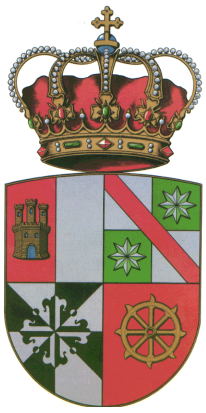
UNIVERSIDAD DE CASTILLA-LA MANCHA

DEPARTAMENTO DE INFORMÁTICA



Una Sencilla Metodología para Garantizar Calidad de Servicio en 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

Presentada por:
Francisco José Alfaro Cortés

Dirigida por:
Dr. D. José Luis Sánchez García
Dr. D. José F. Duato Marín

Albacete, Septiembre de 2003

A mis padres

Agradecimientos

Durante el desarrollo de esta tesis he recibido la ayuda y apoyo de mucha gente, sin la cual no hubiera sido posible llegar hasta aquí. Es justo ahora echar la vista atrás y dedicar unas líneas a todos aquellos que lo han hecho posible.

En primer lugar, me gustaría agradecer toda la confianza depositada en mí por mis directores de tesis, José Luis y José Duato, sin cuya guía, paciencia y apoyo no hubiera sido posible recorrer este camino. En especial, quiero agradecer el continuo ánimo de José Luis, que ha sabido guiarme hasta aquí y que, además de director, también es un gran compañero y amigo. Por otro lado, José Duato, a pesar de la distancia, siempre me ha aportado su experiencia, sus geniales ideas y sabios consejos. A ambos, gracias por todas las cosas que me habéis enseñado.

Tengo que agradecer también la gran ayuda recibida por parte de Manolo Menduiña, que ha aportado su experiencia, ideas y ganas de trabajar, además de darme siempre continuos ánimos para seguir adelante. Su ayuda ha sido fundamental en el desarrollo de toda la parte formal de esta tesis.

También me gustaría agradecer el apoyo y ayuda de todos los miembros del Departamento de Informática. Especialmente a los compañeros del Grupo de Redes y Arquitecturas de Altas Prestaciones (RAAP), cuyo esfuerzo y excelente trabajo, está consiguiendo que se valore al grupo a nivel nacional e internacional. Buena parte de este mérito lo tienen los jefes del grupo, Paco y Antonio, que han sabido crear de la nada un grupo de investigación ahora consolidado.

No quisiera olvidar en este momento a los que siempre consideraré mis compañeros del Departamento de Ingeniería y Tecnología de Computadores (DITEC) de la Universidad de Murcia, con los que tuve el privilegio de compartir casi tres años de trabajo. Entre ellos cuento a algunos de los que considero buenos amigos. En especial, quiero agradecer a José Manuel su continuo apoyo y la confianza que depositó en mí, en un momento personal muy difícil. Sin su apoyo y confianza en aquel momento, estoy seguro de que no habría llegado hasta aquí.

También quiero tener ahora un recuerdo muy especial para el grupo de compañeros de las "mazmorras". Algunos me han demostrado ser, más que compañeros, amigos. Muchos de ellos ya han pasado este momento, pero quiero dar desde aquí mi ánimo al resto, para que pronto puedan ellos también estar escribiendo la sección de agradecimientos de sus respectivas tesis doctorales.

Por último, pero muy en especial para mí, quisiera agradecer, de una manera muy especial, a mis padres y hermanos el constante apoyo que me han prestado, aguantándome los momentos más difíciles, mis malos humores y cabreos. A pesar de ello, siempre me han aportado una desmesurada comprensión y voluntad para que siguiera adelante en mi carrera.

A todos ellos, y muchos más que no he nombrado pero que me han ayudado a llegar hasta aquí, gracias.

Resumen

Con este trabajo se pretende proporcionar garantía de QoS a las aplicaciones en InfiniBand, siendo éste el objetivo principal de esta tesis, y el tema en el que se ha venido trabajando durante los últimos años.

En la tesis, en primer lugar, se realiza un estudio de las especificaciones de InfiniBand para conocer en profundidad sus características. Además, se realiza una clasificación de las posibles aplicaciones que pueden desarrollarse en InfiniBand, y una categorización en función de sus necesidades de QoS. Para ello se parte de una clasificación previamente propuesta, sobre la que se realizan algunas modificaciones.

Seguidamente se aborda cómo garantizar ancho de banda a las aplicaciones que lo requieran, y se proponen estrategias para conseguirlo. Para ello es necesario diseñar una metodología para el funcionamiento tanto de las aplicaciones como de las entidades encargadas de la administración de la red. En concreto, se proponen los pasos que deberán seguir las aplicaciones que deseen tener garantías de QoS, y cómo deben las entidades de gestión usar los mecanismos encargados de proporcionar esa QoS. Específicamente, se detalla cómo se debe modificar la tabla de arbitraje para conseguir dar la garantía de QoS solicitada.

Por otra parte, se estudia también cómo proporcionar garantía de latencia máxima. Para ello se hace un estudio previo de los diversos factores que influyen en la latencia final que sufre una aplicación. Se estudian los modelos de conmutador más comunes, detallando en cada caso el grado de contribución de los diversos elementos en la latencia final observada.

Seguidamente, se plantea un marco global en el que proporcionar garantía tanto de ancho de banda como de latencia máxima. Se muestran las ventajas de este marco global y los posibles problemas que pudiera plantear. Así mismo, se analiza detalladamente la influencia de las decisiones tomadas, y se comparan con otras alternativas.

Una vez estudiado el funcionamiento que deben tener las aplicaciones y las entidades de gestión de InfiniBand para garantizar QoS, se desarrolla un modelo formal sobre la base de la metodología propuesta. Dentro de este modelo formal se enuncia un algoritmo que establece la forma en la que se reservan los recursos para proporcionar la QoS requerida por las aplicaciones. Como consecuencia de todo esto se derivan una serie de propiedades y teoremas que son debidamente demostrados.

Finalmente, se realiza una evaluación exhaustiva de las propuestas presentadas. Esta evaluación se lleva a cabo mediante un simulador de InfiniBand desarrollado en

buena parte por el autor de la tesis. Mediante este estudio experimental se comprueba como, incluso en situaciones de carga máxima de la red de interconexión, todas las aplicaciones obtienen los requisitos de QoS que previamente habían solicitado.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se indican los motivos que han llevado a realizar este trabajo, así como los objetivos que se pretenden cubrir con éste. Para entender mucho mejor dichos motivos, se describe en primer lugar cuál es la situación en el momento presente de las arquitecturas de interconexión, tanto para interconexión de dispositivos de entrada/salida (E/S), como de sistemas de procesamiento entre sí.

Como quedará reflejado a lo largo de este capítulo, la situación actual muestra una clara deficiencia en la capacidad que los sistemas de interconexión poseen para satisfacer las nuevas exigencias de las aplicaciones que constituyen el futuro más inmediato.

Se describirán, por tanto, las características de las aplicaciones actuales, para las cuales la tendencia es demandar cada vez más recursos. Esto es debido a que, cada vez más, las aplicaciones integran audio, vídeo, interacción con el usuario, etc., necesitando para ello una gran cantidad de recursos. De esta forma, para que algunas de estas aplicaciones funcionen adecuadamente, es necesario que la red sea capaz de proporcionarles una serie de garantías.

Todos estos nuevos requisitos que demandan las aplicaciones actuales se reflejan en lo que se suele denominar calidad de servicio (QoS). Es decir, esas aplicaciones necesitan algo más de la red que simplemente entregar los paquetes en el destino tan pronto como se pueda. También, en este capítulo, se indicará cómo se está actualmente proporcionando calidad de servicio a las aplicaciones. Así, se describirán los dos enfoques seguidos para ello en Internet: servicios diferenciados y servicios integrados.

En la parte final del capítulo, y una vez dado un repaso al estado del arte y mostradas las deficiencias en lo que se refiere a la arquitectura de interconexión, se detallarán, como se indicó más arriba, la motivación que inició este trabajo y los objetivos que se pretendían cubrir cuando se comenzó. Finalmente, se presentará el desarrollo y un breve resumen de los capítulos posteriores.

1.1. Redes de altas prestaciones

Durante los últimos años, los avances en teoría, arquitectura y tecnología de las redes de computadores han permitido mejorar sus prestaciones enormemente. Se han desarrollado algoritmos de encaminamiento más eficientes y fiables, mejores topologías de red, mecanismos de control de flujo y de la congestión, esquemas de utilización de los medios compartidos, y técnicas de conmutación más eficientes. Las redes de interconexión de multicomputadores también se han desarrollado en paralelo, siguiendo el camino marcado por las redes generales de amplio alcance, mejorando su arquitectura de una forma significativa. Más aún, durante la pasada década muchas técnicas empleadas tradicionalmente en redes de interconexión de multicomputadores y multiprocesadores, han saltado las barreras de dichas máquinas extendiendo su aplicación al ámbito de las redes de área local (LAN).

Esta importante mejora es, en buena parte, debida a la aparición de los *grupos de estaciones de trabajo* (*Clusters of Workstations*), también denominados redes de estaciones de trabajo (NOWs), o simplemente clusters. Las NOWs se presentan como alternativa de bajo coste a los computadores paralelos, para una extensa gama de aplicaciones que exhiben un grado de paralelismo medio [ACP95]. En estas NOWs, se emplean estaciones de trabajo u ordenadores personales de propósito general como elementos de procesamiento, unidos habitualmente mediante una red de altas prestaciones. Estas redes también se han planteado como alternativa al tradicional bus del sistema dando lugar a las *redes de área de sistema* (*System Area Network*, o SAN) [Hor95, PE00].

Estas redes de altas prestaciones proporcionan una alternativa, mucho más eficiente, a la clásica y barata topología de bus, tanto si se trata del bus del sistema (por ejemplo, un bus PCI), o de la típica configuración de red local Ethernet.

En el campo de las tecnologías de la información, en las últimas décadas se ha producido una importante evolución en varios aspectos:

- En velocidades de transmisión de información. Los tradicionales 10 Mbps de las LANs de hace una década se han superado en tres órdenes de magnitud, estando hoy día disponible la tradicional Ethernet, pero ahora a 10 Gbps.

- En velocidad de funcionamiento de los circuitos integrados. En tan sólo una década, hemos pasado de procesadores a 50 ó 66 MHz a los más de 3 GHz de algunos procesadores actuales. Entre otras causas, esto ha sido debido al cambio de filosofía que se ha producido en el diseño de los procesadores (CISC vs RISC), y en el uso intensivo del paralelismo existente entre las propias instrucciones.

- En las cantidades masivas de información a manejar. Hace una década Internet era algo restringido a la comunidad científica. Aún así, era impensable poder difundir documentos que incluyeran audio, vídeo, etc. Hoy en día, cualquier persona desde el ordenador de su casa, con una simple línea telefónica, puede distribuir documentos que integren gran cantidad de información en varios formatos distintos.

Todos estos factores han puesto de manifiesto las carencias de las topologías basadas en bus. Concretamente, y refiriéndose al bus de sistema PCI, en [PE00] se citan los siguientes inconvenientes:

- Plantea contención en el acceso a los recursos. El acceso a los recursos se hace de forma desordenada por parte de periféricos, memoria y CPUs, y este desorden lleva a la ineficiencia y a un descenso en las prestaciones resultantes.
- Puesto que todo el sistema se monta alrededor del bus, y si éste es único, un sólo fallo en el mismo deshabilita el sistema completo. Más aún, si una tarjeta conectada al bus falla, puede hacer que el sistema entero deje de funcionar, y la única manera de descubrir cuál fue la tarjeta causante del fallo es mediante el método de "ensayo y error". Esto es inadmisible en servidores que necesitan una alta disponibilidad.
- Presenta importantes limitaciones físicas, puesto que al incrementar la longitud del bus para acomodar más dispositivos, las señales eléctricas pierden estabilidad. Lo mismo ocurre al incrementar la frecuencia de reloj, pues cuanto mayor sea la frecuencia, menor es la longitud posible del bus, y menos periféricos se pueden conectar. En un caso extremo, la especificación de PCI-X para 133 MHz sólo permite un único conector por bus.

Compaq [Sch00] también señala las limitaciones de las típicas LAN que dieron auge a los sistemas distribuidos a mediados de los años 80:

- Todas las LANs estaban basadas en la difusión, donde cada mensaje transmitido se transportaba a todos los hosts conectados a la red. Esto significa que la productividad de la red se ve limitada por el ancho de banda del medio. Cuando con el tiempo se van incrementando tanto el número de usuarios de la LAN como el volumen de información transferida, este tipo de red no puede soportar esa demanda.
- En sus configuraciones más simples, las redes LAN convencionales exhibían un único punto de fallo. Tal y como se ha explicado antes para el caso del bus PCI, en este caso también podía ocurrir que el fallo del bus común, o incluso el de alguno de los interfaces de red conectados, dejara inoperante la red.

- En las redes basadas en bus, un incremento en velocidad implica decrementar el tamaño máximo de los segmentos. Esto es debido al retardo de propagación y al tiempo máximo establecido para los periodos de contención.

Así pues, tanto en la interconexión de máquinas para formar un *cluster* dedicado al procesamiento distribuido, como en la misma arquitectura de los computadores convencionales, el principal cuello de botella, al menos en lo que a capacidad de transferencia de información se refiere, se hallaba en la topología de bus. En el ámbito de las LAN, la industria empezó a trabajar en dispositivos como puentes, concentradores inteligentes y encaminadores que aliviasen el problema. Todos ellos se basaban en el uso de conmutadores o *switches* que evitaban que todos los mensajes llegasen a todas partes, y hacían que cada mensaje se dirigiera únicamente hacia su destino.

Como consecuencia de todo lo anterior, a principios de los años noventa se desarrolló en el *Systems Research Center* de Digital la red *Autonet*, o AN1 [SBB90]. Esta red estaba basada enteramente en conmutadores, con enlaces punto a punto entre éstos y los *hosts*. En esta red los mensajes se encaminaban por los conmutadores hacia su destino, a lo largo de un camino que era determinado en cada conmutador con arreglo a la dirección de destino almacenada en la cabecera del mensaje.

Debido a que Autonet usaba conmutadores, muchos mensajes diferentes podían estar atravesando la red por enlaces distintos en un momento dado. Los conmutadores de esta red empleaban como técnica de conmutación *virtual cut-through* [KK79], es decir, tan pronto como el conmutador ha analizado la cabecera del mensaje y ha calculado el puerto de salida correspondiente, el mensaje se hace avanzar a través de ese puerto de salida, si está libre, incluso aunque el mensaje completo no se haya recibido todavía en el conmutador. Esta forma de transmisión segmentada disminuye notablemente la latencia de los mensajes respecto a técnicas tipo *store-and-forward* [Tan88].

En el diseño de esta red se aplicaron muchas de las ideas originalmente desarrolladas y aplicadas en el ámbito de los computadores paralelos, como por ejemplo los enlaces punto-a-punto, la conmutación *cut-through* o los conmutadores basados en *crossbar*.

Sin embargo, también fue necesario desarrollar nuevas técnicas para afrontar las características diferenciales del nuevo entorno: algoritmo de encaminamiento *up*/down** para poder encaminar mensajes sin provocar bloqueos en la topología irregular, *buffers* de mayor tamaño y control de flujo *stop-and-go* para aprovechar mejor el ancho de banda de los enlaces largos, algoritmo de reconfiguración distribuido para reconfigurar las tablas de encaminamiento ante la conexión o desconexión de elementos de la red (que, al contrario de lo que sucedía en los computadores paralelos, ahora son independientes), etc.

Autonet fue la pionera de este tipo de redes, introduciendo muchos de los conceptos y principios presentes en otras redes posteriores, como Myrinet [BCF95] o ServerNet [Hor95].

La red Myrinet, de Myricom Inc., tuvo su germen en dos proyectos financiados por el gobierno de EE.UU.: los proyectos Mosaic, del CalTech (California Institute of Technology), y Atomic LAN, del USC/ISI (University of Southern California/Information Sciences Institute). El objeto del primero era el desarrollo de un multicomputador experimental de grano fino, mientras que en el segundo se perseguía construir una LAN empleando componentes Mosaic. Los orígenes de Myrinet explican el uso de técnicas empleadas en redes de interconexión de multicomputadores en su diseño: conmutación *wormhole*, conmutadores basados en *crossbar*, encaminamiento fuente, etc.

En el mismo año apareció la red ServerNet (originalmente denominada TNet), la primera propuesta para reemplazar el bus del sistema por una red conmutada hecha por Tandem para sus servidores de gama alta, con fuertes requisitos de funcionamiento ininterrumpido. Se trata por tanto de la primera propuesta de una SAN aparecida en la literatura. En la red ServerNet se introducía por primera vez la idea de dotar a los periféricos de cierta inteligencia para ser capaces de enviar y recibir mensajes a través de la red, sin necesidad de que la CPU participe en todos los movimientos de datos. La idea subyacente era que empezaban a aparecer aplicaciones que requerían de una transferencia masiva de datos sin necesidad de ningún procesamiento (telecompra, educación a distancia, vídeo bajo demanda, etc.). Por tanto, era necesario un servidor capaz de satisfacer esas demandas masivas de información puesto que, en la organización tradicional, el bus del sistema y la CPU se convertían en cuellos de botella insalvables para ofrecer las prestaciones en cuanto a transferencia de datos de las nuevas aplicaciones.

