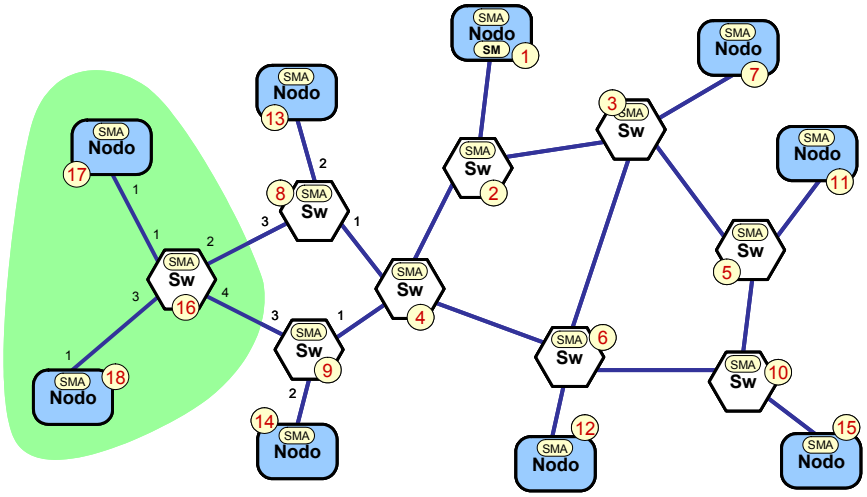


Figura 68. Adición de tres dispositivos a la subred de la Figura 67.

Un par de comentarios más sobre este ejemplo. Lógicamente, aunque el proceso de exploración se dispara a partir de dos puertos distintos (en los conmutadores *8* y *9*) los nuevos dispositivos no serán visitados dos veces. En cuanto a la asignación de identificadores locales a los nuevos nodos terminales, podría ser justo la contraria, en el caso de que el SM recibiera los paquetes de respuesta desde el nodo *18* en primer lugar.

5.4.2 Desactivación de dispositivos

Si el cambio topológico consiste en la desactivación de un puerto, una porción de la subred dejará de ser alcanzable. Existen varios motivos por los que un dispositivo ya no es alcanzable. En primer lugar, puede que el dispositivo haya sido desactivado, o haya fallado. Además, es posible que el dispositivo y el SM hayan quedado físicamente desconectados, en subredes distintas. Por último, puede ocurrir que, aunque el dispositivo y el SM se encuentren en la misma subred, al menos uno de ellos no pueda alcanzar al otro empleando el conjunto de rutas previamente configuradas en los conmutadores de la subred. En este caso, se dice que están "lógicamente" desconectados, pero no "físicamente" desconectados.

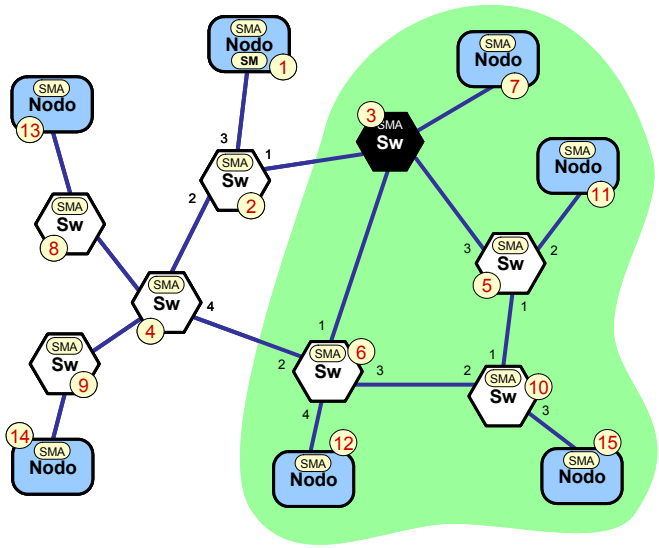


Figura 69. Desactivación de un conmutador en la subred de la Figura 67.

La Figura 69 muestra la topología de la Figura 67, en la que el conmutador *3* ha sido desactivado. Supongamos que las tablas de encaminamiento han sido configuradas de acuerdo a las "rutas originales" de la tabla Tabla 13. Tras el cambio, el SM no podrá alcanzar el conmutador *3* debido al primero de los motivos anteriores. El nodo terminal *7* tampoco podrá ser alcanzado debido al segundo motivo. Finalmente, los dispositivos *5*, *6*, *10*, *11*, *12* y *15* serán inalcanzables debido al tercer motivo. En este caso, el mecanismo de descubrimiento deberá descartar la información topológica sobre los dispositivos *3* y *7*, y actualizar las rutas empleadas para alcanzar el resto de los dispositivos citados.

Tabla 13. Actualización de las rutas para SMPs tras el cambio.

	Rutas originales		Rutas nuevas	
LID	**Ruta**	**Dispositivos implicados**	**Ruta**	**Dispositivos implicados**
2	1→2	1–2	1→2	1–2
3	1→3	1–2–3	(dispositivo desactivado)	
4	1→4	1–2–4	1→4	1–2–4
5	1→5	1–2–3–5	**1→4, (4,3,1)**	**1–2–4–6–10–5**
6	1→6	1–2–3–6	**1→4, (4)**	**1–2–4–6**
7	1→7	1–2–3–7	(físicamente desconectado)	
8	1→8	1–2–4–8	1→8	1–2–4–8
9	1→9	1–2–4–9	1→9	1–2–4–9
10	1→10	1–2–3–5–10	**1→4, (4,3)**	**1–2–4–6–10**
11	1→11	1–2–3–5–11	**1→4, (4,3,1,2)**	**1–2–4–6–10–5–11**
12	1→12	1–2–3–6–12	**1→4, (4,4)**	**1–2–4–6–12**
13	1→13	1–2–4–8–13	1→13	1–2–4–8–13
14	1→14	1–2–4–9–14	1→14	1–2–4–9–14
15	1→15	1–2–3–5–10–15	**1→4, (4,3,3)**	**1–2–4–6–10–15**

Hasta que las nuevas tablas de encaminamiento hayan sido distribuidas a los conmutadores de la subred, el SM debe ser capaz de alcanzar los dispositivos que han quedado lógicamente desconectados, mediante el empleo de rutas dirigidas. Estas rutas pueden emplear un segmento inicial con encaminamiento basado en destino para llegar a un conmutador intermedio. En la Tabla 13, todas las "rutas nuevas" resaltadas incluyen un segmento inicial hasta el conmutador *4*. En la tabla, las rutas dirigidas están expresadas mediante listas de puertos de conmutador entre paréntesis, mientras que las rutas para encaminamiento en base a destino se identifican mediante una flecha y dos valores (SLID y DLID) en sus extremos.

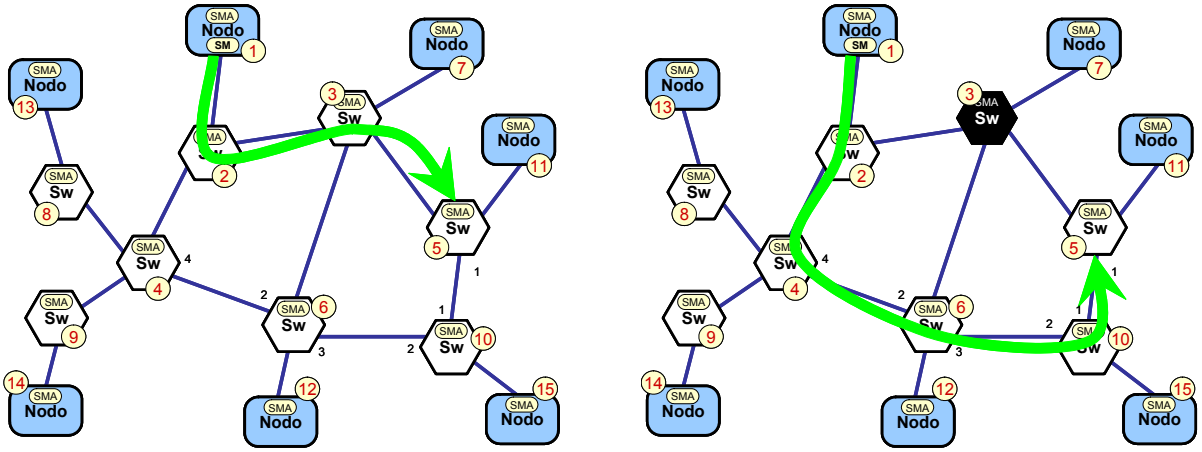


Figura 70. Ruta hasta el conmutador *5* antes y después del cambio.

Fijémonos, por ejemplo, en la primera de las rutas de la Tabla 13 que debe ser actualizada. Se trata de la ruta para alcanzar el conmutador *5*. Como muestra la Figura 70 (izquierda), antes del cambio topológico la ruta atravesaba el conmutador *3*. Como hemos dicho, tras la desactivación de *3*, el conmutador *5* queda lógicamente desconectado. Por tanto, el mecanismo de descubrimiento debe buscar una ruta alternativa. Para ello, en este caso se ha encontrado un camino (a la derecha en la Figura 70) a través del conmutador *4*, para el cual se sigue

manteniendo la ruta original. Desde ahí, una ruta dirigida de longitud 3 permite alcanzar el conmutador *5*.

5.4.3 Protocolo propuesto

El mecanismo de descubrimiento parcial presentado en esta sección se comporta de la manera que acabamos de describir. En primer lugar, el SM envía un SMP de barrido a cada conmutador de la subred y lo marca como *waiting* (en espera), para indicar que la respuesta al sondeo está pendiente. Cada vez que se recibe una respuesta al sondeo, el conmutador emisor se marca como *reachable* (lógicamente alcanzable).

Si durante el barrido se detecta la activación de un puerto, el proceso de exploración se dispara desde dicho puerto, haciendo uso de SMPs con encaminamiento dirigido y segmentos iniciales con encaminamiento en base a destino. Los SMPs de exploración no se propagan a través de dispositivos ya conocidos. Durante el proceso, cada nuevo dispositivo descubierto es marcado también como *reachable*.

Adicionalmente, el SM debe almacenar para cada conmutador de la subred una lista (*destination devices*) con todos los dispositivos en cuya ruta (desde o hacia el SM) está implicado el conmutador. Por ejemplo, la lista para el conmutador *3* antes de su desactivación (Figura 69) incluye los dispositivos *5*, *6*, *7*, *10*, *11*, *12* y *15*. En las "rutas originales" de la Tabla 13, puede verse que el conmutador *3* está involucrado en las rutas hacia todos estos dispositivos.

Por otro lado, si durante el proceso de barrido se detecta la desactivación de un puerto, el SM elimina el enlace correspondiente de la topología y marca como *missing* (desaparecido) el dispositivo al otro extremo. Además, todos los dispositivos incluidos en su lista *destination devices* pasan a estar lógicamente desconectados, y por tanto deben ser marcados también como *missing*. Esto significa que, tras el cambio, el conjunto de rutas originales no permite al SM alcanzar estos dispositivos, y viceversa[1].

Siguiendo el algoritmo anterior, todo dispositivo pasará desde el estado *waiting* al estado *reachable* o al estado *missing*. Una vez que ya no queden conmutadores en el estado *waiting*, el SM comprueba la existencia de algún dispositivo *d* en estado *missing* conectado directamente a un conmutador *s* en estado *reachable*. Si se da el caso, el SM construye una nueva ruta hacia *d*, concatenando a la ruta hasta el conmutador *s* el puerto que conecta *s* con *d*. Si *d* es un conmutador, éste será marcado como *waiting*, y el SM le enviará un nuevo SMP (empleando la nueva ruta) para sondear individualmente su estado. Si, por el contrario, *d* es un nodo terminal, éste será marcado como *reachable*.

[1] Estamos considerando una situación genérica en la que la ruta desde un dispositivo *a* hasta otro dispositivo *b* no tiene por qué coincidir con la ruta desde *b* hasta *a*.

Finalmente, cuando no queden dispositivos que satisfagan la condición del párrafo anterior, cada dispositivo que permanezca en el estado *missing* será eliminado de la base de datos topológica del SM.

Para terminar con la descripción del protocolo de descubrimiento parcial, los diagramas de flujo de la Figura 71 resumen las tareas ejecutadas por el SM cada vez que éste recibe una notificación de cambio y ante la llegada de un paquete de respuesta a una petición previa. Por simplicidad, en la figura no se incluyen los cambios en el estado de los dispositivos que acabamos de detallar.

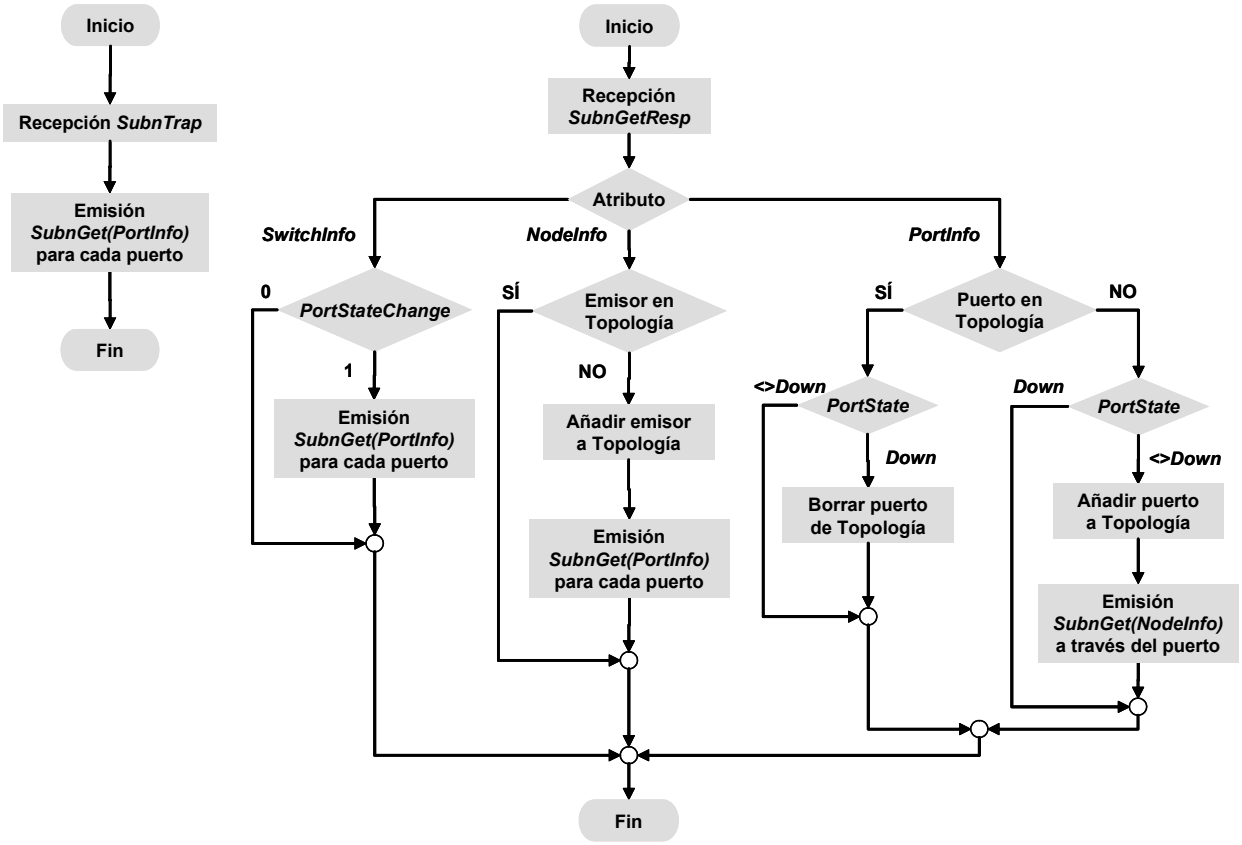


Figura 71. Comportamiento del SM tras la recepción de una notificación y de una respuesta a un SMP de tipo *SubnGet*.

Tras la detección de un cambio en el estado de alguno de los puertos de un conmutador, el SM explora individualmente cada uno de los puertos en dicho conmutador, empleando SMPs *SubnGet(PortInfo)*. Las correspondientes respuestas servirán para detectar cuál de sus puertos ha sido activado o desactivado. La activación de un puerto desencadenará la emisión de un SMP *SubnGet(NodeInfo)* con encaminamiento dirigido y segmento inicial basado en destino. La desactivación de un puerto provocará su eliminación de la topología y el resto de acciones arriba descritas.

Obsérvese que cuando el paquete de respuesta contiene el atributo *NodeInfo*, se ejecutan las mismas acciones que en el caso del descubrimiento completo. El único matiz es que

cuando se pregunta si el emisor del paquete está en la topología, en realidad nos estamos refiriendo a la información topológica correspondiente a la configuración anterior (antes del cambio).

Tabla 14. SMPs empleados para detectar el cambio y explorar los nuevos dispositivos.

	Peticiones (*SubnGet*, *SubnSet*)					Respuestas (*SubnGetResp*)	
Transacción	**Método**	**Atributo**	**Modif.**	**Componentes de interés**	**Ruta de ida**	**Valor de los componentes**	**Ruta de retorno**
338	SubnGet	SwitchInfo	n/a	PortStateChange	1→2	PortStateChange=0	2→1
339	SubnGet	SwitchInfo	n/a	PortStateChange	1→3	PortStateChange=0	3→1
340	SubnGet	SwitchInfo	n/a	PortStateChange	1→4	PortStateChange=0	4→1
341	SubnGet	SwitchInfo	n/a	PortStateChange	1→5	PortStateChange=0	5→1
342	SubnGet	SwitchInfo	n/a	PortStateChange	1→6	PortStateChange=0	6→1
343	SubnGet	SwitchInfo	n/a	PortStateChange	1→8	**PortStateChange=1**	8→1
344	SubnGet	SwitchInfo	n/a	PortStateChange	1→9	**PortStateChange=1**	9→1
345	SubnGet	SwitchInfo	n/a	PortStateChange	1→10	PortStateChange=0	10→1
346	SubnSet	SwitchInfo	n/a	PortStateChange	1→8	PortStateChange=0	8→1
347	SubnGet	PortInfo	1	PortState	1→8	PortState<>DOWN	8→1
348	SubnGet	PortInfo	2	PortState	1→8	PortState<>DOWN	8→1
349	SubnGet	PortInfo	3	PortState	1→8	**PortState<>DOWN**	8→1
350	SubnGet	PortInfo	4	PortState	1→8	PortState=DOWN	8→1
351	SubnSet	SwitchInfo	n/a	PortStateChange	1→9	PortStateChange=0	9→1
352	SubnGet	PortInfo	1	PortState	1→9	PortState<>DOWN	9→1
353	SubnGet	PortInfo	2	PortState	1→9	PortState<>DOWN	9→1
354	SubnGet	PortInfo	3	PortState	1→9	**PortState<>DOWN**	9→1
355	SubnGet	PortInfo	4	PortState	1→9	PortState=DOWN	9→1
356	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	1→8, (3)	NodeGUID=108, NodeType=Switch, NumPorts=4,…	(2), 8→1
357	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	1→9, (3)	NodeGUID=108, NodeType=Switch, NumPorts=4,…	(4), 9→1
358	SubnGet	SwitchInfo	n/a	LinearFDBCap, LIDsPerPort,…	1→8, (3)	…	(2), 8→1
359	SubnGet	PortInfo	0	CapabilityMask, RespTimeValue,…	1→8, (3)	…	(2), 8→1
360	SubnGet	PortInfo	1	PortState, VLCap,…	1→8, (3)	PortState<>DOWN,…	(2), 8→1
361	SubnGet	PortInfo	2	PortState, VLCap,…	1→8, (3)	PortState<>DOWN,…	(2), 8→1
362	SubnGet	PortInfo	3	PortState, VLCap,…	1→8, (3)	PortState<>DOWN,…	(2), 8→1
363	SubnGet	PortInfo	4	PortState, VLCap,…	1→8, (3)	PortState<>DOWN,…	(2), 8→1
364	SubnSet	PortInfo	0	LID, MasterSMLID,…	1→8, (3)	**LID=16**, MasterSMLID=1,…	(2), 8→1
365	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	1→8, (3,1)	NodeGUID=208, NodeType=Switch, NumPorts=4,…	(1,2), 8→1
366	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	1→8, (3,2)	NodeGUID=101, NodeType=Switch, NumPorts=4,…	(3,2), 8→1
367	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	1→8, (3,3)	NodeGUID=209, NodeType=Switch, NumPorts=4,…	(1,2), 8→1
368	SubnGet	NodeInfo	n/a	NodeGUID, NodeType, NumPorts,…	1→8, (3,4)	NodeGUID=103, NodeType=Switch, NumPorts=4,…	(3,2), 8→1
369	SubnGet	PortInfo	1	PortState, VLCap,…	1→8, (3,1)	PortState<>DOWN,…	(1,2), 8→1
370	SubnGet	PortInfo	1	PortState, VLCap,…	1→8, (3,3)	PortState<>DOWN,…	(1,2), 8→1
371	SubnSet	PortInfo	1	LID, MasterSMLID,…	1→8, (3,1)	**LID=17**, MasterSMLID=1,…	(1,2), 8→1
372	SubnSet	PortInfo	1	LID, MasterSMLID,…	1→8, (3,3)	**LID=18**, MasterSMLID=1,…	(1,2), 8→1

En el ejemplo de la Figura 68, el SM emplea 35 SMPs para detectar el cambio (mediante barrido) y explorar los nuevos dispositivos (*16*, *17* y *18*). La Tabla 14 muestra una po-

sible ordenación para estos 35 SMPs. El mecanismo de descubrimiento completo requiere exactamente 123 SMPs para ejecutar las mismas tareas.

Tras recibir las respuestas a los SMPs de sondeo 343 y 344, el SM detecta dos cambios en la subred, que en realidad se corresponden con el mismo cambio topológico; la activación del conmutador *16*. Los SMPs 346 y 351 son empleados por el SM para borrar el valor local de *PortStateChange* en los conmutadores en los que se ha detectado el cambio. Este comportamiento no se ha reflejado en los diagramas de flujo de la Figura 71 (aunque sí se detalló en la Figura 55). Las respuestas a los SMPs 349 y 354 permiten averiguar qué puertos se han activado. Los SMPs 356 y 357 alcanzarán al nuevo conmutador (con GUID *108*).

Para detectar el cambio de la Figura 69 y ejecutar el protocolo de descubrimiento parcial, el SM genera los 26 SMPs que aparecen en la Tabla 15.

Tabla 15. SMPs empleados para detectar el cambio y actualizar la información topológica.

	Peticiones (*SubnGet*, *SubnSet*)					**Respuestas (*SubnGetResp*)**	
Transacción	**Método**	**Atributo**	**Modif.**	**Componentes de interés**	**Ruta de ida**	**Valor de los componentes**	**Ruta de retorno**
338	SubnGet	SwitchInfo	n/a	PortStateChange	1→2	**PortStateChange=1**	2→1
339	SubnGet	SwitchInfo	n/a	PortStateChange	1→3	(no hay respuesta)	
340	SubnGet	SwitchInfo	n/a	PortStateChange	1→4	PortStateChange=0	4→1
341	SubnGet	SwitchInfo	n/a	PortStateChange	1→5	(no hay respuesta)	
342	SubnGet	SwitchInfo	n/a	PortStateChange	1→6	(no hay respuesta)	
343	SubnGet	SwitchInfo	n/a	PortStateChange	1→8	PortStateChange=0	8→1
344	SubnGet	SwitchInfo	n/a	PortStateChange	1→9	PortStateChange=0	9→1
345	SubnGet	SwitchInfo	n/a	PortStateChange	1→10	(no hay respuesta)	
346	SubnSet	SwitchInfo	n/a	PortStateChange	1→2	PortStateChange=0	2→1
347	SubnGet	PortInfo	1	PortState	1→2	**PortState=DOWN**	2→1
348	SubnGet	PortInfo	2	PortState	1→2	PortState<>DOWN	2→1
349	SubnGet	PortInfo	3	PortState	1→2	PortState<>DOWN	2→1
350	SubnGet	PortInfo	4	PortState	1→2	PortState=DOWN	2→1
351	SubnGet	SwitchInfo	n/a	PortStateChange	1→4, (4)	**PortStateChange=1**	(2), 4→1
352	SubnSet	SwitchInfo	n/a	PortStateChange	1→4, (4)	PortStateChange=0	(2), 4→1
353	SubnGet	PortInfo	1	PortState	1→4, (4)	**PortState=DOWN**	(2), 4→1
354	SubnGet	PortInfo	2	PortState	1→4, (4)	PortState<>DOWN	(2), 4→1
355	SubnGet	PortInfo	3	PortState	1→4, (4)	PortState<>DOWN	(2), 4→1
356	SubnGet	PortInfo	4	PortState	1→4, (4)	PortState<>DOWN	(2), 4→1
357	SubnGet	SwitchInfo	n/a	PortStateChange	1→4, (4,3)	PortStateChange=0	(2,2), 4→1
358	SubnGet	SwitchInfo	n/a	PortStateChange	1→4, (4,3,1)	**PortStateChange=1**	(1,2,2), 4→1
359	SubnSet	SwitchInfo	n/a	PortStateChange	1→4, (4,3,1)	PortStateChange=0	(1,2,2), 4→1
360	SubnGet	PortInfo	1	PortState	1→4, (4,3,1)	PortState<>DOWN	(1,2,2), 4→1
361	SubnGet	PortInfo	2	PortState	1→4, (4,3,1)	PortState<>DOWN	(1,2,2), 4→1
362	SubnGet	PortInfo	3	PortState	1→4, (4,3,1)	**PortState=DOWN**	(1,2,2), 4→1
363	SubnGet	PortInfo	4	PortState	1→4, (4,3,1)	PortState=DOWN	(1,2,2), 4→1

Tras la detección de la desactivación del conmutador *3*, el SM descartará (dejará de esperar) los SMPs de sondeo enviados por el proceso periódico de barrido a los conmutadores *3*, *5*, *6* y *10*. Para sondear el estado de los conmutadores *6*, *10* y *5*, que han quedado lógicamente desconectados, el SM envía en primer lugar un SMP de sondeo (transacción 351) con un segmento inicial hasta el conmutador *4*, seguido de una ruta dirigida de un solo salto (el puerto *4* en el dicho conmutador *4*). Tras recibir la correspondiente respuesta, inyecta un segundo SMP de sondeo (357) con el mismo segmento inicial, pero empleando una ruta dirigida un salto más larga (el puerto *4* en el conmutador *4* y el puerto *3* en el conmutador *6*). Final-

mente, se generará un tercer SMP de comprobación (transacción 358) para el conmutador *5*, usando los puertos de salida *4*, *3* y *1* en los conmutadores *4*, *6* y *10*.

5.5 Evaluación comparativa

En esta sección mostramos una serie de resultados que nos permitirán analizar comparativamente el comportamiento de las dos estrategias de descubrimiento que acabamos de describir. Todos los resultados han sido obtenidos empleando técnicas de simulación.

5.5.1 Metodología de simulación

El modelo de la arquitectura InfiniBand empleado y la metodología de simulación general han sido presentados en el Capítulo 4 de esta memoria. Por lo tanto, comentaremos aquí únicamente ciertos aspectos específicos en la preparación de este estudio.

Los resultados presentados han sido obtenidos sin considerar tráfico de aplicación en la red. Sin embargo, debido a la existencia de un canal virtual dedicado para el tráfico de gestión de la subred (VL15), los resultados en presencia de tráfico son prácticamente idénticos.

Para cada simulación, hemos programado un cambio topológico en la subred, en tiempo de simulación 60.1 s. Dicho cambio consiste en la activación o desactivación de uno o varios conmutadores. La simulación finaliza cuando el cambio es totalmente asimilado. Durante la simulación, hemos medido el tiempo de exploración, y la cantidad de paquetes de gestión (SMPs) requeridos por los mecanismos de descubrimiento.

En caso de activación o desactivación individual, el experimento es repetido para cada uno de los conmutadores de la subred, salvo para el conmutador que aloja al SM. Los resultados mostrados en las gráficas son valores medios. Por el contrario, cuando el cambio consiste en la activación o desactivación simultánea de múltiples conmutadores, cada punto en las gráficas representa una sola simulación.

Finalmente, el mecanismo de detección de cambios empleado en las simulaciones no afecta a los resultados mostrados. Esto se debe a que las estadísticas que nos interesan (tiempo y paquetes de exploración) comienzan a ser recogidas una vez que el cambio ha sido detectado.

5.5.2 Tiempos de procesamiento de SMPs

Para comenzar la evaluación, la Figura 72 muestra el tiempo empleado por el SM para procesar cada SMP recibido durante el proceso de descubrimiento, en función del tamaño de la subred y considerando las dos técnicas presentadas. En la sección 4.3.5 se detalla la forma en la que han sido obtenidos estos resultados.

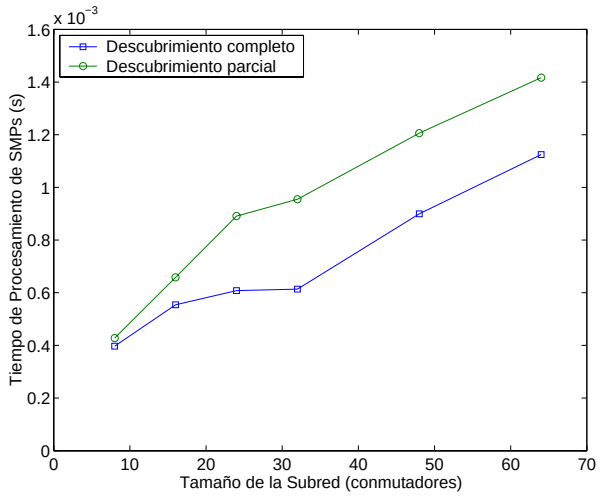


Figura 72. Tiempo medio requerido para procesar un SMP de descubrimiento.

Obviamente, el tratamiento de cada SMP requiere más tiempo en el caso del protocolo de descubrimiento parcial, debido a que la complejidad de las tareas asociadas es mayor que en el caso del descubrimiento completo. Sin embargo, como veremos en las secciones siguientes, este efecto se compensa con una reducción en el número de SMPs de exploración y el empleo de segmentos iniciales con encaminamiento basado en destino.

5.5.3 Cambio individual

La Figura 73 presenta el tiempo y SMPs de exploración requeridos por los dos mecanismos de descubrimiento evaluados, considerando un cambio consistente en la activación (a la izquierda) o desactivación (a la derecha) de uno de los conmutadores de la subred.

Podemos ver que el mecanismo de descubrimiento parcial obtiene mejores resultados en ambos casos. Además, la mejora se acentúa a medida que aumenta el tamaño de la subred.

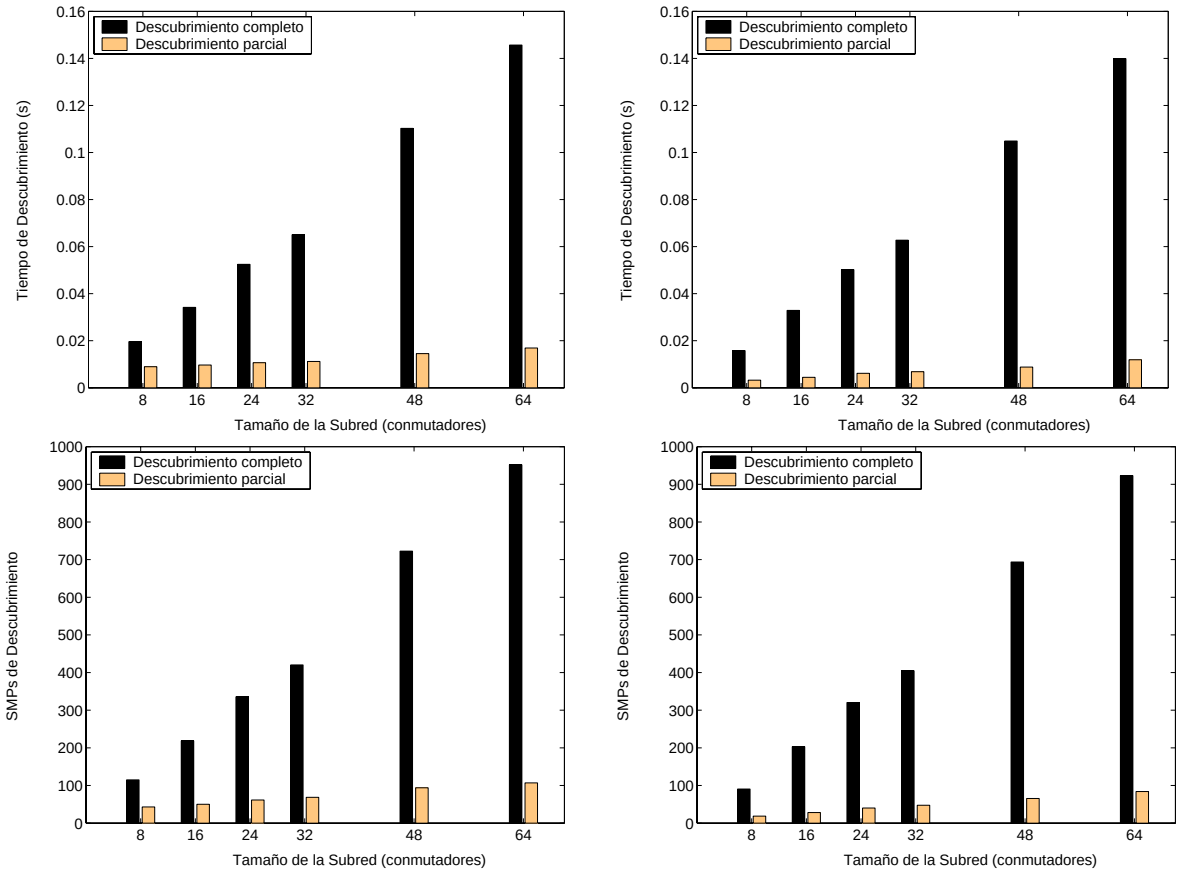


Figura 73. Tiempo y SMPs de exploración en función del tamaño de la subred (cambio individual).

La Tabla 16 muestra los tiempos medios de descubrimiento representados en la Figura 73, junto con la desviación estándar asociada a cada una de las medias.

Tabla 16. Valores medios y desviaciones estándar para los tiempos de descubrimiento.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Mecanismo de descubrimiento**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Completo	0.019588	0.000515	0.015764	0.001835
	Parcial	0.008969	0.001439	0.003229	0.002005
16	Completo	0.034173	0.000523	0.032886	0.003246
	Parcial	0.009692	0.001648	0.004448	0.001729
24	Completo	0.052466	0.000795	0.050199	0.002636
	Parcial	0.010668	0.001102	0.006157	0.002054
32	Completo	0.065045	0.001060	0.062717	0.004423
	Parcial	0.011206	0.001956	0.006832	0.003070
48	Completo	0.109690	0.001757	0.105454	0.005129
	Parcial	0.013631	0.003707	0.008131	0.003752
64	Completo	0.144177	0.002684	0.138611	0.004768
	Parcial	0.015540	0.004144	0.009729	0.004609

Finalmente, la Tabla 17 muestra los valores medios y las desviaciones estándar para los resultados relativos a paquetes de descubrimiento.

Tabla 17. Valores medios y desviaciones estándar para los paquetes de descubrimiento.