Como se ha podido observar, los arquitectos de estas nuevas redes de altas prestaciones se centraron en cómo reservar y utilizar los restringidos recursos de la red de forma eficiente, con el objetivo de proporcionar mayor productividad y menor latencia media que las soluciones basadas en bus que se empleaban anteriormente. Esta aproximación es plenamente válida cuando la información que fluye por la red no necesita obtener prestaciones "especiales", es decir, simplemente es necesario trasegar una cantidad de tráfico suficiente (alta productividad) y procurar que ese tráfico llegue a su destino con el menor retardo posible en término medio. Por ello, a este tipo de tráfico se le denomina como de "mejor esfuerzo" (*best-effort*), pues el sistema sólo se compromete a hacer lo que pueda para que llegue a su destino con buenas prestaciones, pero sin garantizar nada. Sin embargo, cuando las aplicaciones necesitan otra serie de requisitos, como garantía de ancho de banda, latencia, retraso entre paquetes, etc., lo estudiado anteriormente ya no es capaz de proporcionar las prestaciones necesarias.

1.2. Entrada/Salida de altas prestaciones

Durante mucho tiempo los diseñadores de computadores se centraron en mejorar el rendimiento de los procesadores dejando de lado los sistemas de E/S. Una prueba evidente de ello es que llamamos unidad central al procesador, mientras que a las unidades de E/S las llamamos, quizás de forma un poco despectiva, dispositivos periféricos [Pat01].

Otra prueba más de que la E/S ha estado mucho tiempo dejada de lado, es que los populares benchmarks SPEC no incluyen programas que usen más de un 5 % la E/S [Pat01]. De esta forma, no es posible medir las prestaciones del subsistema de E/S con los que, durante mucho tiempo, han sido los benchmarks más usados.

David Patterson dice en [Pat01] que un ordenador sin dispositivos de E/S es como un coche sin ruedas, no puedes ir muy lejos sin ellas. Además, a la persona que compra finalmente un ordenador lo que le importa es la velocidad y productividad final de trabajo, donde evidentemente va a tener una gran importancia la parte de E/S.

Debido a todos estos motivos, uno de los puntos más débiles de las NOWs actuales son sus interfaces de entrada/salida. Tanto la demanda de cada vez mayores potencias de procesamiento, derivadas del empleo de procesadores más potentes, como la existencia de tecnologías de red de altas prestaciones y de aplicaciones que gestionan enormes cantidades de información, están imponiendo cada vez más exigencias sobre el sistema de E/S. Todo esto se traduce en la necesidad de interconectar un mayor número de dispositivos de almacenamiento, así como de incrementar el ancho de banda con el que se acceden.

Los actuales estándares de E/S, utilizados tanto por los PCs como por la mayor parte de las estaciones de trabajo que se usan para construir los servidores de altas prestaciones, están basados en buses. El estándar más conocido y ampliamente utilizado en la actualidad es PCI. Su capacidad, esto es, la velocidad y volumen de la información que puede ser transferida, ha resultado suficiente hasta hace poco para atender las necesidades de las aplicaciones. Sin embargo, no resulta suficiente para atender las nuevas exigencias tecnológicas y la demanda de las futuras aplicaciones. En este contexto, el subsistema de E/S se está convirtiendo en la mayoría de los casos en el auténtico cuello de botella del sistema. Además, el hecho de que la interconexión de cada PC a la red se realice mediante una tarjeta de interfaz a la que se accede a través del mismo bus de E/S, contribuye a agravar el problema.

Los interfaces de E/S basados en buses tienen como principal ventaja su simplicidad y bajo coste. Sin embargo, los buses no resultan eficientes a la hora de transferir datos entre los procesadores y los dispositivos de E/S. La falta de eficiencia de los buses se hace más evidente a medida que avanza la tecnología y se incrementan las exigencias sobre los mismos. Las razones para ello son varias:

- Debido a su naturaleza de medio compartido, el limitado ancho de banda de los buses se convierte en un cuello de botella a medida que aumenta el número de dispositivos conectados al mismo, dificultando la escalabilidad del sistema. Al mismo tiempo, cuanto mayor es el número de dispositivos conectados al bus, mayor es su capacidad parásita, lo que causa mayor consumo y mayores tiempos de propagación.
- Los buses requieren el empleo de mecanismos de arbitraje de acceso al medio, tanto más complejos cuanto mayor sea el número de dispositivos conectados al bus. Ello constituye una importante sobrecarga, cuyos mayores inconvenientes se ponen de manifiesto cuanto mayor es el ancho de banda del bus.
- Los buses no garantizan el nivel de fiabilidad y disponibilidad de funcionamiento requerido por los servidores actuales. Ello se debe a que un simple fallo en el bus o en uno cualquiera de los dispositivos conectados al mismo puede dejar fuera de servicio a todo el sistema. Por otro lado, los mecanismos para identificar el origen del fallo y proceder a su aislamiento suelen requerir un tiempo de búsqueda demasiado elevado, lo que los inhabilita para sistemas que han de garantizar elevada disponibilidad de funcionamiento.

En los últimos años, la investigación en sistemas de E/S de altas prestaciones se ha venido centrando en mejorar la gestión de los buffers y caches [KLS00]. Dependiendo del sistema utilizado, dichos buffers pueden recibir distintos nombres: buffer cache, cache de disco, cache de ficheros, etc. Estas mejoras se han llevado tanto a los controladores de los distintos sistemas operativos, como a los propios dispositivos de E/S.

En este sentido, la necesidad de eliminar el cuello de botella que en la actualidad representa el acceso a los dispositivos de E/S, en el ámbito de PCs y estaciones de trabajo, ha llevado a la aparición recientemente de InfiniBand. Esta constituye la primera apuesta firme por parte de los líderes de la industria informática (IBM, Intel, Compaq, HP, Microsoft, Sun y Dell) por crear un estándar de red que elimine de forma definitiva el problema de la E/S. En este sentido, su principal contribución consiste en sustituir el bus por enlaces punto a punto entre procesadores y dispositivos de E/S a través de conmutadores. Al mismo tiempo, aumenta la escalabilidad y fiabilidad del sistema.

1.3. Nuevas aplicaciones, nuevas necesidades

En los últimos años se ha producido un crecimiento espectacular de las comunicaciones de datos continuas (audio/vídeo) sobre redes de computadores [Tow93, SW97, CJ97]. Las tecnologías de red actuales, como la fibra óptica y la conmutación ATM, hacen que estén disponibles a bajo coste redes de gran ancho de banda, capaces de proporcionar varios gigabits por segundo [Pry91, CL95]. Gracias a la disponibilidad de

este tipo de redes de gran ancho de banda, las aplicaciones multimedia surgen de forma natural como los principales clientes que aprovecharán las mayores capacidades de las redes de computadores. El mayor ancho de banda que requieren las nuevas aplicaciones puede ser satisfecho plenamente con las propuestas introducidas en la década de los noventa. Sin embargo, estas nuevas aplicaciones tienen otros requisitos adicionales completamente diferentes a los requisitos de las aplicaciones tradicionales.

Las aplicaciones multimedia integran varios tipos de medios, como vídeo, audio, imágenes fijas, gráficos, texto, etc. Este tráfico multimedia introduce nuevos requisitos en cuanto a los recursos que la red debe ser capaz de proporcionar a la aplicación. El tráfico multimedia tiene necesidades de ancho de banda mínimo, retardo máximo extremo a extremo, variación en el retardo de los paquetes, tasa de pérdida de los paquetes admisible, etc.

Concretamente, tanto el vídeo como el audio generan un tráfico continuo, en el sentido de que requiere disponibilidad de ancho de banda durante toda la duración de la transmisión para que la aplicación no se vea interrumpida en el equipo receptor. En el caso de la señal de vídeo, se debe mostrar al receptor una imagen cada cierto tiempo de forma continua. Además, estas imágenes, una vez emitidas, se deben presentar en el destino dentro de unos límites de retardo adecuados. Los valores de estos límites dependen del nivel de interacción de la aplicación, y pueden oscilar entre unos 20 milisegundos y varios segundos. La técnica más común es imponer cotas al retardo de acuerdo con los límites perceptivos de la aplicación. Cuando el retardo sufrido por un paquete de información es superior a dicho límite, el receptor lo descarta, provocando una pérdida en la calidad de la señal recibida (posiblemente acentuada por las dependencias entre paquetes introducidas por la codificación de la señal [Cue98]). Además, la transmisión de ese paquete finalmente descartado, también supone un desperdicio de ancho de banda [Kar96].

Todos estos nuevos requisitos se reflejan en lo que se denomina *Calidad de Servicio* (QoS). El término calidad de servicio fue empleado por primera vez hace unos veinte años en el *Modelo de Interconexión de Sistemas Abiertos* (OSI). Hacía referencia a la capacidad de un proveedor de servicios (un operador de red) para soportar los requisitos de las aplicaciones de usuario con respecto a cuatro parámetros de servicio: ancho de banda, latencia o retardo, *jitter* o variación en la latencia y pérdida de datos [Bla00].

Veamos de forma detallada algunos de estos requisitos que solicitan las aplicaciones actuales [GG99, Bla00]:

- La provisión de *ancho de banda* para una aplicación significa que la red tenga la suficiente capacidad para soportar los requisitos de productividad de la aplicación. Es decir, que la red sea capaz de ir trasegando la información que se genera sin provocar congestión en la fuente.

- La *latencia* o *retardo* describe el tiempo que se tarda en enviar un paquete de usuario desde un nodo origen a un nodo destino. Hay algunas aplicaciones, como la transmisión de vídeo o voz, que tienen marcados requisitos en cuanto a latencia, puesto que si el paquete llega demasiado tarde, ya no es útil y se ignora. Por tanto, los paquetes que llegan tarde a su destino se traducen en ancho de banda desperdiciado y en una reducción de la QoS que recibe la aplicación. Sin embargo, los paquetes que llegan demasiado pronto también presentan problemas. Los paquetes prematuros deben almacenarse en el destino hasta que llegue su turno de ser procesados. Por tanto, puede ocurrir que el espacio de buffers en el receptor se desborde, viéndose obligado a descartar información que llegó demasiado pronto. Es decir, al contrario de lo que sucede en la transmisión de datos tradicional, ahora ya no se cumple que a menor retardo necesariamente se obtengan mejores prestaciones, sino que un retardo excesivamente pequeño también puede provocar problemas. Así pues, en este tipo de aplicaciones multimedia, el receptor espera recibir la información a intervalos bastante regulares.

- En cuanto al *jitter*, se trata de la variación en los retardos entre la llegada al receptor de dos paquetes consecutivos. Estas variaciones en los retardos son causadas por esperas provocadas a los paquetes por la contención por recursos que en un momento dado pueden hallarse ocupados. Una variación elevada en los retardos experimentados por los paquetes complica la reproducción en el destino de secuencias de voz o vídeo.

- La *pérdida de datos* es el último de los parámetros generalmente considerados cuando se habla de QoS. La pérdida de información es importante en aplicaciones de voz y vídeo, pues puede afectar al proceso de decodificación, lo que se traduce en una degradación de la señal que recibe el usuario final. En aplicaciones tradicionales, la pérdida de datos también es un serio problema, pero puesto que en general estas aplicaciones no tienen requisitos temporales demasiado estrictos (a fin de cuentas el tráfico que generan se suele catalogar en su inmensa mayoría como de tipo *best-effort*), se pueden aplicar procedimientos de retransmisión para obtener la transferencia fiable de todos los datos. Por el contrario, cuando se trata de aplicaciones multimedia, la retransmisión no suele ser de utilidad debido a los retardos incurridos entre la detección del error y el reenvío de la información.

Es bastante frecuente que los requisitos de calidad de servicio para las aplicaciones multimedia se definan en función de los requisitos perceptivos de las mismas [Hal01]. Es decir, se va a requerir que la red transfiera la información con unas determinadas prestaciones para que los usuarios perciban la señal sin degradaciones. A continuación se comentan brevemente los requisitos perceptivos de algunas de las aplicaciones multimedia más representativas [Cue98]:

Videoconferencia. La videoconferencia es una aplicación interactiva en tiempo real, y por tanto sus requisitos en términos de retardo son muy exigentes. Un retardo en torno a los 200-500 milisegundos puede resultar molesto a los usuarios. En [Gar93] se recomienda un retardo máximo de 150 milisegundos para que el retardo resulte imperceptible para los usuarios. En cuanto a sus requisitos de ancho de banda, las secuencias suelen contener simplemente el busto de una persona, con lo que el movimiento es escaso, y por tanto no requieren un gran consumo de ancho de banda. De hecho, la mayoría de aplicaciones de videoconferencia actuales emplean la red telefónica conmutada, y por tanto se les dedican canales de 64 Kbps.

Difusión de televisión digital. Estas aplicaciones incluyen películas, anuncios, noticias, que en general presentan un mayor grado de movimiento. Esto hace que los requisitos de ancho de banda sean mayores que en el caso de la videoconferencia (en torno a los 4-10 Mbps para la TV convencional, y de 20 a 80 Mbps para la televisión de alta definición). Por el contrario, los requisitos en términos de retardo no son tan estrictos como en el caso anterior. Generalmente, son bien toleradas latencias de varios segundos para las transmisiones en directo. Incluso para las transmisiones de información en diferido no se requiere ningún retardo específico, pues el usuario no lo podrá apreciar. El único requisito al respecto es que una vez que la información empieza a visualizarse en el receptor, ésta debe ser mostrada de manera continua, sin interrupciones. Esto implica que la variación en el retardo o *jitter* será un parámetro importante que deberá tenerse en cuenta.

Vídeo bajo demanda. En esta aplicación la señal de vídeo codificada está almacenada en un servidor, y se transmite al usuario bajo petición de éste. Es el usuario, por tanto, quien controla el comienzo de la transmisión. Además, puede ser que el usuario tenga acceso a funciones de rebobinado o pausa de la señal. Si el usuario no dispone de ese control adicional, el retardo sólo afecta a la latencia con la que empieza a proporcionarse el servicio desde que el usuario lo solicita. En ese caso se toleran retardos de hasta unos pocos minutos. Por el contrario, si el usuario tiene control sobre el avance de la transmisión (rebobinados, pausas, etc.) la aplicación se vuelve más interactiva, y los requisitos en términos de retardo pasan a ser de unos cientos de milisegundos [Dal96].

Telemedicina. Esta aplicación es fuertemente interactiva, entre el centro de diagnóstico y el usuario, y por tanto, los requisitos en términos de retardo son exigentes, en torno a los 100 milisegundos o menos. La resolución de las imágenes es elevada, y las degradaciones en su calidad deben ser imperceptibles. Estas aplicaciones suelen requerir un ancho de banda de entre 1 y 20 Mbps.

Kurose y Ross [KR00] clasifican las aplicaciones multimedia existentes en las siguientes tres categorías:

- **Continuas**: En este tipo de aplicaciones los clientes solicitan ficheros, que suelen ser de audio y/o de vídeo, a servidores, y se segmenta la recepción de los datos desde la red y su visualización. Suelen ser aplicaciones interactivas donde el usuario puede controlar la reproducción, de manera similar a como lo haría con un reproductor de vídeo convencional. En general, son tolerantes a la latencia, de hecho, desde la petición del cliente hasta el inicio de la visualización pueden transcurrir varios segundos. Como ejemplo de este tipo de aplicaciones podemos citar RealPlayer de RealNetworks [Rea] o Windows Media de Microsoft [Med].
- **Unidireccionales de tiempo real**: Este tipo de aplicaciones son similares a las emisiones existentes de radio o televisión, pero empleando la red como medio de transmisión. No son interactivas, pues el usuario simplemente escucha y/o visualiza la señal. Al igual que en el caso anterior, toleran varios segundos de retardo desde que el cliente decide conectarse a la emisión. Como ejemplo de estas aplicaciones podemos citar cualquiera de las emisoras de radio que emiten sus programas vía Internet [Cad, Ser].
- **Interactivas de tiempo real**: Este tipo de aplicaciones permite que los usuarios se comuniquen entre sí mediante audio o vídeo en tiempo real. En este caso, los requisitos de retardo son mucho más exigentes que en las clases anteriores. Suelen ser valores aceptables menos de 150 milisegundos para vídeo y menos de 400 milisegundos para audio. Como ejemplo de aplicaciones englobadas dentro de esta categoría podríamos citar la videoconferencia [Net] o la conversación telefónica usando Internet (VoIP) [IF].

Una característica común a todas estas aplicaciones es que son relativamente insensibles a la pérdida de datos, siempre y cuando no degraden seriamente la calidad de la señal a transmitir. Los niveles tolerables de pérdidas de datos dependen también de la robustez del destino, pues en muchos casos se aplican en el destino técnicas que ocultan las pérdidas de información en la imagen. Un estudio completo sobre el tema se puede encontrar en [Cue98].

A modo de resumen, podemos obtener las siguientes conclusiones en cuanto a las necesidades de las aplicaciones multimedia:

- Los requisitos en cuanto a retardo serán tanto más estrictos cuanto mayor sea el grado de interactividad de la aplicación.
- Los requisitos en cuanto al ancho de banda dependen de la calidad de la señal, y de las características de la misma (imágenes estáticas o con mucho movimiento, resolución de la imagen, etc.).
- En cualquier caso, es importante que, una vez iniciada la visualización de la señal en destino, ésta se haga de manera continua, para que el usuario la perciba sin interrupciones.

Las redes de altas prestaciones empleadas en NOWs y SANs hasta la fecha no incluyen ningún soporte para estas nuevas aplicaciones. Sus objetivos se limitan a ofrecer alta productividad y baja latencia media al tráfico de usuario, y como se ha visto, esto no es suficiente para estas nuevas aplicaciones.

1.4. Clases de servicio

Ya se ha hablado previamente de la calidad de servicio, y se han introducido los parámetros que reflejan los requisitos de las aplicaciones: productividad, retardo, jitter y pérdida de datos. Estos parámetros reflejan la percepción que tiene el usuario de un servicio de red. La red es la responsable de mantener el nivel de QoS esperado por sus usuarios.

Es evidente que considerar por separado las necesidades de cada flujo de información (tráfico) que pueda generar cada aplicación distinta puede ser complicado. Más aún cuando esas necesidades pueden depender no solamente de las aplicaciones en sí, sino de cada usuario. Así pues, considerar por separado cada caso posible es algo completamente inviable dado el número de usuarios, y el amplísimo rango de aplicaciones presentes y futuras que pueden presentarse. Es pues necesario definir un número limitado de clases de servicio. Así, cuando una aplicación desee transferir información escogerá aquella clase de servicio que mejor se adapte a sus necesidades. Si no existiese una diferenciación del tráfico en clases de servicio, la red debería soportar los requisitos de calidad más exigentes para todo el tráfico.

Cada clase de servicio se define mediante unos requisitos específicos de QoS. En la literatura existen diversas propuestas de clasificación del tráfico. Una clasificación clásica es la introducida en ATM [For], basada en cinco categorías de tráfico:

CBR (*Constant Bit Rate*). A esta clase de tráfico pertenecerían conexiones que necesitan un ancho de banda fijo, y unos fuertes requisitos en cuanto a latencia. Este ancho de banda está siempre disponible mientras dure la conexión, por lo que se suele emplear para servicios de emulación de circuitos. Ejemplos de aplicaciones que pueden emplear esta clase de servicio son las conexiones telefónicas, la videoconferencia, el audio o el vídeo sin comprimir, y en general, cualquier tipo de comunicaciones interactivas multimedia, para las que es necesario garantizar cotas en el ancho de banda y/o en los retardos.

rt-VBR (*real time-Variable Bit Rate*). Esta clase de tráfico está pensada para aplicaciones sensibles al retardo que generan el tráfico en ráfagas, es decir, a una tasa de inyección variable. Sobre estas fuentes la red puede aplicar el multiplexado estadístico. Cuando se comprimen flujos de audio y vídeo, como por ejemplo mediante la codificación MPEG-2, se suele generar tráfico perteneciente a esta

categoría. Por tanto, y debido a que cada vez existen más y mejores técnicas de compresión, este tipo de tráfico puede convertirse en el más extendido en un futuro próximo.

nrt-VBR (*non real time-Variable Bit Rate*). A esta categoría de tráfico pertenecerían aquellas fuentes que generan tráfico a ráfagas, pero que no tienen requisitos de retardo o de variación en el retardo. Un ejemplo típico de aplicaciones que generan tráfico de este tipo es la reproducción de temas musicales o películas de vídeo.

ABR (*Available Bit Rate*). Esta clase está pensada para el tráfico normal de datos, tales como transferencia de ficheros y correo electrónico, que no requiere garantías de servicio. Aunque no se requiere garantizar ni el retardo máximo ni un ancho de banda mínimo, es deseable que los conmutadores intenten minimizarlos tanto como sea posible. Por ello, también se suele denominar como *tráfico de mejor esfuerzo* (*best-effort traffic*). A esta clase de tráfico se le aplica control de flujo en caso de que la red esté congestionada. Ejemplos de aplicaciones que pueden utilizar esta clase de servicio son todas las convencionales: transferencia de ficheros, servicios de noticias, etc.

UBR (*Unespecified Bit Rate*). Esta clase de tráfico está diseñada para aplicaciones que utilizan cualquier capacidad sobrante de la red, y que no son sensibles a la pérdida o al retardo de la información. No se le aplica control de flujo, y en caso de congestión sus paquetes se descartan y las fuentes no reducen su tasa de inyección. Las aplicaciones que pueden usar este tipo de servicio serían las mismas que en la clase de servicio ABR.

Diversos autores han propuesto alternativas a esta clasificación. Por ejemplo, en [KLC98] sólo se consideran tres de estas clases, puesto que no hace distinción entre las dos clases de tráfico VBR, y no considera el tráfico UBR. Otro ejemplo aparece en [Pel00], donde se habla de cuatro clases de tráfico:

DBTS (*Dedicated Bandwidth Time Sensitive*). Este tipo de tráfico requiere un ancho de banda mínimo, y una latencia máxima determinada. Sería por tanto equivalente a las clases CBR y rt-VBR de ATM. Ejemplos de esta categoría serían aplicaciones eminentemente interactivas, como la videoconferencia o *Voice over IP* (VoIP).

DB (*Dedicated Bandwidth*). Esta clase de tráfico requiere un ancho de banda mínimo, pero no es demasiado sensible a la latencia. Es por tanto similar a la clase nrt-VBR de ATM.

BE (*Best-Effort*). Por su propia naturaleza, este tipo de tráfico suele tener ráfagas. Además, este tráfico es bastante insensible tanto al ancho de banda como a la

latencia. Se puede asimilar por tanto a la categoría ABR de ATM. En esta clase se pueden encuadrar el tráfico que generan la mayoría de servicios convencionales que se suelen ejecutar en redes de comunicaciones: servicios de ficheros y de impresión, navegación en la web, transferencia de ficheros, correo electrónico, etc.

CH (*Challenged*). El servicio que recibe el tráfico perteneciente a esta clase se degrada intencionadamente para que no afecte al tráfico BE. La idea es que esta clase de servicio sea empleada por tareas de tipo administrativo, que no deben interferir con ninguna de las demás clases empleadas por el tráfico de usuarios. Un ejemplo de aplicación de esta clase de tráfico sería la realización de copias de seguridad en momentos donde no haya interacción con tráfico de usuarios.

Sobre esta clasificación propuesta por Pelissier nosotros hemos propuesto una leve modificación [ASD02]. Se trata de dividir el tráfico BE en dos categorías: PBE (Preferential Best Effort) y el BE usual. Esta nueva categoría PBE engloba al tipo de tráfico que no necesita garantías explícitas en cuanto a latencia o ancho de banda, pero al que se le quiere dar un trato preferencial sobre el tráfico best-effort usual. Como ejemplo de este tipo de tráfico podría citarse el tráfico web, o de acceso a bases de datos. Parece una buena idea que este tipo de tráfico tenga prioridad sobre el resto de tráfico sin garantías, de cara a que tenga un mejor comportamiento. De esta forma, es posible dar trato distinto a aplicaciones que deben servirse sin garantías, pero que se quiere que no obtengan las mismas prestaciones.

Sea cual fuere la clasificación de las clases de tráfico seguida, es responsabilidad del usuario o de las aplicaciones marcar el tráfico generado con un determinado identificador de la clase a la que pertenece para que los medios disponibles en la red (si es que ésta soporta algún tipo de tratamiento con QoS) le den un tratamiento de acuerdo a sus necesidades.

1.5. Proporcionar QoS a las aplicaciones

Como se ha indicado en las secciones anteriores, cada vez es mayor la necesidad de soportar gran variedad de tráfico con una gran diversidad de requisitos en cuanto a QoS. Los sistemas con soporte de QoS traducen una petición de servicio en un conjunto de características de tráfico para ese servicio, como el ancho de banda, retardo máximo, variación máxima en el retardo, pérdidas, tasa de errores, etc.

Los sistemas con soporte de QoS deben ser capaces de medir las características apuntadas anteriormente, y utilizar técnicas de encaminamiento y de tratamiento de colas para conseguir cumplir los requisitos de las aplicaciones. Algunos métodos se basan en tratar de obtener recursos de la red (capacidad y memoria de almacenamiento)

suficientes para garantizar estas características. En este caso, se produce un dimensionado de los recursos y el sistema no se compromete a conseguir unas determinadas prestaciones si no hay recursos suficientes par dar garantías absolutas.

Otros sistemas no tratan de dar garantías absolutas, sino que se comprometen a dar trato preferencial a las clases de tráfico consideradas prioritarias. De esta forma, es seguro que las clases prioritarias obtendrán mejores resultados, pero no está garantizado que vayan a conseguir las prestaciones deseadas. En este caso no se suele producir un dimensionado de los recursos.

El *Internet Engineering Task Force* (IETF) ha propuesto varios modelos y mecanismos para dar servicio a las aplicaciones que necesitan QoS. Los dos principales modelos propuestos son los *Servicios Integrados* [BCS94], y los *Servicios Diferenciados* [BBC98, Ber98]. Veamos cada uno de ellos de manera más amplia.

1.5.1. El modelo de servicios integrados

La arquitectura de servicios integrados [BCS94] es un marco global para proporcionar QoS en un entorno IP. Dentro de este marco se han definido distintos estándares para abordar cada una de las tareas concretas. Cada paquete IP puede estar asociado a un flujo de datos. Los paquetes de un mismo flujo provienen de la misma fuente y todos tienen los mismos requisitos en cuanto a QoS.

El modelo de servicios integrados hace uso de las siguientes funciones para controlar la congestión y proporcionar QoS:

- Control de admisión: Para poder proporcionar QoS a las aplicaciones, éstas deben hacer una reserva previa de recursos. Si los encaminadores por los que debe pasar no tienen recursos suficientes para satisfacer la petición, entonces la petición no será admitida. Para este control de admisión se ha definido el protocolo RSVP [BZB97].

- Algoritmo de enrutamiento: El camino seleccionado para un paquete puede estar basado en alguno de los parámetros de QoS, y no sólo en el criterio de distancia mínima. Por ejemplo, el protocolo de encaminamiento OSPF es capaz de seleccionar unas rutas u otras en función de la QoS que éstas pueden ofrecer.

- Arbitraje: Cuando en un encaminador varios paquetes quieren salir por el mismo puerto, el sistema de arbitraje debe decidir cuál de esos paquetes sale primero. Este arbitraje estará basado en asignar una política de prioridades basada en los requisitos que necesita cada una de las aplicaciones. De esta forma, el sistema de arbitraje permite dar más prioridad a los flujos de datos con más requisitos.

- Descarte de paquetes: En los sistemas donde se permite que en caso de congestión se descarten paquetes, se debe aplicar una política de prioridades a la hora de decidir qué paquetes se descartan en cada momento. De esta forma, los paquetes de los flujos de datos menos prioritarios serán descartados antes que otros paquetes de flujos más prioritarios.

El modelo de servicios integrados propone dos clases de servicio, además del tradicional servicio *best-effort*. Estas clases de servicio nuevas son: Servicio Garantizado (*Guaranteed Service*) [SPG97] para las aplicaciones que necesitan un retraso máximo acotado, y el Servicio con Carga Controlada (*Controlled Load Service*) [Wro97b] para las aplicaciones que sólo necesitan tener asegurado que los paquetes llegarán a su destino y que recibirán un mejor servicio que el tráfico *best-effort*.

El protocolo RSVP [BZB97] se diseñó para permitir a las aplicaciones hacer una reserva de recursos desde el nodo fuente hasta el nodo destino. El nodo fuente envía un paquete hacia el destino especificando los requisitos que deben cumplirse para poder establecer la conexión. Tanto los nodos intermedios como el nodo final analizan la petición realizada y si es posible satisfacerla. En caso de que se pueda aceptar, el mensaje sigue avanzando en su ruta hacia el nodo final. Si en algún nodo intermedio, o en el destino, la petición no puede aceptarse, se rechaza y se informa hacia atrás en la ruta hasta llegar al nodo fuente. Si la petición se acepta, los nodos intermedios reservan el ancho de banda y el espacio de almacenamiento necesario en los buffers.

Hay versiones del protocolo RSVP que permiten que la reserva de recursos sea por flujos agregados, y no por conexiones individuales [GBH97]. De esta manera, lo que se establecen son rutas explícitas con determinadas características de QoS que las aplicaciones pueden solicitar para su uso.

Con la conexión ya establecida, cuando un nodo intermedio recibe un paquete de esa conexión lo almacena donde corresponda según la clasificación de tráfico realizada. El algoritmo de arbitraje se va a encargar de darle salida cuando le corresponda según las prioridades de los paquetes presentes del resto de conexiones.

Los principales problemas que se plantean en la arquitectura de servicios integrados son [XN99]:

- La cantidad de información que cada nodo intermedio de la ruta debe manejar y almacenar crece proporcionalmente con el número de conexiones que esté manejando. Esto supone que los encaminadores deben almacenar una gran cantidad de información, y tener una enorme capacidad para procesar dicha información.
- Los requisitos que se les pide a los encaminadores son muy altos. Todos ellos deben implementar el protocolo RSVP, un control de admisión, planificación de sus colas de salida, etc.

- Es necesario que todos los encaminadores intermedios sean capaces de proporcionar QoS. De esta forma, no es posible una implementación gradual de la arquitectura de servicios integrados. Para un correcto funcionamiento sería necesario incorporar estos servicios en todos los nodos de las redes que se quiera que proporcionen QoS.

Por tanto, la arquitectura no escala bien, y supone un cambio radical con los sistemas que se han venido utilizando hasta la fecha, lo que dificulta su expansión.

1.5.2. El modelo de servicios diferenciados

El modelo de servicios diferenciados [BBC98] proporciona un tratamiento por prioridades. En este caso, los encaminadores no tienen en cuenta el estado actual de la red, ni las características particulares de cada flujo de datos. De esta forma, el servicio puede escalar bastante bien, pues es independiente de la cantidad de flujos que intervengan.

Para poder aplicar el modelo de servicios diferenciados los paquetes IP deben ir marcados en uno de sus campos con el nivel de servicio. Este campo se denomina *DiffServ Code Point* (DSCP), e identifica el comportamiento del paquete en cada encaminador intermedio, ya que será el contenido de este campo el que se utilice en tareas como planificación, elección de la cola en la que hay que situar el paquete, priorización del descarte de paquetes cuando sea necesario, etc.

Al marcarse los paquetes con un valor en su campo DSCP en función de sus características y tratarlos en los encaminadores intermedios de acuerdo a dicho valor, se establecen distintas clases de tráfico que obtendrán prestaciones diferentes. De esta forma, el modelo de servicios diferenciados es básicamente un esquema basado en prioridades relativas.

Para que un cliente reciba servicios diferenciados de su *Internet Service Provider* (ISP), debe tener un acuerdo (*Service Level Agreement*, SLA) con su ISP. Básicamente, un SLA especifica las clases de servicio soportadas y la cantidad de tráfico que se le va a permitir en cada clase. También debe existir este SLA entre dos dominios de Internet distintos, para especificar el trato que debe darse en cada dominio a cada una de las posibles clases de tráfico.

Los usuarios o las aplicaciones van a marcar los paquetes con el tipo de servicio que quieran recibir en función del SLA que tengan. Este SLA será también un compromiso por parte del ISP de las características de la conexión. Por ejemplo, en los sistemas de pago se pueden establecer unos determinados porcentajes de uso de cada una de las clases de tráfico definidas en función de la tarifa contratada por el cliente.

El modelo de servicios diferenciados tiene bastantes diferencias respecto al modelo de servicios integrados. Las principales son [XN99]:

- Hay un número limitado de clases de servicio, y así la cantidad de información que deben almacenar los sistemas intermedios es proporcional al número de clases, en lugar de al número de flujos existentes. De esta forma, el modelo de servicios diferenciados es bastante más escalable.

- Las operaciones más complejas, como las clasificaciones iniciales, el marcado de los paquetes, formación de paquetes, etc., sólo se hacen en las fronteras entre redes. Los nodos interiores en la red sólo tienen que arbitrar entre los paquetes existentes según la clasificación establecida. De esta forma, este modelo es mucho más sencillo de implementar. Es posible además realizar encaminadores frontera de redes y encaminadores internos de la red mucho más sencillos, rápidos y económicos.

- El modelo de servicios integrado permite proporcionar garantía a las aplicaciones, mientras que el modelo de servicios diferenciados sólo es un compromiso a dar trato preferencial a unas clases de tráfico sobre otras, lo cual no supone una garantía.

El modelo de servicios diferenciados define tres tipos de servicios posibles [Ber98], que de menores a mayores requisitos de QoS son: mejor esfuerzo, entrega asegurada (*Assured Forwarding*) y entrega rápida (*Expedited Forwarding*). El tráfico de mejor esfuerzo no tiene ningún tipo de garantía. El ISP intentará entregarlo en su destino, pero no se compromete a nada, ni siquiera a que llegue a su destino, pues si hay congestión en algún punto intermedio de la ruta podrá ser descartado.

El tráfico de entrega asegurada está pensado para los clientes que necesitan un servicio fiable por parte de los ISPs, incluso en situaciones de congestión. Los clientes deben tener un SLA con su ISP donde el cliente adquiere el derecho a enviar una cantidad de información y el ISP se compromete a tener recursos suficientes para poder manejar dicha cantidad. Es responsabilidad del cliente no superar la cantidad de información que tiene contratada, y si esto sucede el ISP puede descartar ese exceso.

El servicio de entrega rápida permite a un cliente tener retrasos y jitters bajos. Es ideal para las aplicaciones que generan un tráfico constante, o con picos de tráfico no muy grandes, como la telefonía por Internet y videoconferencia, o para crear redes privadas virtuales [XN99]. Como en todos los casos, el cliente debe tener un SLA con su ISP especificando la cantidad de información que está habilitado para enviar con este servicio. Al igual que en el caso anterior, es responsabilidad del cliente no superar la tasa de envío contratada para este servicio. El ISP se compromete a que el cliente tenga en la red el ancho de banda que tiene contratado.

1.6. Motivación y objetivos

A lo largo de esta sección se hará una pequeña introducción a los antecedentes sobre nuestro tema de estudio, para pasar a continuación a exponer la motivación que nos ha llevado a desarrollar este trabajo, así como los objetivos que se pretenden cubrir.

1.6.1. Antecedentes

Como ya se ha indicado en las secciones anteriores, proporcionar QoS a las aplicaciones ha sido y es una de las preocupaciones, y como consecuencia un área importante de trabajo, de la comunidad científica. Sin embargo, el hecho de que las aplicaciones no hayan necesitado QoS por parte de la red de interconexión hasta fechas recientes, unido a que los propios elementos de interconexión no disponían de la infraestructura necesaria para soportar esa QoS, ha dificultado la expansión de las técnicas propuestas hasta la fecha para proporcionar QoS.