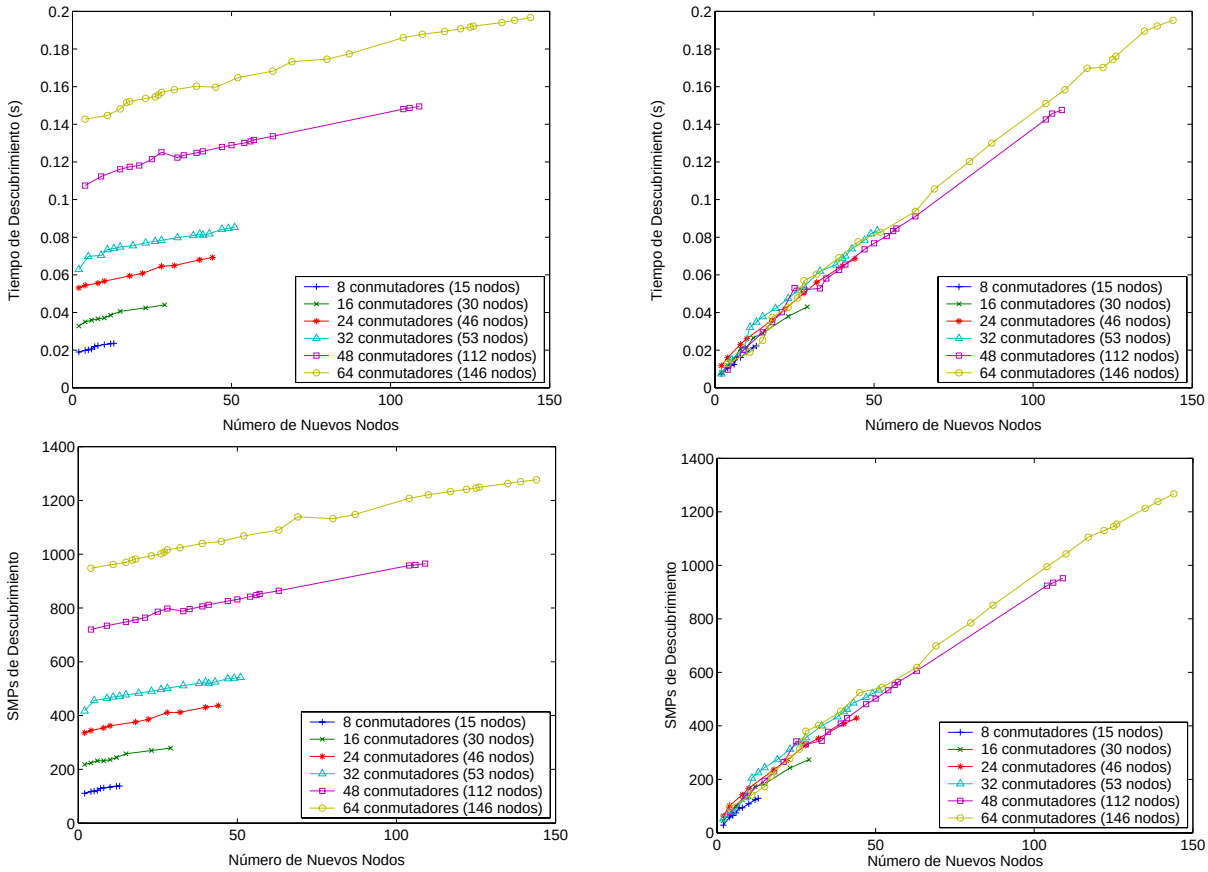
		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Mecanismo de descubrimiento**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Completo	114.86	3.31	90.43	8.48
	Parcial	42.86	11.57	18.57	4.44
16	Completo	218.67	1.74	203.07	6.14
	Parcial	50.00	8.35	28.00	4.00
24	Completo	335.39	1.24	320.26	11.27
	Parcial	61.52	6.63	40.22	4.33
32	Completo	420.23	2.01	404.52	13.73
	Parcial	68.55	9.45	47.55	8.37
48	Completo	722.66	8.71	700.19	24.99
	Parcial	91.94	20.68	64.64	9.21
64	Completo	952.29	12.06	922.63	23.09
	Parcial	109.84	20.69	81.22	11.57

5.5.4 Cambio múltiple

La Figura 74 muestra el tiempo y SMPs de descubrimiento en función de la cantidad de nodos (conmutadores y nodos terminales) que han aparecido en la subred tras un cambio consistente en la activación simultánea de varios conmutadores. A la izquierda se muestran los resultados para descubrimiento completo y a la derecha los resultados para descubrimiento parcial.

En las gráficas, cada serie corresponde a una topología de subred diferente. En todos los casos, todos los nodos de la subred quedan activados tras el cambio. Esto significa que el punto de la izquierda de cada serie refleja la situación en la cual toda la subred está activada, a excepción de uno de los nodos. Entonces, el cambio consiste en la activación de ese nodo. Por otro lado, el punto de la derecha de cada serie corresponde al caso en el que sólo el nodo que aloja el SM está activo, consistiendo el cambio en la activación del resto de la subred.

Si comparamos los puntos de la derecha de cada serie, podemos concluir que no hay diferencia entre los dos mecanismos de descubrimiento cuando el cambio afecta a la subred completa. Éste es el caso, por ejemplo, de la activación inicial de la subred. Sin embargo, la mejora obtenida por el descubrimiento parcial comienza a apreciarse a medida que la porción de la subred que permanece sin cambios crece. El mejor caso corresponden con los puntos de la izquierda de las series, que fueron mostrados por separado en la Figura 73(a).



(a) Descubrimiento completo

(b) Descubrimiento parcial

Figura 74. Tiempo y SMPs de exploración en función del número de nodos que aparecen en la subred tras una activación múltiple.

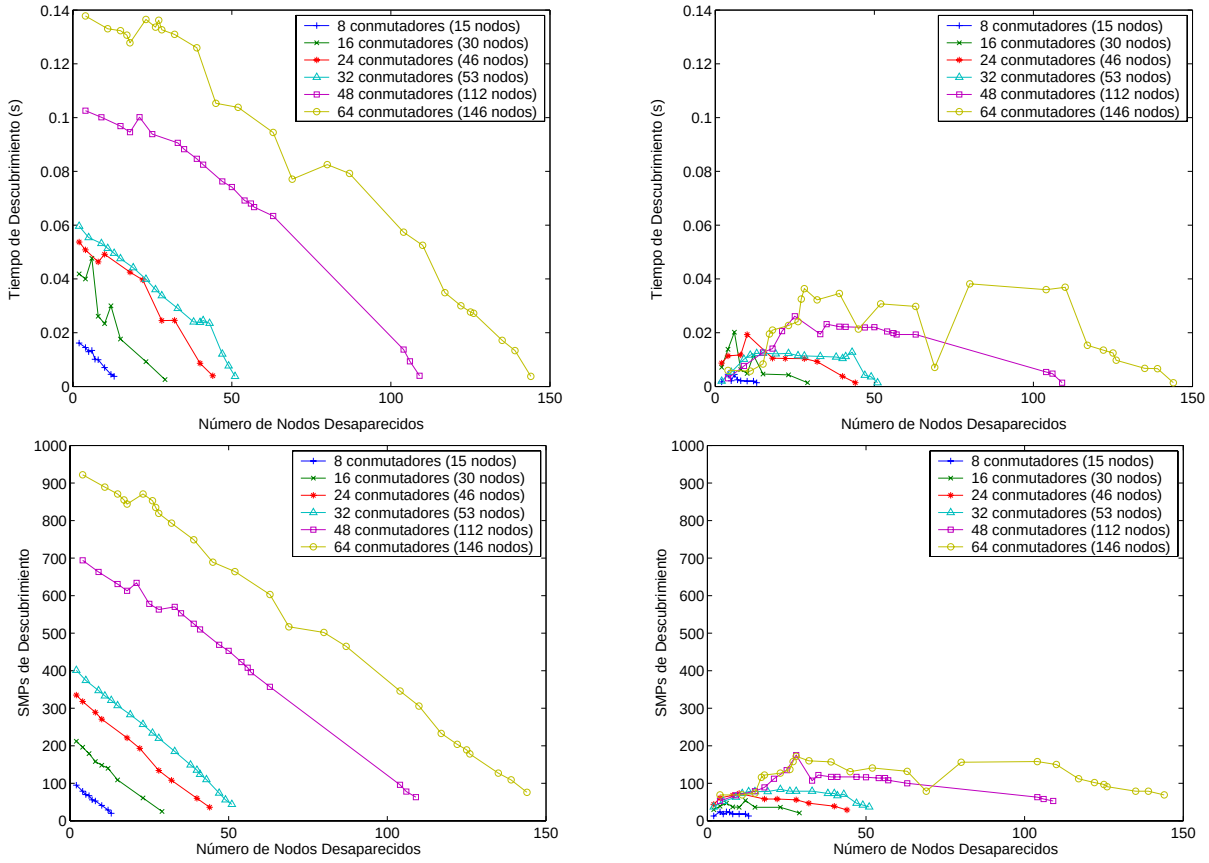
La Figura 75 muestra los mismos resultados que la Figura 74, pero considerando la desactivación simultánea de varios conmutadores en la subred. En todos los casos, la subred completa se encuentra activa antes de la ocurrencia del cambio. El valor representado en el eje horizontal es el número de nodos (conmutadores y nodos terminales) que desaparecen como consecuencia del cambio múltiple.

En la columna de la izquierda de la Figura 75, que muestra los resultados para el mecanismo de descubrimiento completo, podemos observar que el tiempo y el número de SMPs requeridos para asimilar el cambio disminuyen a medida que el número de nodos desaparecidos crece. Esto se debe a que la subred "resultante" es menor que la original. Nótese también que las series mostradas en la Figura 75(a) y en la Figura 74(a) parten de valores muy similares.

El comportamiento del mecanismo de descubrimiento parcial (columna derecha) es bastante distinto. En primer lugar, cuando el número de nodos que desaparecen es pequeño, el mecanismo obtiene resultados muy buenos, si los comparamos con el descubrimiento completo. En concreto, el punto de la izquierda corresponde al caso en el que sólo se desactiva un

nodo, mostrado en la Figura 73(b). En este caso, sólo serán necesarios unos pocos SMPs de sondeo, con encaminamiento dirigido, pero con segmentos iniciales relativamente largos.

Sin embargo, a medida que el número de nodos que desaparece crece, la mejora obtenida por el descubrimiento parcial decrece. El motivo es que se necesitan más SMPs para realizar el sondeo individual de los conmutadores. Además, las rutas dirigidas serán más largas (tendrán más saltos) que antes. Afortunadamente, este efecto se compensa por el hecho de que el tamaño de la subred resultante disminuye al mismo tiempo que crece el número de nodos que desaparecen, siendo los resultados para el descubrimiento parcial mejores en casi todos los casos. Sólo los puntos de la derecha de cada serie (cuando toda la subred menos uno de los nodos es desactivada) son muy similares para ambos mecanismos.



(a) Descubrimiento completo

(b) Descubrimiento parcial

Figura 75. Tiempo y SMPs de exploración en función del número de nodos que desaparecen en la subred tras una desactivación múltiple.

5.6 Conclusiones

En este capítulo hemos estudiado la eficiencia de los dos mecanismos de detección de cambios topológicos previstos en InfiniBand, y su impacto sobre el tráfico de usuario. Hemos comprobado que el mecanismo de notificaciones es siempre más rápido que el mecanismo de barridos periódicos. Por otro lado, hemos visto que la frecuencia con que el mecanismo de barrido comprueba la topología no afecta de manera determinante a las prestaciones ofrecidas por la red.

También se han presentado y evaluado dos mecanismos para el descubrimiento de la topología en una subred InfiniBand. Uno de ellos obtiene nuevamente el mapa de la subred cada vez que se produce un cambio topológico en la misma. El otro aprovecha la información topológica anterior, añadiendo nuevos dispositivos a la base de datos topológica en caso de activación, y descartando aquella información de la base de datos que ha quedado obsoleta en caso de desactivación.

En base a los resultados mostrados, podemos concluir que el proceso de descubrimiento parcial necesita menos tiempo y paquetes de control para asimilar la nueva topología. Los resultados confirman que la diferencia entre las dos estrategias de exploración es mayor a medida que el tamaño de la subred crece y a medida que el cambio afecta a una porción menor de la misma.

Capítulo 6
Cálculo de rutas

6.1 Introducción

Una vez actualizado el mapa de la topología, el siguiente paso en el proceso de asimilación de un cambio consiste en el cálculo de un conjunto de rutas para el encaminamiento de paquetes dentro de la subred. Las rutas obtenidas serán posteriormente distribuidas a los conmutadores en forma de entradas en tablas de encaminamiento.

El tiempo de cálculo de rutas es un factor crítico dentro del proceso de asimilación completo. Un algoritmo de cómputo de rutas que requiera demasiado tiempo no es en absoluto deseable, aún cuando el conjunto de rutas obtenido sea óptimo. Alargar el proceso de cómputo significa que los conmutadores seguirán empleando durante más tiempo unas tablas que no reflejan la nueva situación topológica.

Tras una introducción al algoritmo de encaminamiento *up*/down**, presentaremos dos algoritmos para el cálculo de rutas *up*/down** libres de bloqueo. La principal diferencia entre las dos propuestas radica en el número de entradas que deben ser explícitamente computadas. Posteriormente, veremos que al disminuir el número de entradas a calcular se reduce el tiempo de cálculo total. A continuación propondremos un mecanismo de asimilación de cambios basado en la distribución previa de un conjunto de rutas provisionales no óptimas, pero rápidamente computables. Finalmente, se mostrarán los beneficios derivados del uso del nuevo mecanismo de gestión.

6.1.1 Encaminamiento up*/down* en InfiniBand

Nuestros mecanismos de gestión emplean *up*/down** [80] para calcular las rutas que seguirán los paquetes dentro de la subred. *Up*/down** es un popular algoritmo de encaminamiento libre de bloqueos, adecuado para topologías regulares e irregulares.

En *up*/down** hay dos reglas generales que proporcionan rutas libres de bloqueo para cualquier topología. La primera consiste en la construcción de un grafo dirigido acíclico con un solo nodo sumidero [20]. A cada enlace de la red se le asigna una dirección, identificada por una flecha, de forma que si el enlace se recorre en el sentido de la flecha, hablamos de la dirección *up*, y si se recorre en el sentido contrario, hablamos de la dirección *down*. Como ejemplo, la Figura 76(a) muestra la subred de la Figura 67 (pero con el SM alojado en el conmutador *1*). La Figura 76(b) muestra una posible asignación de direcciones para esta topología. En este caso concreto, las direcciones han sido asignadas en función de la distancia al SM de los nodos en los extremos de los enlaces.

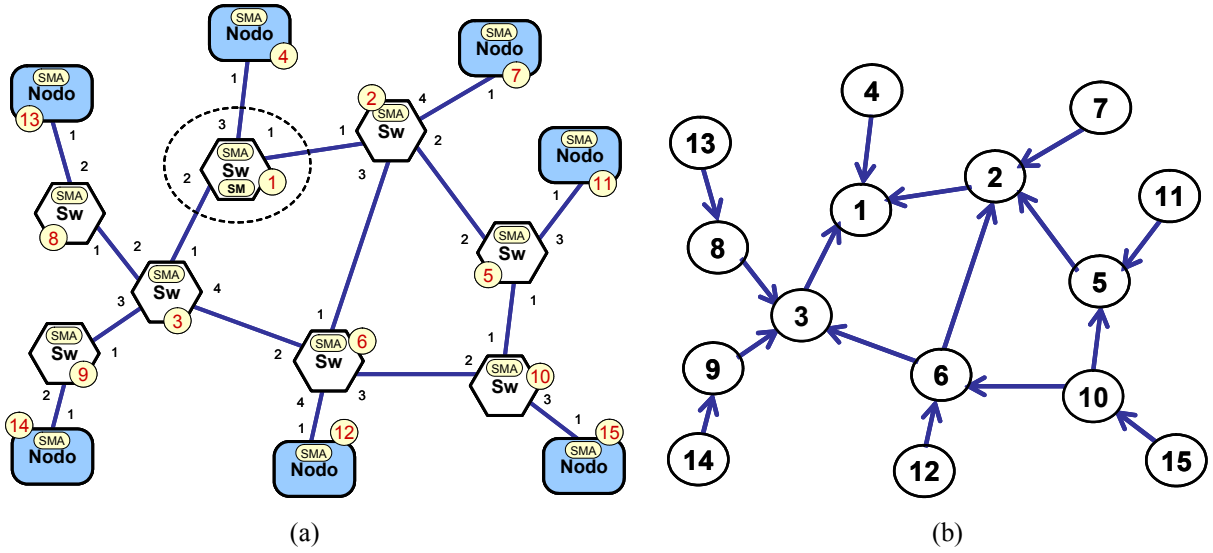


Figura 76. Grafo dirigido asociado a una subred con topología irregular.

En la literatura pueden encontrarse numerosos mecanismos para la obtención del grafo dirigido. Por ejemplo, la primera versión del mecanismo de reconfiguración de Autonet [80] obtenía en primer lugar un árbol de expansión de mínima profundidad (MDST, *Minimum Depth Spanning Tree*), mientras que en la segunda versión [72] el árbol de expansión de la topología se obtenía en orden de propagación (POST, *Propagation-Order Spanning Tree*). Por otro lado, en [77] se propone la construcción del árbol de expansión mediante una búsqueda primero en profundidad (DFS, *Depth First Search*). Como se ha detallado en el capítulo anterior, en nuestro caso el grafo dirigido se obtiene mediante una optimización del mecanismo POST.

Para evitar la aparición de bloqueos, la segunda regla *up*/down** establece que una ruta legal dentro de la red nunca emplea un enlace en la dirección *up* después de haber usado otro en la dirección *down*. Es decir, no se permite realizar transiciones *down-up*. En la Figura 76, un ejemplo de ruta legal desde el nodo terminal *7* al nodo terminal *14* podría ser *7→2→1←3←9←14*. Sin embargo, la ruta *7→2←6→3←9←14* no sería legal, debido a que tras usar un enlace en dirección *down* para alcanzar el conmutador *6*, emplea un enlace en dirección *up* para alcanzar el conmutador *3*.

Desafortunadamente, la aplicación del algoritmo de encaminamiento up*/down* en redes InfiniBand no es inmediata. Esto se debe a que las tablas de encaminamiento de los conmutadores no tienen en cuenta el puerto de entrada del paquete, sólo su destinatario (ver Sección 2.5.1). Rellenar las tablas de encaminamiento sin considerar el puerto de entrada puede conducir a la aparición de situaciones de bloqueo en la red.

En la actualidad, ya existen diversas propuestas para obtener rutas válidas *up*/down** para InfiniBand [23, 44, 75, 78]. Por ejemplo, en el mecanismo *destination renaming* [44], la información sobre el puerto de entrada es almacenada en el propio campo DLID del paquete. En concreto, cuando un conmutador proporciona dos caminos diferentes hacia un mismo destinatario en función del canal de entrada, dicho destinatario es renombrado para uno de esos caminos (haciendo uso del LMC). Como el destino de ambos caminos es distinto desde el punto de vista de la subred, ahora el camino correcto puede seleccionarse en función del DLID, sin necesidad de considerar el canal de entrada. De esta forma, el nodo emisor que use el camino modificado deberá enviar los paquetes al identificador renombrado asociado al destino.

6.2 Algoritmos para la obtención de rutas

A continuación describiremos y evaluaremos dos algoritmos de cálculo de rutas *up*/down** para el encaminamiento en subredes. El primero de ellos obtiene una entrada en cada tabla de encaminamiento para cada posible destinatario en la subred. La segunda propuesta aprovecha la existencia de puertos por defecto en los conmutadores InfiniBand para reducir la cantidad de entradas a computar.

6.2.1 Definiciones y suposiciones de partida

Sea $G(N,A)$ el grafo dirigido *up*/down** asociado a la topología de la subred. N es el conjunto de nodos (conmutadores y nodos terminales) en la subred. A es el conjunto de arcos que representan los enlaces entre nodos. La dirección de cada arco se corresponde con la dirección *up*/down** asignada al enlace correspondiente. La expresión (n_x,n_y) hace referencia al arco en A desde el nodo n_x al nodo n_y. Asumimos que si $(n_x,n_y)\in A$, entonces $(n_y,n_x)\notin A$. Por definición, G es un grafo dirigido acíclico que contiene un solo nodo sumidero.

Cada nodo tiene un conjunto de puertos que lo conectan (mediante enlaces) con otros nodos. Sea P el conjunto de puertos en la subred. La expresión $p_x^y \in P$ hace referencia al puerto en el nodo n_x conectado al nodo n_y. Entonces, $\forall(n_x,n_y)\in A$, p_x^y, $p_y^x \in P$. Además, todo nodo n_x tiene un puerto interno (de gestión), al que nos referiremos con $p_x \in P$. Por conveniencia, asignaremos una dirección a cada puerto (excepto a los puertos internos), que se corresponde con la dirección *up*/down** asignada al enlace conectado a dicho puerto. Sea $D(p)$ la direc-

ción de un puerto p. Si $(n_x,n_y)\in A$, entonces definimos $D(p_x^y) = DOWN$, y $D(p_y^x) = UP$, es decir, la dirección de un puerto es *down* si está conectado al extremo *down* del enlace.

Sea $FT(n_x,n_y)$ la entrada en la tabla de encaminamiento del nodo n_x que contiene el puerto de salida cuando el destinatario del paquete es el nodo n_y. Inicialmente, $\forall n_x,n_y\in N$, $FT(n_x,n_y)$ está asignado a un número de puerto inválido. Además, sea $DEF(n_x)$ el puerto por defecto en el nodo n_x. Como antes, inicialmente, $\forall n_x\in N$, $DEF(n_x)$ es un puerto inválido.

6.2.2 Algoritmo *Fully Explicit Routing* (FERa)

Para generar las rutas a través de la subred, este algoritmo computa todas las entradas en las tablas de encaminamiento. Para describir el funcionamiento del algoritmo, consideremos la situación mostrada en la Figura 77. Para cada nodo *n1* en la subred, se consideran el resto de nodos. Para ello, se lleva a cabo una inundación controlada (marcada como ↷ en la figura), impidiendo aquellos saltos que supongan una transición prohibida *down-up* (por ejemplo, la secuencia *n3*↷*n2*↷*n4*). Durante la inundación, se añade una nueva entrada en la tabla de encaminamiento de cada nodo visitado, considerando el puerto empleado para alcanzarlo como el puerto de salida cuando el nodo de destino es *n1*. Además, se almacena la longitud de la ruta. Si el proceso encuentra más tarde alguna otra ruta, hay que escoger entre la nueva ruta y la anterior, ya que el encaminamiento en InfiniBand es determinista. El criterio que hemos empleado consiste en seleccionar la ruta más corta, o una ruta al azar en caso de empate. En la figura, el algoritmo obtendrá $FT(n2, n1) = 1$.

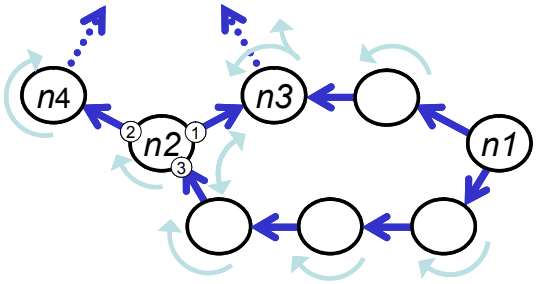


Figura 77. Cálculo de rutas mediante inundación controlada, partiendo del nodo destino *n1*.

Sin embargo, como las tablas en InfiniBand no tienen en cuenta el puerto de entrada a la hora de encaminar los paquetes, podrían aparecer situaciones de bloqueo, si sólo tenemos en cuenta las rutas más cortas y G no está balanceado. En la Figura 77, otro nodo *n4* está conectado a *n2* por medio de un enlace descendente. Este nodo encamina los paquetes dirigidos hacia *n1* a través del nodo *n2*. La consecuencia en este caso es la aparición de una transición prohibida *down-up* (*n4*←*n2*→*n3*), pues *n2* encaminará después estos paquetes hacia *n3*.

Para resolver este tipo de situaciones, el algoritmo procede de la siguiente forma. Si para un destino dado hay algún puerto de salida en dirección *down*, se ignoran todas las opciones de encaminamiento que supongan el empleo de un enlace en dirección *up*, aún cuando

éstas conduzcan a rutas más cortas. El mismo criterio se aplica en [23]. De esta forma, en el ejemplo, el algoritmo escogerá la ruta $FT(n2, n1) = 3$.

La Tabla 18 muestra la secuencia de entradas en tablas de encaminamiento calculadas por el algoritmo FERa para la subred de la Figura 76(a), asumiendo el grafo dirigido de la Figura 76(b). El algoritmo calcula exactamente 120 entradas (15 entradas por tabla).

En la columna "nuevas entradas", cada entrada se representa por medio del LID del conmutador donde la entrada es almacenada, el LID del puerto de destino (DLID), el puerto de salida a emplear para dicho destino (el valor *0* indica que el propio conmutador es el destinatario del paquete, y por tanto éste debe ser enviado al puerto interno de gestión), y finalmente la distancia al destino (entre paréntesis).

Tabla 18. Secuencia de pasos para computar las rutas *up*/down** para la subred de la Figura 76 (algoritmo FERa).

Nodo (DLID)	Nuevas entradas (LID: DLID → P_{OUT})	Nodo (DLID)	Nuevas entradas (LID: DLID → P_{OUT})	Nodo (DLID)	Nuevas entradas (LID: DLID → P_{OUT})
1	1: 1 → 0 (0) 2: 1 → 1 (1) 3: 1 → 1 (1) 5: 1 → 2 (2) 6: 1 → 1 (2) 8: 1 → 1 (2) 9: 1 → 1 (2) 10: 1 → 1 (3)	6	6: 6 → 0 (0) 2: 6 → 3 (1) 3: 6 → 4 (1) 10: 6 → 2 (1) 1: 6 → 1 (2) 5: 6 → 2 (2) 8: 6 → 1 (2) 9: 6 → 1 (2)	11	5: 11 → 3 (1) 10: 11 → 1 (2) 2: 11 → 2 (2) 1: 11 → 1 (3) 6: 11 → 1 (3) 3: 11 → 1 (4) 8: 11 → 1 (5) 9: 11 → 1 (5)
2	2: 2 → 0 (0) 1: 2 → 1 (1) 5: 2 → 2 (1) 6: 2 → 1 (1) 3: 2 → 1 (2) 10: 2 → 1 (2) 8: 2 → 1 (3) 9: 2 → 1 (3)	7	2: 7 → 4 (1) 1: 7 → 1 (2) 5: 7 → 2 (2) 6: 7 → 1 (2) 3: 7 → 1 (3) 10: 7 → 1 (3) 8: 7→ 1 (4) 9: 7 → 1 (4)	12	6: 12 → 4 (1) 2: 12 → 3 (2) 3: 12 → 4 (2) 10: 12 → 2 (2) 1: 12 → 1 (3) 5: 12 → 2 (3) 8: 12 → 1 (3) 9: 12 → 1 (3)
3	3: 3 → 0 (0) 1: 3 → 2 (1) 8: 3 → 1 (1) 9: 3 → 1 (1) 6: 3 → 2 (1) 2: 3 → 1 (2) 10: 3 → 2 (2) 5: 3 → 2 (3)	8	8: 8 → 0 (0) 3: 8 → 2 (1) 1: 8 → 2 (2) 9: 8 → 1 (2) 6: 8 → 2 (2) 2: 8 → 1 (3) 10: 8 → 2 (3) 5: 8 → 2 (4)	13	8: 13 → 2 (1) 3: 13 → 2 (2) 1: 13 → 2 (3) 9: 13 → 1 (3) 6: 13 → 2 (3) 2: 13 → 1 (4) 10: 13 → 2 (4) 5: 13 → 2 (5)
4	1: 4 → 3 (1) 2: 4 → 1 (2) 3: 4 → 1 (2) 5: 4 → 2 (3) 6: 4 → 1 (3) 8: 4 → 1 (3) 9: 4 → 1 (3) 10: 4 → 1 (4)	9	9: 9 → 0 (0) 3: 9 → 3 (1) 1: 9 → 2 (2) 8: 9 → 1 (2) 6: 9 → 2 (2) 2: 9 → 1 (3) 10: 9 → 2 (3) 5: 9 → 2 (4)	14	9: 14 → 2 (1) 3: 14 → 3 (2) 1: 14 → 2 (3) 8: 14 → 1 (3) 6: 14 → 2 (3) 2: 14 → 1 (4) 10: 14 → 2 (4) 5: 14 → 2 (5)
5	5: 5 → 0 (0) 10: 5 → 1 (1) 2: 5 → 2 (1) 1: 5 → 1 (2) 6: 5 → 1 (2) 3: 5 → 1 (3) 8: 5 → 1 (4) 9: 5 → 1 (4)	10	10: 10 → 0 (0) 5: 10 → 1 (1) 6: 10 → 3 (1) 2: 10 → 2 (2) 3: 10 → 4 (2) 1: 10 → 1 (3) 8: 10 → 1 (3) 9: 10 → 1 (3)	15	10: 15 → 3 (1) 5: 15 → 1 (2) 6: 15 → 3 (2) 2: 15 → 2 (3) 3: 15 → 4 (3) 1: 15 → 1 (4) 8: 15 → 1 (4) 9: 15 → 1 (4)

La Figura 78 muestra el orden en que se han explorado los nodos de la subred para calcular las entradas correspondientes al DLID *11*.

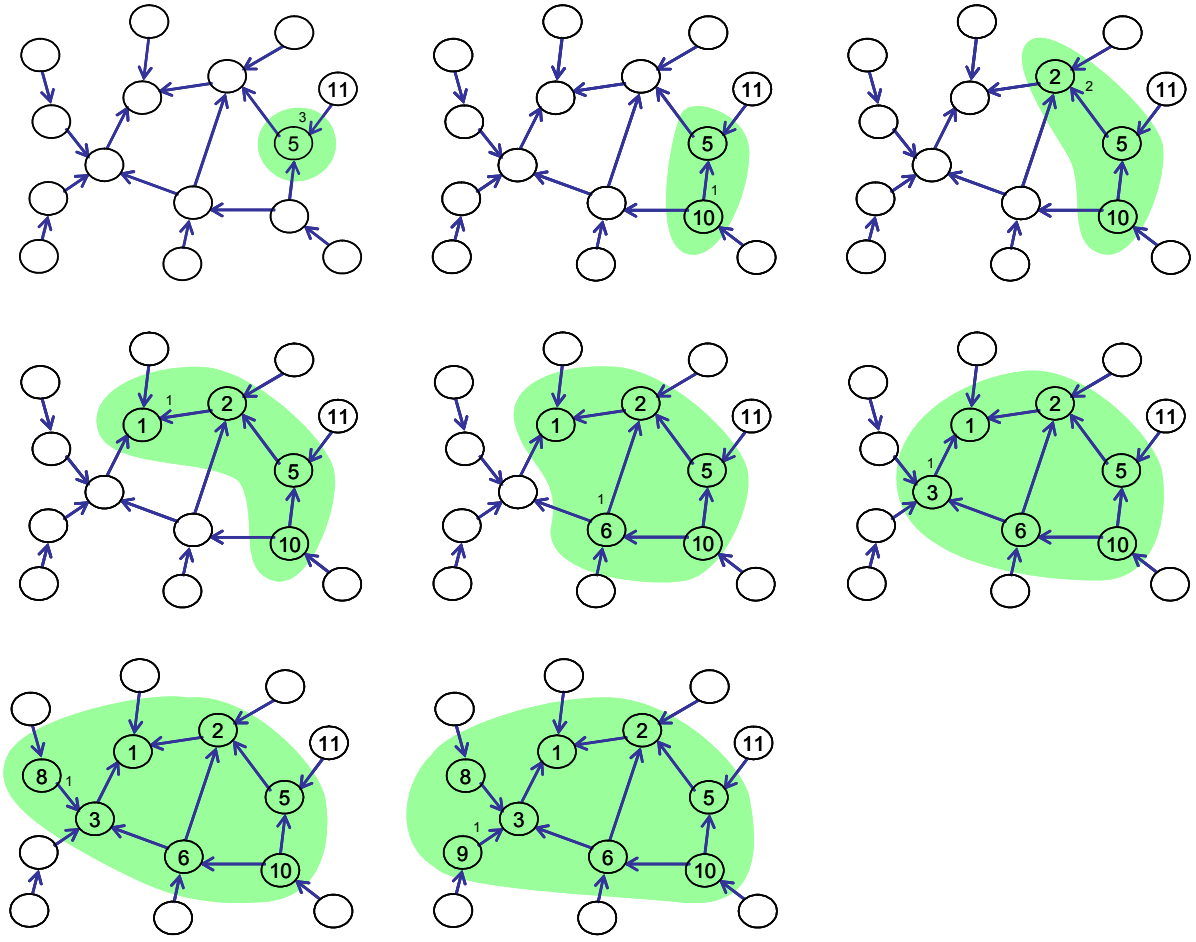


Figura 78. Obtención de entradas para el DLID *11*.

Definición formal

A continuación daremos una definición formal del algoritmo FERa. Consideramos en primer lugar un nodo de destino n_d, y después aplicamos el algoritmo a todos los nodos de la subred.

Sea R^i: $P \times enteros$ el conjunto de puertos (y distancias) escogidas para alcanzar el nodo n_d tras el paso i del algoritmo.

Sea W^i: $P \times enteros$ el conjunto de puertos (y distancias) esperando para ser analizados tras el paso i del algoritmo.

Paso 1

$R^1 = \varnothing$.

$W^1 = \{(p_n^d,1)\}$, $\forall p_n^d \in P$, es decir, W^1 incluye a todos los nodos vecinos de n_d.

Paso *i*

Considerar un $(p_x^y,\alpha)\in W^{i-1}$.

Si $\exists(p_x^z,\beta)\in R^{i-1}$ con $\beta\leq\alpha$ entonces $R^i = R^{i-1}$. Si $\beta>\alpha$ entonces la nueva opción es mejor que la anterior, y $R^i = R^{i-1}-\{(p_x^z,\beta)\}\cup\{(p_x^y,\alpha)\}$.

Si no $\exists(p_x^z,\beta)\in R^{i-1}$, entonces esta opción es directamente seleccionada, y $R^i = R^{i-1}\cup\{(p_x^y,\alpha)\}$.

Si $(p_x^y,\alpha)\notin R^i$ entonces el nodo analizado n_x no ha sido seleccionado y $W^i = W^{i-1}-\{(p_x^y,\alpha)\}$. En otro caso, su tabla de encaminamiento debe ser actualizada y $FT(n_x,n_d) = p_x^y$. Además, cada nodo vecino que no participe en una transición *down-up* debería ser analizado. Es decir, $\forall p_n^x\in P$ tal que $D(p_x^y) = UP$ o $D(p_x^n) = UP$, $W^i = W^{i-1}-\{(p_x^y,\alpha)\}\cup\{(p_n^x,\alpha+1)\}$.

Demostración formal

A continuación probaremos formalmente que el algoritmo FERa converge (explora la subred completa), y genera un conjunto de rutas colectivamente conectadas (proporcionan una ruta para cada par de nodos en la subred) y libres de bloqueo.

Lema 1 (Convergencia) $\forall n_d\in N$ considerado nodo destino, $\exists j>0$ tal que $W^j = \varnothing$.

Demostración: Para cada nodo destino n_d considerado, en cada paso del algoritmo seleccionamos un elemento cualquiera $(p_x^y,\alpha)\in W$ que se elimina de dicho conjunto. Por otra parte, para añadir nuevos elementos a W es necesario que el puerto p_x^y ofrezca una ruta más corta entre n_x y n_d que las contempladas con anterioridad, lo que puede suceder una cantidad de ocasiones limitada. Por tanto, en un número finito de pasos, el conjunto W queda vacío. □

Lema 2 (Conectividad) El algoritmo FERa proporciona una ruta entre cada par de nodos.

Demostración: Demostraremos que hay conectividad entre todos los nodos ya explorados. Procederemos por inducción, demostrando la conectividad de R^1, R^2,…

$R^1 = \varnothing$. En este caso no hacen falta rutas.

$R^2 = \{ (p_v^d,1) \}$, donde n_v es un vecino del nodo destino y la ruta entre ambos emplea el arco que los conecta.

Supongamos la conectividad de R^{i-1}. Tenemos que probar la conectividad de R^i. Si $R^i = R^{i-1}$ la conectividad no varía.

Si añadimos un elemento (p_x^y,α) a R^i es debido a que $\exists(p_y^z,\alpha-1)\in R^{i-1}$. Como existe una ruta conexa en R^{i-1} entre n_y y n_d, y el enlace entre n_x y n_y no genera un corte con dicha ruta, la conectividad queda garantizada en R^i. □

Lema 3 (Libertad de bloqueos) El conjunto de rutas generadas por el algoritmo FERa es libre de bloqueo.

Demostración: Teniendo en cuenta que *G* es un grafo dirigido acíclico y que el encaminamiento *up*/down** es libre de bloqueo, bastará con demostrar que el algoritmo no genera ninguna transición prohibida *down-up*.

Procederemos por reducción al absurdo. Supongamos que existe una transición prohibida *down-up*. Expresaremos los nodos implicados como $n_y \leftarrow n_x \rightarrow n_z$, donde las flechas indican las direcciones asignadas a los enlaces que conectan los nodos. También supondremos que la ruta ilegal parte de n_z hacia n_y. Entonces, al considerar en el Paso *i* el elemento $(p_x^y, \alpha) \in W^{i-1}$, nunca se hubiera incluido el elemento $(p_z^x, \alpha+1)$ en W^i, pues $D(p_x^y) = DOWN$ y $D(p_x^z) = DOWN$, lo que es contrario a la definición. □

6.2.3 Algoritmo *Partially Implicit Routing* (PIRa)

Como se ha descrito en la sección anterior, el algoritmo FERa calcula una ruta individual para cada par de nodos dentro de la subred. En esta sección proponemos un algoritmo alternativo que reduce la cantidad de entradas a computar y, como consecuencia, el tiempo global de cálculo de rutas.

Para conseguir su objetivo, el algoritmo PIRa agrupa varias rutas en cada conmutador aprovechando la existencia del puerto por defecto. Como se describió en la Sección 2.5.2, el valor del puerto por defecto se emplea cuando la búsqueda de un DLID en la tabla de encaminamiento no devuelve ningún resultado[1].

La idea del algoritmo es la siguiente. Si consideramos el grafo dirigido que modela la subred, cualquier ruta válida *up*/down** incluirá cero o más enlaces ascendentes (es decir, en dirección *up*) seguidos de cero o más enlaces descendentes. Como todos los segmentos ascendentes convergen finalmente en el nodo sumidero del grafo dirigido, podemos definirlos empleando el puerto por defecto en cada conmutador. Por otro lado, los segmentos descendentes se definen explícitamente, mediante entradas en las tablas de encaminamiento.