Sin embargo, como ha quedado reflejado en la Sección 1.3, la tendencia actual en cuanto a las necesidades de las aplicaciones, para un correcto funcionamiento, va hacia una mayor disponibilidad de la red y un mayor número de recursos. Estos requisitos que las aplicaciones necesitan pueden ser relativos a un ancho de banda mínimo, una latencia máxima, escasa pérdida de paquetes, etc. Así pues, adaptar los sistemas de interconexión presentes y preparar los futuros para que sean capaces de proporcionar estos requisitos ha sido un tema fundamental de estudio en los últimos años, pero también lo será en los próximos.

Recientemente, un grupo de empresas líderes en el campo de las tecnologías de la información ha propuesto lo que pretende ser un nuevo estándar de interconexión entre nodos de procesamiento y dispositivos de entrada/salida, y para la comunicación entre procesadores. Se trata de la *Arquitectura InfiniBand (IBA)* [IBA99], y está siendo desarrollada por la *InfiniBand^{TM} Trade Association* (IBTA). La IBTA se constituyó en Agosto de 1999, y es un grupo de más de 200 empresas lideradas por un comité compuesto por miembros de Dell, Compaq, HP, IBM, Intel, Microsoft, y Sun Microsystems. Empresas como 3Com, Cisco Systems, Fujitsu-Siemens, Hitachi, Adaptec, Lucent Technologies, NEC y Nortel Networks actúan como patrocinadores. La IBTA publicó la primera especificación de la arquitectura de InfiniBand en octubre del 2000 [Inf00].

El objetivo perseguido por la arquitectura de InfiniBand es proponer una alternativa a los buses estándar, para la interconexión de procesadores entre sí y con dispositivos de entrada/salida. Como ya se ha visto, dichos buses se convierten en el cuello de botella de los servidores, además de presentar problemas de baja disponibilidad [Pfi01]. Por tanto, InfiniBand define una red de área de sistema (SAN) independiente de los sistemas operativos de los *hosts* y de la plataforma en concreto.

La arquitectura de InfiniBand incluye una serie de mecanismos que, convenientemente usados, pueden servir para proporcionar QoS a las aplicaciones actuales y futuras. En el Capítulo 2 se estudiarán detenidamente las principales características que presenta InfiniBand, incluidas aquellas que tienen que ver con QoS.

1.6.2. Motivación

Como se ha visto en las secciones anteriores, en los últimos años han surgido nuevas aplicaciones multimedia, que debido a que integran vídeo, audio, interactividad, etc., presentan nuevas exigencias que las redes de transmisión deben satisfacer. En determinadas ocasiones, esas aplicaciones requieren una garantía absoluta de que la red será capaz de proporcionar los requisitos solicitados. En ese caso, si no es posible satisfacer las exigencias, no llegará a comenzar a funcionar la aplicación.

Por otra parte, las arquitecturas de interconexión actuales, tanto para interconexión de dispositivos de entrada/salida, como de sistemas de procesamiento entre sí, presentan una serie de limitaciones que las hacen, en la mayoría de los casos, incapaces de proporcionar las nuevas exigencias de algunas aplicaciones. Así por ejemplo, uno de los sistemas de interconexión más utilizado es el bus. Este sistema presenta severas limitaciones en cuanto a escalabilidad y fiabilidad. Por tanto, es necesario que los sistemas de interconexión sean capaces de proporcionar una serie de nuevas características.

InfiniBand presenta una serie de propiedades que la hacen apropiada para ser utilizada para satisfacer las necesidades presentadas por las nuevas aplicaciones. Por tanto, la motivación que nos ha llevado a realizar este trabajo es evidente. Todo lo relacionado con InfiniBand es un tema de investigación actual que puede tener gran importancia en los próximos años. Si a esto unimos la relevancia que ya tiene conseguir proporcionar QoS a las aplicaciones, tenemos un campo de investigación puntero y novedoso. Así pues, estudiar cómo conseguir proporcionar QoS a las aplicaciones dentro del marco particular de InfiniBand es un campo de trabajo prometedor.

Desde nuestro punto de vista, el único estudio con cierta relación al que aquí se presenta en el campo de QoS en InfiniBand, es el publicado por Joe Pelissier [Pel00]. Sin embargo, este trabajo sólo se limita a hacer una breve descripción de las características de InfiniBand, particularizando en aquellas que el autor considera van a permitir proporcionar QoS, junto con una breve descripción de cómo implementar el modelo de servicios diferenciados con InfiniBand. En este trabajo no se dan resultados analíticos ni experimentales.

Hasta donde nosotros sabemos, no hay aún otros estudios similares en el campo de QoS en InfiniBand. Evidentemente, muchos han sido los trabajos ya realizados sobre QoS, pero casi siempre referidos a redes IP o ATM. Por otra parte, están empezando a aparecer trabajos sobre temas de investigación en InfiniBand, aunque relacionados con

otras líneas como el encaminamiento [SRF01, FLS02], el modelado de la arquitectura [BCQ03], o el diseño de un API para las aplicaciones [WGA01].

Es de esperar que siendo un tema tan importante, no tarden mucho tiempo en aparecer otros autores con propuestas relativas a cómo proporcionar QoS en InfiniBand. Incluso, consideramos probable que algunas de las propuestas que se realicen próximamente sean trasladadas a productos comerciales de InfiniBand. De esta forma, nos encontramos en el momento adecuado para realizar una investigación puntera, que incluso podría quedar plasmada en alguno de los productos comerciales de InfiniBand que finalmente sean comercializados.

Por otra parte, este trabajo va encaminado a proporcionar *garantía* de QoS a las aplicaciones. Pretendemos proporcionar garantía absoluta y no unas determinadas clases de servicio basadas en un esquema de prioridades. Este otro enfoque lo abordaremos próximamente, pero ya fuera de esta tesis doctoral.

Es evidente que proporcionar garantía de QoS es una tarea mucho más difícil que sólo comprometerse a dar un servicio basado en prioridades. Para garantizar QoS hay que hacer un buen dimensionado de los recursos de la red para distribuirlo entre los solicitantes de manera eficiente incluso en casos extremos de congestión.

1.6.3. Objetivos

Como ya se ha indicado en los párrafos anteriores, con este trabajo se pretende desarrollar el soporte necesario para proporcionar QoS a las aplicaciones en InfiniBand. Éste es el objetivo principal de esta tesis, y para conseguirlo se plantea cubrir una serie de objetivos parciales que se enumeran a continuación:

1. Estudiar las especificaciones de InfiniBand para conocer en profundidad las características de este nuevo estándar. Para ello utilizaremos fundamentalmente tanto las propias especificaciones [Inf00], los pocos libros que hasta la fecha hay publicados sobre el tema [Fut01, Sha02], así como los informes técnicos y el resto de material que hay disponible en la página web de InfiniBand [IBA99].

2. Realizar una clasificación de las posibles aplicaciones que pueden desarrollarse sobre InfiniBand, y categorizarlas en función de sus necesidades de QoS. Para ello partiremos de la clasificación realizada por Pelissier [Pel00], sobre la que propondremos algunas modificaciones.

3. Estudiar cómo garantizar ancho de banda a las aplicaciones que lo requieran, y proponer estrategias para conseguirlo. Para ello será necesario diseñar una metodología para el funcionamiento tanto de las aplicaciones como de las entidades encargadas de la administración de la red. En concreto, propondremos los pasos

que deberán seguir las aplicaciones que deseen tener garantías de QoS, y cómo deben las entidades de gestión usar los mecanismos encargados de proporcionar esa QoS. Específicamente, detallaremos cómo se debe rellenar o modificar la tabla de arbitraje para conseguir dar la garantía de QoS solicitada.

4. Estudiar cómo proporcionar garantía de latencia máxima. Para ello se hará un estudio previo de los diversos factores que influyen en la latencia final de comunicaciones que sufre una aplicación. Se estudiarán los modelos de conmutador más comunes, detallando en cada caso el grado de contribución de los diversos elementos en la latencia final observada.

5. Proponer un marco global donde proporcionar garantía tanto de ancho de banda como de latencia máxima. Estudiaremos las ventajas de este marco global y los posibles problemas que pudiera plantear. Así mismo, se analizará detalladamente la influencia de las decisiones tomadas, comparándolas, cuando sea posible, con otras alternativas.

6. Desarrollar un modelo formal sobre la base de la metodología propuesta. Dentro de este modelo formal, se enunciará un algoritmo para establecer cómo proporcionar la QoS requerida por las aplicaciones. Como consecuencia de todo esto se derivará una teoría que permitirá demostrar propiedades del servicio ofrecido y garantizar QoS a las aplicaciones.

7. Estudiar experimentalmente las propuestas realizadas. Este estudio se llevará a cabo utilizando un simulador de InfiniBand que ha sido desarrollado durante el transcurso de esta tesis doctoral. Este simulador está basado en otros anteriores usados en el grupo de investigación [Cas01, Fli01, San02] que están ampliamente validados. Mediante este estudio experimental comprobaremos como, incluso en situaciones de carga máxima de la red de interconexión, todas las aplicaciones obtienen los requisitos de QoS que previamente habían solicitado.

Estos son los objetivos que pretendemos cubrir con este trabajo. Al final de esta memoria volveremos a repasar dichos objetivos para ver el grado de cumplimiento que se ha obtenido.

1.7. Desarrollo de la Tesis

Para cubrir los objetivos propuestos en este trabajo, se ha estructurado esta memoria en los siguientes capítulos:

- Capítulo 1: Este capítulo introduce, de forma resumida, algunas de las materias relacionadas con este trabajo. Así por ejemplo, en él se comenta la tendencia

actual tanto en las redes como en la entrada/salida de altas prestaciones. También aquí se abordan las nuevas necesidades que plantean las aplicaciones actuales, y cómo se ha venido consiguiendo hasta ahora proporcionar QoS a las aplicaciones en los entornos IP.

- Capítulo 2: Se presenta un resumen detallado de algunas de las características más importantes de InfiniBand. Se detallan las diversas características que incorpora, así como la estructura arquitectónica que plantea. Se estudiará aquí también la infraestructura para la gestión de InfiniBand. Por último, se incluirán las características que plantea InfiniBand para conseguir proporcionar QoS.

- Capítulo 3: Plantea un marco para proporcionar garantía de ancho de banda a las aplicaciones. Se estudia en qué consiste el problema y se plantea una metodología capaz de conseguirlo. A continuación, se aborda el problema de garantizar latencia y se estudia la influencia de cada uno de los elementos que componen el conmutador. Por último. se plantea un marco global capaz de proporcionar ambos tipos de garantía.

- Capítulo 4: En este capítulo se plantea un modelo teórico para tratar el tema de estudio para lo cual se introducirán una serie de definiciones y se plantearán unas hipótesis de partida. En este capítulo se considera que las peticiones se ubican en la tabla, pero no se eliminan de ella, lo que simplifica mucho el tratamiento. Se enunciará un algoritmo para rellenar la tabla de arbitraje y sus condiciones de funcionamiento. Esto nos llevará a proponer y demostrar una serie de teoremas.

- Capítulo 5: En este capítulo se elimina la restricción impuesta en el capítulo anterior, con lo que las peticiones pueden ubicarse y retirarse de la tabla. Esto introduce una serie de consideraciones que son debidamente mostradas. Enunciaremos dos nuevos algoritmos para solucionar los problemas que ahora se plantean. Mediante una serie de teoremas se demuestra que la situación alcanzable tras insertar y eliminar peticiones en la tabla es equivalente a que sólo se hubieran ubicado las peticiones que restan tras las eliminaciones, y por tanto todo lo demostrado en el capítulo anterior es también válido con estas consideraciones. En concreto, se verá que la propuesta realizada es capaz de ubicar cualquier secuencia de peticiones en la tabla, si el número total de entradas no supera las disponibles.

- Capítulo 6: Haremos la evaluación experimental del modelo propuesto. Para ello utilizaremos un simulador desarrollado dentro del trabajo de esta tesis doctoral. Comprobaremos como con la metodología propuesta, incluso en situaciones de carga máxima, todas las aplicaciones consiguen la QoS que previamente habían solicitado.

- Capítulo 7: Para finalizar, se recogen en este capítulo las conclusiones fundamentales extraídas del estudio realizado, así como comentarios sobre la posible

continuidad del mismo. Se recogerán también en este capítulo los artículos publicados en revistas del área, así como en diversos Congresos y Simposios tanto nacionales como internacionales, fruto del trabajo realizado.

Capítulo 2

InfiniBand

Se presenta en este capítulo una breve descripción de InfiniBand, enumerando algunas de sus principales características. Hay que tener en cuenta que las propuestas que se presentan en esta tesis se plantean sobre InfiniBand, y es por tanto necesario indicar en qué consiste, cuál es su origen y el ámbito donde se utilizará. Este capítulo no pretende ser un recorrido exhaustivo por las especificaciones de InfiniBand, sino un resumen de algunas de sus características. Si el lector está interesado, o necesita más detalles sobre alguno de los aspectos aquí tratados, puede consultar las especificaciones de InfiniBand [Inf00] o los libros publicados sobre esta propuesta [Fut01, Sha02].

2.1. Introducción

InfiniBand es una tecnología de interconexión entre sistemas de procesamiento y dispositivos de E/S que permite formar una red de área de sistema (Figura 2.1). La arquitectura definida por InfiniBand es independiente del sistema operativo y de las plataformas que se usen.

Las dos principales metas que pretenden cubrirse con InfiniBand son: salvar las limitaciones que presentan los actuales buses PCI (cuellos de botella, fiabilidad, escalabilidad, etc.), y estandarizar las emergentes tecnologías propietarias en el terreno de los clusters (por ejemplo, Servernet, Myricom, Giganet, etc.).

Sin embargo, InfiniBand es mucho más que una simple sustitución del típico bus PCI. InfiniBand incorpora características que hasta ahora sólo podían encontrarse en grandes y costosos supercomputadores. Estas características son importantes para el montaje de clusters de altas prestaciones y permiten aprovecharse de las posibilidades que ofrece la tecnología actual.

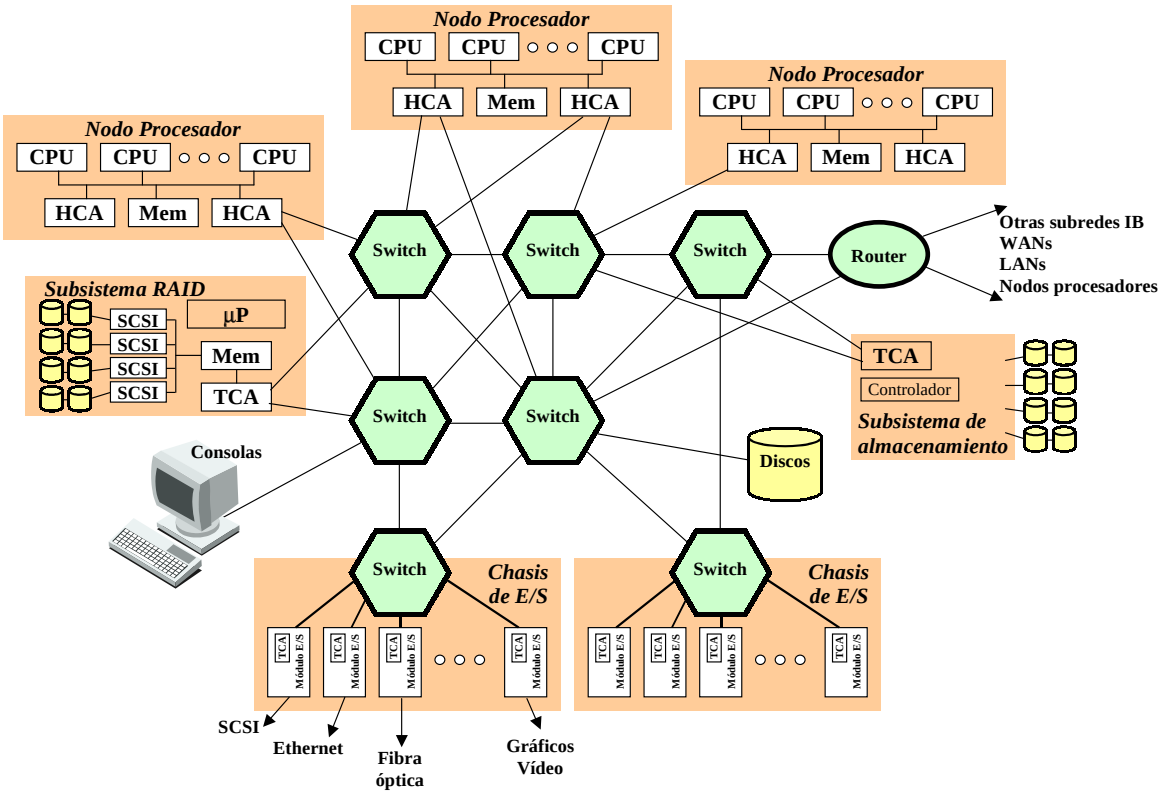


Figura 2.1: Red de área de sistema con InfiniBand.

2.1.1. Ámbito e historia

Las tecnologías de interconexión existentes no han logrado seguir el crecimiento en prestaciones que han experimentado en los últimos años los sistemas de procesamiento. Los servidores y aplicaciones actuales requieren gran ancho de banda y baja latencia, que la tecnología de E/S actual no es capaz de proporcionar. La tendencia actual es poner cada vez más funcionalidad en los dispositivos de E/S, con lo que estos dispositivos requieren sistemas de protección, aislamiento y calidad de servicio, que los sistemas actuales no pueden proporcionar.

Estas fueron las principales razones por las que algunas de las más importantes empresas del sector empezaron a estudiar un cambio de filosofía en las tecnologías de interconexión que se estaban usando hasta entonces. Cada una de estas empresas estaba interesada en resolver su problemática concreta. Así por ejemplo, unas estudiaron como mejorar las prestaciones en la interconexión entre el procesador y los dispositivos, otras como hacerlo en una LAN, etc. Pocos de estos intentos abordaban de forma genérica el problema de mejorar la E/S.

La arquitectura final de InfiniBand surge de la fusión de dos iniciativas existentes a finales de los años noventa: *Next Generation I/O* (NGIO) y *Future I/O* (FIO). Ambas iniciativas tenían en común buena parte de sus metas, y pronto se dieron cuenta que no

había mercado para dos propuestas distintas. Así, a finales de 1999 decidieron converger en un único grupo de trabajo que diera como fruto una única propuesta. De esta forma, en octubre de 1999 se formó la *InfiniBand Trade Association* (IBTA).

No está muy claro de donde surgió el nombre de InfiniBand, pues todos los implicados coinciden en que dicho nombre no es de su agrado. Según algunos autores [Fut01] parece que dicho nombre proviene de infinite-bandwidth. Sin embargo, InfiniBand es una marca registrada que oficialmente no tiene ningún significado.

La arquitectura de InfiniBand está basada en muchos conceptos ya existentes en otras tecnologías, como conmutación, canales virtuales, planificación, etc. También utiliza características de señalización existentes ya en Fiber Channel o Ethernet. De esta manera se consigue reutilizar conceptos ya usados en otros campos, pero ahora con otros objetivos.

En octubre del año 2000 se publicó la primera versión (release 1.0) de las especificaciones de InfiniBand [Inf00]. Consisten en dos volúmenes, estando dedicado el primero de ellos a los aspectos arquitectónicos, mientras que el segundo se centra en aspectos electro-mecánicos. Estas especificaciones han sido desarrolladas por más de 150 ingenieros de más de 30 empresas, y revisadas por personal de otras 150 empresas. En junio de 2001 se produjo la primera modificación a las especificaciones (release 1.0a) que consistió en corrección de errores e introducción de algunas clarificaciones. Está previsto que próximamente esté terminada la siguiente modificación a las especificaciones (release 1.2). También se está trabajando ya en otra actualización más ambiciosa para el año que viene que sería la versión 2.0.

Quizás el punto más importante de toda esta historia es que un gran número de ingenieros de las más importantes empresas del sector han colaborado para definir y hacer converger las mejores ideas y conceptos en una arquitectura común. Si esta arquitectura será de uso común tanto en el sector de E/S como para la construcción de clusters es una cuestión que sólo el tiempo dirá.

2.1.2. Servidores de E/S

Muchos de los problemas que actualmente plantea la E/S son debidos a la diferencia de crecimiento de las prestaciones de los procesadores comparadas con los buses utilizados para dicha E/S. En la Figura 2.2 se puede observar la diferencia en la evolución de los procesadores y del tradicional bus PCI. Otras de las limitaciones que plantea la forma actual de realizar la E/S radica en otra serie de factores como fiabilidad, escalabilidad, seguridad, etc.

Para solucionar estos problemas que se plantean en la E/S se pueden observar varias opciones:

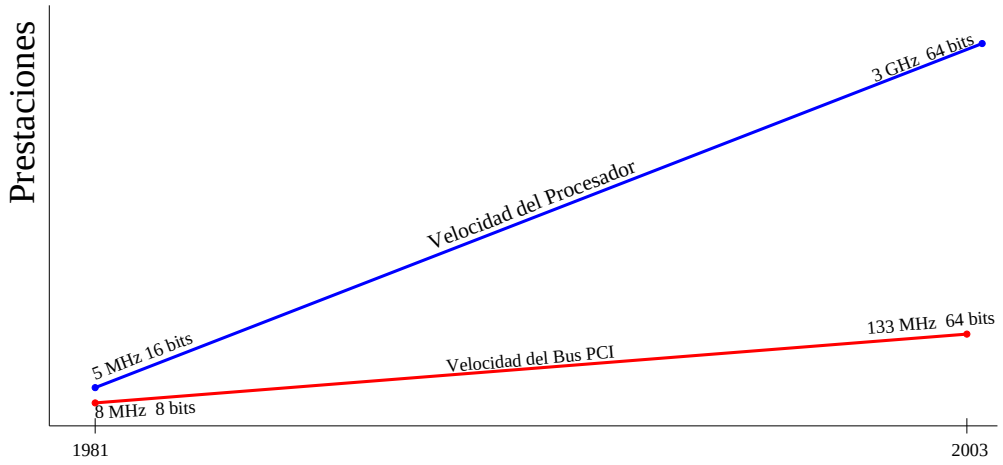


Figura 2.2: Evolución de la velocidad de los procesadores frente a la del bus PCI.

- Incorporar más buses de E/S. Haciendo uso del paralelismo, aprovechar las replicaciones para poder hacer transferencias en paralelo. Sin embargo, esta opción sólo soluciona algunos de los problemas, pero no permite incorporar algunas de las prestaciones que son interesantes y que no incorporan los sistemas actuales.
- Expandir las tecnologías de E/S actuales para aumentar sus prestaciones. De esta forma se podría conseguir aumentar el ancho de banda o mejorar ciertos modos de funcionamiento. Por las mismas razones citadas anteriormente, esta opción tampoco permite incorporar nuevas características demandadas por las aplicaciones y servicios actuales.
- Desarrollar nuevas tecnologías. Esta parece ser la única alternativa para poder hacer frente a las nuevas características que demandan los actuales sistemas.

De esta forma, las tecnologías de E/S actuales (como por ejemplo el bus PCI) ya no son suficientes para los servidores actuales y futuros. En este punto es donde surge InfiniBand como un intento de cubrir este campo, proporcionando las características de velocidad, fiabilidad, escalabilidad y seguridad, demandadas por los sistemas actuales.

La arquitectura que plantea InfiniBand puede considerarse una revolución más que una evolución, pues plantea una ruptura con los sistemas tradicionales de E/S. Sin embargo, pocas son las propuestas realmente nuevas que plantea InfiniBand, ya que la mayoría de ellas ya estaban siendo usadas en otros campos de las redes de interconexión.

En la Tabla 2.1 se recogen las principales diferencias que plantea InfiniBand respecto a los sistemas tradicionales de E/S. Se puede observar que con los cambios que se plantean se consigue mejorar la eficiencia, escalabilidad, aislamiento, recuperación, redundancia, ancho de banda, velocidad, etc.

Como puede observarse, se pasa del tradicional esquema de E/S mapeado en memoria, a un esquema donde las unidades de E/S funcionan con un sistema de canales

Cambio		Beneficios
Desde:	**A:**	
Mapeada en memoria	Basada en canales	Eficiencia, escalabilidad, aislamiento, recuperación
Bus paralelo	Sistema conmutado	Escalabilidad, aislamiento, redundancia, menor número de pines, modularidad, mayor ancho de banda por sección
Acceso a un bus compartido	Conexiones punto a punto	Mayores distancias y velocidades
Cargas y almacenamientos	Planificando el DMA	Mejora la eficiencia de la CPU
Espacio de direccionamiento único	Dominios de direccionamiento independientes	Protección, aislamiento, recuperación, fiabilidad

Tabla 2.1: Resumen de algunas de las diferencias de InfiniBand con los sistemas de E/S tradicionales y los beneficios que plantean.

virtuales y los dispositivos harán uso de las facilidades que proporcionan dichos canales. De esta manera, es posible mejorar la eficiencia de la CPU al poder hacer varias operaciones de E/S con el mismo dispositivo en paralelo, usando distintos canales virtuales. De la misma forma, se puede conseguir mejor escalabilidad, aislar unas operaciones de otras y la recuperación en caso de error en uno de los canales virtuales. Podríamos decir que al dividir el ancho de banda total del enlace entre los distintos canales virtuales se pueden conseguir unas prestaciones escalables, balanceadas y predecibles.

Al ser un sistema conmutado, la interconexión de los dispositivos se realiza a través de los conmutadores. Esto afecta al modo de encaminamiento de los datos entre los distintos dispositivos. Cada dispositivo debe tener una dirección para identificarlo de forma unívoca. De esta forma, en cada petición, se debe hacer constar la dirección del dispositivo al que se desea acceder. Como se indicará más adelante, en cada subred de InfiniBand hay un Subnet Manager que se encarga de construir las rutas válidas para transmitir desde cualquier dispositivo a cualquier otro de los que haya conectados.

Al ser InfiniBand un sistema conmutado se evitan muchas de las limitaciones de los tradicionales sistemas de E/S basados en buses compartidos. Se logra mayor escalabilidad, pues como mucho puede ser necesario añadir más conmutadores para añadir nuevos dispositivos. De esta forma, no se limita el número máximo de dispositivos de E/S a conectar. Por la misma razón, la velocidad de un puerto es independiente del número de nodos conectados al sistema. También se consigue mejorar la redundancia,

pues se pueden tener caminos físicos alternativos para acceder a un mismo dispositivo. Se consigue poder tener aislada una zona del tráfico de otras, así como mejorar la modularidad del sistema. Al ser un sistema conmutado se obtiene un ancho de banda efectivo mucho mayor que el de sólo uno de sus enlaces.

En InfiniBand se puede conseguir protección de los datos, pues cada aplicación decide qué cosas de su espacio de direccionamiento hace accesibles a los procesos que indique. Además, proporciona capacidad de particionamiento lo cual permite al administrador fijar fronteras a los accesos. Estas particiones protegen a los nodos de accesos inapropiados de componentes no deseados.

Por otro lado, buena parte de las tareas de gestión están construidas dentro de la propia arquitectura. De esta forma se intenta que la configuración y gestión sean sencillas para los usuarios y, siempre que sea posible, se realicen directamente por el propio hardware. Así, los dispositivos se detectan automáticamente al ser conectados y además se autoconfiguran.

En la Tabla 2.2 se incluye una comparativa de InfiniBand con otras tecnologías de E/S utilizadas hasta ahora [Mel01a, Mel01b]. Para esta comparativa se han elegido los siguientes sistemas de E/S:

- PCI-X: Es la última versión del tradicional bus PCI. Puede funcionar hasta a 133 MHz con un ancho de banda de 8,5 Gbps. Ya hay propuestas para conseguir la compatibilidad de los actuales sistemas PCI en su migración a InfiniBand. En concreto, Mellanox ha propuesto el $InfiniPCI^{TM}$ [Mel01a] como un puente transparente entre los actuales sistemas PCI y los nuevos sistemas InfiniBand.

- Fiber Channel: Diseñado a principio de los años noventa como un bus estándar para proporcionar un nivel físico de alta velocidad, capaz de satisfacer las necesidades de los servidores de E/S. Funciona a 2 Gbps y permite transferencia de bloques. Algunas de las ideas que incorpora InfiniBand para los servidores de E/S han sido utilizadas previamente en Fiber Channel.

- HyperTransport^{TM}: Es una propuesta de interconexión de circuitos de alta velocidad y altas prestaciones en una placa impresa. Para un número equivalente de líneas, consigue velocidades mayores que el tradicional bus PCI.

- RapidIO^{TM}: Al igual que la anterior propuesta, ésta también se centra en la interconexión de circuitos de alta velocidad y altas prestaciones en una placa impresa. También permite la interconexión de placas entre sí.

- 3GIO [Bha02]: También llamado PCI-Express, es un nuevo estándar que intenta conseguir comunicación con gran ancho de banda usando sólo unas cuantas líneas, frente al poco ancho de banda y gran número de líneas que necesita el tradicional bus PCI. Las metas que intenta conseguir 3GIO son la interconexión chip-to-chip

con un bus local de alta velocidad y mejorar las prestaciones y características de los tradicionales slots PCI.

Característica	InfiniBand	PCI-X	Fiber Channel	Hyper Transport	Rapid I/O	3GIO
Ancho de banda (Gbps)	2,5/10/30	8,51	1/2,1	12,8; 25,6; 51,2	16/32	2,5
Número de líneas	4/16/48	90	4	55, 103, 197	40/76	4
Distancia máxima	Kms	Cms	Kms	Cms	Cms	Cms
Medio físico	Cable de cobre, fibra o en placa	En placa	Cobre y fibra	En placa	En placa	En placa
Comunicaciones simultáneas	15 VLs + VL gestión				3 flujos	
Soporte para RDMA	Sí					
Soporte para interfaz virtual	Sí					
Control de flujo extremo a extremo	Sí			Sí	Sí	Sí
Particionado de la memoria	Sí		Sí			
Calidad de servicio	Sí		Sí		Sí	
Fiabilidad	Sí		Sí	Sí	Sí	Sí
Escalabilidad	Sí		Sí	Sí	Sí	Sí
Tamaño máximo de paquete	4 KB	No basado en paquetes	2 KB	64 bytes	256 bytes	

Tabla 2.2: Comparativa de InfiniBand con otras tecnología de E/S.

A modo de resumen, hay que señalar que InfiniBand no es propiamente una alternativa para realizar la E/S. Mientras que el resto de propuestas analizadas en esta sección son alternativas a los buses locales, InfiniBand es mucho más que eso. El principal fabricante de procesadores, Intel, parece que tiene la intención de apostar por 3GIO como la solución para reemplazar a los tradicionales buses PCI [Bha02]. Sin embargo, según algunos fabricantes de productos InfiniBand, 3GIO no es competencia de

InfiniBand, sino que mejora el ancho de banda de los buses locales de E/S y las características de los slots PCI, que complementan y ayudan a las nuevas características de los conmutadores de InfiniBand [Mel01b].

2.2. InfiniBand, una arquitectura conmutada

InfiniBand, tal y como se muestra en la Figura 2.1, define una red de área de sistema (System Area Network, SAN) para conectar ordenadores, sistemas de E/S y dispositivos de E/S. InfiniBand proporciona la infraestructura adecuada para comunicación y gestión, tanto para transacciones de E/S como para comunicación entre ordenadores. Un sistema InfiniBand puede variar desde un pequeño servidor de un procesador y unos cuantos dispositivos de E/S conectados, hasta un supercomputador masivamente paralelo con miles de procesadores y dispositivos de E/S que está conectado vía Internet a otras plataformas de procesamiento y/o sistemas de E/S.

InfiniBand define una interconexión conmutada que permite a muchos dispositivos intercambiar datos de forma simultánea, con gran ancho de banda y baja latencia. Al ser un sistema conmutado, se pueden conseguir características como protección, fiabilidad, escalabilidad, seguridad, etc., hasta ahora impensables en sistemas de E/S, e incluso en la mayoría de las redes habituales para conexión de computadores. Por otra parte, la plataforma que platea InfiniBand debe permitir una automática configuración y gestión, facilitando al usuario y/o administrador las habituales y tediosas tareas de administración en este tipo de entornos.

Un nodo final en una SAN InfiniBand puede comunicarse por medio de múltiples puertos del conmutador al que está conectado, pudiéndose habilitar de esta manera caminos alternativos. Así, se podría aprovechar la disponibilidad de caminos alternativos tanto para incrementar el ancho de banda real, como para permitir tolerancia a fallos.

Como es conocido, una SAN InfiniBand consiste en nodos de procesamiento y unidades de E/S conectados por medio de un sistema de conmutadores y encaminadores. Por ejemplo, se podría tener interconectado un sistema de procesamiento con un disco SCSI mediante el adaptador de InfiniBand correspondiente. Pues bien, InfiniBand no define el tipo de disco a utilizar, ni las órdenes a usar para intercambiar información con él, ni las características del sistema operativo o a quién está asociado dicho disco. InfiniBand sólo define cómo deben transportarse los datos y órdenes, entre el manejador de E/S en el sistema de procesamiento y el adaptador del disco SCSI.

InfiniBand permite a las unidades de E/S comunicarse entre ellas y con cualquier sistema de procesamiento existente en el sistema. De esta manera, una unidad de E/S tiene la misma capacidad de comunicación que cualquier otro nodo de procesamiento.

2.2.1. Topología

InfiniBand tiene una topología conmutada con conexiones punto a punto, que permite a un nodo acceder a otros nodos independientemente de su situación física en la red. La misión de la red es trasegar paquetes desde un nodo fuente hasta uno, o más, nodos destino.

En InfiniBand están permitidas tanto topologías regulares como irregulares. El sistema de administración será capaz de identificar cualquier topología formada, y construir las tabla de encaminamiento adecuadas para permitir la interconexión entre dos elementos cualquiera conectados a través de la red.

Desde un punto de vista de alto nivel, InfiniBand sólo es un medio para interconectar nodos entre sí, tal y como puede verse en la Figura 2.3, donde un nodo puede ser un sistema de procesamiento, una unidad de E/S o un encaminador hacia otra red.

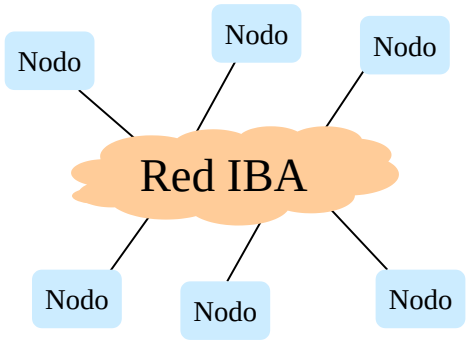


Figura 2.3: Red InfiniBand.

Una red InfiniBand puede subdividirse en subredes interconectadas entre sí mediante routers o encaminadores, tal y como puede verse en la Figura 2.4. Los nodos finales pueden conectarse a una única subred o a múltiples subredes a través de distintos interfaces.

En cada subred debe existir un Subnet Manager, encargado de ciertas tareas de control de la subred. No todos los dispositivos tienen la capacidad para llegar a ser Subnet Manager. Sin embargo, cualquier dispositivo que tenga esa capacidad, puede llegar a actuar de Subnet Manager, aunque sólo uno de ellos lo hará en un momento dado. Como se verá más adelante, la misión de este Subnet Manager es configurar adecuadamente los dispositivos, así como controlar todos los aspectos de gestión y administración de la subred.