El algoritmo requiere una exploración del grafo dirigido partiendo del nodo sumidero. En cada iteración, el proceso toma un nodo *n1* tal que todos sus enlaces ascendentes están conectados a vecinos ya visitados. Si *n1* es un conmutador, se establece como puerto por defecto en *n1* el puerto que lo conecta con alguno de los nodos previamente visitados (al que nos referiremos como *n2*). A continuación, el proceso obtiene una nueva entrada en la tabla de *n1* para cada uno de sus vecinos directos ya visitados (alcanzables a través de enlaces ascenden-

[1] El puerto por defecto sólo está disponible cuando el conmutador implementa una tabla de encaminamiento de tipo RFT. Sin embargo, esto no supone ninguna restricción, ya que si el conmutador implementa una tabla de tipo LFT, el puerto por defecto puede "simularse" fácilmente, programando adecuadamente aquellas entradas (DLIDs) para las que no se ha definido un puerto de salida.

tes). Finalmente, debe añadirse una entrada en la tabla de cada uno de los conmutadores visitados *n3*, considerando como puerto de salida hacia *n1* el mismo puerto empleado para alcanzar *n2*, excepto cuando *n3* y *n1* son vecinos directos. Obsérvese que no es necesario añadir una nueva entrada para un conmutador cuando el puerto de salida escogido coincide con el puerto por defecto del conmutador. En particular, cuando *n3* no dispone de una entrada para alcanzar *n2* (es decir, usa el puerto por defecto), no se añade una nueva entrada en su tabla.

La Tabla 19 muestra la secuencia de entradas computadas por el algoritmo PIRa para la subred de la Figura 76(a), asumiendo el grafo dirigido mostrado en laFigura 76(b). Además de los 7 puertos por defecto, el algoritmo obtiene exactamente 50 entradas en tablas de encaminamiento. Se trata de una cantidad pequeña si la comparamos con las 120 entradas calculadas por el algoritmo FERa para el mismo escenario.

Tabla 19. Secuencia de pasos para computar las rutas *up*/down** para la subred de la Figura 76 (algoritmo PIRa).

LID considerado	Puerto por defecto	Nuevas entradas (LID: DLID → P_{OUT})	
1	-	1: 1 → 0	
2	1	2: 2 → 0	1: 2 → 1
3	1	3: 3 → 0	1: 3 → 2
4	n/a	1: 4 → 3	
5	2	5: 5 → 0 1: 5 → 1	2: 5 → 2
6	2	6: 6 → 0 6: 2 → 1 1: 6 → 2	2: 6 → 3 3: 6 → 4
7	n/a	1: 7 → 1 2: 7 → 4	6: 7 → 1
8	1	8: 8 → 0 1: 8 → 2	3: 8 → 2
9	1	9: 9 → 0 1: 9 → 2	3: 9 → 3
10	2	10: 10 → 0 10: 5 → 1 1: 10 → 2 2: 10 → 3	3: 10 → 4 5: 10 → 1 6: 10 → 3
11	n/a	1: 11 → 1 2: 11 → 2	5: 11 → 3 10: 11 → 1
12	n/a	1: 12 → 2 2: 12 → 3	3: 12 → 4 6: 12 → 4
13	n/a	1: 13 → 2 3: 13 → 2	8: 13 → 2
14	n/a	1: 14 → 2 3: 14 → 3	9: 14 → 2
15	n/a	1: 15 → 2 2: 15 → 3 3: 15 → 4	5: 15 → 1 6: 15 → 3 10: 15 → 3

La Figura 79 muestra gráficamente el conjunto de nodos visitados en cada paso del algoritmo.

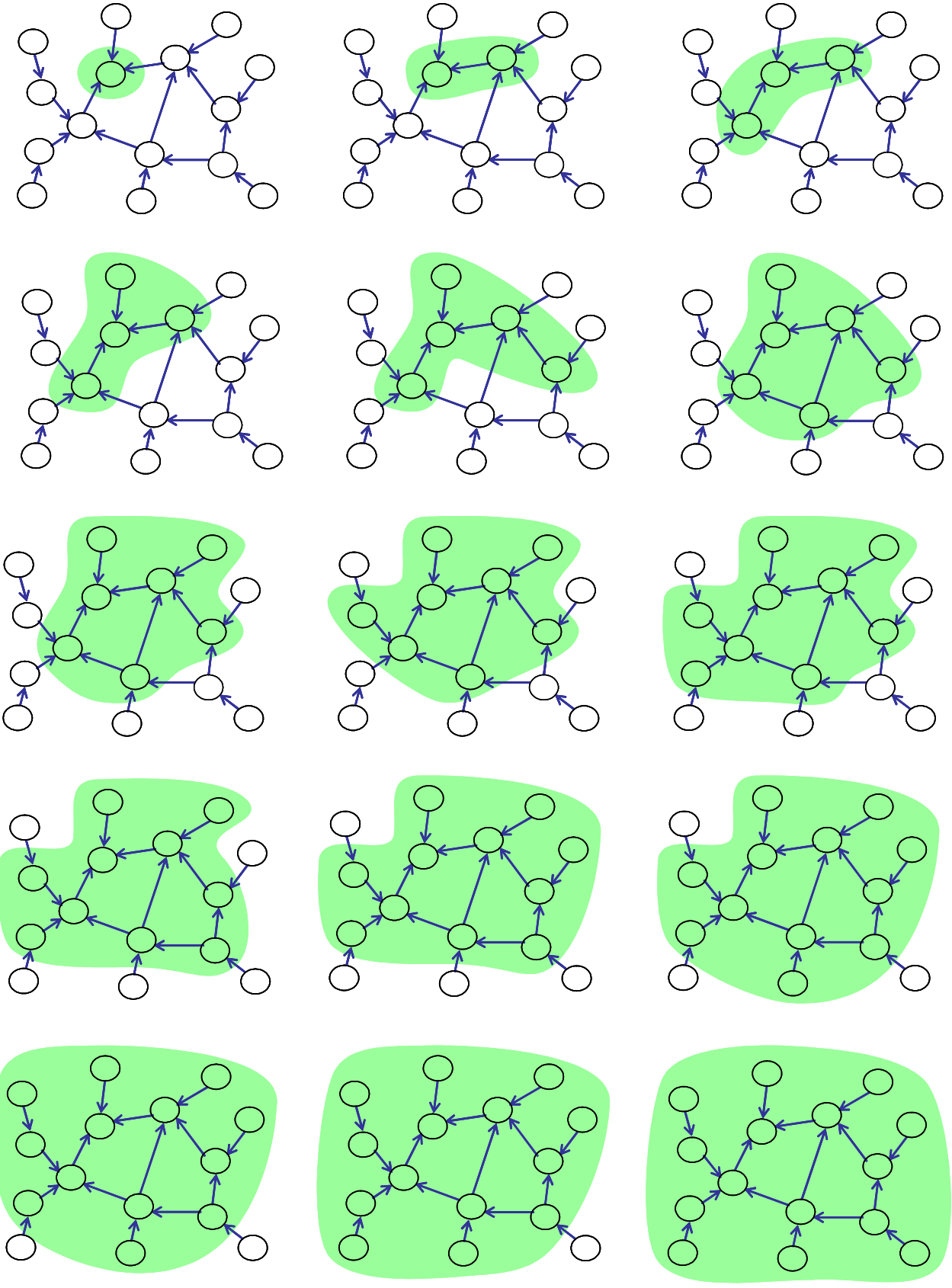


Figura 79. Secuencia de exploración.

Definición formal

Sea $PREV(n_x)$ el conjunto de vecinos de n_x unidos a dicho nodo a través de un enlace descendente. Es decir, $\forall n_y \in N$, $n_y \in PREV(n_x)$ sii $D(p_x^y) = DOWN$. Este conjunto contiene a los nodos vecinos de n_x que han sido explorados antes de considerar n_x, durante el proceso de cómputo de rutas.

Sea X^i el conjunto de nodos explorados tras el paso i del algoritmo.

Paso 1

$X^1 = \{sumidero\}$.

Paso *i*

Buscar un $n_x \notin X^{i-1}$ tal que $PREV(n_x) \subset X^{i-1}$.

Añadir n_x al conjunto de nodos explorados, es decir, $X^i = X^{i-1} \cup \{n_x\}$.

Escoger aleatoriamente un nodo $n_f \in PREV(n_x)$ y establecer $DEF(n_x) = p_x^f$. Diremos que n_f es el *padre* de n_x en el proceso de exploración.

Actualizar la tabla de encaminamiento de n_x para alcanzar a sus vecinos explorados, $\forall n_y \in PREV(n_x)$, $n_y \neq n_f$, $FT(n_x,n_y) = p_x^y$.

Actualizar las tablas de encaminamiento de los vecinos explorados de n_x, $\forall n_y \in PREV(n_x)$, $FT(n_y,n_x) = p_y^x$.

Actualizar las tablas de encaminamiento del resto de nodos explorados, $\forall n_y \in X^{i-1}$ tal que $n_y \notin PREV(n_x)$, $FT(n_y,n_x) = FT(n_y,n_f)$.

Demostración formal

A continuación probaremos formalmente que el algoritmo PIRa converge (explora la subred completa), y genera un conjunto de rutas colectivamente conectadas (proporcionan una ruta para cada par de nodos en la subred) y libres de bloqueo.

Lema 1 (Convergencia) $X^{\text{card}(N)} = N$.

Demostración: Probemos previamente que, en el Paso i del algoritmo, si $X^{i-1} \neq N$, es posible encontrar un $n_x \notin X^{i-1}$ tal que $PREV(n_x) \subset X^{i-1}$.

Procederemos por reducción al absurdo. Supongamos que $\forall n_x \notin X^{i-1}$, $PREV(n_x) \not\subset X^{i-1}$. Entonces, $\exists n_y \in PREV(n_x)$ tal que $n_y \notin X^{i-1}$. Si n_y cumple que $PREV(n_y) \subset X^{i-1}$, hemos encontrado un nodo para explorar en el Paso i y por tanto hemos terminado. En otro caso, continuamos la búsqueda, considerando n_y en lugar de n_x. Procediendo de esta forma iterativa, y teniendo en cuenta que G es finito, en un número finito de pasos encontraremos:

- Un nodo n_z que cumple que $n_z \notin X^{i-1}$ y $PREV(n_z) \subset X^{i-1}$.

- Un nodo sumidero no incluido en X^{i-1}. Esta opción contradice la suposición inicial acerca de la existencia de un único nodo sumidero en G, el cual fue incluido en X^{i-1} en la primera iteración del algoritmo.

- Un nodo previamente explorado. Esto contradice la suposición inicial acerca de la inexistencia de ciclos en G.

Por lo tanto, en cada Paso i, si $X^{i-1} \neq N$, es posible encontrar un $n_x \notin X^{i-1}$ tal que $PREV(n_x) \subset X^{i-1}$ y añadirlo al conjunto de nodos ya explorados, es decir, $X^i = X^{i-1} \cup \{n_x\}$. Tras *card(N)* pasos, tendremos $X^{card(N)} = N$. □

Lema 2 (Conectividad) El algoritmo PIRa proporciona una ruta entre cada par de nodos.

Demostración: Teniendo en cuenta el Lema 1, podemos probar la conectividad demostrando que, en el Paso i del algoritmo, hay conectividad entre todos los nodos ya explorados. Procederemos por inducción, demostrando la conectividad de X^1, X^2,…

$X^1 = \{sumidero\}$. En este caso no hacen falta rutas.

$X^2 = \{sumidero, n_1\}$, donde n_1 es un vecino del nodo sumidero. La ruta entre ambos nodos se establece, por definición, empleando el arco que los conecta.

Supongamos la conectividad de X^{i-1}. Tenemos que probar la conectividad de $X^i = X^{i-1} \cup \{n_x\}$. Para cada vecino de n_x perteneciente a X^{i-1}, se establece una ruta entre ellos a través del arco que conecta al vecino con n_x. Para el resto de nodos en X^{i-1}, el algoritmo establece conectividad desde todos ellos hacia n_x empleando la ruta ya existente hacia el *padre* de n_x (con independencia de que dicho camino emplee puertos por defecto o no), a la que se le añade el arco entre este nodo y n_x. Obsérvese que el arco desde el *padre* de n_x hacia n_x es siempre un enlace descendente, y por tanto no contradice las reglas del encaminamiento *up*/down**. De manera similar, el algoritmo establece conectividad desde n_x al resto de nodos en X^{i-1} empleado la ruta ya existente desde el *padre* de n_x hasta el nodo correspondiente (con independencia de que dicho camino emplee puertos por defecto o no), y usando el puerto por defecto en n_x para alcanzar a su *padre*. □

Lema 3 (Libertad de bloqueos) El conjunto de rutas generadas por el algoritmo PIRa es libre de bloqueo.

Demostración: Teniendo en cuenta que G es un grafo dirigido acíclico y que el encaminamiento *up*/down** es libre de bloqueo, bastará con demostrar que el algoritmo no genera ninguna transición prohibida *down-up*.

Procederemos por reducción al absurdo. Supongamos que existe una transición prohibida *down-up*. Expresaremos los nodos implicados como $n_x \leftarrow n_y \rightarrow n_z$, donde las flechas indican las direcciones asignadas a los enlaces que conectan los nodos. Por definición, en algún Paso i, n_y

fue añadido a X^i tras añadir n_x y n_z. Antes de añadir n_y a X^i, la conectividad entre n_x y n_z estaba garantizada por el Lema 2. Por lo tanto, n_x y n_z no usarán n_y para alcanzarse, y las rutas entre ellos no emplearán esta transición prohibida *down-up*. □

6.2.4 Evaluación comparativa

En esta sección evaluaremos los dos algoritmos de cómputo de tablas que acabamos de describir, mostrando el número de entradas calculadas por cada uno de ellos, el tiempo necesario para obtenerlas, y las prestaciones del conjunto de rutas obtenidas en cada caso. El modelo de InfiniBand y la metodología de simulación utilizada fueron descritos en el Capítulo 4 de esta memoria.

Número de entradas computadas

La Tabla 20 muestra el número de entradas en tablas de encaminamiento computadas por los algoritmos FERa y PIRa, considerando diversas subredes con topología irregular y distinto número de conmutadores y nodos terminales.

Tabla 20. Número de entradas calculadas por cada algoritmo.

Tamaño de la subred	FERa	PIRa
8 conmutadores (15 nodos)	120	50
12 conmutadores (23 nodos)	276	85
16 conmutadores (30 nodos)	482	122
20 conmutadores (40 nodos)	800	188
24 conmutadores (46 nodos)	1108	262
32 conmutadores (72 nodos)	2304	435
40 conmutadores (68 nodos)	2740	515
48 conmutadores (112 nodos)	5417	927
64 conmutadores (146 nodos)	9353	1220
128 conmutadores (218 nodos)	27936	2592

En la Figura 80 se representan gráficamente estos resultados. Como vemos, el algoritmo PIRa reduce de forma notable el número de entradas a computar.

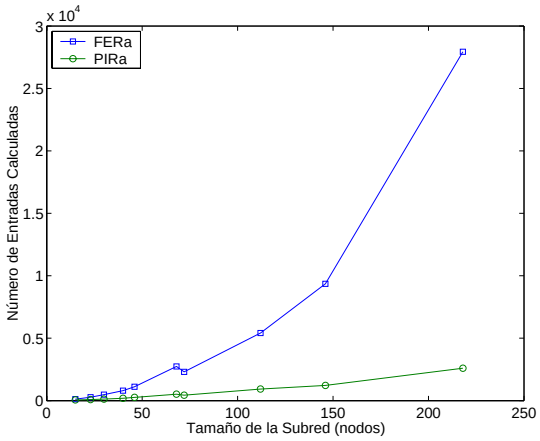
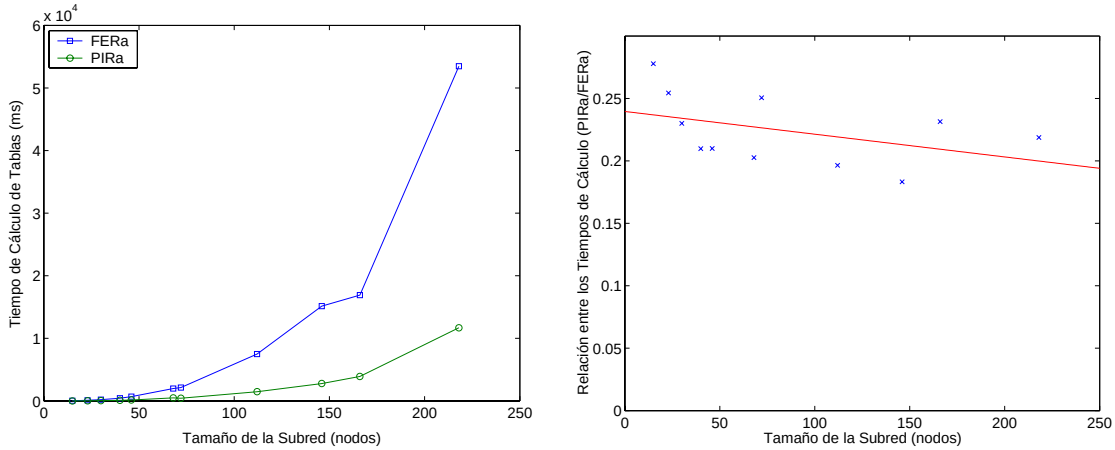


Figura 80. Número de entradas calculadas por cada algoritmo (Tabla 20).

Tiempo de cálculo de tablas

La Figura 81 muestra el tiempo de ejecución de los algoritmos FERa y PIRa, en función del tamaño de subred. La manera en la que hemos obtenido estos tiempos se detalla en la sección 4.3.5.



(a) (b)

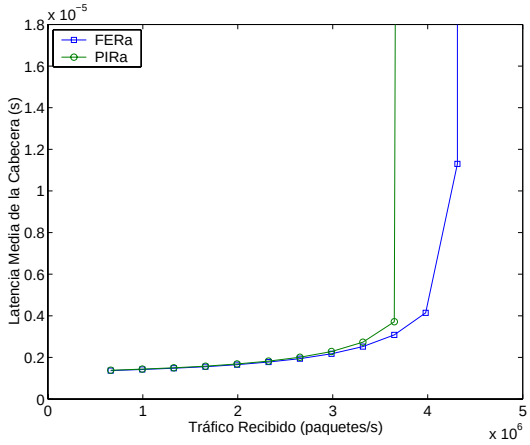
Figura 81. Tiempo requerido por los algoritmos para computar las rutas.

La Figura 81(b) nos indica que el algoritmo PIRa es en torno a 4 o 5 veces más rápido que el algoritmo FERa. Además, a medida que el tamaño de la subred crece, las diferencias son más apreciables. Esto es debido a la gran diferencia de complejidad de ambos algoritmos.

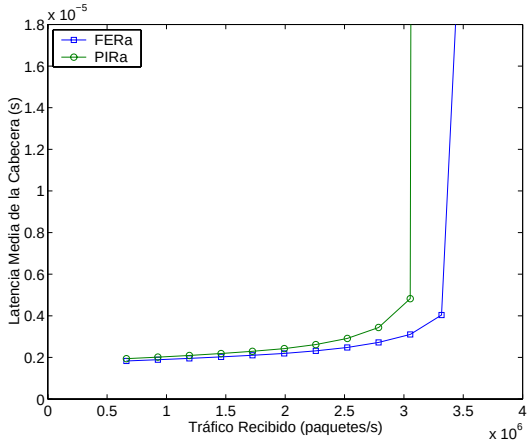
Prestaciones de las rutas obtenidas por los algoritmos

Para concluir esta evaluación comparativa, la Figura 82 presenta las prestaciones obtenidas por los conjuntos de rutas generados por cada algoritmo. En particular, se muestran las curvas de latencia para algunas subredes irregulares. Como era de esperar, las rutas obtenidas

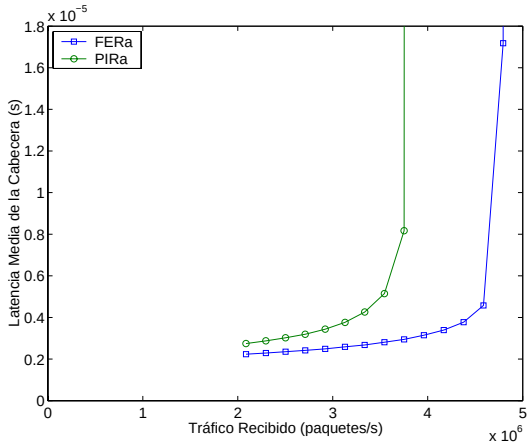
por el algoritmo PIRa ofrecen unas prestaciones ligeramente inferiores a las proporcionadas por el algoritmo FERa.



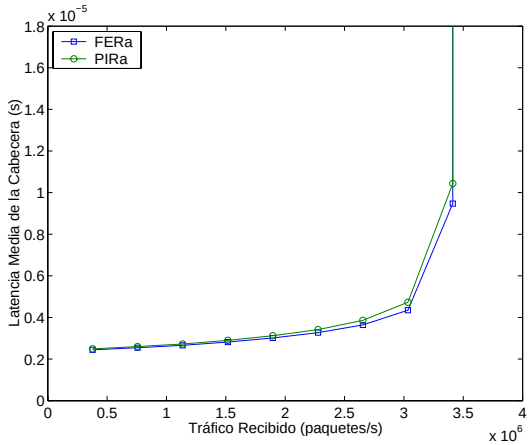
(a) 8 conmutadores (15 nodos)



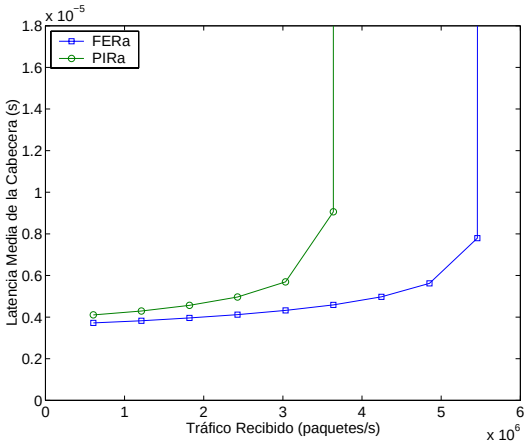
(b) 16 conmutadores (30 nodos)



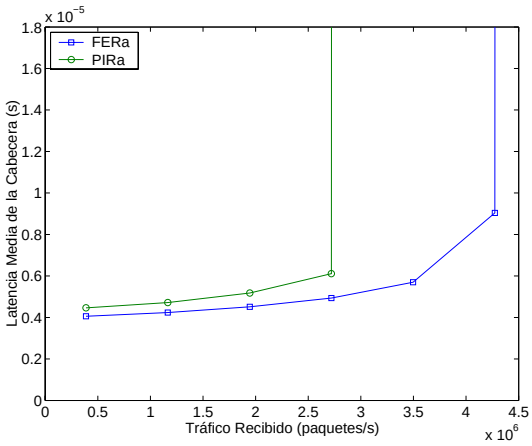
(c) 24 conmutadores (46 nodos)



(d) 32 conmutadores (72 nodos)



(d) 48 conmutadores (112 nodos)



(e) 64 conmutadores (146 nodos)

Figura 82. Prestaciones de los algoritmos de cálculo de rutas evaluados para diversas topologías.

6.3 Mecanismo de gestión basado en rutas provisionales

Los resultados de la sección anterior nos permiten derivar un protocolo optimizado para la asimilación de cambios. La idea es aprovechar la rapidez del algoritmo PIRa para distribuir en relativamente poco tiempo un conjunto de rutas provisionales acordes con la nueva topología.

6.3.1 Procedimiento de asimilación de cambios

Una vez que el mecanismo de detección de cambios descubre la ocurrencia de un cambio topológico y que la base de datos topológica es convenientemente actualizada, el SM calcula y distribuye a los conmutadores de la subred el conjunto de puertos por defecto y las entradas calculadas por el algoritmo PIRa. A partir de este momento, los conmutadores podrán comenzar a emplear este conjunto de rutas, de manera provisional.

Para evitar la aparición de bloqueos durante la distribución de las rutas provisionales, las tablas de encaminamiento serán distribuidas de forma estática, es decir, en ausencia de tráfico de aplicación en la red. En particular, los puertos de la subred deben ser desactivados antes de comenzar el envío de las tablas de encaminamiento. Una vez que todos los conmutadores han recibido sus tablas, el tráfico de aplicación es reactivado de nuevo[1].

El siguiente paso, una vez restablecida la conectividad en la subred a través de rutas provisionales, es la obtención y distribución de un conjunto de rutas definitivas, calculadas con el algoritmo FERa. Como acabamos de ver, las prestaciones ofrecidas por estas rutas son superiores.

En este caso, las rutas definitivas pueden ser distribuidas sin necesidad de aplicar ninguna técnica de control de bloqueos (como detener el servicio de la red). En otras palabras, las rutas provisionales y las definitivas pueden coexistir sin riesgo de bloqueo. Esto se debe a que ambos conjuntos de rutas emplean un subconjunto de las rutas proporcionadas por el algoritmo de encaminamiento *up*/down** original.

Sin embargo, este proceso de distribución dinámico puede provocar el descarte de algunos paquetes de datos. En particular, el descarte puede aparecer cuando la ruta definitiva hacia un destinatario no coincide exactamente con el camino provisional. Entonces, es posible que algún paquete procedente de un conmutador configurado (todavía) con la ruta provisional llegue a otro conmutador que deba emplear como puerto de salida el mismo puerto por el que recibió el paquete. Como se describe en la Sección 2.5.2, en estos casos el paquete debe ser descartado.

[1] En el próximo capítulo detallaremos la implementación del proceso de distribución estático en InfiniBand, y propondremos algunas optimizaciones para el mismo.

Para evitar estas situaciones, las entradas de las tablas definitivas deben distribuirse siguiendo algún orden. En nuestra propuesta, estas entradas son entregadas a los conmutadores de la subred a medida que son obtenidas. Como el algoritmo FERa calcula las nuevas rutas hasta un determinado nodo de destino mediante una exploración que parte desde dicho nodo, es obvio que si se actualizan las tablas en ese mismo orden, ningún conmutador obligará a "dar media vuelta" a ningún paquete entrante.

Finalmente, para acelerar el mecanismo de asimilación, la tarea de cálculo de rutas definitivas puede solaparse en el tiempo con su distribución. Así, cada vez que el algoritmo FERa obtiene una nueva entrada, el SM envía un SMP conteniendo dicha entrada.

La Figura 83 muestra todas las fases del mecanismo de gestión basado en rutas provisionales (derecha), comparando su comportamiento con un mecanismo básico (izquierda) que directamente calcula y distribuye (también de forma estática) un conjunto de rutas definitivas.

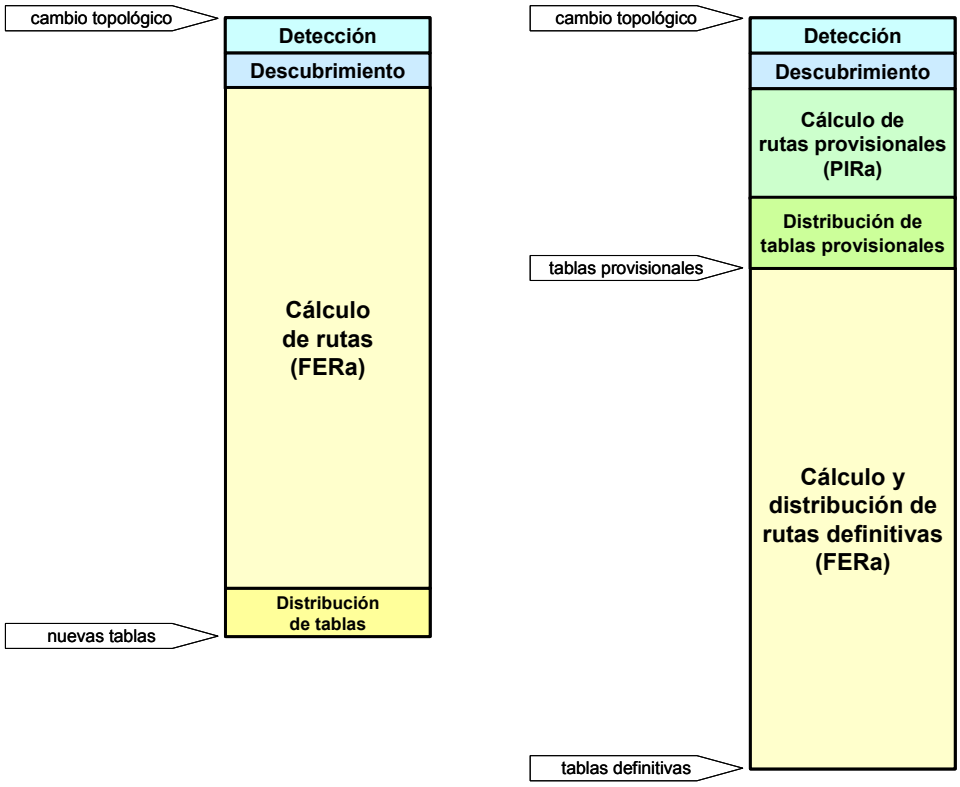


Figura 83. Mecanismos de asimilación básico y basado en rutas provisionales.

Obviamente, el protocolo basado en rutas provisionales conduce a un incremento en el tiempo y el número de SMPs requeridos para asimilar un cambio. Sin embargo, como veremos después, la principal ventaja es que un conjunto válido de rutas está disponible en un periodo de tiempo mucho menor, reduciendo así la cantidad de paquetes afectados por el cambio.

6.3.2 Evaluación de prestaciones

Los resultados que presentamos a continuación muestran las ventajas de usar el mecanismo basado en la obtención y envío de rutas provisionales para asimilar un cambio topológico en la subred.

Compararemos este mecanismo con el mecanismo básico (sin rutas provisionales). En ambos casos, tras la detección del cambio se aplica la técnica de descubrimiento parcial, descrita en el Capítulo 5.

Metodología de simulación

De nuevo, hemos aplicado técnicas de simulación para realizar la evaluación de los mecanismos de gestión. El modelo de la arquitectura InfiniBand empleado y la metodología de simulación general han sido presentados en el Capítulo 4 de esta memoria.

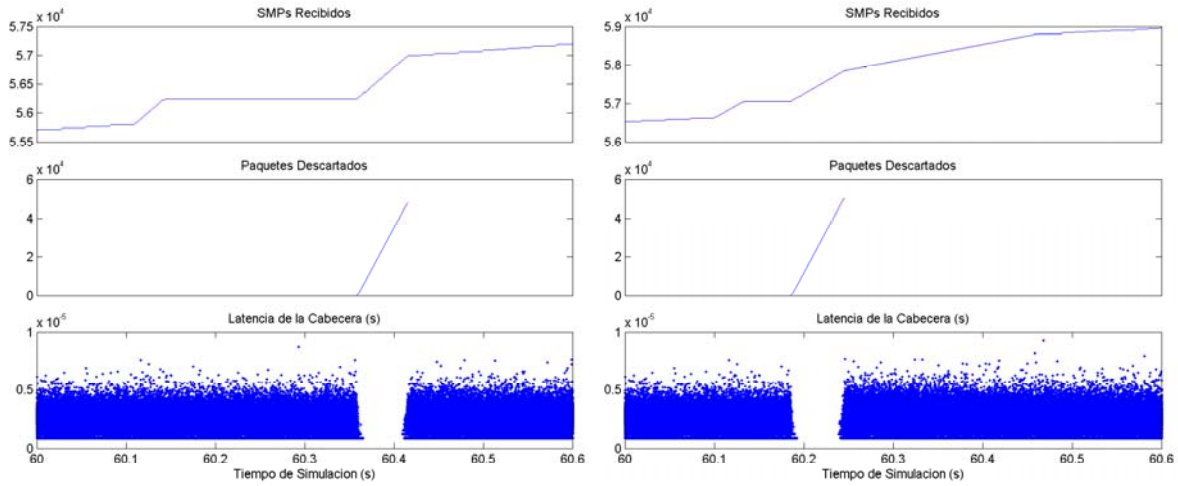
Dado que queremos evaluar el impacto del proceso de asimilación del cambio sobre las aplicaciones, hemos considerado la existencia de paquetes de datos en la subred. En concreto, la tasa de generación de paquetes (en los consumidores) es diferente para cada topología, y se corresponde con el 25% de la tasa que satura la subred. En la Sección 4.2.4 pueden encontrarse más detalles sobre el modelo de tráfico empleado.

En cada simulación, tras un periodo transitorio hemos programado un cambio topológico. En particular, el cambio se produce en tiempo de simulación 60.1 s, y consiste en la activación o desactivación de uno de los conmutadores de la subred. La simulación finaliza cuando el cambio es completamente asimilado. Este experimento ha sido repetido para cada conmutador en la subred, salvo para el conmutador que aloja al SM. Los resultados mostrados en las gráficas son valores medios.

Aunque en las gráficas de tiempo de asimilación del cambio no incluiremos la fase de detección, como mecanismo de detección de cambios se ha empleado únicamente el barrido periódico de la subred. La frecuencia de los barridos se establece en función de la topología de la subred. El factor de barrido (descrito en la Sección 5.2) es de 100.

Comportamiento instantáneo

Para comenzar, la Figura 84 y la Figura 85 muestran algunos resultados instantáneos que nos van a ayudar a comprender el comportamiento de los mecanismos de asimilación evaluados, y su impacto sobre el tráfico de aplicación. En concreto, el cambio simulado consiste en la activación (Figura 84) y la desactivación (Figura 85) de un conmutador en una subred irregular compuesta por 16 conmutadores y 14 nodos terminales.



(a) Sin rutas provisionales (b) Con rutas provisionales

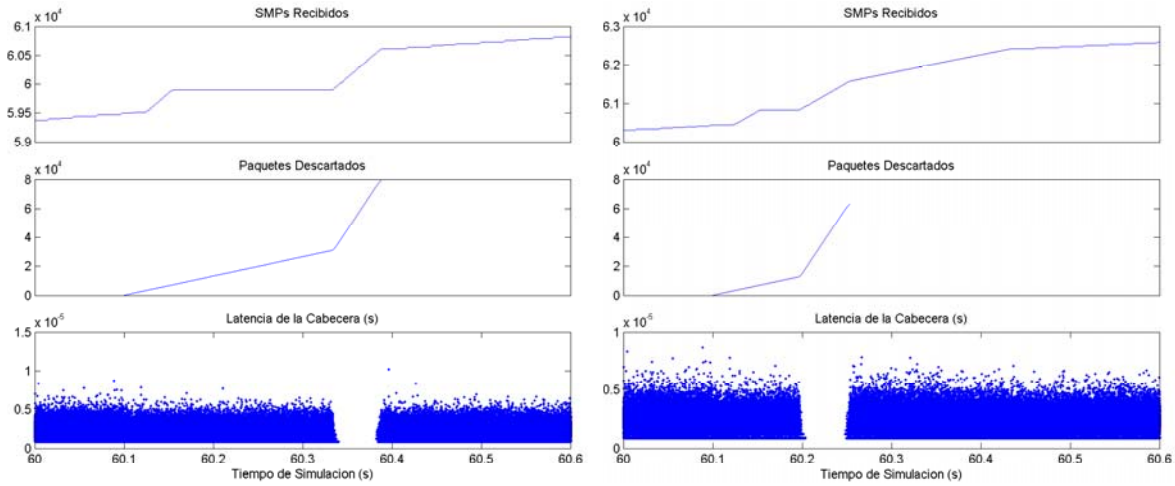
Figura 84. Resultados instantáneos para una subred con 30 nodos y un evento de activación.

En todas las gráficas, el eje horizontal representa el tiempo de simulación. La primera gráfica de cada figura muestra el número de SMPs intercambiados por las entidades de gestión (SM y SMAs) desde el comienzo de la simulación.

El primer "escalón" en esta gráfica coincide en todos los casos con la fase de descubrimiento de la topología, una vez detectado el cambio. Hay que comentar aquí que para las simulaciones presentadas en estas dos figuras, hemos empleado un factor de barrido de 10, para conseguir que el tiempo de detección sea muy similar para ambos mecanismos. De ahí que la cifra de SMPs recibidos alcance valores tan altos. Además, para poder localizar fácilmente la fase de descubrimiento, hemos empleado la técnica de descubrimiento completo del Capítulo 5.

El siguiente "escalón" en la gráfica de SMPs recibidos corresponde a la distribución de tablas de encaminamiento definitivas cuando el mecanismo de gestión no emplea rutas provisionales (Figura 84(a) y Figura 85(a)). Cuando se emplean rutas provisionales, puede apreciarse en esta gráfica una fase de envío de tablas provisionales seguida de otra fase para la distribución de las tablas definitivas.