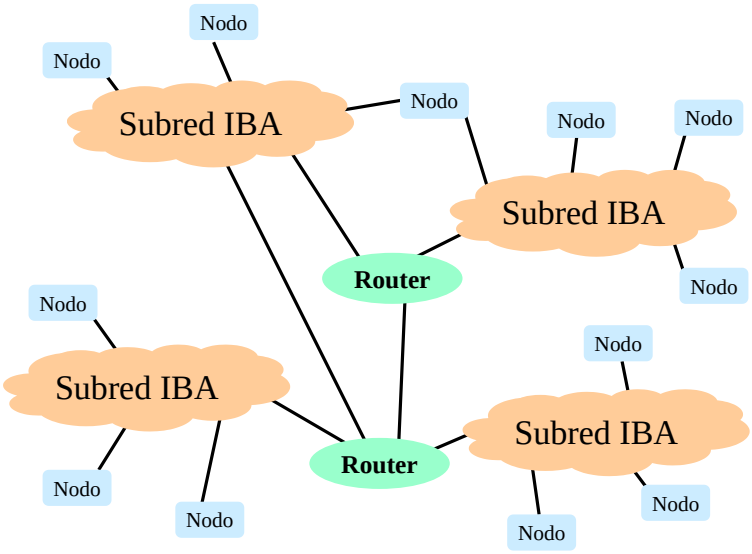


Figura 2.4: Componentes en una red InfiniBand.

2.2.2. Componentes en InfiniBand

Como se ha visto en la sección anterior, una subred InfiniBand puede estar compuesta por nodos finales, conmutadores, encaminadores y el Subnet Manager, interconectados todos ellos por enlaces. Este esquema es el que se ilustra en la Figura 2.5. Estos enlaces pueden ser de cable, de fibra óptica o grabados en la placa base.

Entre cualquiera de los componentes de la red pueden existir múltiples enlaces, consiguiendo de esta forma incrementar el ancho de banda, así como mejorar la tolerancia a fallos. Además, cualquiera de los componentes de la red InfiniBand puede estar conectado a un único conmutador, a varios, o directamente con cualquier otro dispositivo sin pasar por un conmutador. Sin embargo, la conexión directa entre dos nodos finales forma una subred independiente, sin conexión con el resto de dispositivos. En este caso, ambos se pondrán de acuerdo para que uno de ellos actúe como Subnet Manager.

Un *channel adapter* es el componente en un nodo final que lo conecta a la red. Un channel adapter es un dispositivo DMA programable con características especiales de protección, que permite que las operaciones de DMA sean iniciadas de forma local o remota.

Cada channel adapter tiene uno o varios puertos, tal y como puede verse en la Figura 2.6. Normalmente cada uno de ellos suele estar conectado a un puerto de un conmutador, aunque también es posible conectarlos entre sí directamente.

Cada puerto del channel adapter tiene su propia dirección local configurada por el Subnet Manager. La entidad local que se comunica con el Subnet Manager para la configuración del channel adapter es el *Subnet Management Agent*. Por otra parte, cada puerto tiene su propio conjunto de buffers de envío y recepción, con lo que es capaz de

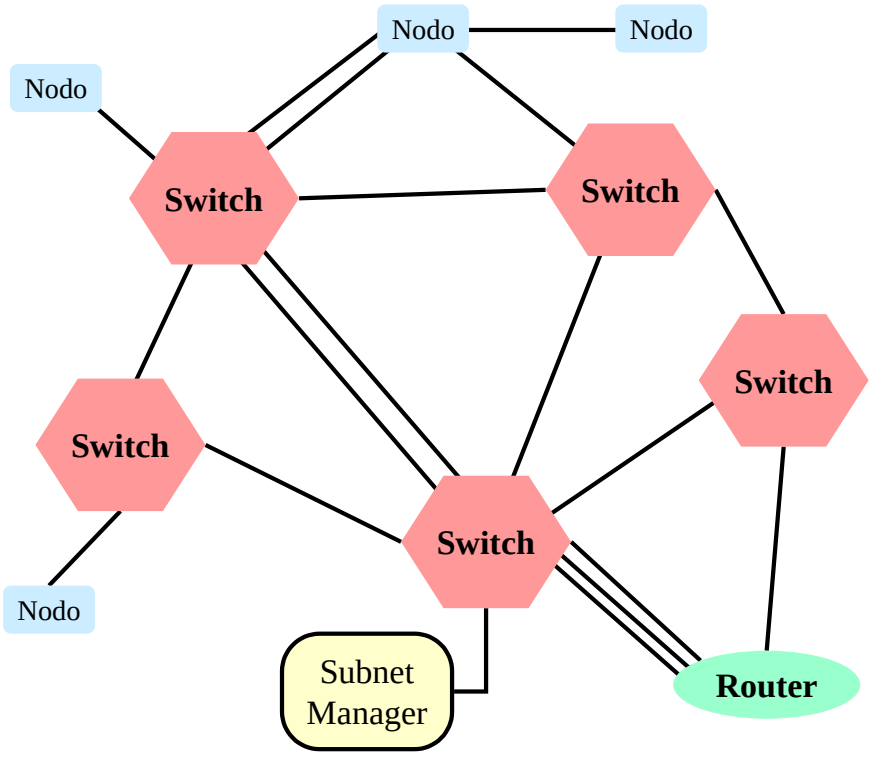


Figura 2.5: Componentes en una subred InfiniBand.

estar enviando y recibiendo información al mismo tiempo. El almacenamiento de los buffers está distribuido en varios canales virtuales (VL), teniendo cada VL su propio control de flujo.

InfiniBand permite el particionado virtual de la red por medio de la definición de dominios de acceso lógico. Cada puerto de cada channel adapter pertenece al menos

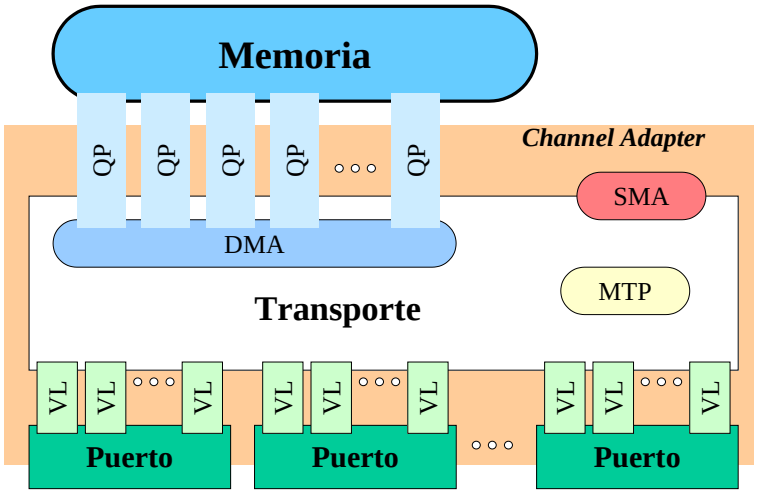


Figura 2.6: Estructura de un channel adapter.

a una partición y puede comunicarse sólo con otros puertos que también pertenezcan a alguna de sus particiones. De esta forma, la propia red proporciona un nivel de protección.

Los channel adapters pueden clasificarse en *Host Channel Adapter* (HCA) o en *Target Channel Adapter* (TCA). Un HCA es para sistemas de procesamiento con gran capacidad para ejecutar distinto tipo de software. Por otra parte, un TCA es para dispositivos de E/S, que por su simplicidad, no suelen tener capacidad para ejecutar software. Sin embargo, conceptualmente tanto el HCA como el TCA son idénticos. Desde un punto de vista arquitectónico, el HCA difiere del TCA porque el HCA tiene una interfaz más estructurado que ofrece al sistema operativo.

En contraste con los channel adapters, los conmutadores ni generan ni consumen paquetes, salvo los propios de gestión de la subred. Los conmutadores simplemente transmiten paquetes desde un puerto de entrada hacia un puerto de salida, en base a la dirección destino que el paquete lleva en su cabecera.

Los conmutadores son el eje central de una subred InfiniBand. Disponen de una serie de puertos donde se conectan los enlaces. La estructura de un conmutador es la mostrada en la Figura 2.7. A todos los efectos, los conmutadores son completamente transparentes para los nodos finales. Es decir, ellos no saben de su existencia, ni mucho menos les son accesibles. Tan sólo serán accedidos para tareas de gestión y configuración y siempre por las entidades correspondientes.

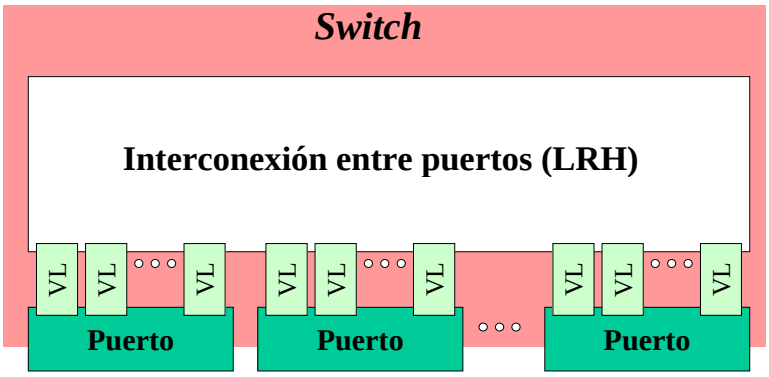


Figura 2.7: Estructura de un conmutador de InfiniBand.

Los paquetes que cruzan los conmutadores permanecen inalterados en su contenido, aunque sí puede que sufran cambios en su cabecera (CRC, VL, etc.). Sin embargo, estos cambios de la cabecera no influyen en el concepto de la comunicación. Es decir, la información que envía un nodo se empaqueta y se le añade una cabecera. Este paquete viajará a través de uno o varios conmutadores hasta llegar al destino, entregándose en este momento al nodo final tal y como había salido del origen, sin que éste sea consciente de los cambios que haya sufrido la cabecera a lo largo del trayecto.

Los conmutadores tienen unas tablas de encaminamiento para decidir por qué puerto sacar cada paquete. Estas tablas se cargan en el momento del arranque, o tras un cambio en la topología, por el Subnet Manager. El conmutador, en base a esas tablas y a la dirección destino que viene especificada en la cabecera del paquete, debe decidir por qué puerto reenviar cada paquete que le llega. Según las especificaciones, no todos los conmutadores deben tener capacidad de reenvío multidestino, dejando este aspecto a criterio de los fabricantes.

Como sabemos, cualquier nodo puede estar conectado a un conmutador por varios enlaces para aumentar el ancho de banda, disponibilidad, fiabilidad y tolerancia a fallos. En ese caso, el conmutador podría usar esos enlaces redundantes para balancear la carga de los distintos puertos del mismo nodo final, o para tener redundancia.

Por otra parte, los encaminadores o routers se encargan de comunicar entre sí distintas subredes. Los encaminadores tampoco tienen capacidad para generar o consumir paquetes (sin contar los de control). Nuevamente, se limitan a trasvasarlos desde uno de sus puertos de entrada a uno de los de salida. En este caso la decisión del puerto por donde salir se toma en base a una dirección global (GID). Esta dirección global es única en todas las redes InfiniBand, mientras que la dirección local (LID) es propia de cada subred.

El esquema básico de un encaminador es el mostrado en la Figura 2.8. Al igual que los conmutadores, los encaminadores tienen varios puertos con capacidad de recibir y transmitir de forma simultánea. El espacio para almacenamiento también está distribuido entre varios canales virtuales.

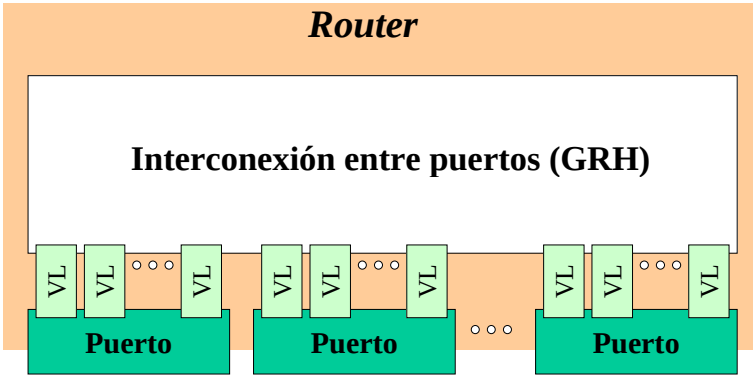


Figura 2.8: Estructura de un encaminador de InfiniBand.

Los encaminadores no son completamente transparentes para los nodos finales, puesto que el nodo fuente debe especificar la dirección local (LID) del encaminador de salida al exterior de su subred, además de indicar la dirección global (GID) del nodo destino.

Cada encaminador envía los paquetes hacia la siguiente subred a otro encaminador, hasta que el paquete llega a la subred destino. El último encaminador envía el paquete al nodo final utilizando la dirección local (LID) que se especificó en el origen.

Al igual que los conmutadores, los encaminadores también pueden estar interconectados por más de un enlace. En este caso también se pueden usar esos enlaces adicionales para repartir el ancho de banda o para cuestiones de redundancia para tolerancia a fallos.

El encargado de configurar los encaminadores es el Subnet Manager. Éste se encarga de configurar toda la información relativa a la subred, como direcciones asignadas, qué VLs usar o información sobre las particiones.

2.2.3. Características de InfiniBand

A lo largo de las próximas secciones se indican detenidamente las características fundamentales de InfiniBand. Así, algunos de los temas que tratar son: las colas, los tipos de servicio, las claves para protección, el direccionamiento de memoria virtual, los dominios de protección, las particiones, los canales virtuales, el control de la tasa de inyección, el modo de direccionamiento, el envío multidestino (multicast) y los verbos de la capa de transporte.

2.2.3.1. Pares de colas

Los *pares de colas* (queue pairs, o QP) son la interfaz virtual que el hardware proporciona a un productor de información en InfiniBand, y el puerto de comunicación virtual que proporciona para el consumidor de dicha información. De esta forma, la comunicación tiene lugar entre un QP fuente y un QP destino. La arquitectura permite hasta 2^{24} QPs por channel adapter, de forma que las operaciones en cada QP son independientes de las del resto de QPs. La interfaz para la comunicación que proporciona InfiniBand sería el mostrado en la Figura 2.9.

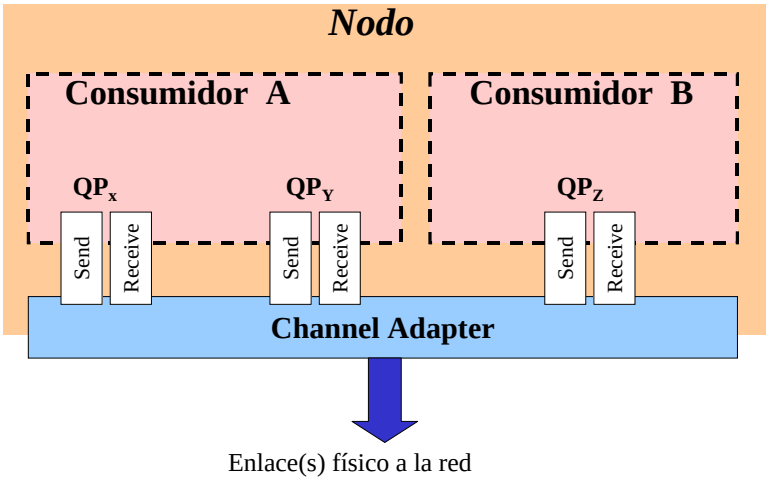


Figura 2.9: Interfaz para la comunicación en InfiniBand.

Cada QP permite tener aislada y protegida esa comunicación de la del resto de QPs u otros consumidores. Un QP puede considerarse un recurso privado asignado a un único consumidor. Sin embargo, un consumidor puede consumir de múltiples QPs, tal y como se muestra en la Figura 2.9.

Todos los tipos de servicio, salvo el *raw datagram*, necesitan tener establecido un QP asociado. Para los tipos de servicio con conexión, se establece una comunicación previa entre ambos extremos para asociar un QP origen con un único QP en el nodo destino. Los tipos de servicio sin conexión utilizan un QP genérico compartido por varias aplicaciones.

2.2.3.2. Tipos de servicio

Cada QP debe ser configurado para proporcionar un determinado tipo de servicio. Dicho tipo de servicio está basado en cómo interactúan los QPs fuente y destino, que deben estar configurados con el mismo tipo de servicio. Hay tres características que identifican a cada tipo de servicio:

- Servicio orientado a conexión o no orientado a conexión. El servicio no orientado a conexión suele llamarse también *datagrama*. En un tipo de servicio orientado a conexión cada QP fuente está asociado con un único QP destino, y viceversa. En un tipo de servicio no orientado a conexión está permitido que un QP envíe/reciba paquetes a/desde cualquier otro QP en cualquier nodo.

- Servicio con confirmación o sin confirmación. En un servicio confirmado, cuando un QP recibe un paquete debe confirmar al QP origen que lo ha recibido correctamente. Estos mensajes de confirmación pueden ir integrados en otro paquete con información, o en un mensaje ACK o NAK (negative acknowledged) propio. El servicio confirmado se dice que es fiable, pues el protocolo de transporte garantiza un envío sin errores y con entrega en orden para el posterior reensamblado de los paquetes en un mensaje de nivel superior. Por contra, el servicio sin confirmación se dice que es no fiable pues el protocolo de transporte no asegura que la información llegue a su destino.

- Servicio de transporte de InfiniBand u otro tipo de transporte. Como se verá más adelante, el servicio de transporte de InfiniBand permite transmitir paquetes *en bruto* encapsulando paquetes de otros protocolos de transporte, como por ejemplo IPv6, o de otros tipos de redes.

Así pues, los tipos de servicios definidos en InfiniBand, y sus características, son los mostrados en la Tabla 2.3.