La segunda gráfica en cada figura representa la cantidad (acumulada) de paquetes descartados durante la simulación. Para el caso de activación (Figura 84), sólo se descartan paquetes durante la fase de distribución. Como hemos dicho, la distribución de las tablas se lleva a cabo deteniendo el servicio en la red. Cuando se trata de una desactivación (Figura 85), los paquetes comienzan a ser descartados en el mismo momento en el que se produce el cambio topológico. En ambos casos, el descarte de paquetes se detiene en el momento en que las nuevas tablas de encaminamiento terminan de ser distribuidas.



(a) Sin rutas provisionales (b) Con rutas provisionales

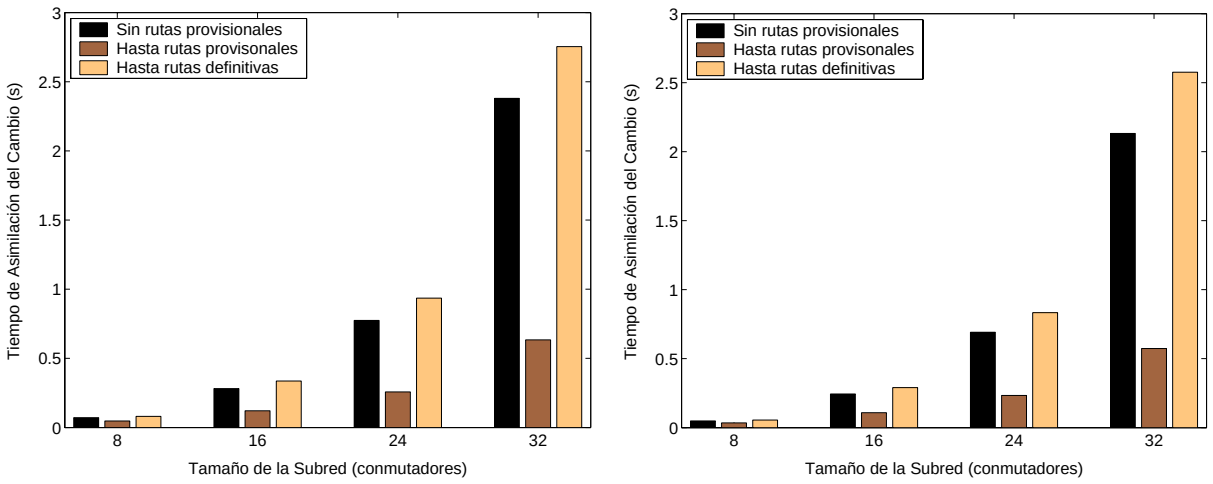
Figura 85. Resultados instantáneos para una subred con 30 nodos y un evento de desactivación.

El mecanismo basado en rutas provisionales no evita este descarte masivo de paquetes durante la fase de distribución de las tablas. Sin embargo, podemos que ver que este mecanismo comienza a distribuir las tablas antes que el mecanismo básico. Consecuentemente, el número total de paquetes descartados deberá verse también reducido. Lo veremos con más detalle más adelante.

Finalmente, la tercera gráfica en la Figura 84 y la Figura 85 muestra la latencia (desde generación) de la cabecera de cada paquete de datos. Los "valles" en estas gráficas aparecen como consecuencia de la distribución estática de las tablas.

Tiempo de asimilación del cambio

La Figura 86 muestra el tiempo de asimilación del cambio en función del tamaño de la subred y del tipo de cambio. La primera serie corresponde a la ejecución completa del proceso de asimilación que no usa rutas provisionales. La segunda y la tercera serie corresponden al tiempo transcurrido hasta la distribución de las tablas provisionales y definitivas cuando el mecanismo emplea rutas provisionales. Nótese que, aunque el mecanismo completo requiere más tiempo, los conmutadores disponen de unas tablas actualizadas mucho antes.



(a) Activación (b) Desactivación

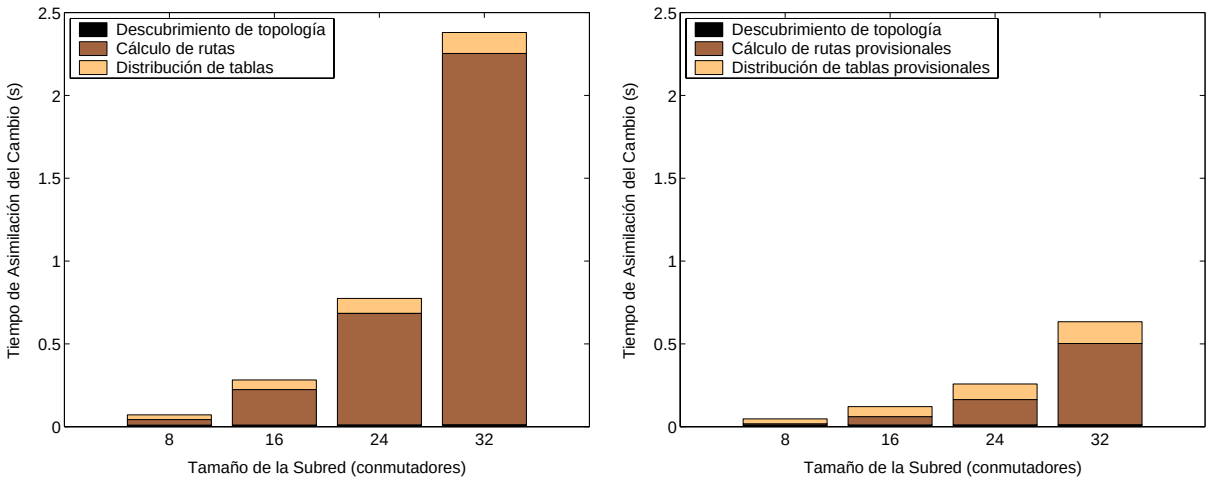
Figura 86. Tiempo total de asimilación del cambio para distintos tamaños de subred.

La Tabla 21 muestra los valores medios y las desviaciones estándar para los resultados mostrados en la Figura 86.

Tabla 21. Valores medios y desviaciones estándar para el tiempo de asimilación del cambio.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Mecanismo de gestión**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Sin rutas prov.	0.071062	0.001066	0.048837	0.007014
	Provisionales	0.047234	0.001090	0.034208	0.004257
	Definitivas	0.080632	0.001140	0.055266	0.008109
16	Sin rutas prov.	0.281818	0.001582	0.243874	0.026223
	Provisionales	0.120736	0.001594	0.108436	0.014962
	Definitivas	0.336237	0.002164	0.289714	0.029751
24	Sin rutas prov.	0.774523	0.001184	0.691443	0.011749
	Provisionales	0.257605	0.001192	0.233344	0.003749
	Definitivas	0.935637	0.003801	0.833029	0.013784
32	Sin rutas prov.	2.379867	0.002089	2.132043	0.139874
	Provisionales	0.633388	0.002090	0.573658	0.031950
	Definitivas	2.754186	0.139629	2.575702	0.169099

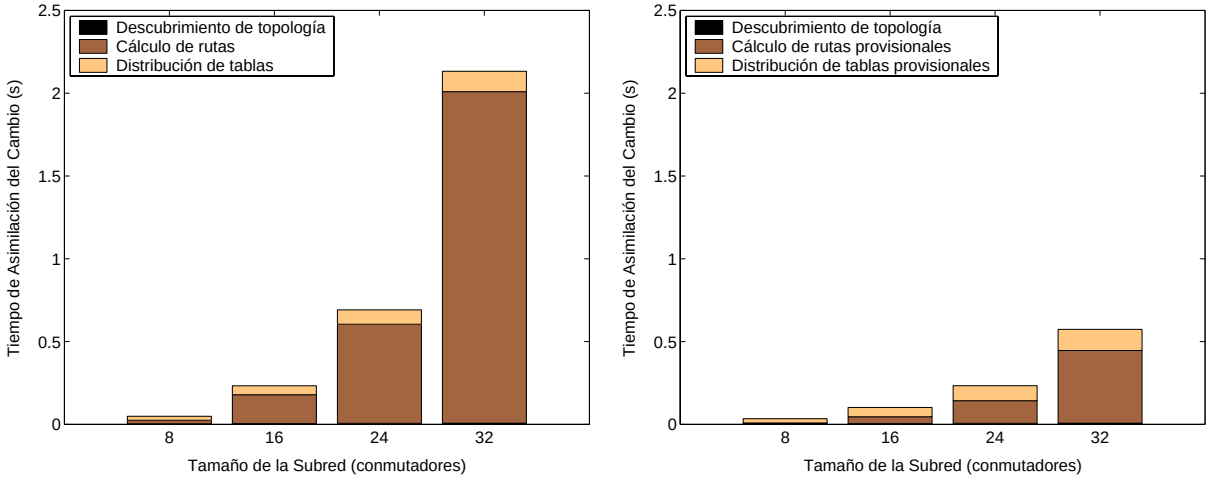
Para verlo con más detalle, la Figura 87 y la Figura 88 muestran el tiempo transcurrido desde la detección del cambio hasta la distribución de un conjunto válido de rutas, para un cambio consistente en la activación (Figura 87) y la desactivación de un conmutador (Figura 88). La diferencia en este caso es que se identifica la aportación de cada una de las tareas ejecutadas; descubrimiento de topología, cálculo de rutas y distribución de tablas.



(a) Sin rutas provisionales (b) Con rutas provisionales

Figura 87. Tiempo hasta la distribución de las nuevas tablas (activación).

Obviamente, el tiempo de descubrimiento de la nueva topología es independiente del empleo de rutas provisionales. Sin embargo, el tiempo necesario para obtener un nuevo conjunto de rutas válidas se reduce significativamente al emplear el algoritmo PIRa (en lugar del algoritmo FERa) para calcular las tablas provisionales.



(a) Sin rutas provisionales (b) Con rutas provisionales

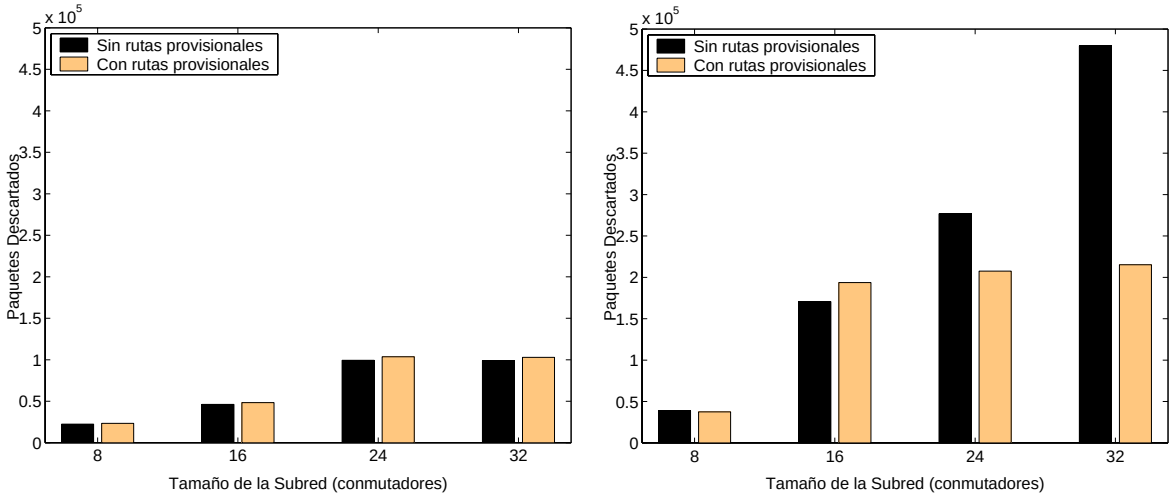
Figura 88. Tiempo hasta la distribución de las nuevas tablas (desactivación).

Con respecto al proceso de distribución de tablas, es ligeramente más largo cuando se emplean rutas provisionales. Esto se debe a que cuando se distribuyen las tablas provisionales, además de los SMPs que contienen las entradas para las nuevas tablas de encaminamien-

to, el SM debe enviar SMPs adicionales para establecer el puerto por defecto en cada conmutador[1].

Paquetes descartados

La Figura 89 presenta la cantidad total de paquetes de datos descartados por cada mecanismo, en función del tamaño de la subred y del tipo de cambio.



(a) Activación (b) Desactivación

Figura 89. Número total de paquetes descartados para distintos tamaños de subred.

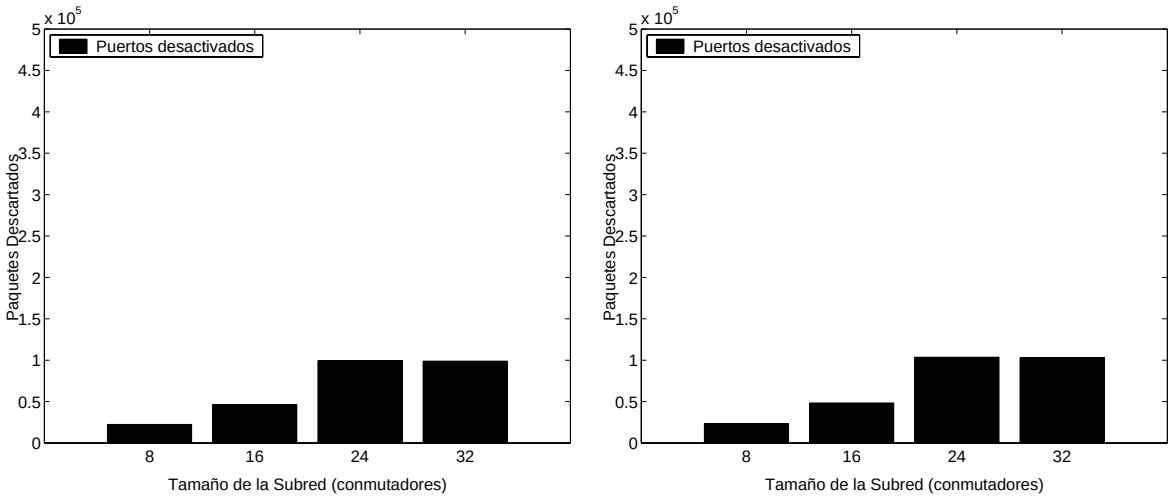
La Tabla 22 muestra los valores medios y las desviaciones estándar para los resultados mostrados en la Figura 89.

Tabla 22. Valores medios y desviaciones estándar para el número de paquetes descartados.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Mecanismo de gestión**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Sin rutas prov.	22,416.29	1,831.84	38,641.86	5,129.06
	Con rutas prov.	23,380.43	1,803.21	37,446.29	4,022.70
16	Sin rutas prov.	46,272.07	1,112.61	170,542.47	82,092.45
	Con rutas prov.	48,313.53	1,141.32	193,609.60	174,839.06
24	Sin rutas prov.	99,295.22	1,096.14	276,721.57	79,850.60
	Con rutas prov.	103,668.39	1,090.88	207,460.26	46,395.80
32	Sin rutas prov.	98,899.97	4,258.03	479,645.65	295,568.24
	Con rutas prov.	102,888.87	4,338.56	215,175.77	85,081.91

Para poder analizar estos resultados con mayor precisión, la Figura 90 y la Figura 91 muestran los mismos resultados de la Figura 89(a) y la Figura 89(b), respectivamente, pero distinguiendo las principales causas de descarte de paquetes.

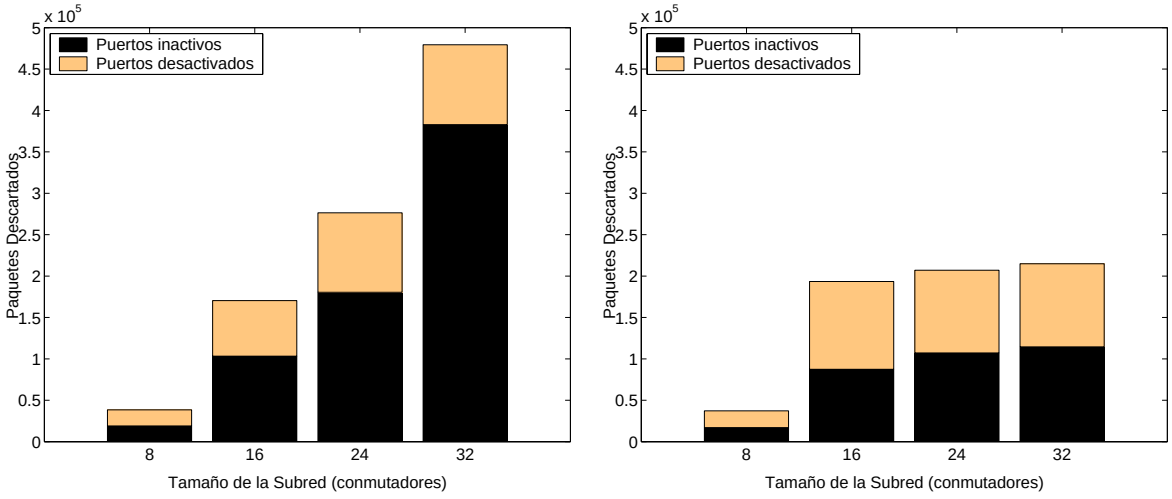
[1] En concreto, se trata de SMPs de tipo *SubnSet(SwitchInfo)*.



(a) Sin rutas provisionales (b) Con rutas provisionales

Figura 90. Paquetes descartados en función de la causa de descarte (activación).

En estas gráficas, la leyenda "Puertos desactivados" hace referencia a los paquetes generados por los nodos terminales durante la distribución estática de las tablas de encaminamiento. Todos estos paquetes acaban perdiéndose, ya que los puertos de la subred han sido desactivados por el SM y sólo admiten paquetes de gestión. Por otro lado, en el caso de desactivación (Figura 91), la leyenda "Puertos inactivos" se refiere a aquellos paquetes dirigidos al conmutador desactivado, y a los paquetes que deberían cruzarlo para alcanzar su destino, según la configuración anterior. Los paquetes descartados no son reinyectados en la red. En nuestro modelo asumimos que esta función es llevada a cabo por mecanismos en niveles superiores.



(a) Sin rutas provisionales (b) Con rutas provisionales

Figura 91. Paquetes descartados en función de la causa de descarte (desactivación).

En caso de activación (Figura 90), podemos ver que el número de paquetes descartados es ligeramente superior para el mecanismo basado en rutas provisionales. Esto se debe a que los puertos de la subred permanecen desactivados durante más tiempo. Como hemos detallado antes, la distribución de las tablas provisionales requiere un periodo de tiempo mayor.

En caso de desactivación (Figura 91), apreciamos de nuevo un ligero incremento en la cantidad de paquetes descartados en puertos desactivados. Sin embargo, hay una reducción significativa en el número de paquetes descartados por la existencia de puertos sin actividad en la subred. El mecanismo basado en rutas provisionales asimila el cambio más rápidamente, reduciendo así la cantidad de paquetes descartados por esta causa.

6.4 Conclusiones

En este capítulo hemos presentado y evaluado dos algoritmos para la obtención de rutas libres de bloqueo para InfiniBand. El primero de ellos (*fully explicit routing*) computa un puerto de salida en cada conmutador para cada posible destinatario. El segundo algoritmo (*partially implicit routing*) reduce significativamente el número de entradas en tablas a calcular, y por tanto el tiempo de cómputo de las rutas. Se ha demostrado formalmente la conectividad total entre los elementos de la subred, a través del conjunto de rutas libres de bloqueo proporcionadas por ambos algoritmos.

Se ha presentado un mecanismo de gestión basado en el empleo del algoritmo PIRa para calcular un conjunto de rutas provisionales adaptadas a la nueva situación topológica. La principal ventaja del mecanismo basado en rutas provisionales es que aúna las ventajas de los algoritmos PIRa y FERa simultáneamente. Los conmutadores disponen mucho antes de un conjunto de rutas válidas, reduciendo significativamente los efectos del proceso de asimilación del cambio sobre las aplicaciones.

Capítulo 7
Distribución de tablas de encaminamiento

7.1 Introducción

El envío de las nuevas tablas de encaminamiento hacia los conmutadores de la subred constituye la última fase en el proceso de asimilación de un cambio topológico. Una vez configuradas las tablas, los paquetes de datos serán encaminados a través de rutas acordes a la nueva configuración topológica.

La especificación de InfiniBand define completamente los SMPs que el SM debe emplear para actualizar las tablas de encaminamiento. Sin embargo, el orden de actualización no ha sido especificado. Como veremos seguidamente, aunque la configuración previa y la nueva sean libres de bloqueo, una distribución descontrolada de las tablas puede dar lugar a la aparición de bloqueos entre paquetes de datos en la subred [65].

Las técnicas de distribución tradicionalmente empleadas en redes de altas prestaciones detienen el servicio en la red durante el proceso de envío, con el fin de evitar la aparición de situaciones de bloqueo. En otras palabras, se trata de separar temporalmente los paquetes encaminados según distintas configuraciones de tablas. Como es lógico, esta forma de proceder, denominada en la literatura reconfiguración estática, tiene efectos muy negativos sobre las aplicaciones [12, 19], sobre todo cuando éstas tienen fuertes requisitos de calidad de servicio (QoS), o se trata de aplicaciones en tiempo real.

Para paliar los efectos del mecanismo de distribución estático, en la bibliografía pueden encontrarse numerosas propuestas para llevar a cabo la reconfiguración dinámica de la red (sin detener el tráfico), manteniendo al mismo tiempo la libertad de bloqueos.

Sin embargo, la adaptación a InfiniBand de estos protocolos de reconfiguración dinámica no es inmediata. Su complejidad obliga a introducir modificaciones en la especificación. Por ejemplo, habría que añadir algunos elementos nuevos, como es el caso de los paquetes de control específicos del protocolo de reconfiguración. También es posible que el protocolo requiera emplear alguno de los elementos proporcionados por la especificación para fines

totalmente diferentes a los que en principio estaban destinados. En otros casos, el protocolo de reconfiguración dinámica podría necesitar que los agentes de gestión (SMAs) llevaran cabo alguna tarea que en principio no tienen asignada, es decir, que estuvieran dotados de cierta "inteligencia" (recordemos que se trata de entidades pasivas).

En este capítulo, se presentan dos mecanismos alternativos de distribución de tablas que, en realidad, son variantes de la reconfiguración estática "tradicional". La ventaja de estos mecanismos con respecto a las técnicas de reconfiguración dinámica es que son muy fácilmente implementables en InfiniBand, ya que se adaptan perfectamente a su especificación. Además, como veremos, nuestras propuestas reducen significativamente el impacto del proceso de asimilación del cambio sobre las aplicaciones. Finalmente, veremos algunas propuestas para acortar la duración de la fase de distribución de tablas, haciendo el proceso de asimilación todavía más transparente a las aplicaciones.

7.1.1 Bloqueos durante la distribución de las tablas

Emplearemos de nuevo el ejemplo de la Figura 92 para ilustrar cómo una distribución asíncrona de las tablas de encaminamiento puede generar situaciones de bloqueo. Como en el capítulo anterior, asumiremos que las tablas de encaminamiento de la subred se obtienen empleando el algoritmo de encaminamiento libre de bloqueo *up*/down**.

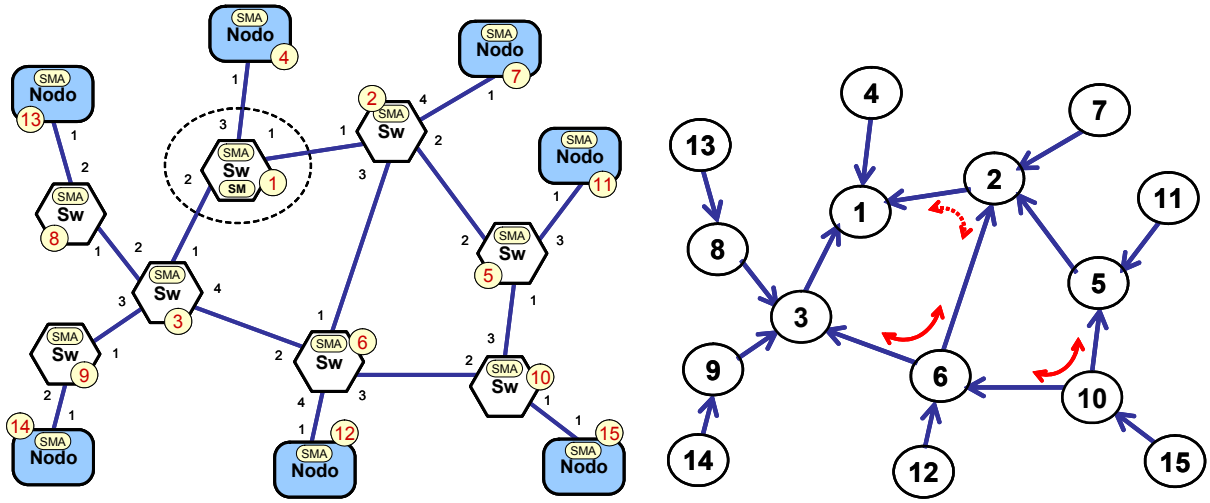


Figura 92. Grafo dirigido asociado a una subred con topología irregular.

En un grafo dirigido *up*/down**, un nodo corte [20] es aquel del que parten dos o más arcos. En cada ciclo del grafo hay exactamente un nodo corte. Los nodos corte impiden que determinadas transiciones (puerto de entrada – puerto de salida) sean empleadas por los paquetes que los cruzan. Estas restricciones en el encaminamiento son necesarias para garantizar la libertad de bloqueos.

En la Figura 92(b), todas las transiciones prohibidas están marcadas con una flecha en línea continua. En concreto, las transiciones $6_{1\rightarrow2}$ y $6_{2\rightarrow1}$ (desde el puerto *1* al puerto *2* en el nodo *6*, y viceversa) están prohibidas. Lo mismo sucede con las transiciones $10_{2\rightarrow3}$ y $10_{3\rightarrow2}$. También se dice que las correspondientes dependencias entre canales [26] están desactivadas.

Durante el proceso de distribución de tablas, el bloqueo puede aparecer cuando el nuevo nodo corte de un ciclo cambia de posición, con respecto al grafo dirigido anterior. Supongamos que, en la vieja configuración, el nodo corte en el ciclo de la izquierda de la Figura 92(b) era el nodo *2*. Eso significa que la dirección asignada al enlace que conecta los nodos *6* y *2* era la opuesta a la actual. Por lo tanto, las transiciones $2_{1\rightarrow3}$ y $2_{3\rightarrow1}$ (marcadas con una flecha en línea discontinua en la figura) no estaban permitidas.

Como el proceso de distribución es asíncrono, es decir, no sigue ningún orden concreto, podría activar las dependencias en el nodo *2* antes de desactivar las dependencias en el nodo *6*. Este escenario no es libre de bloqueos. En particular, el nodo *2* podría emplear su nueva tabla para encaminar los paquetes procedentes del nodo *1* hacia el nodo *6* y, simultáneamente, el nodo *6* podría usar su tabla (aún sin actualizar) para encaminar paquetes desde *2* hacia *3*, cerrando así el ciclo. Análogamente, el bloqueo podría aparecer en el sentido opuesto.

En este ejemplo, las viejas dependencias en el nodo *6* (ilegales en la nueva configuración) han interactuado con las nuevas dependencias en el nodo *2* para formar un bloqueo. En la literatura, las primeras se conocen como *zombie* [20] o *ghost* [65] *dependencies*.

7.1.2 Reconfiguración dinámica. Implementación en InfiniBand

Los mecanismos de reconfiguración dinámica permiten la existencia de tráfico de aplicación en la red durante la actualización de las tablas de encaminamiento, asegurando la libertad de bloqueos por medio de técnicas de evitación [26]. En la bibliografía puede encontrarse un marco formal para la reconfiguración dinámica [46, 63, 65]. Como veremos, la implementación de estos mecanismos en InfiniBand no es trivial, debido a su complejidad.

Reconfiguración parcial progresiva

La reconfiguración parcial progresiva [17, 19, 20, 24] (PPR, *Partial Progressive Reconfiguration*) fue la primera propuesta para la reconfiguración dinámica de la red. El mecanismo hace evolucionar el grafo dirigido *up*/down** hacia un grafo acorde a la nueva configuración de la red. Para ello, emplea una secuencia ordenada de actualizaciones parciales de las tablas de encaminamiento. Tras cada actualización parcial, cada conmutador debe sincronizarse con algunos de sus vecinos.

El algoritmo se aplica cuando, como consecuencia del cambio topológico, el grafo dirigido deja de cumplir las condiciones mencionadas en la Sección 6.1.1 (página 119). En concreto, el cambio puede dar lugar a la aparición de múltiples nodos sumidero en el grafo. Uno

de ellos se convertirá en el sumidero del nuevo grafo dirigido, mientras que los demás dejarán de realizar esa función. Para ello, el mecanismo define una secuencia de modificaciones en la dirección de parte de los enlaces de la red.

El proceso de traslado de la función de nodo corte dentro de un ciclo es el procedimiento más delicado del mecanismo, puesto que las dependencias en el nodo corte final deben desactivarse antes de la activación de las dependencias en el nodo corte inicial. En caso contrario, podrían darse situaciones como la descrita en la Sección 7.1.1. Para garantizar esta secuenciación, es necesaria una sincronización entre los nodos corte inicial y final, a través de un mensaje especial (denominado mensaje *escoba*).

En su forma original, el mecanismo PPR no puede ser implementado en InfiniBand, pues el proceso de activación y desactivación de dependencias requiere información explícita sobre el puerto de entrada. Sin embargo, como veremos más tarde, este problema puede resolverse manipulando convenientemente las tablas de mapeo.

Double scheme

El mecanismo *double scheme* (y sus variantes) [56, 64] emplea dos conjuntos disjuntos de canales virtuales para separar espacialmente los paquetes encaminados con cada una de las funciones de encaminamiento (la previa y la nueva). De esta forma, un paquete nunca podrá ser encaminado bajo ambas configuraciones.

Básicamente, tras la detección del cambio y la obtención de las nuevas rutas, el procedimiento drena (vacía de paquetes) uno de los conjuntos de canales virtuales, y comienza a utilizarlos para los paquetes encaminados con la nueva función de encaminamiento. Después, cuando las nuevas tablas han sido distribuidas y todos los paquetes utilizan el primer conjunto de canales virtuales, drena el otro conjunto, que pasará finalmente a ser usado también por la nueva función de encaminamiento.

Para implementar este mecanismo, es necesario que la red ofrezca soporte para encaminar simultáneamente unos paquetes con una función de encaminamiento y otros paquetes con una función distinta, soporte para implementar el mecanismo de drenaje de canales virtuales, y soporte para conmutar de una función de encaminamiento a otra sin tener que descartar paquetes.

En [66, 92] se describe y analiza una implementación del mecanismo *double scheme* en InfiniBand. En esta propuesta, la separación espacial de los recursos (canales virtuales) se realiza asignando dos LIDs a cada GUID. Esto permite definir dos funciones de encaminamiento en la red, una con cada conjunto de LIDs. Nótese que esto significa que sólo la mitad de los posibles LIDs y entradas en tablas de encaminamiento estarán disponibles durante la operación normal de la red.

Para drenar cada conjunto de canales virtuales, el proceso reserva un conjunto de SLs para la vieja función de encaminamiento, y modifica el esquema de mapeo de SL a VL (Sec-

ción 2.4.5), de forma que estos paquetes usen un conjunto restringido de canales virtuales. Nótese que esto puede afectar al esquema de calidad de servicio, si éste está empleando los 16 niveles de servicio contemplados en la especificación de InfiniBand. Además, el SM debe emplear unos paquetes de control especiales para comprobar el nivel de ocupación de los VLs. Actualmente, estos paquetes no existen en la especificación, si bien existe soporte para su definición[1].

Finalmente, el cambio de una función de encaminamiento a otra puede ser implementado mediante la actualización de las tablas de encaminamiento y la modificación del esquema de mapeo de GUID a LID.

Otros mecanismos de reconfiguración dinámica

NetRec [3] restaura la conectividad de la red construyendo un árbol de expansión a partir de los vecinos del nodo que ha fallado. Las rutas obtenidas a partir de este árbol se emplean para redirigir el tráfico que atravesaba el nodo que ha caído (en la configuración previa). Este mecanismo es totalmente distribuido y requiere que cada conmutador mantenga cierta información sobre los conmutadores de su entorno. En principio, InfiniBand no proporciona mecanismos para almacenar esta información. Además, *NetRec* sólo es aplicable a redes *wormhole*.

El mecanismo *skyline* [45] sólo reconfigura una parte de la red tras el cambio. En realidad, parte de la misma idea que PPR, es decir, identifica varios nodos sumidero en el grafo y modifica la dirección de una serie de enlaces para corregir esta situación.

Por último, recientemente se ha propuesto el mecanismo de reconfiguración dinámica *LORE* (*LOcal REconfiguration*) [86]. *LORE* reconfigura únicamente una pequeña zona alrededor del fallo. En esta zona, se utilizan algunos canales virtuales para reencaminar los paquetes que van hacia el fallo, evitando así su descarte. Uno de los requisitos para la implementación de *LORE* es que los elementos de encaminamiento conozcan la configuración topológica de la red, lo cual no es inmediato en InfiniBand.

7.2 Mecanismos de distribución propuestos

Esta sección describe, en primer lugar, la forma en la que el mecanismo de distribución estático "tradicional" puede implementarse en InfiniBand. Después, se presentan de manera informal dos técnicas de distribución optimizadas. La idea es relajar la técnica de reconfiguración estática, mediante la reducción del número de puertos de la subred que realmente

[1] En concreto, habría que emplear un GMP de la clase de gestión *vendor specific* (ver Capítulo 3).

deben ser desactivados, o evitando el empleo de determinadas transiciones entre enlaces durante la distribución de las tablas.

7.2.1 Mecanismo tradicional. Desactivación completa

La distribución estática ha sido tradicionalmente empleada en redes como Autonet [80], Myrinet [14] o Fibre Channel [90]. Con objeto de evitar la aparición de bloqueos, el tráfico de aplicación se detiene durante el proceso de envío de las tablas de encaminamiento. La red completa deja de estar operativa, aún en el caso de que el cambio sólo afecte a una porción reducida de la misma.

La distribución estática puede llevarse a cabo en InfiniBand en tres pasos secuenciales. En primer lugar, todos los puertos activos en la subred deben ser desactivados por parte del SM. En concreto, el SM emplea un SMP de tipo *SubnSet(PortInfo)* para establecer el estado de cada puerto a *Initialize*[1]. Como se describe en la Sección 2.4.1, en este estado el puerto sólo puede recibir y transmitir SMPs y paquetes de control de flujo, descartando el resto de paquetes recibidos o presentados para su transmisión.

El siguiente paso en el proceso de distribución es el envío de las tablas de encaminamiento propiamente dicho. Esta fase se lleva a cabo empleando SMPs *SubnSet(LinearForwardingTable)* y/o *SubnSet(RandomForwardingTable)*, en función del tipo de tablas que implementen los conmutadores de la subred.

Finalmente, una vez que el SM ha comprobado que todos los conmutadores han recibido correctamente sus tablas (a través de los correspondientes reconocimientos), el tráfico de aplicación puede ser permitido de nuevo en la subred. En otras palabras, la subred debe ser activada. Por medio de SMPs *SubnSet(PortInfo)*, el SM establece el estado de cada puerto a *Active*. El proceso de activación de la subred se detalla en la Sección 3.2.5.

Los SMPs empleados en los dos primeros pasos del mecanismo (desactivación de puertos y envío de tablas) deben emplear encaminamiento dirigido, dado que las nuevas tablas de encaminamiento todavía no han sido cargadas. Por el contrario, y como ya comentamos en la Sección 3.2.5, los SMPs para la fase de activación de la subred pueden emplear también encaminamiento en base a destino. Como vimos en la Sección 3.1.3, en principio esta forma de encaminamiento es más rápida que el encaminamiento dirigido, ya que evita el procesamiento del SMP en cada SMI intermedio.

[1] En realidad, el SM usa el SMP para poner el puerto a *Down*. Como hay actividad en el enlace, el puerto cambia automáticamente al estado *Initialize*.

7.2.2 Desactivación de puertos en nodos corte

El mecanismo de desactivación completa es demasiado restrictivo, es decir, impide que cualquier paquete cruce la subred durante el proceso de distribución. Un mecanismo de distribución optimizado podría basarse en desactivar únicamente las dependencias entre canales en los nuevos nodos corte, en lugar de desactivar todos los puertos de la subred, antes de comenzar a enviar las tablas de encaminamiento. De esta forma, se mantendría la libertad de bloqueos durante la distribución, pues situaciones como la descrita en la Sección 7.1.1 no podrían llegar a darse.

Lo anterior implicaría no permitir determinadas transiciones puerto de entrada – puerto de salida en los nodos corte. Desafortunadamente, el puerto de entrada del paquete no es empleado cuando éste es encaminado. En InfiniBand, las tablas de encaminamiento proporcionan un puerto de salida para el paquete independientemente del puerto por el que llegó al conmutador.

Como paso intermedio, propondremos un mecanismo de distribución de tablas (libre de bloqueo) que desactive únicamente los puertos de los conmutadores que actuarán como nodos corte en la nueva configuración. Obsérvese que ni siquiera es necesario desactivar todos los puertos en estos nodos. Sería suficiente con seleccionar aquellos puertos conectados a enlaces ascendentes (en dirección *up*), y desactivarlos todos, excepto uno. De esta forma, nos aseguramos de que las transiciones prohibidas (*down-up*) no serán empleadas por ningún paquete, al tiempo que estamos permitiendo el uso de muchas rutas a través de la subred durante el proceso de distribución.

Considerando el grafo dirigido de la Figura 92(b), para el nodo corte *6* el proceso de distribución sólo debe desactivar el puerto 6_1 (el puerto *1* en el nodo *6*) o el puerto 6_2. De la misma forma, para el nodo corte *10* sólo es necesario desactivar el puerto 10_2 o el puerto 10_3.

Finalmente, como sucede en el proceso de desactivación completa, es necesario un tercer paso en el proceso, tras la fase de envío de las tablas de encaminamiento, en el cual los puertos de los nodos corte son nuevamente activados, recuperando totalmente la conectividad entre todos los elementos de la subred. De nuevo, es posible emplear encaminamiento basado en destino para los SMPs de esta tercera fase.

7.2.3 Desactivación de dependencias en nodos corte

El mecanismo anterior es fácil de implementar y, como veremos, reduce significativamente el impacto del proceso de distribución basado en la desactivación completa de la subred. Sin embargo, todavía hay muchas rutas en la subred que podrían ser empleadas mientras se están distribuyendo las tablas, sin introducir bloqueos en la subred.

Por ejemplo, supongamos que, en el ejemplo de la Figura 92(b), el puerto 10_2 ha sido seleccionado para su desactivación. En este caso, estamos descartando innecesariamente (entre otros) todos aquellos paquetes generados en el nodo *10* (o destinados hacia él) que deban emplear el enlace que conecta *10* con *6*.

Como se ha comentado en la sección anterior, podemos mejorar el mecanismo de distribución de tablas impidiendo únicamente unas pocas combinaciones puerto de entrada – puerto de salida, y permitiendo el resto de opciones de encaminamiento. Para lograrlo, podemos programar de forma adecuada las tablas de mapeo de SL a VL (SLtoVLMT) de los conmutadores de la subred (Sección 2.4.5). De hecho, la propia especificación de InfiniBand sugiere que estas tablas pueden ser empleadas para evitar bloqueos.

Cuando una consulta a la tabla de mapeo (que se establece para cada combinación puerto de entrada – puerto de salida) devuelve el valor VL15, el paquete que está siendo encaminado debe ser descartado. Esto se debe a que en el canal virtual de gestión no se permiten paquetes de datos. Éste es el mecanismo que emplearemos para impedir el uso de transiciones prohibidas.

La técnica de distribución resultante no requiere desactivar ninguno de los puertos de la subred, ni siquiera los puertos de los nodos corte. La fase previa de desactivación de puertos es sustituida por el envío de tablas SLtoVLMT a los nuevos nodos corte. En concreto, es necesario enviar a cada nodo corte una tabla SLtoVLMT para cada transición puerto de entrada – puerto de salida prohibida en dicho nodo. En esta tabla, todas las entradas contendrán el valor VL15. Consecuentemente, todos los paquetes que intenten realizar esta transición serán automáticamente descartados.

En el ejemplo de la Figura 92(b), es necesario configurar únicamente cuatro tablas de mapeo, de forma que las dependencias en los nodos *6* y *10* (los nuevos nodos corte) queden desactivadas. Más concretamente, se trata de las tablas SLtoVLMT para las transiciones $6_{1\rightarrow2}$, $6_{2\rightarrow1}$, $10_{2\rightarrow3}$ y $10_{3\rightarrow2}$. El SM emplea cuatro SMPs de tipo *SubnSet(SLtoVLMappingTable)* para llevar a cabo el envío de estas tablas.

Una vez configuradas las tablas de mapeo, puede procederse al envío asíncrono de las tablas de encaminamiento. Obsérvese que, en este caso, no es estrictamente necesario ejecutar una última etapa para configurar nuevamente las tablas de mapeo en los nodos corte, ya que las nuevas tablas de encaminamiento no contemplarán en ningún caso esas transiciones prohibidas.

A modo de resumen, la Figura 93 muestra el comportamiento de cada una de las tres estrategias de distribución de tablas descritas en esta sección, una vez que las nuevas rutas han sido obtenidas.

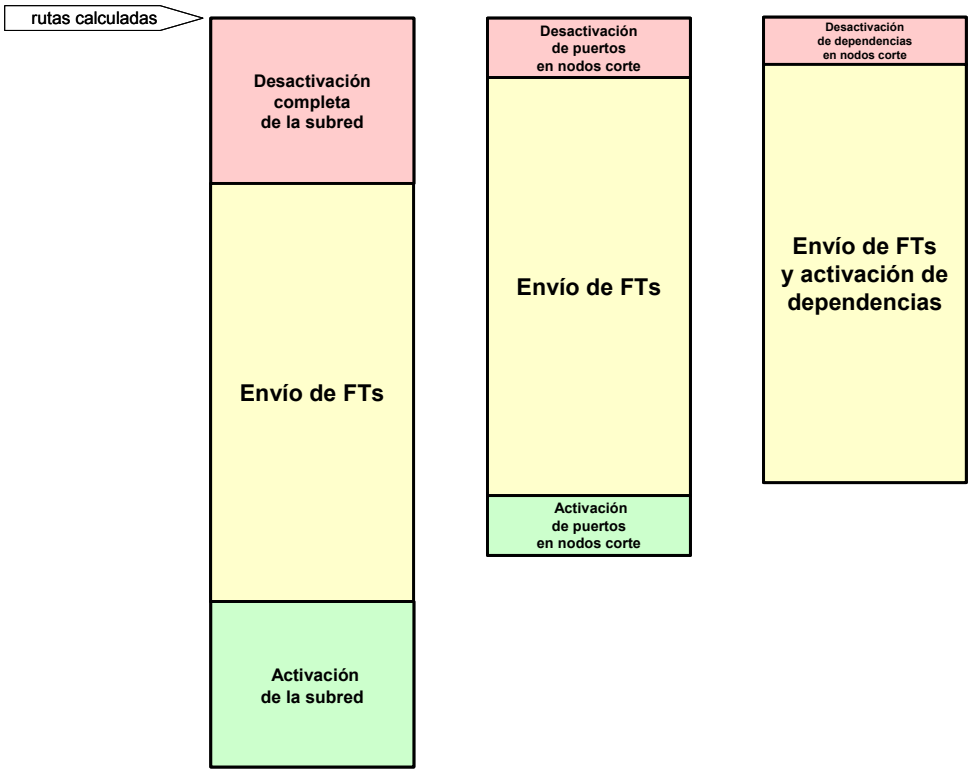


Figura 93. Mecanismos para la distribución de las tablas de encaminamiento.

7.3 Evaluación comparativa

En esta sección analizaremos comparativamente las distintas técnicas de distribución de tablas propuestas, a través de un conjunto de resultados de simulación.

De nuevo, hemos aplicado la técnica de descubrimiento parcial de la topología del Capítulo 5. Sin embargo, por simplicidad a la hora de analizar los resultados de simulación, no emplearemos el mecanismo de gestión basado en el uso de rutas provisionales, descrito en el Capítulo 6. Obviamente, las técnicas de distribución descritas pueden aplicarse con independencia del tipo de tablas que están siendo distribuidas (provisionales o definitivas).

La metodología de simulación empleada en este estudio es idéntica a la descrita en la Sección 6.3.2.

7.3.1 Resultados de simulación

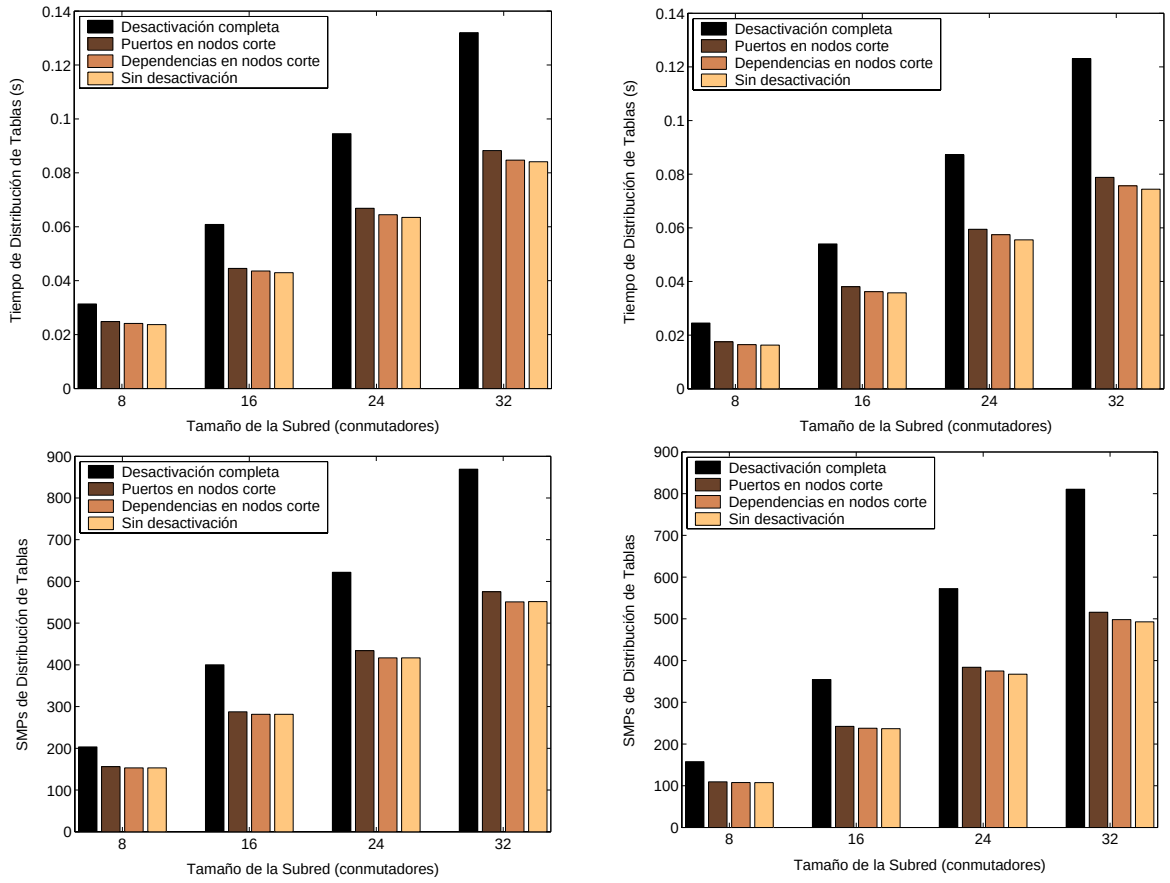
Analizaremos en primer lugar el comportamiento de las técnicas de distribución. Posteriormente, nos centraremos en evaluar su impacto sobre las aplicaciones.

Como referencia a la hora de realizar las comparaciones, hemos añadido a algunas de las gráficas una cuarta serie (etiquetada “Sin desactivación”) que muestra los resultados para un proceso de distribución dinámico “básico”. Este proceso constaría de una única fase, en la cual las tablas de encaminamiento son asíncronamente enviadas a los conmutadores de la subred, sin una desactivación previa de puertos o dependencias. Obsérvese que, por tanto, este mecanismo es propenso al bloqueo.

Tiempo y SMPs de distribución de tablas

La Figura 94 muestra el tiempo y los SMPs requeridos por los mecanismos de distribución evaluados para actualizar las tablas de encaminamiento de la subred tras la ocurrencia de un cambio topológico, en función del tamaño de la subred y del tipo de cambio.

No se incluyen los procesos previos de detección del cambio, descubrimiento de la topología o cálculo de nuevas rutas. Sin embargo, los resultados sí que tienen en cuenta las etapas inicial y final del proceso de distribución (desactivación y activación de puertos o dependencias).



(a) Activación (b) Desactivación

Figura 94. Tiempo y SMPs empleados por los distintos mecanismos de distribución, en función del tamaño de la subred y del tipo de cambio.

La Tabla 23 y la Tabla 24 muestran los valores medios y las desviaciones estándar para los resultados mostrados en la Figura 94.

Tabla 23. Valores medios y desviaciones estándar para los tiempos de distribución.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Técnica de distribución**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Desactivación completa	0.031375	0.001238	0.024500	0.002423
	Puertos en nodos corte	0.024821	0.001761	0.017499	0.001768
	Depend. en nodos corte	0.024136	0.001997	0.016462	0.001825
	Sin desactivación	0.023703	0.001747	0.016287	0.001751
16	Desactivación completa	0.060855	0.001147	0.056231	0.004512
	Puertos en nodos corte	0.044557	0.001435	0.042236	0.009882
	Depend. en nodos corte	0.043594	0.001726	0.040644	0.010744
	Sin desactivación	0.042955	0.001472	0.040250	0.010657
24	Desactivación completa	0.094511	0.001151	0.087236	0.000822
	Puertos en nodos corte	0.066868	0.001322	0.059473	0.000630
	Depend. en nodos corte	0.064460	0.001436	0.057425	0.001725
	Sin desactivación	0.063467	0.001346	0.055523	0.000495
32	Desactivación completa	0.132666	0.003887	0.123100	0.001547
	Puertos en nodos corte	0.090293	0.011544	0.078830	0.001042
	Depend. en nodos corte	0.084724	0.001809	0.075661	0.002053
	Sin desactivación	0.086265	0.012283	0.074411	0.000873

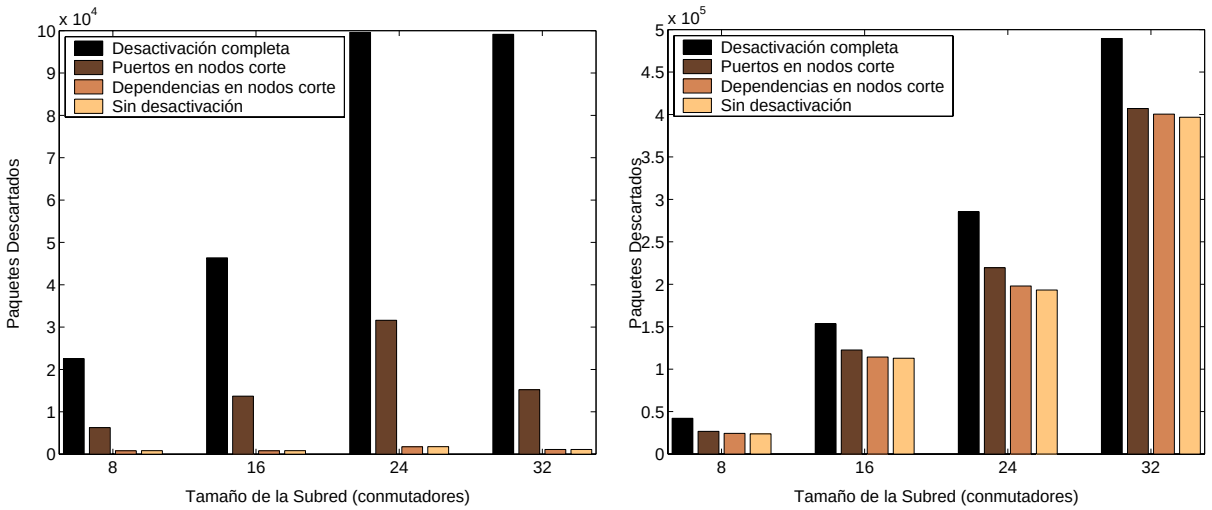
Tabla 24. Valores medios y desviaciones estándar para los SMPs de distribución.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Técnica de distribución**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Desactivación completa	203.43	5.83	157.71	15.69
	Puertos en nodos corte	156.00	9.86	109.43	11.30
	Depend. en nodos corte	153.14	10.19	107.71	11.34
	Sin desactivación	153.142	10.19	107.43	11.20
16	Desactivación completa	400.13	5.14	369.87	30.52
	Puertos en nodos corte	287.47	7.06	271.60	68.53
	Depend. en nodos corte	281.47	8.11	267.20	70.24
	Sin desactivación	281.47	8.11	266.40	70.56
24	Desactivación completa	621.91	5.32	572.52	4.91
	Puertos en nodos corte	434.00	6.62	383.91	3.61
	Depend. en nodos corte	416.70	7.57	375.04	8.92
	Sin desactivación	416.70	7.57	367.30	3.26
32	Desactivación completa	873.55	26.86	810.58	11.06
	Puertos en nodos corte	589.10	78.36	515.74	6.32
	Depend. en nodos corte	550.97	8.90	498.06	12.15
	Sin desactivación	566.19	82.59	492.90	6.32

Podemos ver que la técnica basada en la desactivación de puertos en nodos corte logra una importante reducción en los dos parámetros evaluados. Además, se observa una reducción adicional para el mecanismo basado en la desactivación de dependencias en nodos corte. De hecho, este mecanismo obtiene, en general, unos resultados muy cercanos a la distribución dinámica.

Paquetes descartados

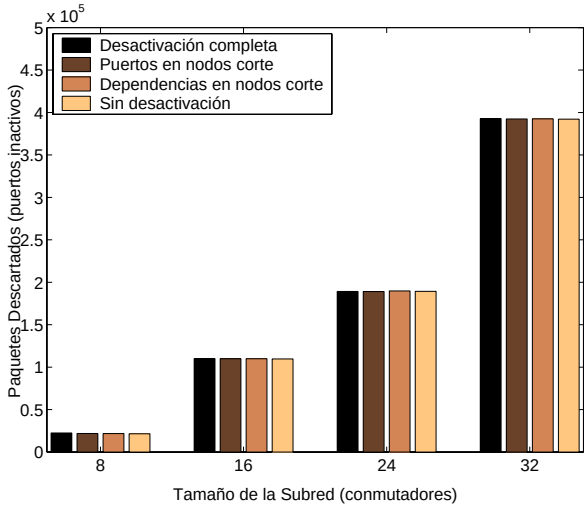
La Figura 95 muestra la cantidad de paquetes descartados en función del mecanismo de distribución empleado, el tipo de cambio topológico y el tamaño de la subred. Nótese que las escalas para activación y desactivación no coinciden.



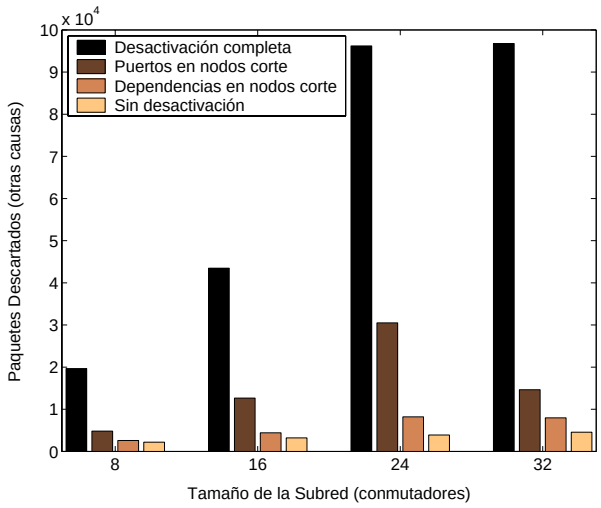
(a) Activación (b) Desactivación

Figura 95. Número total de paquetes descartados para distintos tamaños de subred.

En el caso de la desactivación, hay una gran cantidad de paquetes que son descartados debido a que muchas de las rutas dentro de la subred desaparecen físicamente a consecuencia del cambio, al menos hasta que las nuevas tablas de encaminamiento hayan sido cargadas y proporcionen caminos alternativos. Este descarte masivo de paquetes es inevitable, con independencia de la técnica de distribución aplicada, debido a que el encaminamiento en InfiniBand es determinista. La Figura 96 distingue entre aquellos paquetes descartados en puertos físicamente inactivos y los paquetes descartados por otras causas (que veremos seguidamente), ante un evento de desactivación.



(a) Puertos inactivos



(b) Otras causas de descarte

Figura 96. Número de paquetes descartados (a) en puertos inactivos y (b) por otras causas, para el caso de desactivación.

La Tabla 25 muestra los valores medios y las desviaciones estándar para los resultados mostrados en la Figura 95.

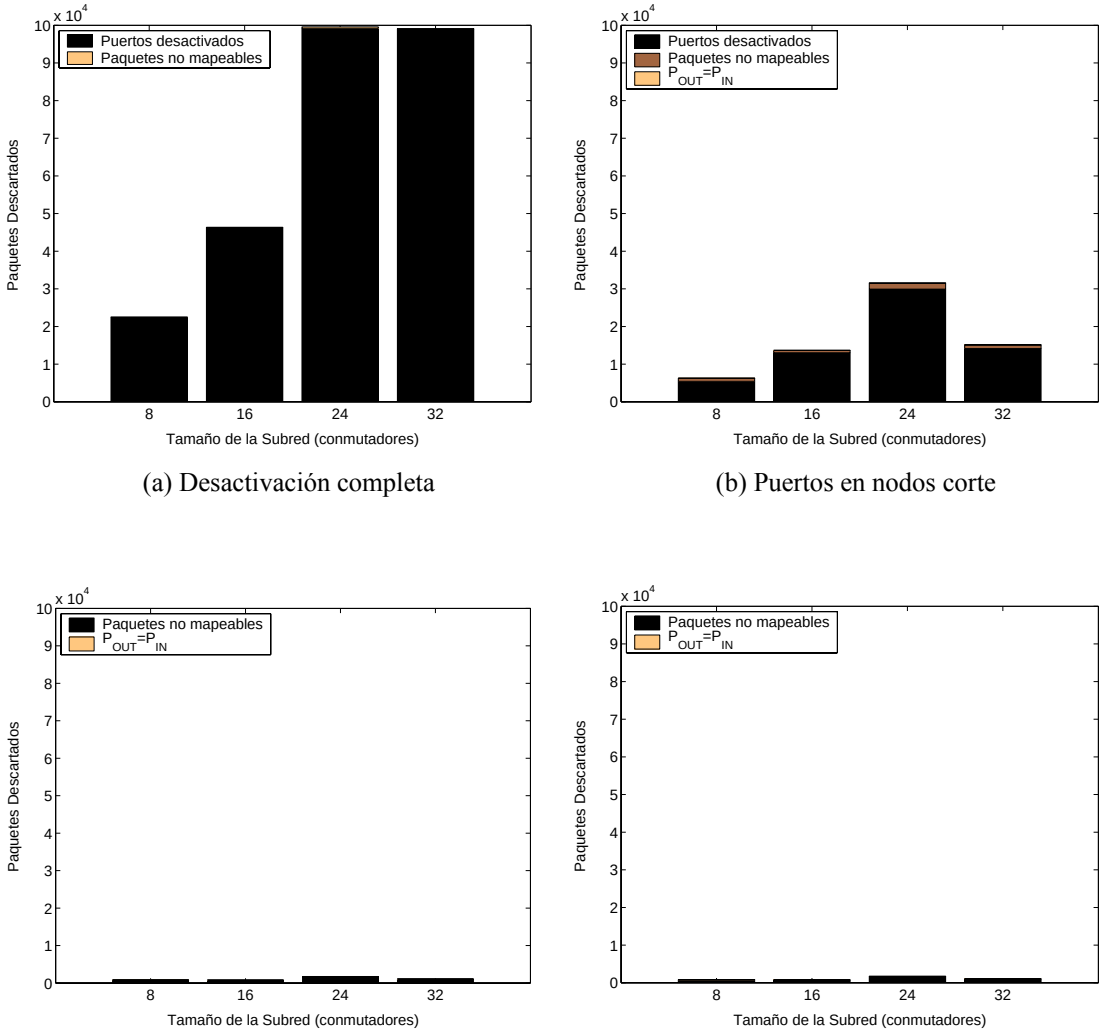
Tabla 25. Valores medios y desviaciones estándar para el número de paquetes descartados.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Técnica de distribución**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Desactivación completa	22,547.00	1,875.26	42,040.71	5,670.88
	Puertos en nodos corte	6,252.57	1,400.69	26,641.29	4,607.88
	Depend. En nodos corte	1,853.00	221.97	24,353.71	6,960.63
	Sin desactivación	1,917.67	197.56	23,713.71	6,102.96
16	Desactivación completa	46,365.80	1,079.83	172,831.67	78,518.34
	Puertos en nodos corte	13,682.67	3,075.68	128,875.13	55,562.64
	Depend. En nodos corte	1,993.50	1,429.88	117,324.47	55,492.40
	Sin desactivación	2,022.17	1,426.19	112,078.00	49,791.93
24	Desactivación completa	99,590.43	1,135.89	285,353.17	83,774.09
	Puertos en nodos corte	31,600.22	3,598.76	219,504.61	87,123.53
	Depend. En nodos corte	2,216.83	1,523.64	197,950.91	91,525.49
	Sin desactivación	2,242.89	1,556.80	193,230.65	86,470.67
32	Desactivación completa	99,143.30	4,400.65	489,517.29	303,020.09
	Puertos en nodos corte	15,202.47	8,109.04	407,008.87	310,187.77
	Depend. En nodos corte	2,651.92	1,569.12	400,437.03	313,356.84
	Sin desactivación	2,814.75	1,629.16	396,729.77	308,098.70

Tanto en la Figura 95(a) como en la Figura 96(b) podemos apreciar una mejora considerable cuando sólo se desactivan los puertos de los nodos corte, debido principalmente a que el número de paquetes descartados en puertos que han sido "manualmente" desactivados disminuye significativamente. La cantidad de paquetes descartados disminuye todavía más cuando se emplea la técnica basada en la desactivación de dependencias en nodos corte. En este caso, los paquetes descartados en puertos desactivados han sido sustituidos por aquellos paquetes descartados por el proceso de mapeo de SL a VL (transiciones prohibidas). Como su-

cedía con el tiempo y los SMPs de distribución de tablas, los resultados proporcionados por esta técnica son muy similares a los obtenidos aplicando una técnica de distribución dinámica.

Es importante señalar que existen otras causas "menores" de descarte. Por ejemplo, durante el periodo transitorio, hay paquetes que no pueden ser encaminados porque el DLID correspondiente no aparece en la tabla de encaminamiento, o porque el puerto de salida devuelto por la tabla coincide (temporalmente) con el puerto de entrada empleado por el paquete. La Figura 97 y la Figura 98 muestran los resultados de la Figura 95(a) y la Figura 95(b), respectivamente, pero distinguiendo la aportación de las distintas causas de descarte en cada caso.



(a) Desactivación completa

(b) Puertos en nodos corte

(c) Dependencias en nodos corte

(d) Sin desactivación

Figura 97. Paquetes descartados por los distintos mecanismos de distribución, distinguiendo las causas de descarte (activación).

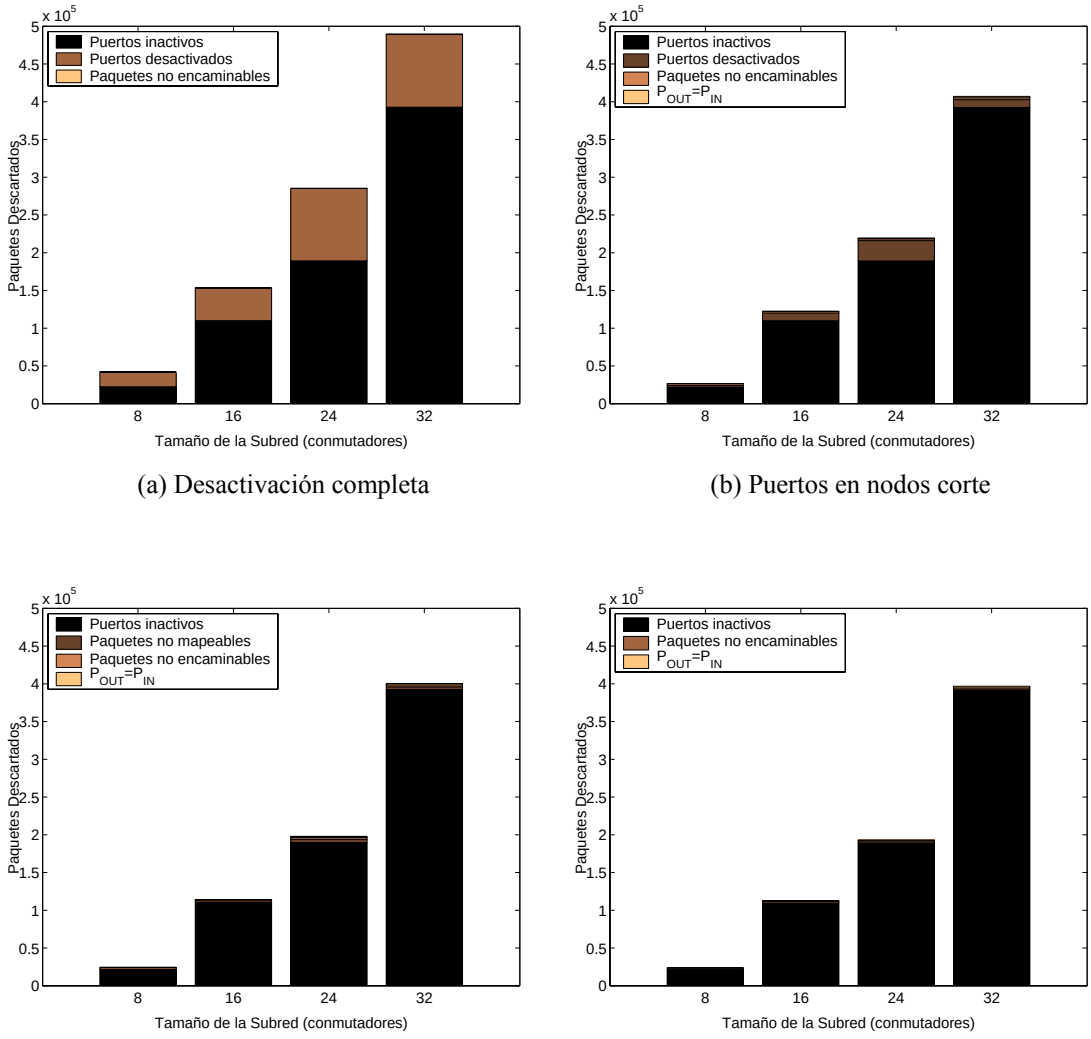


Figura 98. Paquetes descartados por los distintos mecanismos de distribución, distinguiendo las causas de descarte (desactivación).

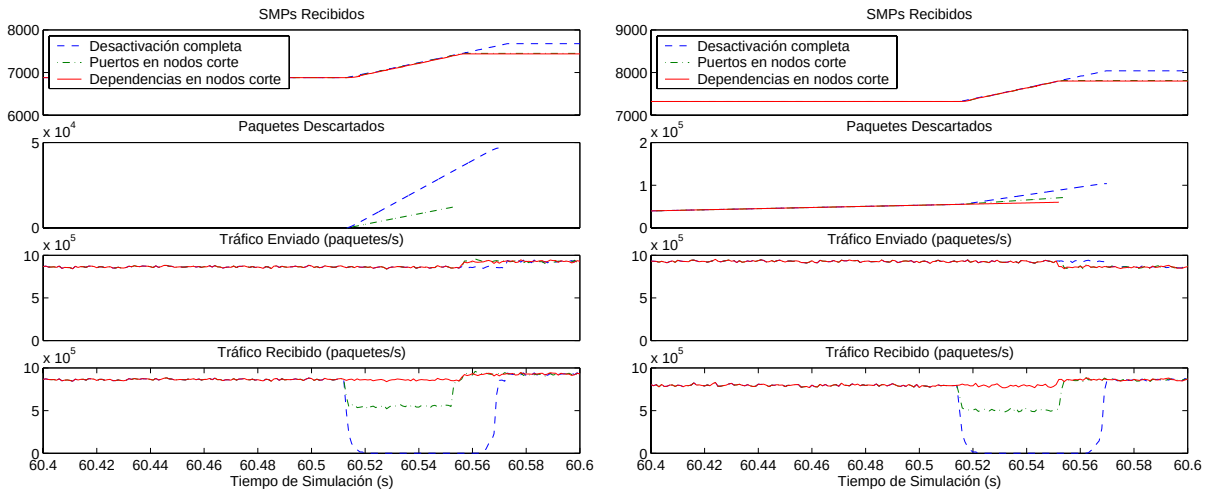
Comportamiento instantáneo

Las siguientes figuras muestran cómo afecta la técnica de distribución al tráfico de aplicación, para tres subredes irregulares con 30 (Figura 99), 46 (Figura 100) y 72 (Figura 101) nodos en total, y ante la ocurrencia de los dos tipos de cambio considerados (activación y desactivación de un conmutador). Obsérvese que estas gráficas sólo muestran un detalle de la fase de distribución de las tablas, en lugar del proceso de asimilación completo (el cambio topológico se produce en el instante 60.1 s).

En todas las gráficas, el eje horizontal representa el tiempo de simulación. Como en la Figura 84 y en la Figura 85, la primera y segunda gráficas representan, respectivamente, la cantidad de SMPs intercambiados por las entidades de gestión y la cantidad de paquetes descartados durante la simulación.

El "escalón" en la gráfica de SMPs recibidos corresponde exactamente con la fase de distribución de tablas. En cuanto a la gráfica de paquetes descartados, cuando el cambio considerado es una activación, el periodo de tiempo durante el cual los paquetes están siendo descartados coincide con el proceso de distribución. Sin embargo, cuando el cambio consiste en una desactivación, los paquetes comienzan a descartarse desde el mismo momento en que se produce el cambio (debido a la existencia de puertos inactivos en la subred).

Las dos últimas gráficas en estas figuras muestran la productividad instantánea de la red, a través del tráfico enviado y del tráfico recibido en cada momento.

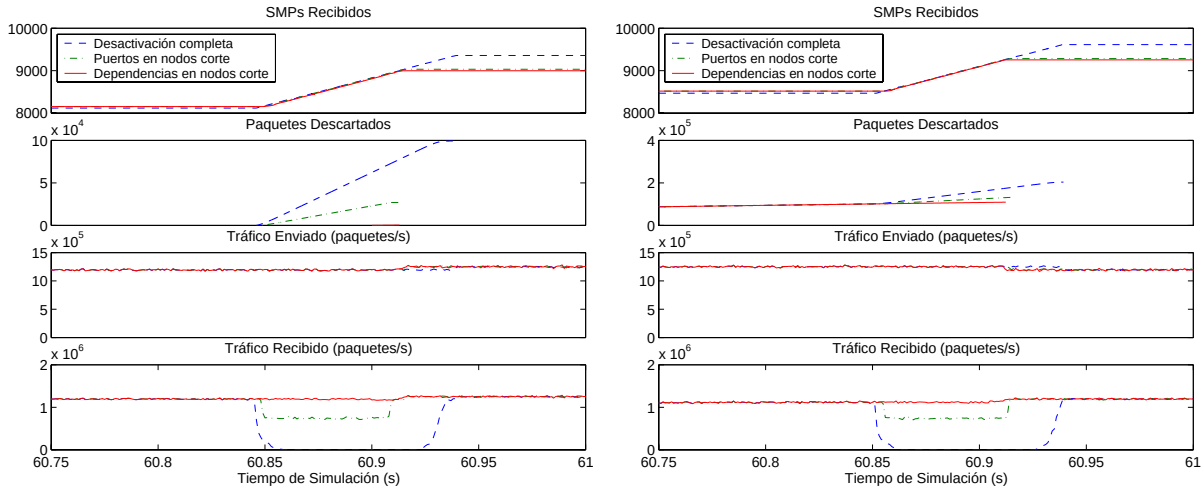


(a) Activación (b) Desactivación

Figura 99. Impacto del proceso de distribución de tablas sobre el tráfico de aplicación (subred con 16 conmutadores y 14 nodos terminales).

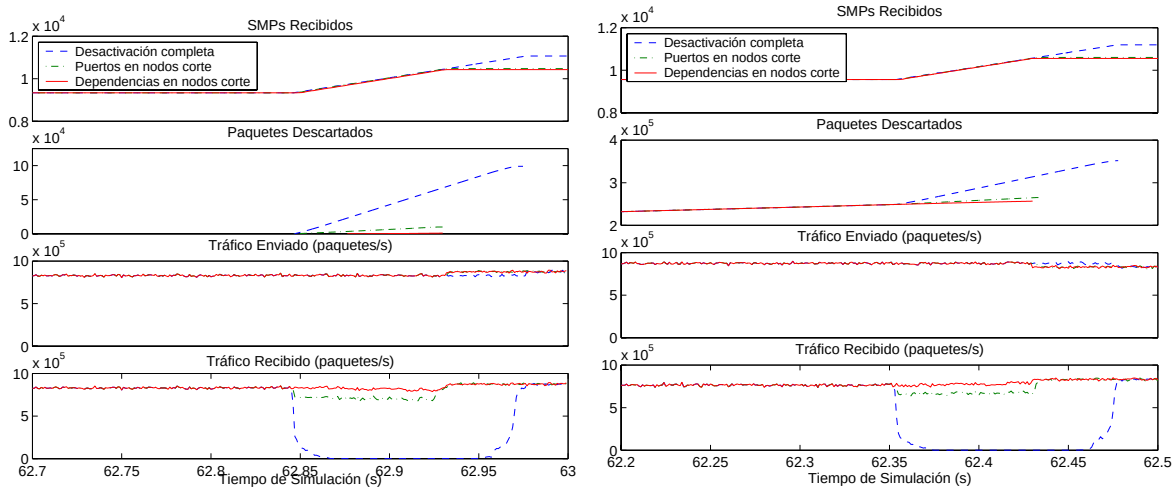
Los resultados mostrados en estas tres figuras son muy similares (cualitativamente). En las dos gráficas superiores observamos (de nuevo) una importante reducción en la cantidad de SMPs y tiempo de distribución, y sobre el número de paquetes descartados, cuando se aplican los mecanismos de distribución propuestos.