Tipo de Servicio	Orientado a Conexión	Confirmado	Transporte
Reliable Connection	Sí	Sí	InfiniBand
Unreliable Connection	Sí	No	InfiniBand
Reliable Datagram	No	Sí	InfiniBand
Unreliable Datagram	No	No	InfiniBand
Raw Datagram	No	No	En bruto

Tabla 2.3: Tipos de servicio de transporte en InfiniBand.

Evidentemente, los tipos de servicio orientados a conexión requieren que la aplicación origen inicie un proceso de establecimiento de conexión. De esta forma se asocia un QP del nodo origen con un QP del nodo destino. Dependiendo del tipo de servicio establecido serán válidas unas operaciones u otras, de las definidas por InfiniBand.

2.2.3.3. Claves

Las *claves* (keys) permiten a InfiniBand proporcionar un cierto nivel de aislamiento y protección del tráfico. Estas claves son unos valores asignados por las entidades de administración y que luego se insertarán en los paquetes según la función y destino al que vayan dirigidos. Las aplicaciones sólo podrán acceder a los paquetes que contengan claves para los que ellas estén habilitadas. Los diferentes tipos de claves son:

- **Management Key** (M_Key). Esta clave se usa para tareas de gestión. Las administra el Subnet Manager, que puede asignar una distinta a cada puerto. Una vez hecha la asignación, todo el tráfico de control con ese puerto deberá llevar esa clave insertada en los paquetes.
- **Baseboard Management Key** (B_Key). Permite que actúe el gestor de subred en placa (subnet baseboard manager). Esta clave la contienen cierto tipo de paquetes de gestión de la subred.
- **Partition Key** (P_Key). Permite la división lógica de la subred en distintas zonas. Cada adaptador contiene una tabla de claves de partición que define las particiones para las que ese adaptador está habilitado. Hay un gestor de particiones (Partition Manager, PM) único, que se encarga de gestionar las claves de las particiones.
- **Queue Key** (Q_Key). Permite controlar el derecho de acceso a las colas para los servicios sin conexión (datagram). De esta forma, dos nodos que no hayan establecido previamente una conexión pueden intercambiar información de forma que esta clave identifique unívocamente a los interlocutores.

- **Memory Keys** (L_Key y R_Key). Permite el uso de direcciones de memoria virtuales y dota al consumidor de un mecanismo para controlar el acceso a dicha memoria. El consumidor le especifica al adaptador una zona de memoria y recibe de éste una L_Key y otra R_Key. El consumidor usa la L_Key en las gestiones locales de memoria, y pasa la R_Key a los consumidores remotos para que la usen en las operaciones remotas de DMA (RDMA).

Por sí solas, las claves no proporcionan seguridad pues dichas claves están disponibles en la cabecera de los paquetes que fluyen por la red, con lo que físicamente podrían ser accedidas por cualquier entidad presente en la subred. Sin embargo, sí que permiten hacer un aislamiento virtual de forma que cada adaptador sepa qué mensajes le atañen directamente a él, o van dirigidos a otro tipo de entidades.

2.2.3.4. Direcciones de memoria virtual

La arquitectura de InfiniBand está preparada para manejar directamente direcciones virtuales. Una aplicación de InfiniBand puede especificar direcciones virtuales y es el adaptador el que convierte la dirección virtual a una dirección física. Cada consumidor registra regiones de memoria virtual con su adaptador, y éste le devuelve dos claves llamadas L_Key y R_Key. A partir de ese momento el consumidor utilizará L_Key en las peticiones que requieran un acceso a la memoria de esa región.

InfiniBand también permite el acceso remoto (RDMA) a una región de memoria previamente registrada en el adaptador. Para el RDMA el consumidor pasa a otro consumidor remoto la R_Key y una dirección virtual de un buffer en esa región de memoria. Ese consumidor remoto utilizará en sus accesos a esa zona remota de memoria la R_Key que le han proporcionado.

2.2.3.5. Dominios protegidos

Los dominios protegidos permiten conceder distintos modos de acceso a las zonas de memoria registrada. Al registrar con el adaptador una zona de memoria se consigue también proporcionar un alto nivel de protección contra accesos inadvertidos o no autorizados. Mediante estos dominios, un consumidor puede controlar qué zonas de memoria de las que tiene registradas son accesibles por sus colas.

Antes que un consumidor reserve un QP o registre una zona de memoria, debe crear uno o más dominios protegidos. Tanto los QPs como las zonas de memoria registradas están asociadas a un dominio protegido. Tanto las claves L_Keys como las R_Keys de un dominio de memoria concreto sólo son válidas para los QPs creados para el mismo dominio protegido.

2.2.3.6. Particiones

Las particiones permiten aislar de forma virtual zonas de una subred InfiniBand. Este particionamiento no está ligado a límites establecidos por subredes, conmutadores o encaminadores. Son zonas virtuales que permiten aislar ciertas zonas de la subred.

Cada puerto o interfaz de la subred pertenece al menos a una partición y puede pertenecer a varias de ellas. El gestor de particiones asigna claves de partición (P_Keys) a cada adaptador de la subred. Estas claves se insertan en los paquetes que se envían luego a la red.

Cuando una entidad de la subred recibe un paquete con una P_Key para la que no está habilitado debe descartar dicho paquete. Los conmutadores y los encaminadores pueden ser configurados por el gestor de particiones para que también descarten paquetes aislando físicamente zonas de la subred.

2.2.3.7. Canales virtuales

Los canales virtuales (virtual lanes, VL) constituyen un mecanismo para crear múltiples enlaces virtuales con un único enlace físico. Un canal virtual representa un conjunto de buffers de transmisión y recepción en un puerto, tal y como puede verse en la Figure 2.10.

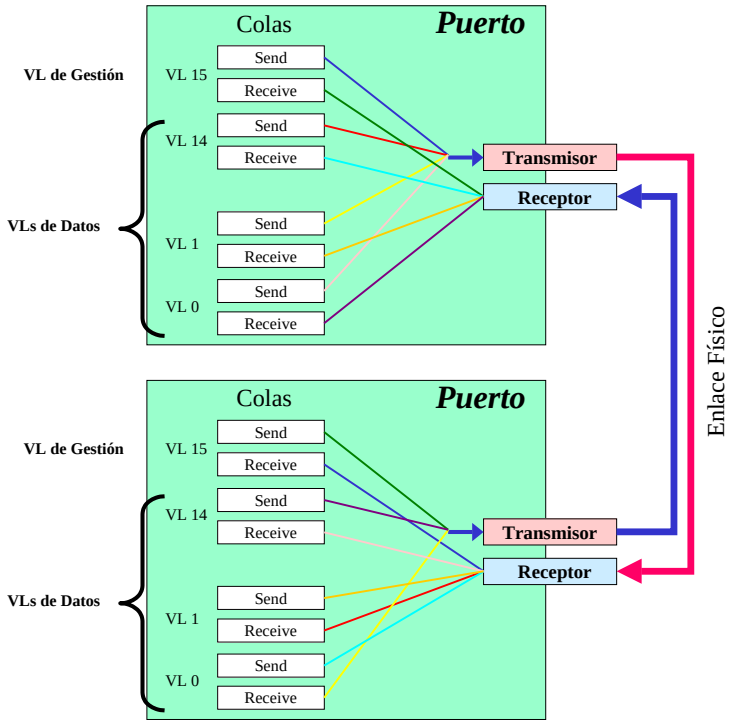


Figura 2.10: Multiplexación del enlace físico en canales virtuales.

Todos los puertos deben soportar un VL_{15} que está reservado en exclusividad para la gestión de la subred. También deben soportar como mínimo un canal virtual para datos, que es el VL_0. El resto de canales virtuales de datos (VL_1 a VL_{14}) son opcionales. Dependiendo de la implementación, un puerto puede tener 16, 8, 4 ó 2 canales virtuales implementados.

El VL que en cada momento se usa viene configurado por el Subnet Manager, y esta decisión está basada en el campo nivel de servicio (Service Level, SL) que llevan los paquetes. En concreto, existe una tabla *SLtoVLMappingTable* que especifica la correspondencia entre los SLs y VLs.

Cada puerto mantiene control de flujo separado para cada VL de datos. De esta forma, el tráfico de cada VL no influye con el tráfico de otros VLs. Cada paquete incorpora en su cabecera un SL. Por otra parte, cada puerto mantiene la tabla de correspondencia entre SLs y VLs. Así, cuando el paquete va fluyendo por la subred es ese SL que tiene en su cabecera el que va indicando el VL a usar en cada caso.

Como el número de canales virtuales que incorpora cada conmutador o encaminador de la red puede ser diferente, cada puerto debe adaptarse al mínimo número entre los que él incorpora y los que tiene el puerto con el que está conectado. Esto se realiza durante la fase de establecimiento de la subred.

2.2.3.8. Control de la tasa de inyección

InfiniBand utiliza enlaces serie punto a punto full-duplex funcionando a una frecuencia de 2,5 GHz. Obviamente, con esta frecuencia la velocidad de transmisión que se obtiene es 2,5 Gb/seg, que se suele denominar 1x. Sin embargo, InfiniBand permite alcanzar mayores velocidades usando varios de esos enlaces en paralelo. De esta forma, otras velocidades que también están definidas en InfiniBand son 10 Gb/seg (4x) y 30 Gb/seg (12x).

De cara a ser capaz de utilizar componentes que funcionen a diferentes velocidades, InfiniBand define un mecanismo de control de inyección que previene que un puerto pueda llegar a saturar a otro más lento. Un ejemplo de esta situación puede observarse en la Figura 2.11.

El control de flujo no soluciona este problema, pues aunque evitará que se pierdan paquetes, provocará congestión hacia atrás en la ruta. De esta forma, se llegarán a congestionar incluso aquellos enlaces que si no fuera por este cuello de botella no se habrían visto afectados. InfiniBand soluciona este problema por medio de un mecanismo de control de inyección que deben incorporar los puertos que funcionen a velocidades mayores de 1x.

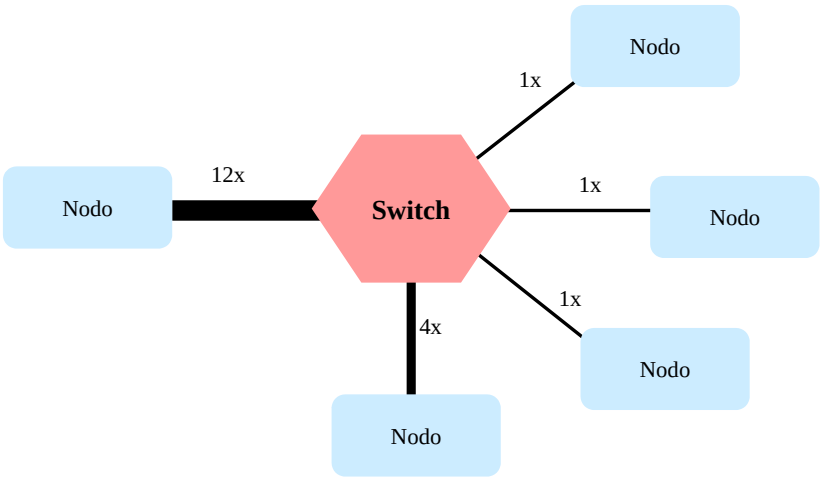


Figura 2.11: Ejemplo de una subred con enlaces de diferentes velocidades.

Cada fuente tiene un tiempo de time-out asociado a cada destino. Ese valor ha sido calculado como el ratio entre las velocidades de la fuente y de los enlaces intermedios hasta llegar al destino. De esta forma, cada vez que el origen envía un paquete, debe esperar a que transcurra ese tiempo antes de enviar otro paquete a la misma fuente.

2.2.3.9. Direccionamiento

Cada nodo puede tener uno o más adaptadores y a su vez éstos pueden tener uno o más puertos. Además, cada adaptador tiene un número variable de pares de colas (QPs). Cada QP tiene asignado un número (queue pair number, QPN) por parte del adaptador, que lo identifica de forma única.

Por otra parte, cada puerto tiene un identificador local (LID) asignado por el Subnet Manager. Dentro de la subred estos LIDs son únicos y se usan para encaminar los paquetes. Cada conmutador de la subred tiene una tabla de encaminamiento que está basada en esos identificadores locales. De esta forma, cada paquete cuando viaja por la subred lleva incorporadas la dirección del puerto fuente (SLID) y la del puerto destino (DLID).

Cada puerto también tiene asignada una dirección global (GID) que sigue el formato de las direcciones IPv6. Estas GIDs son únicas en cualquier subred InfiniBand. Los paquetes que vayan a viajar entre distintas subredes deberán incorporar las direcciones globales fuente (SGID) y destino (DGID), que serán usadas por los encaminadores intermedios.

Así pues, la dirección que identifica a dos entidades que se comunican será la combinación de las direcciones de los puertos (GID + LID) y de los pares de colas que intervienen (QPN).

2.2.3.10. Multicast

InfiniBand permite la comunicación multidestino. No es obligatorio que los fabricantes lo incorporen, pero si lo hacen, sí está especificado cómo debe ser su comportamiento.

Cada grupo multidestino está identificado por una dirección LID y GID única. De esta forma, puede haber grupos multidestino en la misma subred o también entre distintas subredes. Cada nodo se une a un grupo multidestino indicando los LID de los puertos que van a participar. Esta información se distribuye a los conmutadores, que se configuran para saber a qué puertos deben viajar los paquetes de ese grupo multidestino. Evidentemente, se tiene en cuenta que no se formen ciclos en esas rutas multidestino.

Cuando un conmutador o un encaminador recibe un paquete multidestino lo reenvía por todos aquellos puertos a través de los que se accede a puertos que participen en el grupo multidestino, salvo por el puerto por donde llegó dicho paquete. De esta forma, el paquete llegará una sola vez a cada miembro del grupo multidestino.

2.2.3.11. Verbos

InfiniBand describe el comportamiento de la interfaz entre el adaptador del host y el sistema operativo por medio de un conjunto de comportamientos llamados verbos. La intención de los verbos no es definir un API, sino el comportamiento de la interfaz. De esta forma, los diseñadores de los sistemas operativos deberán definir APIs adecuados que aprovechen las ventajas de esta arquitectura.

Los verbos describen los parámetros necesarios para configurar y gestionar el adaptador, crear y liberar colas, configurar operaciones con las colas, hacer peticiones de envío a las colas, consultar el estado de las peticiones, etc.

2.3. Estructura arquitectónica

El funcionamiento de InfiniBand puede describirse mediante la interacción de una serie de niveles. El protocolo que gobierna cada nivel es independiente del resto de niveles. Cada nivel usa los servicios que le proporciona el nivel inferior y a su vez proporciona una serie de servicios al nivel inmediatamente superior.

Los distintos niveles que distingue InfiniBand, y sus modos de actuación, pueden observarse en la Figura 2.12. Se distinguen los siguientes niveles: físico, enlace, red y transporte. Por encima del nivel de transporte ya estarían las aplicaciones o los manejadores de los sistemas operativos.

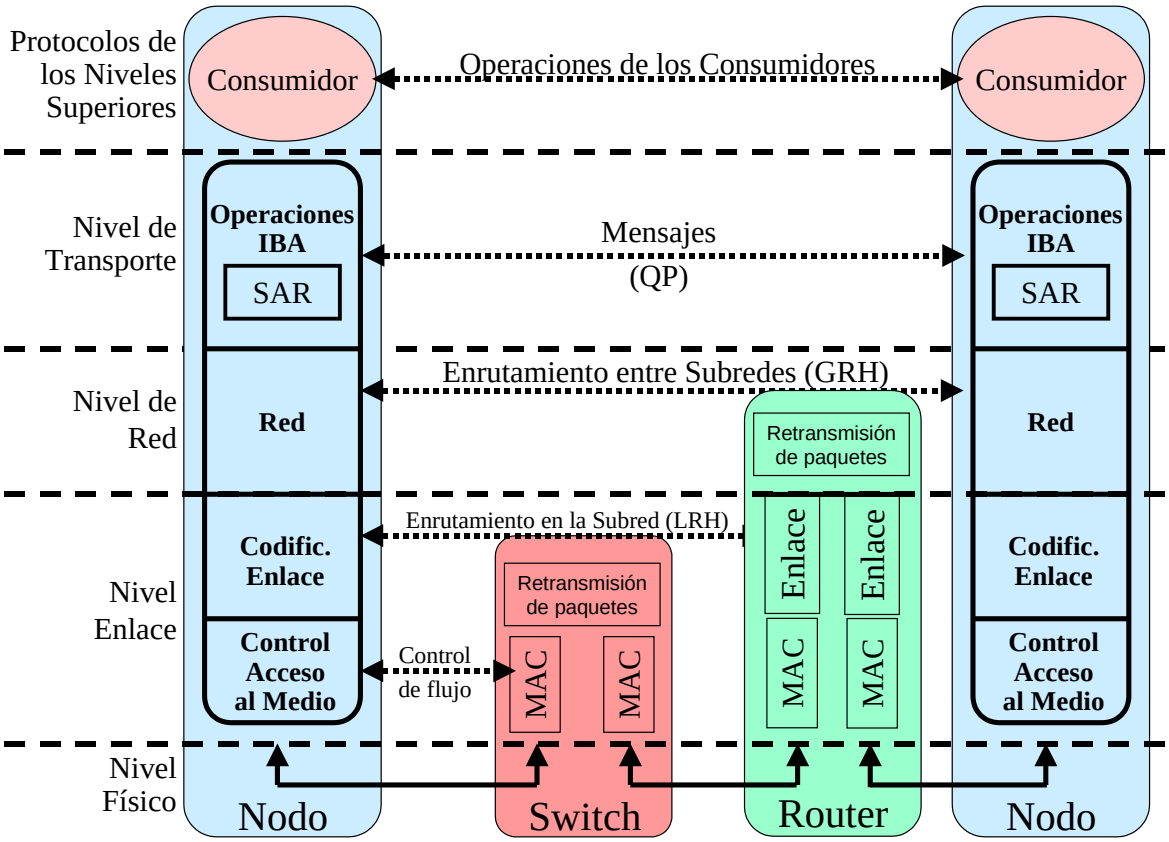


Figura 2.12: Niveles en la arquitectura de InfiniBand.