En las dos gráficas inferiores podemos ver que la distribución estática de las tablas (desactivación completa) tiene un efecto muy negativo sobre la productividad de la red. Podemos apreciar además que las técnicas de distribución optimizadas mejoran considerablemente este comportamiento. De hecho, cuando se emplea el mecanismo basado en la desactivación de dependencias en nodos corte, el "valle" en la gráfica de paquetes recibidos desaparece completamente, tal y como sucedería si empleásemos una técnica de distribución dinámica.



(a) Activación (b) Desactivación

Figura 100. Impacto del proceso de distribución de tablas sobre el tráfico de aplicación (subred con 24 conmutadores y 22 nodos terminales).



(a) Activación (b) Desactivación

Figura 101. Impacto del proceso de distribución de tablas sobre el tráfico de aplicación (subred con 32 conmutadores y 40 nodos terminales).

7.4 Aceleración del proceso de distribución

Todos los resultados presentados hasta el momento han sido obtenidos asumiendo que el proceso de activación de la subred se lleva a cabo por medio de SMPs con encaminamiento dirigido. Además, también hemos supuesto que el SM envía a cada conmutador tantos SMPs de distribución como sean necesarios para actualizar todas las entradas implementadas en su

tabla de encaminamiento. En esta sección intentaremos acelerar el proceso de distribución de tablas empleando técnicas alternativas en cada caso.

7.4.1 Activación de la subred con encaminamiento basado en destino

Podría pensarse que una forma de acelerar el proceso de distribución de tablas (y por tanto el proceso completo de asimilación del cambio) podría consistir en emplear SMPs con encaminamiento en base a destino en la última etapa del mismo, es decir, la activación de puertos (o dependencias). Dado que las nuevas tablas de encaminamiento ya han sido cargadas en los conmutadores, en principio no habría ningún problema en emplearlas a la hora de encaminar los SMPs de activación de la subred.

La Figura 102 (izquierda) compara la duración del proceso de distribución completo al emplear los dos tipos de encaminamiento permitidos para los SMPs, a la hora de activar los puertos de la subred, asumiendo el mecanismo de distribución de tablas basado en la desactivación completa de la subred, y ante un evento de desactivación de uno de los conmutadores. Los resultados para activación son muy similares.

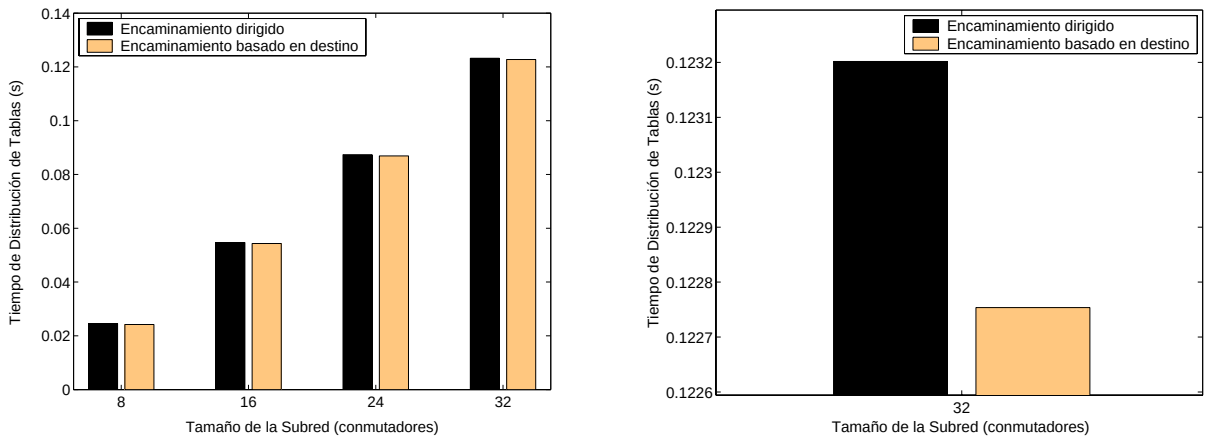


Figura 102. Tiempo de distribución de tablas en función del mecanismo de encaminamiento de los SMPs de activación de la subred y del tamaño de la misma.

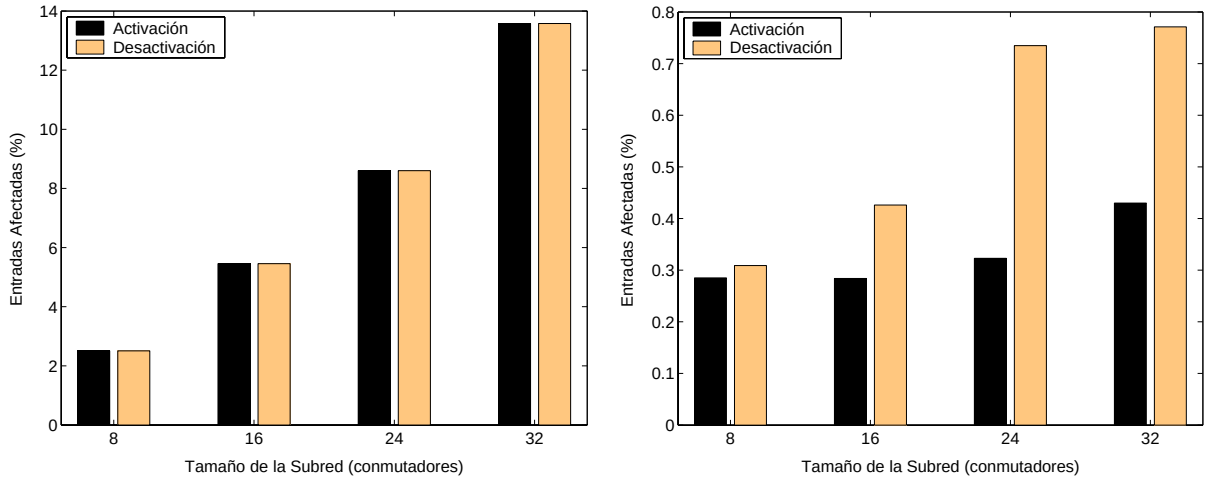
Las dos series alcanzan valores bastante parecidos. En la Figura 102 (derecha) se muestra un detalle de los resultados para una red con 32 conmutadores. En este caso, puede apreciarse una ligera reducción del tiempo cuando el encaminamiento está basado en el destino del SMP.

Por lo tanto, vemos que hay una aportación al proceso de distribución completo, pero muy reducida, ya que estamos acelerando únicamente una de sus etapas. Además, no se trata precisamente de la etapa de mayor duración.

7.4.2 Distribución parcial de entradas

Como es lógico, la etapa de mayor duración dentro del proceso de distribución coincide con el envío de las tablas propiamente dicho. Por otro lado, hemos observado que la cantidad de entradas en tablas de encaminamiento que realmente varían con respecto a la configuración anterior es muy reducida.

En este sentido, la Figura 103 muestra la cantidad de entradas en tablas afectadas por el cambio, en función del mecanismo de descubrimiento de la topología (Capítulo 5), el tipo de cambio considerado y el tamaño de la subred. Estamos considerando que cada conmutador implementa una tabla de tipo LFT con 1024 entradas. Distinguimos aquí entre descubrimiento completo y parcial porque al aplicar el primero de ellos, el SM asigna nuevos LIDs a los dispositivos de la subred, independientemente de si éstos existían o no en la configuración anterior. Por tanto, los resultados para este caso serán claramente superiores.



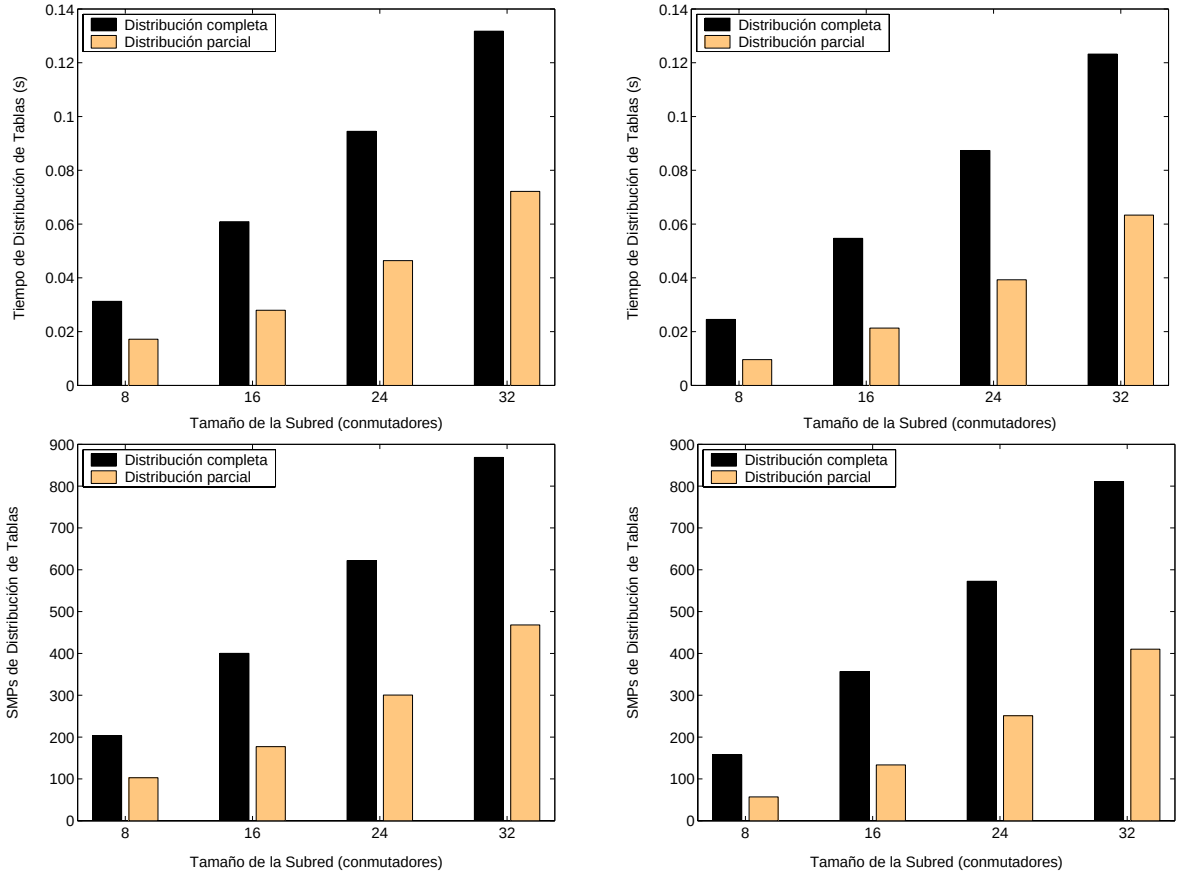
(a) Descubrimiento completo de la topología (b) Descubrimiento parcial de la topología

Figura 103. Porcentaje medio de entradas en tablas de encaminamiento afectadas por el cambio topológico.

En cualquier caso, se observa que el porcentaje de entradas afectadas por el cambio es pequeño, o muy pequeño (en el caso del descubrimiento parcial). Por tanto, parece que no tiene sentido volver a distribuir todas las entradas cada vez que se produce un cambio topológico en la subred. Bastaría con distribuir únicamente aquellas entradas que han cambiado. En concreto, las entradas correspondientes a aquellos destinatario (DLIDs) que han aparecido o desaparecido en la nueva configuración, o a aquellos destinatarios para los que el nuevo puerto de salida difiere del anteriormente asignado. Evidentemente, esto implica que el SM almacene completamente la configuración previa de las tablas, y haga las pertinentes comparaciones a la hora de distribuir las nuevas.

La Figura 104 muestra una importante reducción (en torno al 50%) en el tiempo y la cantidad de SMPs de distribución cuando se lleva a cabo una distribución parcial de las entra-

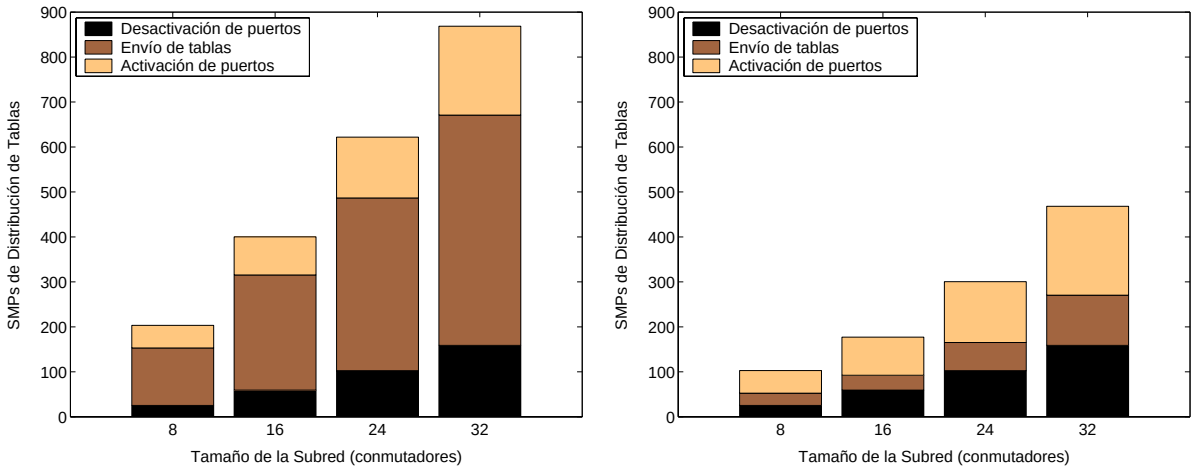
das. De nuevo, hemos considerado el mecanismo de distribución basado en la desactivación completa de la subred. Obviamente, la distribución parcial de entradas puede aplicarse con independencia de la técnica de evitación de bloqueos que se emplee durante la distribución de las tablas.



(a) Activación (b) Desactivación

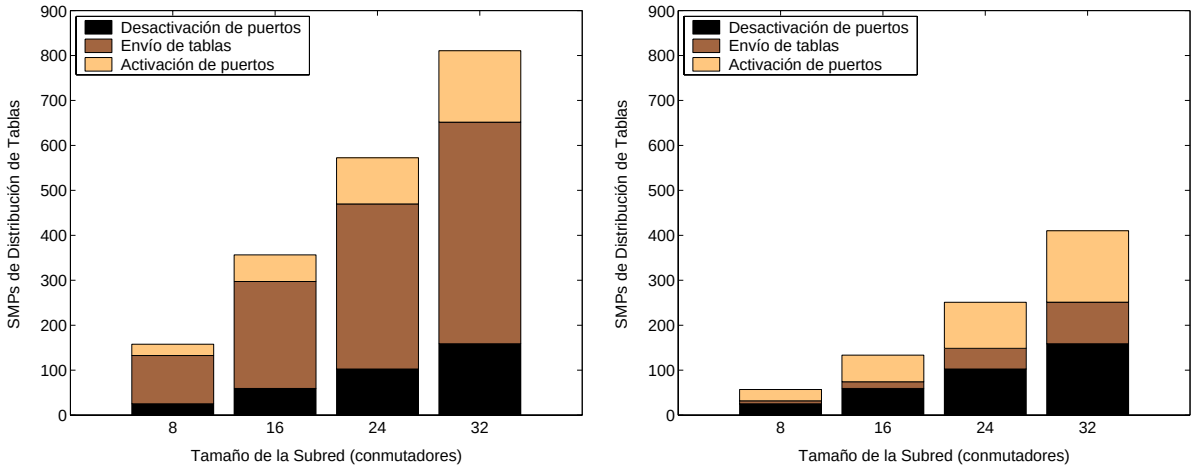
Figura 104. Tiempo y SMPs de distribución de tablas,
en función del tamaño de la subred y del tipo de cambio.

Esta mejora en el tiempo total de distribución se debe a una importante reducción en el número de SMPs de la etapa de envío de tablas. La Figura 105 y Figura 106 muestran nuevamente los SMPs empleados durante el proceso completo de distribución de tablas (Figura 104), pero distinguiendo entre las distintas etapas del proceso.



(a) Distribución completa (b) Distribución parcial

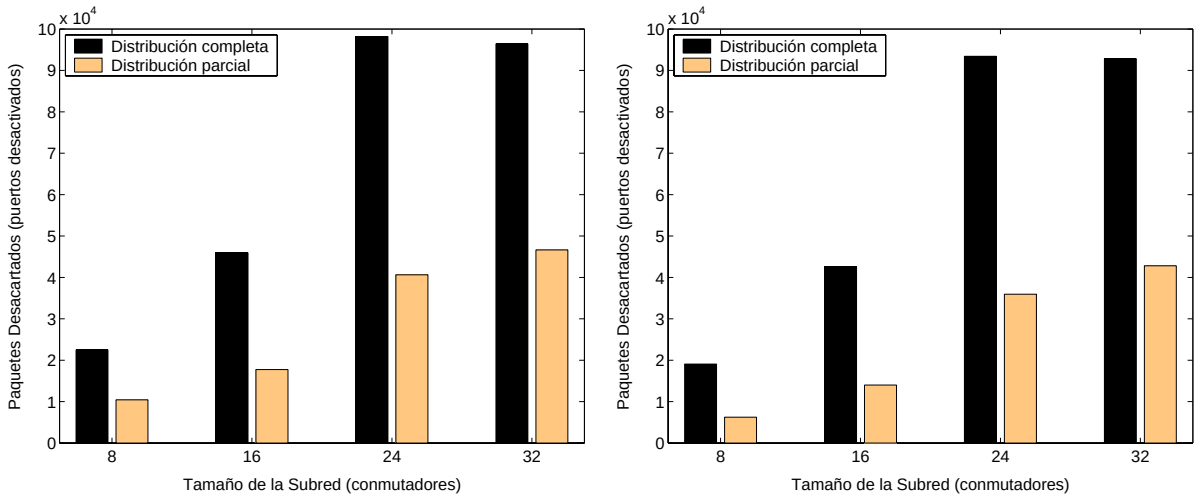
Figura 105. SMPs empleados en cada una de las fases del proceso de distribución (activación).



(a) Distribución completa (b) Distribución parcial

Figura 106. SMPs empleados en cada una de las fases del proceso de distribución (desactivación).

Lógicamente, al acortar el proceso de envío de las tablas, el número de paquetes descartados en aquellos puertos desactivados por el proceso de distribución también se reduce. La Figura 107 muestra la cantidad de paquetes descartados en puertos desactivados durante el proceso de distribución de tablas, para el mecanismo de distribución basado en la desactivación completa de la subred. Se aprecia una mejora superior al 50%.



(a) Activación (b) Desactivación

Figura 107. Paquetes descartados en puertos desactivados.

7.5 Conclusiones

En este capítulo hemos presentado dos mecanismos libres de bloqueo para la distribución de las tablas de encaminamiento en InfiniBand. Estos mecanismos han sido directamente derivados a partir del proceso de reconfiguración estática "tradicional", alcanzando unas prestaciones casi idénticas a la reconfiguración dinámica.

Hemos comprobado que el impacto del proceso de distribución sobre el tráfico de aplicación es prácticamente anulado. En otras palabras, la ejecución del proceso de asimilación del cambio es mucho más transparente para las aplicaciones. Esto se debe a que nuestras propuestas logran una conectividad casi total entre los elementos de la subred, durante el periodo de distribución de las nuevas tablas de encaminamiento.

Además, las técnicas propuestas se ajustan perfectamente a la especificación de InfiniBand, pudiendo ser fácilmente implementadas en sistemas reales, sin necesidad de introducir modificación alguna en el estándar.

Finalmente, hemos visto dos formas de acelerar el proceso de distribución de tablas de encaminamiento. Obviamente, al reducir el tiempo de distribución, se reduce todavía más el impacto que el proceso de asimilación completo pudiera tener sobre las aplicaciones.

Capítulo 8
Evaluación conjunta de las propuestas

8.1 Introducción

En los capítulos anteriores de esta memoria se han descrito una serie de propuestas encaminadas a optimizar las tareas más importantes del proceso de gestión de cambios en InfiniBand; la exploración de la nueva topología, la obtención de rutas acordes a dicha topología y la distribución de las tablas de encaminamiento. Las técnicas propuestas han sido evaluadas, en mayor o menor medida, de forma individual.

En este capítulo evaluaremos el comportamiento de un mecanismo de gestión que incorpora simultáneamente todas estas mejoras. Este mecanismo, que denominaremos "mejorado", será comparado con un mecanismo "básico", compatible también con la especificación de InfiniBand. La Tabla 26 resume las características diferenciadoras de ambos esquemas. Nótese que el mecanismo mejorado obtiene y distribuye dos conjuntos de rutas. Las rutas provisionales se distribuyen empleando el mecanismo basado en la desactivación de dependencias en nodos corte, mientras que las definitivas son enviadas dinámicamente.

Tabla 26. Características de los mecanismos de gestión evaluados.

Mecanismo	Exploración	Cálculo de rutas	Distribución de tablas
Básico	Completa	FERa	Desactivación completa de la subred
Mejorado	Parcial	1) PIRa ; 2) FERa	1) Dependencias en nodos corte ; 2) Dinámica

En cuanto a la técnica de detección de los cambios, en primer lugar se mostrarán los resultados obtenidos asumiendo que ambos mecanismos utilizan un protocolo basado únicamente en barridos periódicos de la topología. Al final del capítulo veremos en qué medida afecta al proceso de asimilación la activación del mecanismo de notificaciones.

8.2 Metodología de simulación

La metodología empleada en esta evaluación comparativa es muy similar a la utilizada en el resto de estudios de este trabajo. La Tabla 27 recopila los datos más relevantes.

Tabla 27. Parámetros de simulación.

Categoría	Parámetro	Valor
Topología de la subred	Tipo	redes irregulares red clos (3, 1, 4)
	Tamaño (conmutadores)	8, 16, 24, 32 (redes irregulares) 7 (red clos)
Enlaces	Ancho de banda	1X / 4X
	Retardo de propagación	100 ns
Puertos en HCAs y conmutadores	Velocidad	1X / 4X
	Número de VLs de datos	2 (VL0, VL1)
	MTU soportada	256 bytes
	Tamaño de la tabla VLAT	64 + 64 entradas
	Política de asignación del enlace	*round-robin*
	Tiempo de arbitraje (enlace)	40 ns
Tarjetas de interfaz (HCAs)	Número de puertos	1
	Tamaño de los buffers asociados a los VLs	(ilimitado)
	Esquema de mapeo de SL a VL	asignación cíclica
	Tiempo de mapeo	20 ns
Conmutadores	Número de puertos	4
	Tamaño de los buffers asociados a los VLs	4 Kbytes
	Ancho de banda interno	1X / 4X
	Tamaño de la tabla LFT	1024 entradas
	Tiempo de encaminamiento	40 ns
	Esquema de mapeo de SL a VL	asignación cíclica
	Tiempo de mapeo	20 ns
	Tiempo de arbitraje (*crossbar*)	40 ns
	Tiempo de configuración del *crossbar*	2 ns
	Tiempo de cruce	4 ns
Fuentes de paquetes	Distribución de destinos (DLID)	Uniforme
	Nivel de servicio (SL)	Aleatorio
	MTU de los paquetes	256 bytes
	Tasa de generación de paquetes	25% de la tasa de saturación 50% de la tasa de saturación
Eventos durante la simulación	Encendido completo de la subred	10 s
	Activación de las fuentes	60 s
	Cambio topológico (activación/desactivación)	60.1 s
Detección de cambios	Mecanismo de detección	Barrido Barrido y notificaciones
	Tiempo entre barridos	100 x tiempo de barrido de la subred

8.3 Comportamiento instantáneo

Las gráficas de la Figura 108 y la Figura 109 muestran el comportamiento instantáneo y el efecto sobre el tráfico de aplicación de los mecanismos de gestión básico y mejorado. En ambos casos, los resultados corresponden a una subred con topología irregular, compuesta por 32 conmutadores y 40 nodos terminales, y con una tasa de generación de paquetes equivalente al 25% de la tasa de saturación de la subred. El evento considerado es la activación (Figura 108) y desactivación (Figura 109) de uno de los conmutadores de la subred. A diferencia de los resultados mostrados en la Figura 99 (página 160), mostramos el proceso de asimilación completo.

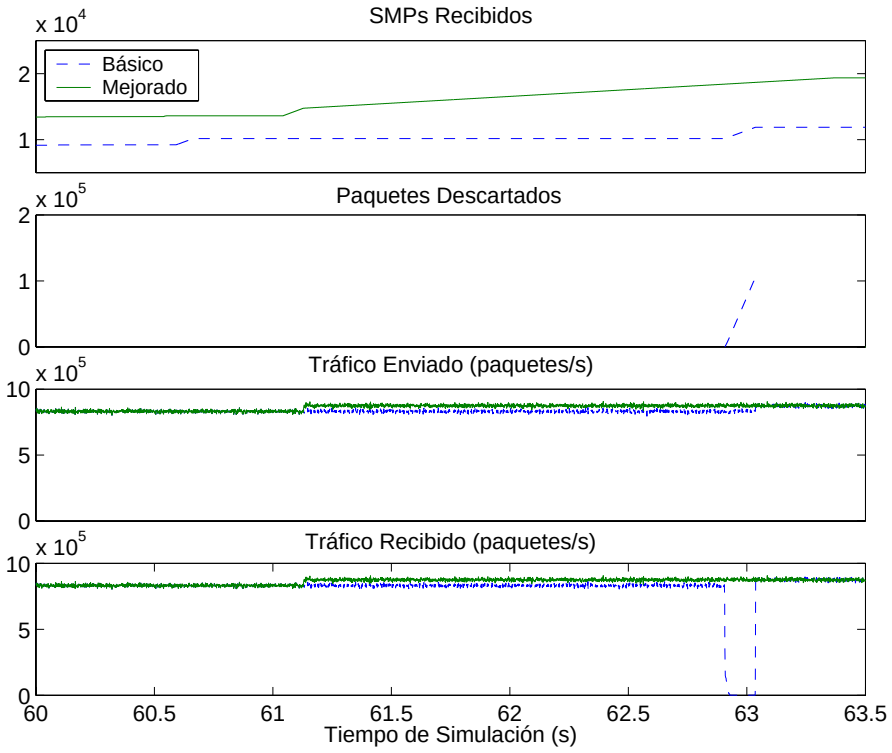


Figura 108. Comportamiento de los mecanismos de gestión ante un evento de activación.

En las gráficas de SMPs recibidos por el mecanismo básico podemos apreciar claramente los paquetes correspondientes a las fases de exploración (primer "escalón") y de distribución de tablas (segundo "escalón"). En el caso del mecanismo mejorado, apenas pueden apreciarse los SMPs de exploración (parcial) de la topología. Por otro lado, los procesos de distribución, primero de tablas provisionales y después de tablas definitivas, están claramente identificados mediante dos cambios en la pendiente de la gráfica.

Hay que matizar aquí que, en estas primeras simulaciones, la distribución de las tablas provisionales se lleva a cabo de forma completa. Es decir, el SM envía a cada conmutador todas las entradas implementadas. En una sección posterior veremos los resultados cuando sólo se actualizan las entradas que han cambiado con respecto a la configuración previa.

En cuanto al descarte de paquetes, en caso de activación (Figura 108), el mecanismo básico descarta, aproximadamente, 106000 paquetes. Este descarte coincide con la ejecución de la tarea de distribución de las nuevas tablas (en torno al instante 63 s). Por otra parte, en esta simulación concreta el mecanismo mejorado no descarta ningún paquete.

En caso de desactivación (Figura 109), ambos mecanismos comienzan a descartar paquetes exactamente en el instante 60.1 s, coincidiendo con la ocurrencia del cambio. El mecanismo básico sigue descartando paquetes (361000 en total) hasta que distribuye las nuevas tablas, en torno al instante 62.5. El mejorado deja de descartar paquetes (84000 en total) una vez distribuidas las tablas provisionales, poco antes del instante 61 s.

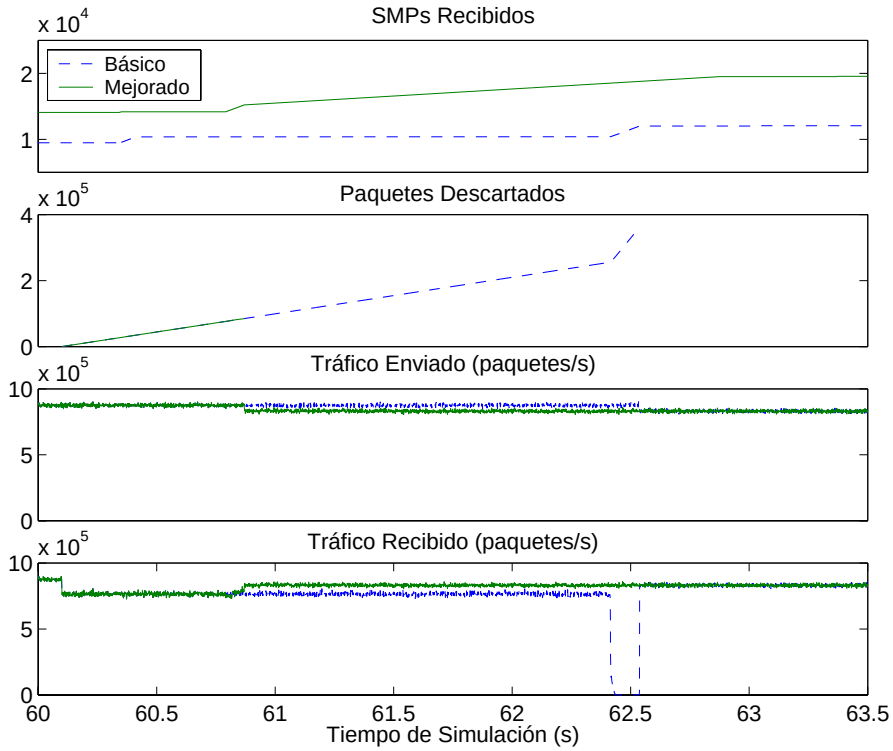
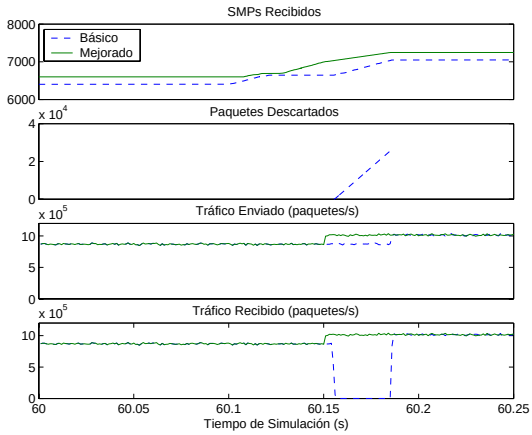


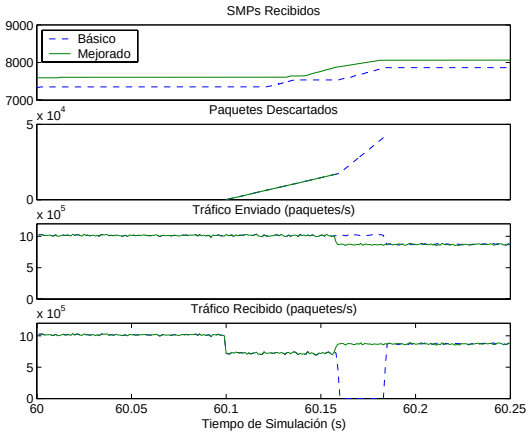
Figura 109. Comportamiento de los mecanismos de gestión ante un evento de desactivación.

En las gráficas de productividad instantánea de la Figura 108 y la Figura 109 vemos que el mecanismo de gestión mejorado no sólo elimina el "valle" en la gráfica de tráfico recibido, sino que recupera la conectividad de la subred mucho antes (casi 2 segundos antes en caso de activación y más de 1.5 segundos antes en caso de desactivación).

Para terminar, mostramos también los resultados instantáneos para subredes irregulares con 15 (Figura 110), 30 (Figura 111) y 46 (Figura 112) nodos en total. Los comentarios que acabamos de realizar para la subred de 32 conmutadores son extrapolables a estas otras topologías. Nótese que, aunque el factor de barrido programado para todas las simulaciones sea el mismo, el tiempo de detección del cambio es distinto en casi todos los casos, dado que depende del momento concreto en que se realiza el siguiente barrido de la topología. En cualquier caso, las ventajas del mecanismo mejorado se siguen apreciando claramente.

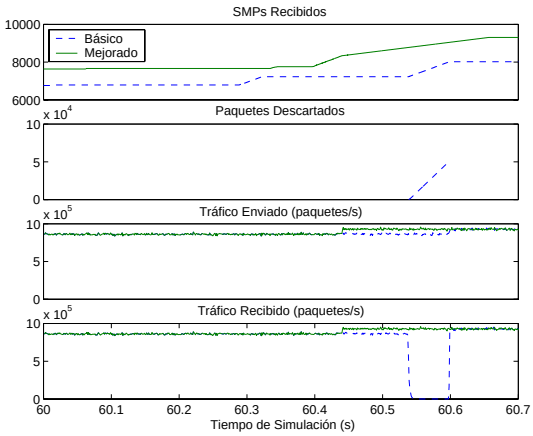


(a) Activación

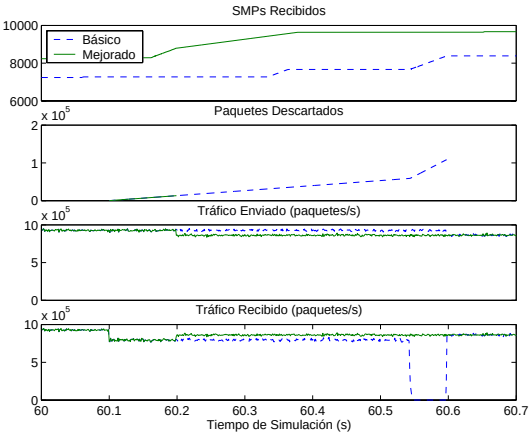


(b) Desactivación

Figura 110. Comportamiento instantáneo para una subred con 8 conmutadores y 7 terminales.

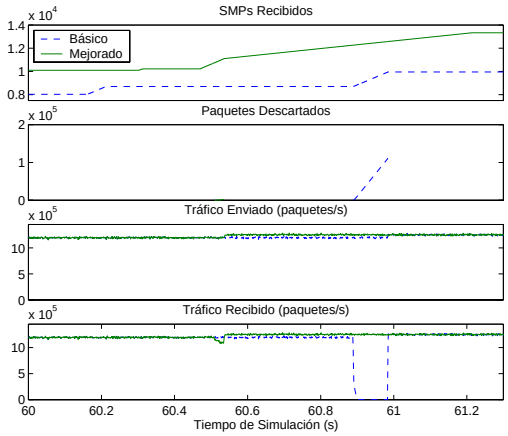


(a) Activación

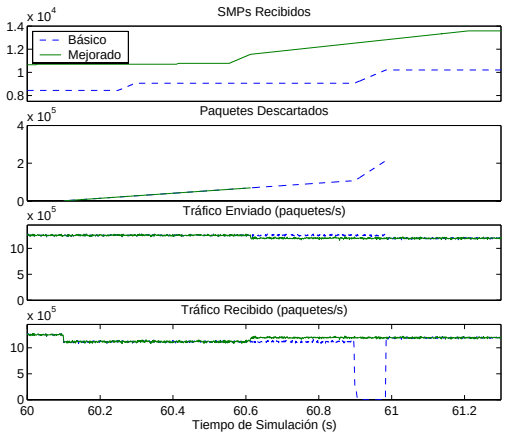


(b) Desactivación

Figura 111. Comportamiento instantáneo para una subred con 16 conmutadores y 14 terminales.



(a) Activación

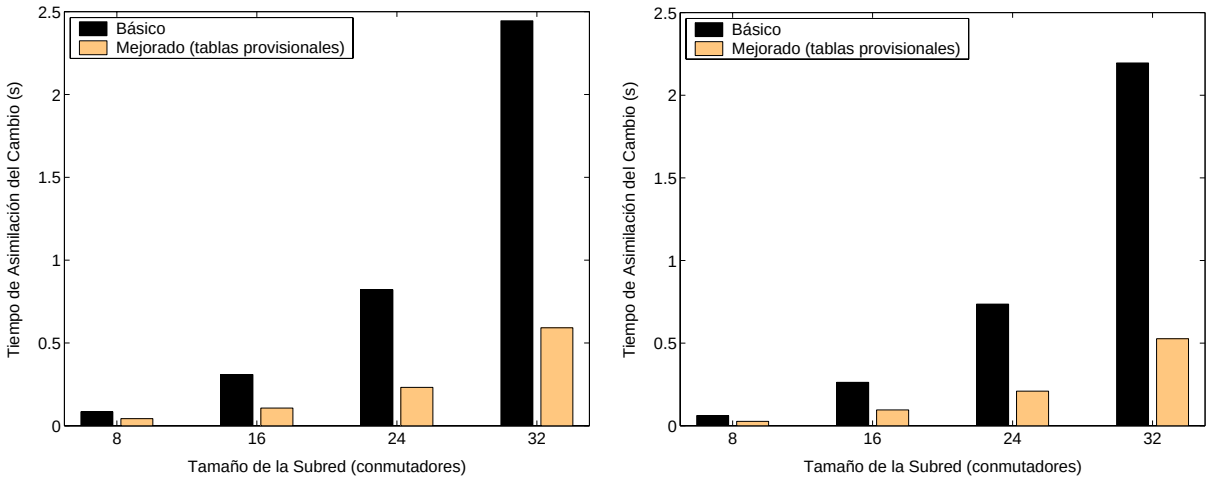


(b) Desactivación

Figura 112. Comportamiento instantáneo para una subred con 24 conmutadores y 22 terminales.

8.4 Tiempo de asimilación y paquetes de gestión

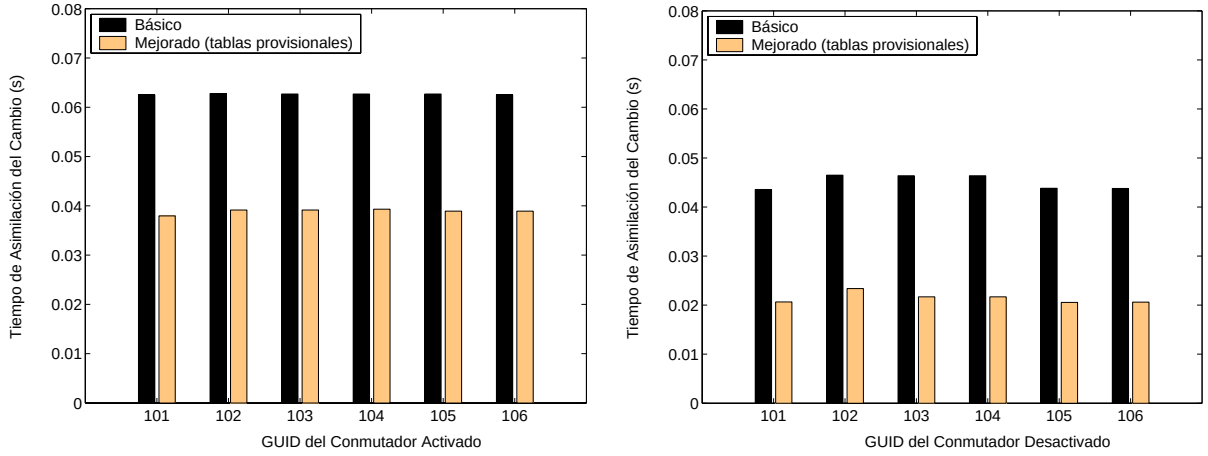
La Figura 113 muestra el tiempo de asimilación del cambio, una vez detectado éste, para diversas topologías irregulares. Para el mecanismo mejorado se ha considerado hasta el momento en que las tablas provisionales han sido completamente distribuidas.



(a) Activación (b) Desactivación

Figura 113. Tiempo hasta la distribución de las nuevas tablas, en función del tamaño de la subred y del tipo de cambio.

La Figura 114 muestra el mismo parámetro, pero en este caso los resultados corresponden a la red clos (3, 1, 4) mostrada en la Figura 62 (página 94). Los valores en el eje horizontal hacen referencia al GUID del conmutador que se activa o desactiva en cada simulación.



(a) Activación (b) Desactivación

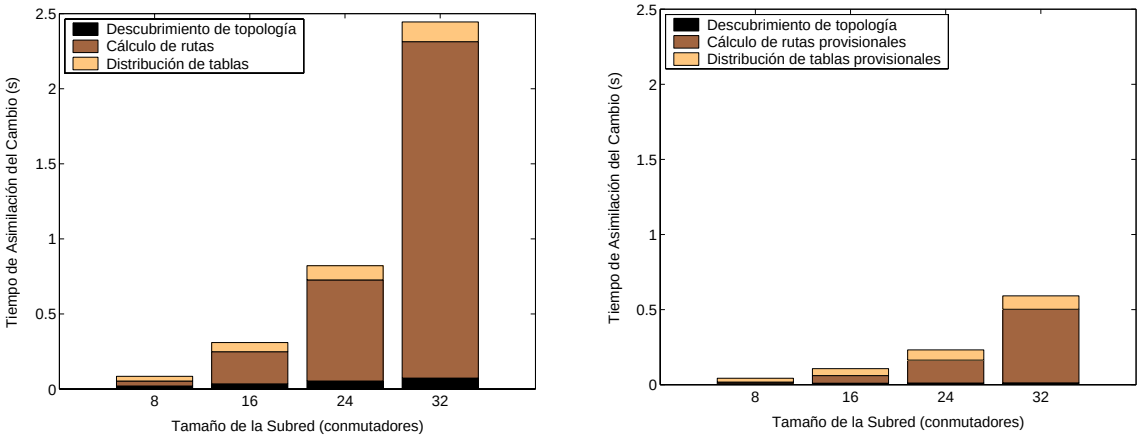
Figura 114. Tiempo hasta la distribución de las nuevas tablas para la red clos (3, 1, 4).

La Tabla 28 recoge los valores medios y las desviaciones estándar para los valores mostrados en la Figura 113.

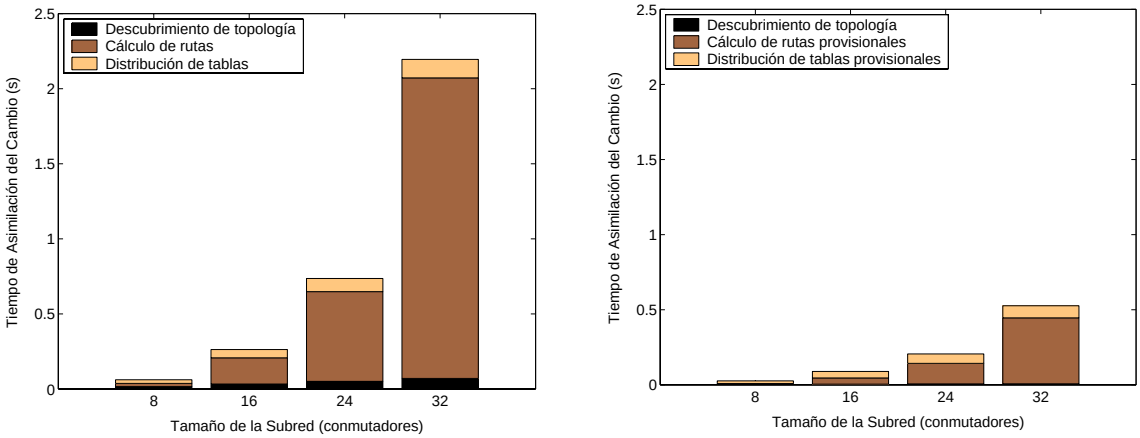
Tabla 28. Valores medios y desviaciones estándar para el tiempo de asimilación del cambio.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Mecanismo de gestión**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Básico	0.084322	0.001215	0.061461	0.007597
	Mejorado	0.043018	0.003124	0.026079	0.004010
16	Básico	0.309162	0.001077	0.262362	0.009467
	Mejorado	0.106540	0.003142	0.095271	0.027424
24	Básico	0.820756	0.001267	0.735567	0.011557
	Mejorado	0.231911	0.002292	0.209227	0.028033
32	Básico	2.444927	0.001758	2.195207	0.139962
	Mejorado	0.591356	0.003656	0.526156	0.032089

La aportación de cada una de las tareas de gestión al tiempo total de asimilación del cambio se muestra en la Figura 115, para el caso de las topologías irregulares evaluadas. Como hemos visto en los capítulos anteriores, el tiempo requerido por cada tarea se reduce. La mejora que más influye es la reducción en el tiempo de cálculo de las nuevas rutas, el proceso de mayor duración en ambos mecanismos.



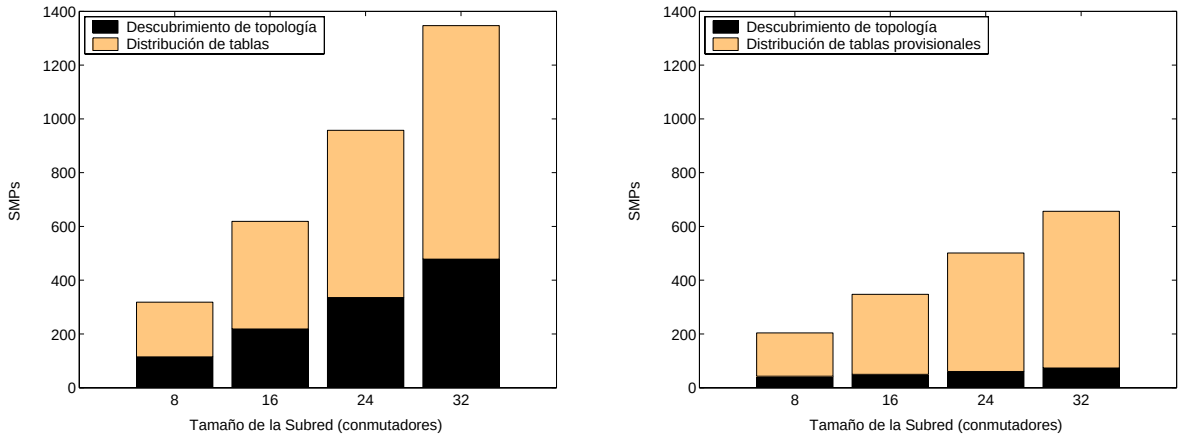
(a) Activación



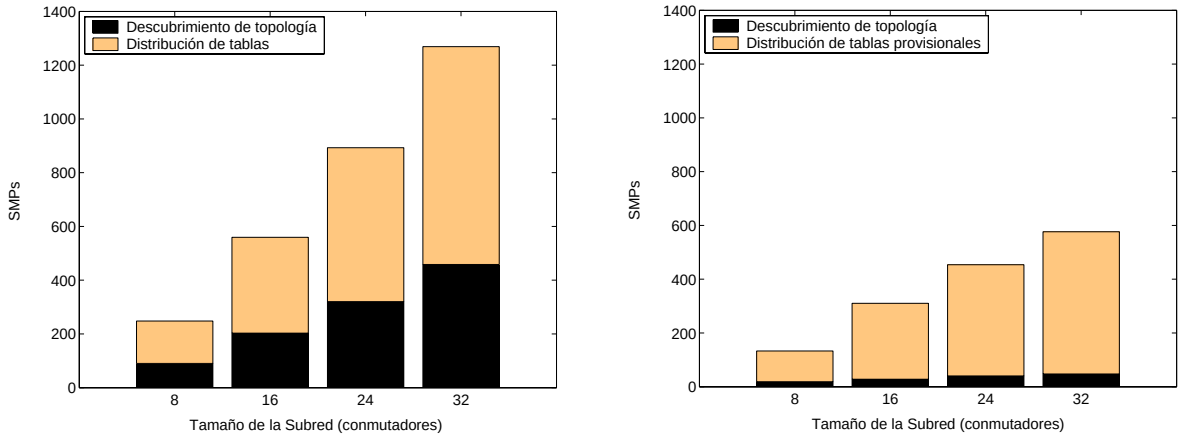
(b) Desactivación

Figura 115. Tiempo de asimilación para los mecanismos básico (izquierda) y mejorado (derecha), en función del tamaño de la subred y del tipo de cambio.

Por otro lado, la Figura 116 muestra el número de SMPs empleados para asimilar el cambio (una vez detectado) por parte de los dos mecanismos de gestión evaluados. En las gráficas se distingue entre los SMPs de descubrimiento de la topología y los SMPs de distribución de tablas. El mecanismo mejorado reduce ambos tipos de SMPs, si bien la mejora es más importante en el caso de los SMPs de descubrimiento.



(a) Activación

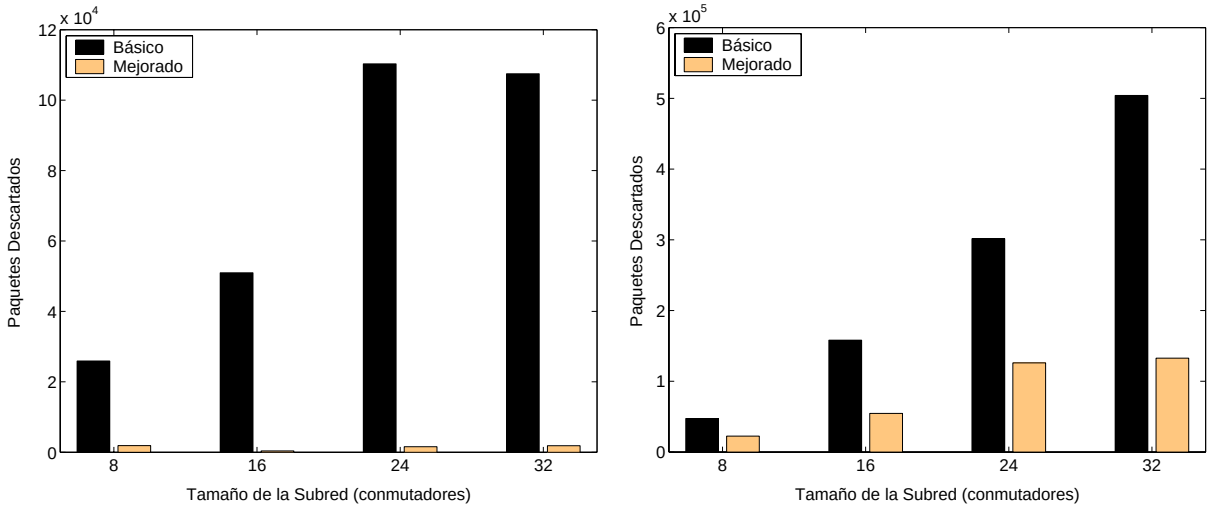


(b) Desactivación

Figura 116. SMPs utilizados por los mecanismos básico (izquierda) y mejorado (derecha), en función del tamaño de la subred y del tipo de cambio.

8.5 Paquetes descartados

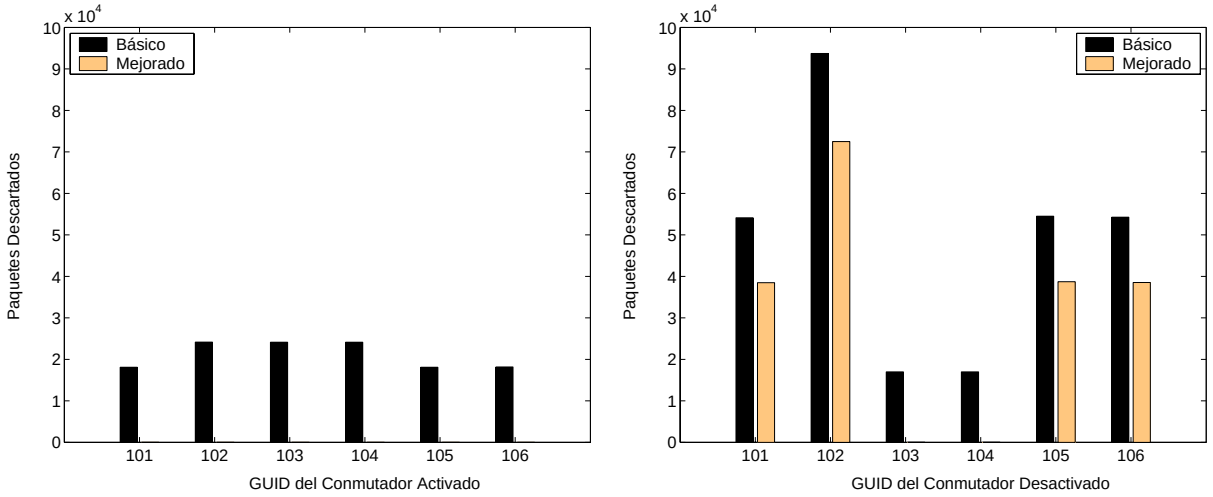
La Figura 117 muestra la cantidad total de paquetes descartados por los mecanismos de gestión básico y mejorado, para diversas topologías irregulares.



(a) Activación (b) Desactivación

Figura 117. Número de paquetes descartados por los mecanismos básico y mejorado, en función del tamaño de la subred y del tipo de cambio.

De nuevo, mostramos también los resultados para una red clos, en la Figura 118[1].



(a) Activación (b) Desactivación

Figura 118. Número de paquetes descartados para la red clos (3, 1, 4).

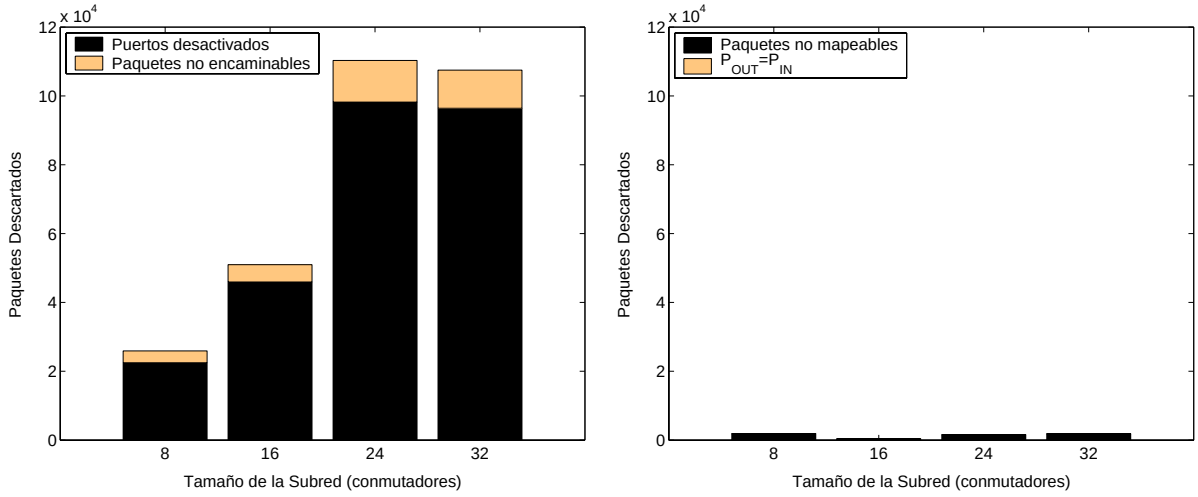
La Tabla 29 recoge los valores medios y las desviaciones estándar para los valores mostrados en la Figura 117.

[1] Obsérvese que, en los casos de desactivación de los conmutadores con GUIDs *103* y *104*, se obtienen unos valores anormalmente bajos. Esto se debe a que las rutas que siguen los paquetes dentro de la subred no atraviesan estos conmutadores, sino el conmutador *102*.

Tabla 29. Valores medios y desviaciones estándar para el número de paquetes descartados.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Mecanismo de gestión**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Básico	25,906.71	1,682.72	47,417.14	6,427.53
	Mejorado	4,400.33	764.90	22,447.86	6,434.12
16	Básico	50,959.20	1,283.82	158,143.00	48,846.44
	Mejorado	1,911.33	1,370.90	54,601.47	82,310.87
24	Básico	110,293.09	1,801.13	301,600.22	88,941.79
	Mejorado	4,012.22	2,405.09	126,093.39	59,216.76
32	Básico	107,490.55	4,611.99	504,065.81	311,468.81
	Mejorado	5,727.50	3,048.31	132,758.71	103,695.04

Para verlo con más detalle, la Figura 119 y la Figura 120 muestran los mismos resultados de la Figura 117, pero distinguiendo entre las distintas causas de descarte.

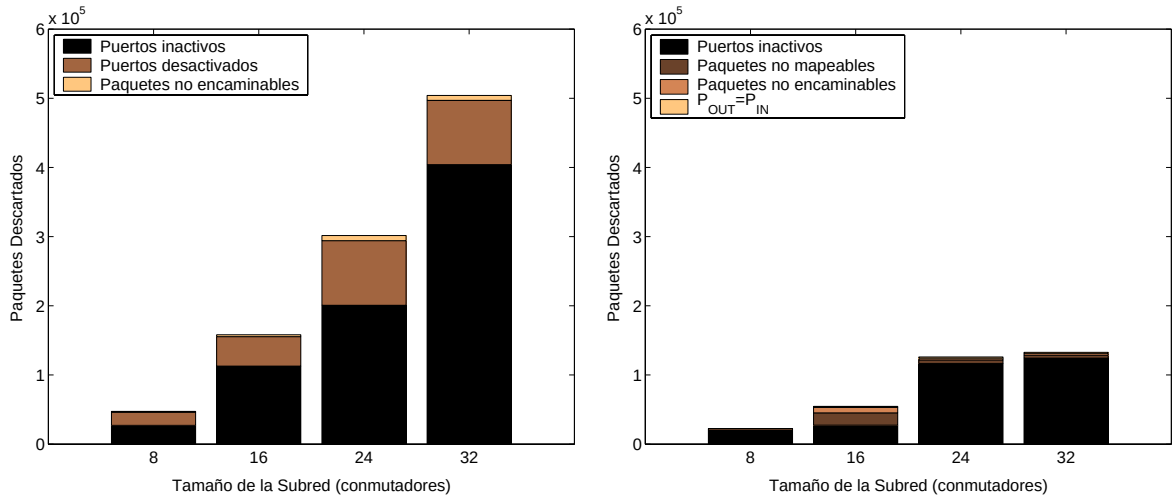


(a) Mecanismo básico (b) Mecanismo mejorado

Figura 119. Paquetes descartados por los mecanismos básico y mejorado, distinguiendo las causas de descarte (activación).

Vemos que, en caso de activación, el descarte prácticamente se anula, si bien todavía se pierden algunos paquetes. Este descarte residual se debe, principalmente, a la manipulación de las tablas de mapeo durante el proceso de distribución de tablas. Obsérvese que en el caso particular de la red clos (Figura 118), el mecanismo mejorado no descarta ningún paquete.

En caso de desactivación, el mecanismo mejorado consigue una reducción bastante notable, pero sigue existiendo un descarte apreciable, debido sobre todo a la presencia de puertos inactivos en la subred (antes del envío de las nuevas rutas).

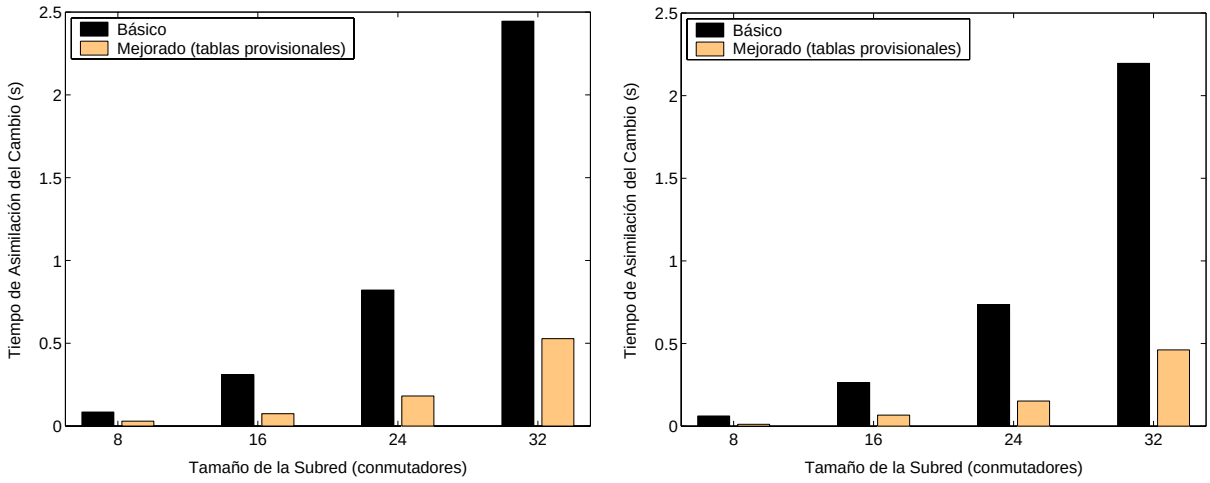


(a) Mecanismo básico (b) Mecanismo mejorado

Figura 120. Paquetes descartados por los mecanismos básico y mejorado, distinguiendo las causas de descarte (desactivación).

8.6 Distribución parcial de tablas provisionales

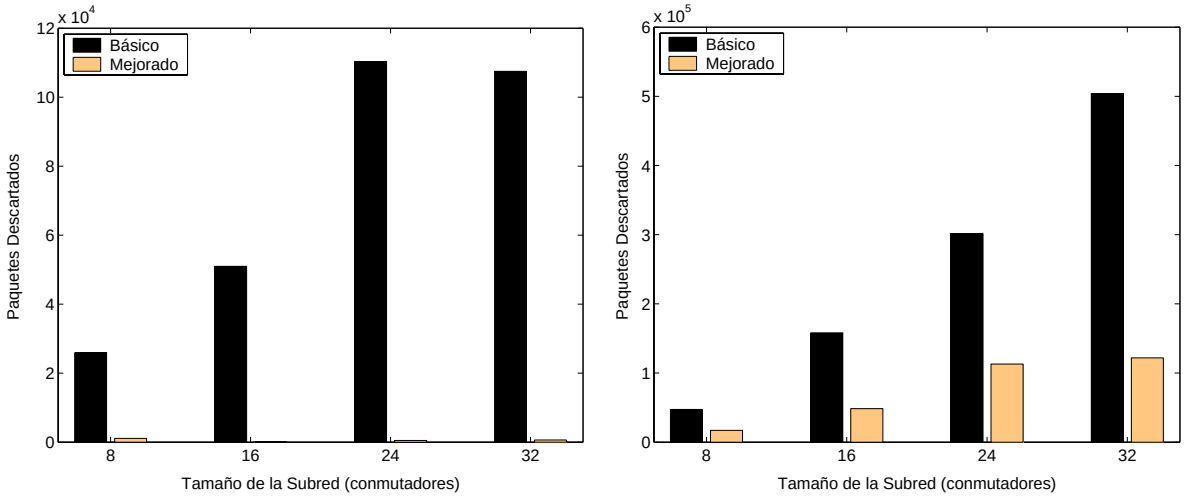
La Figura 121 y la Figura 122 muestran los mismos resultados que la Figura 113 y la Figura 117, respectivamente, pero empleando la mejora propuesta en la Sección 7.4.2 (página 163) en el caso del mecanismo de gestión mejorado.



(a) Activación (b) Desactivación

Figura 121. Tiempo hasta la distribución de las nuevas tablas, en función del tamaño de la subred y del tiempo de cambio.

Si comparamos estas figuras con las anteriores, podemos observar una ligera reducción en ambos parámetros, debida al envío parcial de las tablas. Sin embargo, dentro del proceso completo de asimilación del cambio, esta mejora no es demasiado importante.



(a) Activación (b) Desactivación

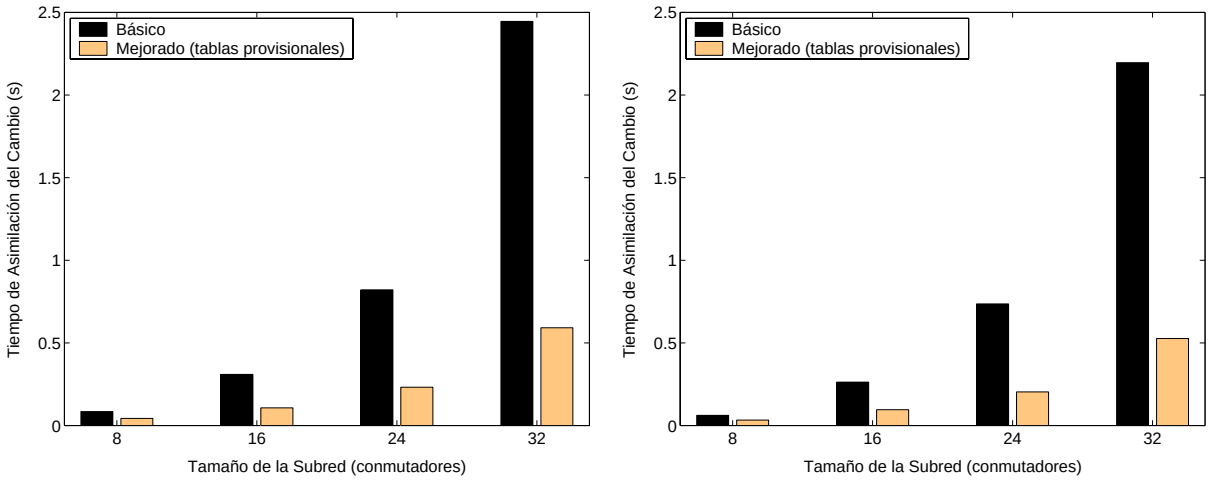
Figura 122. Número de paquetes descartados por los mecanismos básico y mejorado, en función del tamaño de la subred y del tipo de cambio.

8.7 Influencia de la carga en la subred

Todos los resultados mostrados hasta el momento han sido obtenidos empleando una tasa de generación de paquetes que representa el 25% de la tasa que satura cada topología. Para ver la influencia de la carga en la subred sobre las prestaciones del mecanismo de gestión, hemos repetido el estudio empleando una tasa de generación del 50% de la tasa de saturación.

La Figura 123 muestra los tiempos de asimilación del cambio obtenidos por los mecanismos básico y mejorado, para las topologías irregulares analizadas. La Tabla 30 muestra los correspondientes valores medios y desviaciones estándar.

Si comparamos estos resultados con los de la Figura 113 (y la Tabla 28), veremos que son prácticamente idénticos. Por tanto, parece que duplicar la carga en la red no afecta en absoluto al tiempo requerido por el mecanismo de gestión.



(a) Activación (b) Desactivación

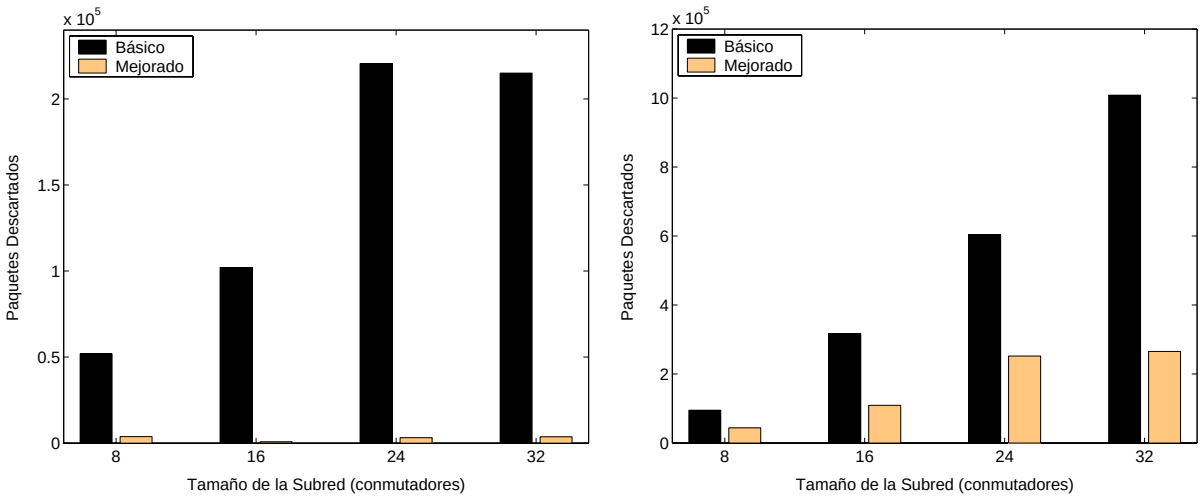
Figura 123. Tiempo hasta la distribución de las nuevas tablas, en función del tamaño de la subred y del tipo de cambio.

Tabla 30. Valores medios y desviaciones estándar para el tiempo de asimilación del cambio.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Mecanismo de gestión**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Básico	0.084184	0.001406	0.061422	0.007599
	Mejorado	0.043004	0.003144	0.033075	0.014054
16	Básico	0.309141	0.001068	0.266308	0.017794
	Mejorado	0.106548	0.003151	0.095274	0.027423
24	Básico	0.820750	0.001241	0.735569	0.011544
	Mejorado	0.231935	0.002305	0.203523	0.004737
32	Básico	2.444983	0.001719	2.195328	0.140058
	Mejorado	0.591355	0.003682	0.526184	0.032115

Por otra parte, la Figura 124 muestra el número total de paquetes descartados por los mecanismos de gestión. La Tabla 31 muestra los valores medios y desviaciones estándar correspondientes.

Las cantidades representadas en la Figura 124 son (aproximadamente) el doble de las mostradas en la Figura 117. Nótese que aunque las gráficas parezcan idénticas, las escalas en los ejes verticales cambian. Además, pueden compararse los valores recogidos en la Tabla 31 con los de la Tabla 29. Como era de esperar, duplicar la carga en la red supone el descarte del doble de paquetes de aplicación.



(a) Activación (b) Desactivación

Figura 124. Número de paquetes descartados por los mecanismos básico y mejorado, en función del tamaño de la subred y del tipo de cambio.

Tabla 31. Valores medios y desviaciones estándar para el número de paquetes descartados.

		Activación		Desactivación	
Tamaño de la subred (conmutadores)	**Mecanismo de gestión**	**Media**	**Desviación estándar**	**Media**	**Desviación estándar**
8	Básico	51,865.43	3,395.59	94,864.57	12,864.39
	Mejorado	8,754.33	1,402.11	43,835.71	12,669.07
16	Básico	102,008.47	2,581.77	298,304.13	118,133.17
	Mejorado	3,776.00	2,716.83	109,209.60	164,392.84
24	Básico	220,441.26	3,594.66	603,357.87	177,942.39
	Mejorado	8,060.00	4,834.46	252,187.26	118,501.05
32	Básico	214,994.13	9,292.47	1,008,262.00	622,710.33
	Mejorado	11,359.00	6,083.30	265,387.97	207,377.79

8.8 Influencia del ancho de banda del enlace

También hemos analizado las prestaciones de los mecanismos de gestión básico y mejorado cuando la red usa enlaces con un ancho de banda distinto a 1X. La Figura 125 y la Figura 126 muestran los mismos resultados que la Figura 113 y la Figura 117, respectivamente, pero asumiendo el empleo de enlaces 4X y un evento de desactivación.

Estos resultados son prácticamente idénticos a los obtenidos para enlaces 1X. Esto quiere decir que aumentar el ancho de banda del enlace no afecta al tiempo de asimilación del cambio, ni a la cantidad de paquetes de aplicación descartados durante el proceso. Lo mismo se puede concluir para el caso de la activación de dispositivos (no mostrado aquí). Como es lógico, las diferencias se aprecian cuando comparamos parámetros como la latencia de los paquetes de aplicación.

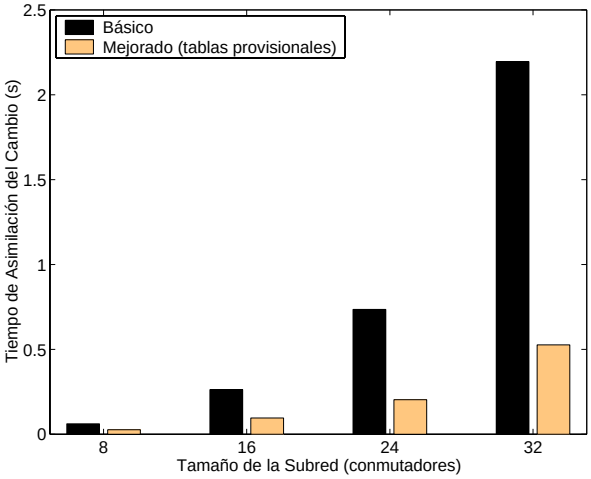


Figura 125. Tiempo hasta la distribución de las nuevas tablas, en función del tamaño de la subred (desactivación).

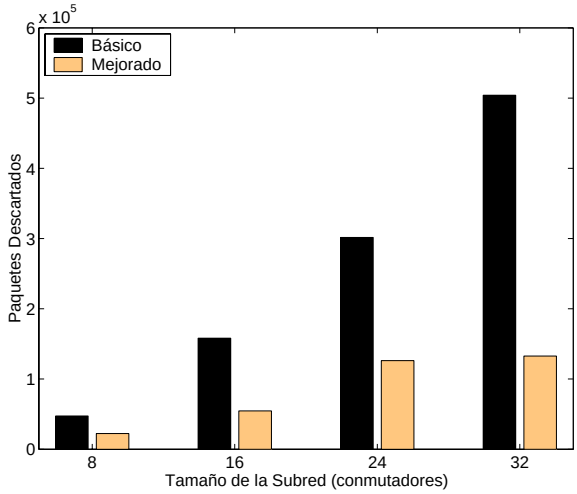


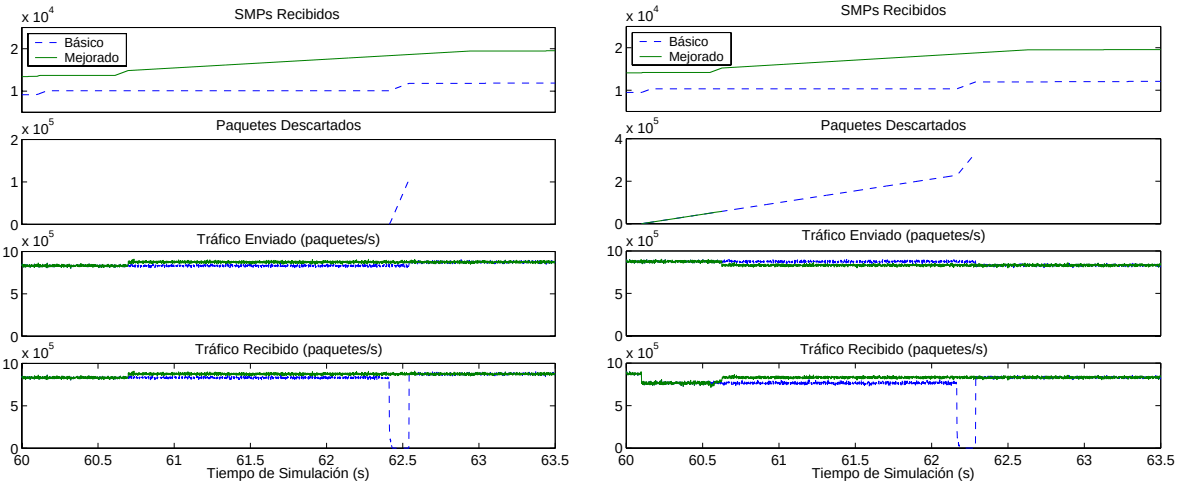
Figura 126. Número de paquetes descartados por los mecanismos básico y mejorado, en función del tamaño de la subred (desactivación).

8.9 Detección de cambios mediante barrido y notificaciones

Los resultados mostrados a continuación han sido obtenidos asumiendo que los SMAs en los conmutadores de la subred son capaces de notificar al SM la ocurrencia de algún cambio a nivel local. En lugar de mostrar nuevamente todos los resultados de las secciones anteriores, destacaremos únicamente aquellos en los que se observa alguna variación.

Los resultados mostrados en la Figura 127 han sido obtenidos en las mismas condiciones que los presentados en la Figura 108 y la Figura 109. Si comparamos con aquellas figuras,

vemos que la distribución de las nuevas tablas se adelanta ligeramente, dado que la detección del cambio se produce antes. Por lo demás, el comportamiento de las gráficas es muy similar.



(a) Activación (b) Desactivación

Figura 127. Impacto del proceso de asimilación del cambio sobre el tráfico de aplicación, para una subred con topología irregular compuesta por 32 conmutadores y 40 nodos terminales.

Aunque en la Figura 127(b) no se aprecia, el número de paquetes descartados (por ambos mecanismos) es ligeramente inferior al mostrado en la Figura 109. Como el mecanismo de notificaciones adelanta en el tiempo el proceso de asimilación completo, el número de paquetes descartados debido a la existencia de puertos inactivos en la subred se reduce. La Figura 128 muestra los paquetes descartados por esta causa, para el mecanismo de gestión básico. La cantidad de paquetes descartados por el resto de causas permanece inalterada.

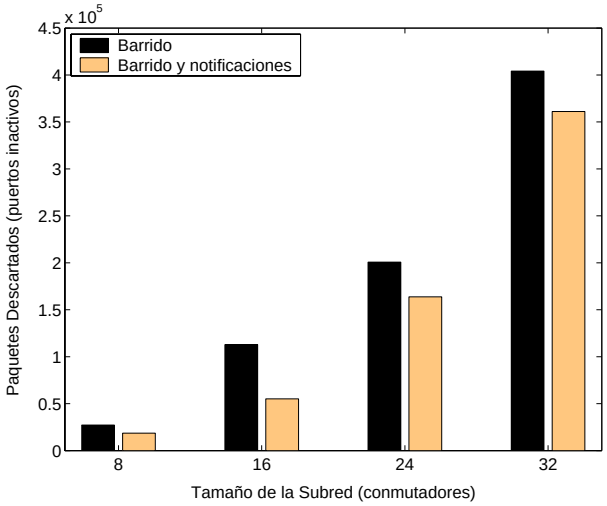


Figura 128. Número de paquetes descartados en puertos inactivos.

8.10 Conclusiones

En este capítulo hemos presentado un análisis detallado de un mecanismo de gestión de la subred que soporta todas las optimizaciones parciales desarrolladas en este trabajo.

Los resultados de simulación obtenidos nos permiten concluir que, independientemente de la carga de trabajo existente en la subred, el mecanismo mejorado gestiona el cambio topológico (distribuye las nuevas tablas) más rápidamente que un mecanismo simplificado.

Hemos comprobado que el impacto de la ejecución del proceso sobre el tráfico de aplicación se reduce significativamente. En particular, la red mantiene el servicio en todo momento. Además, el número de paquetes de aplicación descartados se reduce de manera notable, sobre todo ante un evento de activación de un nuevo dispositivo. El impacto del cambio se reduce todavía más si se habilita el mecanismo opcional de notificaciones.

Capítulo 9
Conclusiones finales

A continuación enumeramos las principales conclusiones y aportaciones derivadas de este estudio. Posteriormente se presentan los trabajos publicados en diferentes foros nacionales e internacionales. Finalmente, se describen posibles líneas de investigación originadas a consecuencia de este trabajo, susceptibles de ser exploradas en el futuro.

9.1 Conclusiones y aportaciones

En la Sección 1.2 (página 3) nos planteábamos una serie de objetivos a cubrir al comienzo de este trabajo. Revisaremos ahora estos objetivos y comprobaremos en qué medida han sido cubiertos.

1. **Estudiar la arquitectura InfiniBand.** En el Capítulo 2 y el Capítulo 3 se ha presentado un resumen de esta arquitectura, resaltando aquellos elementos que más nos interesaban, desde el punto de vista del mecanismo de gestión de la subred.

2. **Desarrollar y validar un modelo de los componentes propios de una subred InfiniBand.** Dicho modelo ha sido descrito con detalle en el Capítulo 4. En la actualidad, el simulador continúa siendo revisado y mejorado, y se encuentra disponible para realizar cualquier tipo de investigación. De hecho, recientemente ha sido empleado por el grupo *SMART Interconnects* para analizar el mecanismo *double scheme* (Sección 7.1.2).

3. **Diseñar, desarrollar y evaluar un prototipo de mecanismo de gestión de la subred.** Una vez que disponíamos de un modelo de la arquitectura, nuestra labor se centró en el diseño de un mecanismo de gestión completo y compatible con la especificación de InfiniBand. Su evaluación nos permitió determinar sus cuellos de botella y enfocar nuestra investigación. Se trata del mecanismo de gestión "básico" empleado en la comparativa presentada en el Capítulo 8 de esta memoria.

4. **Ajustar los mecanismos de detección de cambios topológicos definidos en la especificación.** En la Sección 5.2 hemos analizado los mecanismos de barrido de la topología y notificaciones. Se ha observado que la elección de la frecuencia de sondeo para el mecanismo de barrido no afecta de manera determinante al tráfico de las aplicaciones. Además, hemos comprobado que el mecanismo de notificaciones detecta el cambio con mayor rapidez, independientemente de la frecuencia de barrido aplicada.

5. **Proponer y evaluar un mecanismo para la adquisición de la topología de la subred.** En el Capítulo 5 se describen dos implementaciones para esta tarea, una que descarta totalmente la información topológica previa al cambio y otra que la aprovecha al máximo. Hemos comprobado que el mecanismo de descubrimiento mejorado reduce el tiempo y paquetes de exploración necesarios. Los mejores resultados se obtienen para redes grandes y cambios puntuales.

6. **Diseñar y analizar un algoritmo para la obtención de rutas para encaminamiento dentro de la subred.** En el Capítulo 6 hemos presentado dos implementaciones alternativas para el algoritmo de cálculo de rutas *up*/down**. Una de ellas obtiene un conjunto de rutas que ofrece peores prestaciones, pero reduce significativamente el número de entradas en tablas de encaminamiento a computar, y, por tanto, el tiempo de cómputo total. Este algoritmo, cuya validez se ha demostrado formalmente, se emplea para diseñar un mecanismo de gestión de cambios rápido que reduce de manera notable el impacto del proceso de asimilación.

7. **Mejorar el proceso de actualización de las tablas de encaminamiento de la subred.** En el Capítulo 7 se presentan dos variantes del mecanismo de distribución estática tradicional que mantienen la libertad de bloqueos, son fácilmente implementables en InfiniBand y obtienen unos resultados muy similares a los mecanismos de distribución dinámica de tablas. Las propuestas permiten el empleo de una gran cantidad de rutas dentro de la subred durante el proceso de actualización de las tablas. También se han presentado algunas optimizaciones adicionales que aceleran el proceso de envío de tablas, haciendo la asimilación del cambio más transparente a las aplicaciones.

8. **Evaluar mediante simulación el mecanismo de gestión mejorado completo.** El mecanismo mejorado reúne las optimizaciones parciales analizadas previamente por separado. Su evaluación ha sido presentada en el Capítulo 8 de esta memoria.

A modo de resumen, las principales aportaciones de esta Tesis son las siguientes:

1. Se ha desarrollado una herramienta de simulación que modela los dispositivos de una subred InfiniBand. Esta herramienta nos ha facilitado enormemente la tarea de implementar y evaluar los mecanismos de gestión propuestos.

2. Se han propuesto y analizado diversas implementaciones para cada una de las tareas involucradas en el proceso de asimilación de un cambio topológico en InfiniBand.

3. Se ha diseñado un mecanismo completo de gestión para la arquitectura InfiniBand, totalmente compatible con su especificación. Para cada tarea, se ha escogido la implementación que mejores prestaciones ofrecía.

4. Se ha realizado una evaluación exhaustiva de las prestaciones ofrecidas por el mecanismo de gestión mejorado. Hemos demostrado que este mecanismo gestiona el cambio topológico con rapidez y, lo que es más importante, afectando de forma mínima al tráfico de las aplicaciones.

9.2 Trabajos publicados

En cuanto a la producción científica, en primer lugar destacaremos algunas publicaciones internacionales relacionadas con el trabajo desarrollado dentro de la línea de reconfiguración dinámica en redes de altas prestaciones del *Grupo de Redes y Arquitecturas de Altas Prestaciones* (RAAP) de la Universidad de Castilla-La Mancha.

- R. Casado, A. Bermúdez, F. J. Quiles, J. L. Sánchez, J. Duato. *Performance evaluation of Dynamic reconfiguration in high-speed local area networks*. In proceedings of 6th International Symposium on High-Performance Computer Architecture, pp. 85-96. January 2000.

- A. Bermúdez, F. J. Alfaro, R. Casado, F. J. Quiles, J. L. Sánchez, J. Duato. *Extending dynamic reconfiguration to NOWs with adaptive routing*. In proceedings of 4th Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing. Springer-Verlag, Lecture Notes in Computer Science, vol. 1797. July 2000.

- F. J. Alfaro, A. Bermúdez, R. Casado, F. J. Quiles, J. L. Sánchez, J. Duato. *On the performance of up*/down* routing*. In proceedings of 4th Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing. Springer-Verlag, Lecture Notes in Computer Science, vol. 1797. July 2000.

- R. Casado, A. Bermúdez, F. J. Quiles, J. L. Sánchez, J. Duato. *A protocol for deadlock-free dynamic reconfiguration in high-speed local area networks*. Special Issue on Dependable Network Computing, IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 2, pp. 115-132. IEEE Computer Society Press. ISSN: 1045-9219. February 2001.

- R. Casado, A. Bermúdez, F. J. Quiles, J. Duato. *Influence of network size and load on the performance of reconfiguration protocols*. In proceedings of the IEEE International Symposium on Network Computing and Applications. February 2002.

Por otro lado, versiones preliminares del trabajo desarrollado en esta Tesis Doctoral han sido presentadas en diferentes congresos internacionales y nacionales de reconocido prestigio.

- A. Bermúdez, R. Casado, F. J. Quiles, T. M. Pinkston, J. Duato. *Modeling InfiniBand with OPNET*. 2nd Annual Workshop on Novel Uses of System Area Networks (SAN-2) (HPCA workshop). February 2003.

- A. Bermúdez, R. Casado, F. J. Quiles, T. M. Pinkston, J. Duato. *Evaluation of a subnet management mechanism for InfiniBand networks*. In proceedings of the IEEE International Conference on Parallel Processing (ICPP 2003). October 2003.

- A. Bermúdez, R. Casado, F. J. Quiles, T. M. Pinkston, J. Duato. *On the InfiniBand subnet discovery process*. In proceedings of the IEEE International Conference on Cluster Computing (Cluster 2003). December 2003.

- A. Bermúdez, R. Casado, F. J. Quiles, J. Duato. *Use of provisional routes to speed-up change assimilation in InfiniBand networks*. In proceedings of the Workshop on Communication Architecture for Clusters (CAC'04) (IPDPS workshop). April 2004.

- A. Bermúdez, R. Casado, F. J. Quiles. *Distributing InfiniBand forwarding tables*. In proceedings of the Euro-Par Conference. Springer-Verlag, Lecture Notes in Computer Science. September 2004.

- A. Bermúdez, R. Casado, F. J. Quiles, J. Duato. *Administración de la subred en InfiniBand*. Actas de las XII Jornadas de Paralelismo, pp. 131-136. Septiembre 2001.

- A. Bermúdez, R. Casado, P. E. García, F. J. Quiles, J. Duato. *Modelado de InfiniBand sobre OPNET*. Actas de las XIII Jornadas de Paralelismo, pp. 193-198. Septiembre 2002.

- A. Bermúdez, R. Casado, F. J. Quiles, J. Duato. *Diseño y evaluación de mecanismos de administración para subredes InfiniBand*. Actas de las XIV Jornadas de Paralelismo, pp. 59-64. Septiembre 2003.

Finalmente, destacar que, como fruto de la colaboración con el grupo *SMART Interconnects* de la Universidad del Sur de California (EE.UU.), hemos obtenido una publicación en una de las revistas del área.

- B. Zafar, T. M. Pinkston, A. Bermúdez, J. Duato. *Deadlock-free dynamic reconfiguration over InfiniBand*. Special Issue on Parallel and Distributed Algorithms, International Journal of Parallel Algorithms and Applications. 2004.

9.3 Trabajos futuros

El trabajo futuro en esta línea se centrará fundamentalmente en la mejora del modelo de la arquitectura InfiniBand, la implementación y evaluación del mecanismo de gestión de la subred propuesto en un sistema real, su refinamiento y la migración hacia entornos similares.

9.3.1 Mejora del modelo de InfiniBand

El modelo de InfiniBand presentado en esta memoria es susceptible de ser mejorado en varias direcciones. La idea es obtener un entorno de trabajo más completo y realista, que nos ayude en la obtención de objetivos posteriores. Seguidamente damos algunos ejemplos.

- Modelar el resto de niveles y dispositivos contemplados en la arquitectura InfiniBand. En la actualidad, el modelo sólo contempla algunos elementos de los niveles físico y de enlace. A nivel de red no sería complicado incluir los elementos necesarios para el encaminamiento entre subredes. Sería interesante considerar también los elementos de nivel de transporte, como los pares de colas y los mecanismos para el establecimiento de las conexiones. En cuanto al modelo de nodo terminal, habría que mejorar el nodo de procesamiento e incorporar también unidades de E/S con sus correspondientes tarjetas de interfaz (TCAs).

- Mejorar el modelo de tráfico de aplicación. El modelo de tipo Poisson que hemos empleado debe ser sustituido por modelos más apropiados para nuestro entorno, como el tráfico *self-similar* [42]. En este sentido, también pueden incorporarse los modelos de aplicación que incorpora *OPNET Modeler*, que simulan el comportamiento de aplicaciones reales como el FTP o la videoconferencia.

- Modelar otros elementos de gestión, como el administrador de la subred (SA) y el gestor de las comunicaciones (CM), necesarios para establecer las conexiones entre los pares de colas y ejecutar el protocolo de migración automática de rutas (Sección 3.3.2, página 51).

9.3.2 Implementación de las propuestas en un sistema real

En este trabajo hemos evaluado nuestras propuestas mediante simulación. Los modelos tienden a simplificar el sistema original que representan, lo que permite avanzar en su desarrollo. No obstante, sería conveniente en este punto trasladar nuestras propuestas a un sistema real. Nuestra intención es emplear los productos de la empresa *Mellanox* (Sección 2.8.1,

página 28) como entorno de desarrollo, debido a su disponibilidad y prestaciones. En la actualidad, el grupo de investigación RAAP dispone de estos componentes entre su equipamiento.

9.3.3 Optimizaciones adicionales del mecanismo de gestión

Nos plantearemos aquí el estudio de otras vías que nos permitan mejorar todavía más las prestaciones ofrecidas por el mecanismo propuesto. Lógicamente, seguiremos imponiendo la condición de que los refinamientos se ajusten a las normas recogidas en la especificación de InfiniBand.

- Mejorar los mecanismos de detección de cambios. Una propuesta podría estar basada en generar un conjunto de rutas cerradas que, partiendo y llegado al SM, recorrieran toda la subred empleando muy pocos paquetes (si comparamos con el proceso de barrido normal).

- Emplear SMs distribuidos. Si asumimos que la funcionalidad del SM está distribuida en varios nodos dentro de la subred, podemos acelerar claramente los procesos de detección del cambio y de adquisición de la nueva topología.

- Solapar las tareas de gestión. Por ejemplo, el cálculo de rutas sigue siendo el proceso más largo. Aquí hemos optimizado esta tarea por separado, pero quizás podría solaparse con la tarea de exploración. La idea es que a medida que se van descubriendo nuevos dispositivos, éstos puedan ser añadidos a las tablas de los dispositivos ya conocidos. Por otro lado, también se puede solapar el cálculo y la distribución de las entradas de las tablas de encaminamiento, tal y como hace el mecanismo mejorado con las rutas definitivas.

- Mejorar el proceso de cálculo de rutas. Por ejemplo, trabajando con nuevas estructuras de datos que reflejen la información topológica de forma más accesible. También nos proponemos incorporar algún criterio de balanceo de carga a la hora de construir las nuevas rutas a través de la subred. Por último, y dado que *up*/down** no es el algoritmo que obtiene las mejores prestaciones para la red [2], nos planteamos sustituirlo por otros esquemas de encaminamiento que resulten más eficientes.

- Reducir el número de paquetes descartados en el caso de la desactivación de dispositivos. Hemos visto que, aún con el mecanismo mejorado, sigue existiendo un importante descarte. Esto se debe a que algunas de las viejas rutas han dejado de estar operativas. Habría que idear algún mecanismo para redirigir los paquetes, en lugar de proceder a su descarte al llegar a un puerto desconectado.

9.3.4 Desarrollo de propuestas de gestión para otros entornos

Por un lado, nos centraremos en PCI Express Advanced Switching, un estándar emergente que se plantea como sustituto del bus PCI (y sus variantes) para la conexión de dispositivos de E/S, aunque pretende ser mucho más ambicioso. De hecho, básicamente se trata de una extensión de PCI Express que soporta multiprocesamiento y comunicaciones *peer-to-peer*.

La arquitectura PCI Express AS tiene unas características muy similares a InfiniBand. Se trata igualmente de una interconexión serie punto a punto, altamente escalable, que ofrece canales virtuales y clases de tráfico, control de flujo basado en créditos, mecanismos robustos de control de la congestión, encaminamiento en base a destino y fuente, posibilidad de *multicast*, etc.

Por supuesto, el estándar también ofrece una completa infraestructura para la gestión de la red. En este caso, el *fabric master* es la entidad encargada de realizar las acciones llevadas a cabo en InfiniBand por el gestor de la subred (SM).

Otro entorno hacia el que nos planteamos trasladar toda nuestra experiencia en el desarrollo de mecanismos de gestión y reconfiguración es el de las redes inalámbricas *ad hoc*. Aunque se trata de un ámbito con unas características muy diferentes a InfiniBand o PCI Express AS, es obvio que la naturaleza de estas redes, inherentemente dinámica, obliga a contemplar este tipo de mecanismos.

Referencias

[1] F. J. Alfaro. *Una sencilla metodología para garantizar calidad de servicio en subredes InfiniBand*. Tesis Doctoral. Universidad de Castilla-La Mancha. Septiembre 2003.

[2] F. J. Alfaro, A. Bermúdez, R. Casado, F. J. Quiles, J. L. Sánchez, J. Duato. *On the performance of up*/down* routing*. In proceedings of the 4th Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing. Springer-Verlag, Lecture Notes in Computer Science, vol. 1797. July 2000.

[3] D. Avresky, N. Natchev, V. Shurbanov. *Dynamic reconfiguration in high-speed computer clusters*. In proceedings of the 2001 IEEE International Conference on Cluster Computing (CLUSTER'01). October 2001.

[4] Banderacom Silicon Solutions. http://www.banderacom.com.

[5] A. Bermúdez, F. J. Alfaro, R. Casado, F. J. Quiles, J. L. Sánchez, J. Duato. *Extending dynamic reconfiguration to NOWs with adaptive routing.* In proceedings of the 4th Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing. Springer-Verlag, Lecture Notes in Computer Science, vol. 1797. July 2000.

[6] A. Bermúdez, R. Casado, P. E. García, F. J. Quiles, J. Duato. *Modelado de InfiniBand sobre OPNET.* Actas de las XIII Jornadas de Paralelismo, pp. 193-198. Septiembre 2002.

[7] A. Bermúdez, R. Casado, F. J. Quiles. *Distributing InfiniBand forwarding tables.* In proceedings of the Euro-Par Conference. Springer-Verlag, Lecture Notes in Computer Science. September 2004.

[8] A. Bermúdez, R. Casado, F. J. Quiles, J. Duato. *Administración de la subred en InfiniBand.* Actas de las XII Jornadas de Paralelismo, pp. 131-136. Septiembre 2001.

[9] A. Bermúdez, R. Casado, F. J. Quiles, J. Duato. *Diseño y evaluación de mecanismos de administración para subredes InfiniBand.* Actas de las XIV Jornadas de Paralelismo, pp. 59-64. Septiembre 2003.

[10] A. Bermúdez, R. Casado, F. J. Quiles, J. Duato. *Use of provisional routes to speed-up change assimilation in InfiniBand networks.* In proceedings of the Workshop on Communication Architecture for Clusters (CAC'04). April 2004.

[11] A. Bermúdez, R. Casado, F. J. Quiles, T. M. Pinkston, J. Duato. *Modeling InfiniBand with OPNET*. 2nd Annual Workshop on Novel Uses of System Area Networks (SAN-2). February 2003.

[12] A. Bermúdez, R. Casado, F. J. Quiles, T. M. Pinkston, J. Duato. *Evaluation of a subnet management mechanism for InfiniBand networks.* In proceedings of the IEEE International Conference on Parallel Processing (ICPP 2003). October 2003.

[13] A. Bermúdez, R. Casado, F. J. Quiles, T. M. Pinkston, J. Duato. *On the InfiniBand subnet discovery process.* In proceedings of the IEEE International Conference on Cluster Computing (Cluster 2003). December 2003.

[14] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kuawik, C. L. Seitz, J. N. Seizovic, W. Su. *Myrinet: A gigabit per second LAN*. IEEE Micro, vol. 15, no. 1, pp. 29-36. February 1995.

[15] R. Budruk, D. Anderson, T. Shanley. *PCI Express System Architecture*. Mindshare, Inc., 2003.

[16] R. Buyya. *High Performance Cluster Computing: Architectures and Systems*. ISBN 0-13-013784-7, Prentice Hall PTR, NJ, USA, 1999.

[17] R. Casado. *Mecanismo de reconfiguración eficiente en redes de interconexión con topología irregular y encaminamiento up*/down**. Tesis Doctoral. Universidad de Castilla-La Mancha. Julio 2001.

[18] R. Casado, A. Bermúdez, F. J. Quiles, J. Duato. *Influence of network size and load on the performance of reconfiguration protocols.* In proceedings of the IEEE International Symposium on Network Computing and Applications. February 2002.

[19] R. Casado, A. Bermúdez, F. J. Quiles, J. L. Sánchez, J. Duato. *Performance evaluation of Dynamic reconfiguration in high-speed local area networks*. In proceedings of the 6th International Symposium on High-Performance Computer Architecture, pp. 85-96. January 2000.

[20] R. Casado, A. Bermúdez, F. J. Quiles, J. L. Sánchez, J. Duato. *A protocol for deadlock-free dynamic reconfiguration in high-speed local area networks.* Special Issue on Dependable Network Computing, IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 2, pp. 115-132. IEEE Computer Society Press. ISSN: 1045-9219. February 2001.

[21] C. Clos. *A study of non-blocking switching networks*. Bell System Technical Journal 32, 406-424. March 1953.

[22] W. J. Dally, C. L. Seitz. *Deadlock-free message rouing in multiprocessor interconnection networks*. IEEE Transactions on Computers, vol. c-36, no. 5, pp. 547-553. May 1987.

[23] J. Duato. *An efficient methodology for the implementation of up*/down* routing on InfiniBand networks*. Tech. Report, DISCA, UPV. January 2001.

[24] J. Duato, R. Casado, F. J. Quiles, J. L. Sánchez. *Dynamic reconfiguration in high-speed local area networks.* D. Avresky (ed.). Dependable network computing. Kluwer Academic Publishers, pp. 207-232. 2000.

[25] J. Duato, A. Robles, F. Silla, R. Beivide. *A comparison of router architectures for virtual cut-through and wormhole switching in an NOW environment*. Journal on Parallel and Distributed Computing, 61, pp 224-253. 2001.

[26] J. Duato, S. Yalamanchili, L. Ni, *Interconnection networks: An engineering approach*. Morgan Kaufmann Publishers. August 2002.

[27] C. Eddington (Mellanox Technologies). *InfiniBridge: an InfiniBand channel adapter with integrated switch*. IEEE Micro. March-April 2002.

[28] Fibre Channel Industry Association. *Fibre Channel Standard*. http://www.fibrechannel.org.

[29] W. T. Futral. *InfiniBand Architecture. Development and Deployment.* Intel Press. Agosto 2001.

[30] P. E. García. *Modelado de la arquitectura InfiniBand con OPNET*. Proyecto Fin de Carrera. Directores: R. Casado y A. Bermúdez. Departamento de Informática. Universidad de Castilla-La Mancha. Diciembre 2002.

[31] D. García, W. Watson. *ServerNet II*. In proceedings of the 1997 Parallel Computing, Routing and Communication Workshop. June 1997.

[32] Gigabit Ethernet Alliance Home Page. http://www.gigabit-ethernet.org.

[33] A. Heirich, D. Garcia, M. Knowles, R. Horst. *ServerNet II: a reliable interconnect for scalable high performance cluster computing*. Paper draft. Compaq Computer Corporation, Tandem Division. September 1998.

[34] Horst, R. W. *TNet: a reliable system area network*. IEEE Micro, vol. 15, no. 1, pp. 37-45. February 1995.

[35] HyperTransport Consortium. http://www.hypertransport.org.

[36] InfiniBand™ Architecture Specification Release 1.1. InfiniBand℠ Trade Association. November 6, 2002. http://www.infinibandta.org.

[37] InfiniBand Management Working Group. *InfiniBand Management.* IBTA Spring 2001 Developers' Conference. June 19, 2001. http://www.infinibandta.org.

[38] InfiniCon Systems. http://www.infinicon.com.

[39] InfiniSwitch Corporation. http://www.infiniswitch.com.

[40] P. Kermani, L. Kleinrock. *Virtual cut-though: A new computer communication switching technique*. Computer Networks, vol. 3, pp. 267-286. 1979.

[41] E. J. Kim, K. H. Yum, C. R. Das, M. Yousif, J. Duato. *Performance Enhancement Techniques for InfiniBand Architecture*. In proceedings of the 9th International Symposium on High-Performance Computer Architecture (HPCA9'03). 2003.

[42] W. E. Leland, M. S. Taqqu, W. Willinger, D. V. Wilson. *On the self-similar nature of Ethernet traffic*. IEEE Transactions on Networking, vol. 2, no. 1, pp. 1-15. February 1994.

[43] J. Liu *et al*. *Performance comparison of MPI Implementations over InfiniBand, Myrinet and Quadrics*. In proceedings of the Supercomputing Conference. November 2003.

[44] P. López, J. Flich, J. Duato. *Deadlock-free routing in InfiniBand™ through destination renaming*. In proceedings of the 2001 International Conference on Parallel Processing (ICPP). September 2001.

[45] O. Lysne, J. Duato. *Fast dynamic reconfiguration in irregular networks*. In proceedings of the 2000 International Conference on Parallel Processing (ICPP'00).

[46] O. Lysne, T. M. Pinkston, J. Duato. *A methodology for developing dynamic network reconfiguration processes*. In proceedings of the IEEE International Conference on Parallel Processing (ICPP 2003). October 2003.

[47] A. M. Mainwaring, B. N. Chun, S. Schleimer, D. S. Wilkerson. *System Area Network Mapping*. In proceedings of the 9th Annual Symposium on Parallel Algorithms and Architectures. 1997.

[48] J. C. Martínez, J. Flich, A. Robles, P. López, J. Duato. *Supporting Adaptive Routing in InfiniBand Networks*. In proceedings of the 7th Euromicro Conference on Parallel, Distributed and Network-Based Processing (Euro-PDP'03).

[49] J. C. Martínez, J. Flich, A. Robles, P. López, J. Duato. *Supporting Fully Adaptive Routing in InfiniBand Networks*. In proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03).

[50] D. Mayhew, V. Krishnan. *PCI Express and Advanced Switching: evolutionary path to building next generation interconnects*. In proceedings of the 11th Symposium of High Performance Interconnects (HOTI'03). August 2003.

[51] P. Mehra. *Trends in system area networking*. IEEE Symposium on Network Computing and Applications, pp. 3-8. October 2001.

[52] Mellanox Technologies. http://www.mellanox.com.

[53] Microsoft Corporation. http://www.microsoft.com.

[54] J. M. Montañana, J. Flich, A. Robles, P. López, J. Duato. *A transition-based fault-tolerant routing methodology for InfiniBand networks.* In proceedings of the Workshop on Communication Architecture for Clusters (CAC'04). April 2004.

[55] *OPNET Modeler Documentation*. OPNET Technologies, Inc. http://www.opnet.com/

[56] R. Pang, T. M. Pinkston, J. Duato. *The double scheme: deadlock-free dynamic reconfiguration on cut-through netowrks*. In proceedings of the 2000 International Conference on Parallel Processing. August 2000.

[57] L. D. Paulson. *The ins and outs of new local I/O trends*. Computer. July 2003.

[58] PCI-X 2.0 Specifications. PCI Special Interest Group. http://www.pcisig.com/specifications/pcix_20.

[59] R. Perlman. *An algorithm for distributed computation of a spanning tree in an extended LAN*. In proceedings of the 9th Data Communications Symposium. 1985. ACM, pp. 44-53.

[60] F. Petrini *et al*. *The Quadrics network: high-performance clustering technology*. IEEE Micro. January 2002.

[61] F. Petrini, M. Vanneschi. *k-ary n-trees: high performance networks for massively parallel architectures*. In proceedings of the 11th International Parallel Processing Symposium. 1997.

[62] G. P. Pfister. *High performance mass storage and parallel I/O*. Chapter 42: *An Introduction to the InfiniBand Architecture*. Wiley-IEEE Press. November 2001.

[63] T. M. Pinkston, J. Duato, O. Lysne, R. Pang. *Theoretical support for deadlock-free dynamic network reconfiguration*. Workshop on self-healing, adaptive, and self-managed systems. June 2002.

[64] T. M. Pinkston, R. Pang, J. Duato. *Dynamic reconfiguration of networks with distributed routing: the single scheme*. In proceedings of the 2001 International Conference on Parallel and Distributed Processing Techniques and Applications. June 2001.

[65] T. M. Pinkston, R. Pang, J. Duato. *Deadlock-free dynamic reconfiguration schemes for increased network dependability*. IEEE Transactions on Parallel and Distributed Systems, vol. 14, no. 6. June 2003.

[66] T. M. Pinkston, B. Zafar, J. Duato. *A method for applying Double Scheme dynamic reconfiguration over InfiniBand*. In proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. June 2003.

[67] QLogic Corporation. http://www.qlogic.com.

[68] Rapid I/O Specifications. Rapid I/O Trade Association. http://www.rapidio.org/specs.

[69] Rational Software Corporation. http://www.rational.com.

[70] R. J. Recio. *Server I/O networks past, present, and future*. In proceedings of the ACM SIGCOMM 2003 Workshops. August 2003.

[71] RedSwitch, Inc. http://www.redswitch.com.

[72] T. L. Rodeheffer, M. D. Schroeder. *Automatic reconfiguration in Autonet*. SRC Research Report 77 of the ACM Symposium on Operating Systems Principles. October 1991.

[73] P. J. Sáez. *Modelado y evaluación de algoritmos de encaminamiento para la arquitectura InfiniBand*. Proyecto Fin de Carrera. Directores: A. Bermúdez y R. Casado. Departamento de Informática. Universidad de Castilla-La Mancha. Diciembre 2003.

[74] I. Sánchez-Ferrer. *Modelado de encaminamiento multidestino en InfiniBand*. Proyecto Fin de Carrera. Directores: A. Bermúdez y R. Casado. Departamento de Informática. Universidad de Castilla-La Mancha. Junio 2004.

[75] J. C. Sancho. *Contribución al diseño de algoritmos de encaminamiento en redes de estaciones de trabajo*. Tesis Doctoral. Universidad Politécnica de Valencia. Septiembre 2002.

[76] J. C. Sancho, J. Flich, A. Robles, P. López, J. Duato. *Analyzing the influence of virtual lanes on the performance of InfiniBand networks*. Workshop on Communication Architecture for Clusters (held in conjunction with IPDPS '02). Fort Lauderdale, Florida, April 15, 2002.

[77] J. C. Sancho, A. Robles, J. Duato. *A new methodology to compute deadlock-free routing tables for irregular networks*. In proceedings of the 4th Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing. January 2000.

[78] J. C. Sancho, A. Robles, J. Duato. *Effective Strategy to Compute Forwarding Tables for InfiniBand Networks*. In proceedings of the 2001 International Conference on Parallel Processing (ICPP). September 2001.

[79] J. C. Sancho, A. Robles, J. Flich, P. López, J. Duato. *Effective Methodology for Deadlock-free Minimal Routing in InfiniBand Networks*. In proceedings of the 2002 International Conference on Parallel Processing (ICPP). 2002.

[80] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, T. L. Rodeheffer, E. H. Satterthwate, C. P. Thacker. *Autonet: A high-speed, self-configuring local area network using point-to-point links*. IEEE Journal on Selected Areas in Communications, vol.9, no. 8. October 1991.

[81] C. L. Seitz. *Recent advances in cluster networks*. In proceedings of the 2001 IEEE International Conference on Cluster Computing (CLUSTER'01). October 2001.

[82] T. Shanley. *InfiniBand Network Architecture*. Mindshare, Inc., 2003.

[83] SourceForge. *Linux InfiniBand Project*. http://sourceforge.net/projects/infiniband.

[84] SourceForge. *Open InfiniBand Stack*. http://sourceforge.net/projects/openib.

[85] Tanenbaum, A. *Redes de Computadoras*. Prentice Hall, 1997.

[86] I. Theiss, O. Lysne. *LORE – Local reconfiguration for fault management in irregular interconnects*. In proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'04). April 2004.

[87] Top500 Supercomputer Sites. http://www.top500.org.

[88] Vieo. http://www.vieo.com.

[89] Voltaire, Inc. http://www.voltaire.com.

[90] T. Weimer. *Fibre Channel Fundamentals*. Spectra Logic Corporation. http://www.spectralogic.com/fibre/fcfund.htm.

[91] J. Wu, A. Gulati, B. Abali, D. Panda. *Design of an InfiniBand emulator over Myrinet: challenges, implementation, and performance evaluation*. Ohio State University, Tech. Report OSU-CISRC-2/01-TR03.

[92] B. Zafar, T. M. Pinkston, A. Bermúdez, J. Duato. *Deadlock-free dynamic reconfiguration over InfiniBand*. Special Issue on Parallel and Distributed Algorithms, International Journal of Parallel Algorithms and Applications. 2004.