En esta arquitectura la interacción se produce siempre entre dos niveles contiguos. Sin embargo, puede asumirse como si hubiera una comunicación virtual directa entre los mismos niveles de distintas entidades. Evidentemente, esa comunicación será encapsulada en los paquetes de los niveles inferiores y transmitida por la red. Será cuando llegue al destino cuando será desencapsulada y entregada al nivel correspondiente, tal y como la envió el mismo nivel del origen.

Veamos a continuación de manera detallada cada uno de esos niveles que se pueden distinguir en la arquitectura de InfiniBand, y su interacción, tanto con los niveles inferiores y superiores, como con su mismo nivel en otras entidades.

2.3.1. Nivel físico

El nivel físico especifica como se trasladan los bits al cable. InfiniBand utiliza el sistema de codificación 8B/10B, que es el esquema de codificación usado también por Gigabit Ethernet y por Fiber Channel. También depende de este nivel definir las características de la transmisión y la recepción, de los cables usados y de la ecualización realizada.

El nivel físico también especifica los símbolos que se usarán para indicar principio y final de paquete, datos, espacio entre paquetes, etc. El nivel físico especifica el protocolo de señalización a utilizar, y por tanto qué constituye un paquete correcto.

El nivel físico proporciona al nivel de enlace la funcionalidad de poner un paquete de ese nivel en el nivel de enlace del nodo que está al otro lado del enlace, tanto si éste es un host terminal, un conmutador o un encaminador. Obviamente, el nivel físico variará dependiendo del medio físico que se vaya a utilizar. InfiniBand especifica tres tipos de medios físicos: par trenzado, fibra óptica o circuito en placa.

Esquemáticamente, la estructura del nivel físico sería como la mostrada en la Figura 2.13. En dicha figura se puede observar que hay ciertas partes que variarán dependiendo del medio físico concreto que se utilice. También puede verse la interacción de este nivel con el nivel de enlace.

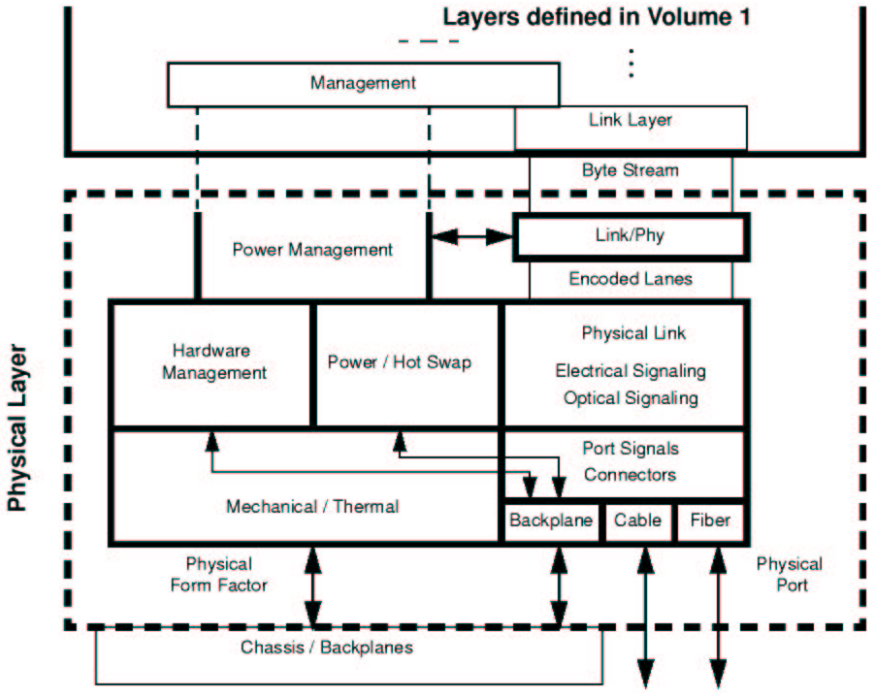


Figura 2.13: Estructura del nivel físico de InfiniBand.

El nivel físico también define la estructura y funcionamiento que deben tener los conectores. Algunos de los conectores definidos en InfiniBand son los mostrados en la Figura 2.14. De esta forma, se asegura la compatibilidad del nivel físico que puedan implementar los distintos fabricantes.

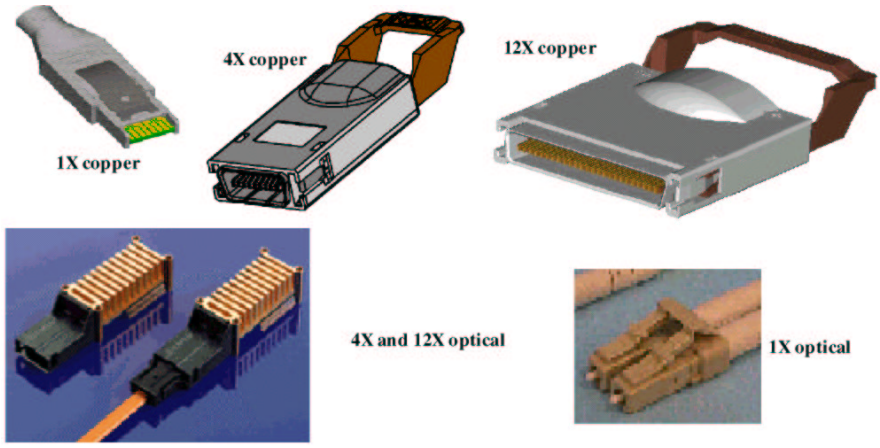


Figura 2.14: Algunos de los conectores definidos en InfiniBand.

2.3.2. Nivel de enlace

El nivel de enlace describe los formatos de paquete a usar y los protocolos para las operaciones con ellos. También son tareas de este nivel el control de flujo y el encaminamiento de los paquetes dentro de la misma subred.

Se pueden distinguir dos tipos de paquetes:

Paquetes de gestión. Se utilizan para mantenimiento de las operaciones de enlace. Estos paquetes se crean y se consumen en el nivel de enlace, y no están sujetos a control de flujo. Se suelen usar para operaciones de negociación de parámetros entre los puertos de cada lado del enlace, como velocidad, tasa de inyección, etc. También son estos paquetes los que se utilizan para control de flujo y mantenimiento de la integridad del enlace. Estos paquetes sólo se envían entre ambos extremos de un enlace, y nunca se retransmiten hacia otro destino.

Paquetes de datos. Estos son los paquetes que intercambian las operaciones de InfiniBand y pueden contener varias cabeceras, algunas de las cuales son opcionales.

El formato del paquete del nivel de enlace es el mostrado en la Figura 2.15. En dicha figura puede observarse como se encapsulan los paquetes de los niveles superiores en los paquetes que envían los niveles inferiores de la arquitectura.

La **Local Route Header** (LRH) siempre está presente e identifica los puertos origen y destino donde los conmutadores intermedios encaminarán el paquete. Esta cabecera también contiene el nivel de servicio y el canal virtual que debe utilizar el paquete. Este canal virtual podría cambiar a lo largo de la ruta, pero el resto de campos de esta cabecera permanecerán inalterados durante todo el trayecto.

El paquete también contiene dos CRCs: el **Invariant CRC** (ICRC) y el **Variant CRC** (VCRC). El ICRC cubre aquellos campos del paquete que no van a variar a

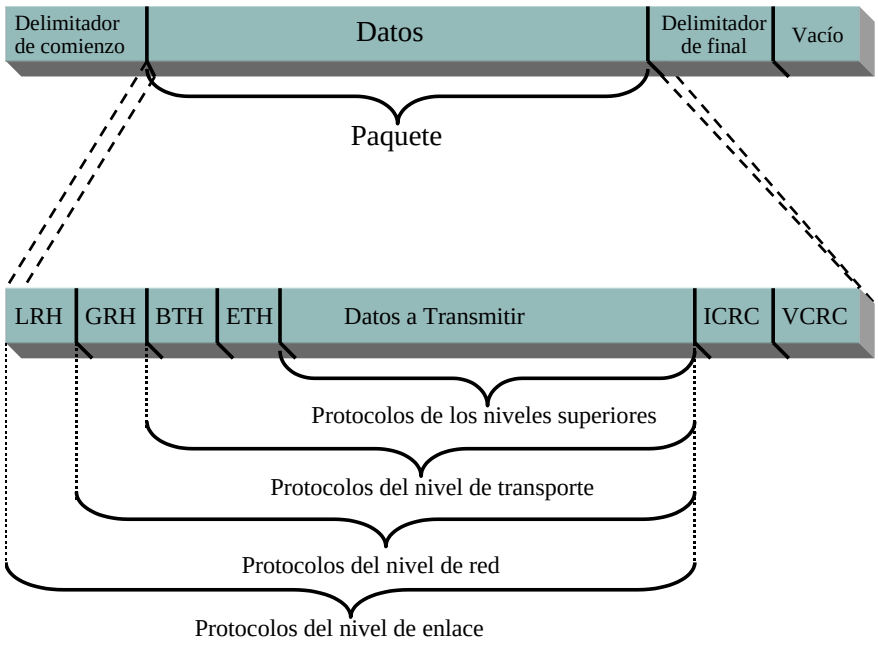


Figura 2.15: Formato del paquete de datos en InfiniBand.

lo largo de la ruta. Sin embargo, el VCRC cubre todo el paquete, con lo que deberá ser recalculado en cada conmutador o encaminador intermedio. La combinación de ambos CRCs permite a los nodos intermedios modificar apropiadamente las cabeceras, controlando entre ellos los posibles errores locales, y sin embargo seguir manteniendo el control de la integridad del paquete extremo a extremo.

El control de flujo que realiza la capa de enlace está basado en créditos. En cada enlace, el extremo receptor envía al origen los créditos disponibles. Estos créditos se contabilizan para cada canal virtual, e indican el número de paquetes de datos que el receptor puede aceptar en un canal virtual. El extremo origen no enviará paquetes a menos que el receptor le haya indicado previamente que tiene espacio disponible para alojarlos. Por otra parte, el canal virtual número 15 (el canal virtual de gestión) no está sujeto a control de flujo.

El formato de la cabecera LRH es el mostrado en la Figura 2.16. Como puede comprobarse, esta cabecera tiene un tamaño de 8 bytes. Los campos que incluye son los siguientes:

- VL (4 bits): Identifica el canal virtual que el paquete está usando.
- LVer (4 bits): Este campo identifica el protocolo de nivel de enlace al que pertenece este paquete.
- SL (4 bits): Este campo indica el nivel de servicio que el paquete lleva marcado, y que será usado para decidir el canal virtual a usar durante su viaje por la subred.

- 2 bits reservados para uso futuro.
- LNH (2 bits): Identifica las cabeceras que siguen al LRH.
- DLID (16 bits): Este campo identifica el puerto destino en la subred.
- 5 bits reservados para uso futuro.
- Longitud del paquete (11 bits): Este campo identifica el tamaño del paquete medido en número de palabras de cuatro bytes que contiene. Incluye desde el primer byte de la LRH hasta el último byte antes del CRC variante, pero no incluye a éste.
- DLID (16 bits): Este campo identifica el puerto destino de la subred local.
- SLID (16 bits): Este campo identifica el puerto origen de la subred local.

bits / bytes	31–24		23–16			15–8	7–0
0–3	VL	LVer	SL	2 Rsv	LNH	Identificador Local del Destino (DLID)	
4–7	5 Reservados		Longitud del Paquete (11 bits)			Identificador Local del Emisor (SLID)	

Figura 2.16: Formato de la cabecera local (LRH) de los paquetes en InfiniBand.

2.3.3. Nivel de red

El nivel de red describe el protocolo para encaminar un paquete entre distintas subredes. Este nivel define una **Global Route Header** (GRH) que debe estar presente en los paquetes que tengan que ser encaminados entre dos o más subredes. El formato de la GRH es el mostrado en la Figura 2.17. La GRH identifica los puertos origen y destino usando direcciones globales (GID) en el formato de una dirección IPv6.

Los encaminadores se basan en la información que contiene la GRH para encaminar los paquetes hacia su destino. Conforme los paquetes van avanzando entre subredes los encaminadores van modificando el contenido del GRH y reemplazando el LRH. Sin embargo, los identificadores globales de los nodos fuente y destino no variarán a lo largo de toda la ruta, y están protegidos por el campo ICRC.

El nodo origen pone el identificador global (GID) del destino en la GRH pero el identificador local (LID) del encaminador en la LRH. Cada conmutador intermedio hace avanzar el paquete hacia la siguiente subred hasta que llega a la subred destino. El último encaminador reemplaza la LRH usando el identificador local del destino.

El formato de la cabecera GRH contiene los siguientes campos:

bits / bytes	31–24	23–16	15–8	7–0
0–3	IPVer	TClass	FlowLabel	
4–7	PayLen		NxtHdr	HopLmt
8–11	SGID [127–96]			
12–15	SGID [95–64]			
16–19	SGID [63–32]			
20–23	SGID [31–0]			
24–27	DGID [127–96]			
28–31	DGID [95–64]			
32–35	DGID [63–32]			
36–39	DGID [31–0]			

Figura 2.17: Formato de la cabecera global (GRH) de los paquetes en InfiniBand.

- IPVer (4 bits): Indica la versión que se está usando en cuanto al formato de la cabecera GRH. Para la versión actual este campo estará siempre fijo con el valor 6 (0110).
- TClass (8 bits): Este campo se usa para comunicar el nivel de servicio extremo a extremo.
- FlowLabel (20 bits): Este campo se usa para identificar una secuencia de paquetes que debe entregarse en el destino en orden.
- PayLen (16 bits): Especifica la longitud del paquete en bytes. Incluye desde el primer byte después de la GRH hasta el último byte anterior al VCRC. De esta forma, ni la GRH ni el VCRC están incluidos.
- NxtHdr (8 bits): Este campo indica qué cabecera (si es que hay alguna) viene justo después de la GRH.
- HopLmt (8 bits): Este campo especifica el número de saltos (por ejemplo el número de encaminadores por los que se pasa) que el paquete puede dar antes de ser descartado. De esta forma se asegura que, aunque hubiera bucles entre encaminadores, un paquete no va a estar circulando por la red de forma indefinida buscando el destino. Se puede fijar este campo a 0 para asegurarse que el paquete no va a ser transmitido fuera de la subred.
- SGID (64 bits): Identifica el puerto que envía el paquete. Esta dirección es única en cualquier subred de InfiniBand.
- DGID (64 bits): Identifica el puerto destino del paquete. Esta dirección también es única en cualquier subred de InfiniBand.

Como puede apreciarse, existen muchas similitudes entre este formato y el utilizado en las cabeceras de IPv6.

2.3.4. Nivel de transporte

Los protocolos de enlace y de red llevan un paquete al destino deseado. La parte de nivel de transporte del paquete se encarga de hacer que se entregue el paquete en la cola (QP) apropiada, indicándole además cómo procesar los datos contenidos en el paquete.

El nivel de transporte es el responsable de segmentar una operación en varios paquetes si los datos a transmitir exceden el tamaño máximo de paquete (MTU). El QP en el extremo final reensambla los paquetes para formar la secuencia de datos que se quiso enviar.

La **Base Transport Header** (BTH) está en todos los paquetes, salvo en los datagramas RAW. El formato de la BTH es el mostrado en la Figura 2.18. Esta cabecera especifica el QP destino e indica el código de operación, el número de secuencia del paquete y la partición a la que pertenece.

bits / bytes	31–24	23–16	15–8	7–0
0–3	OpCode	SE | M | Pad | TVer	Partition Key	
4–7	8 Reservados	QP Destino		
8–11	A | 7 Reservados	PSN – Nº Secuencia del Paquete		

Figura 2.18: Formato de la cabecera básica de transporte (BTH) de los paquetes en InfiniBand.

El formato de la cabecera BTH incluye los siguientes campos:

- OpCode (8 bits): Define la interpretación que debe darse al resto de la cabecera y a la información que contiene el paquete.
- SE (Solicited Event, 1 bit): El emisor activa este bit si considera que el receptor cuando reciba este paquete debe llamar al manejador de cola completa.
- M (MigReq, 1 bit): Usado para comunicar estados de migración debido al *Automatic Path Migration*.
- Pad (2 bits): Se usa para forzar que el tamaño total de paquete sea un múltiplo de 4 bytes. De esta forma, este campo indica el número de bytes (entre 0 y 3) que se han añadido al final del campo de datos para forzar a un múltiplo de 4. Hay que señalar que la longitud total del paquete se indica mediante el número de bloques de 4 bytes que tiene.

- TVer (4 bits): Indica la versión del protocolo de transporte de InfiniBand usado por el paquete. Si un nodo recibe un paquete de transporte cuyo protocolo no soporta deberá descartarlo. Para la versión actual se utiliza el valor 0.
- Partition Key (16 bits): Identifica la partición de la que es miembro el QP destino.
- 8 bits reservados para uso futuro.
- A (AckReq, 1 bit): Con este campo el emisor solicita que el receptor confirme la llegada correcta del paquete.
- 7 bits reservados para uso futuro.
- PSN (Packet Sequence Number, 24 bits): Este campo se utiliza para identificar la posición que ocupa este paquete dentro de la secuencia global. Dependiendo del tipo de servicio utilizado el destino utilizará este campo para detectar paquetes perdidos en la secuencia.

El número de secuencia del paquete se inicializa durante el proceso de establecimiento de conexión y se incrementa cada vez que el QP crea un nuevo paquete. El QP receptor utiliza este número de secuencia para detectar si se ha perdido algún paquete. En el caso de los servicios fiables, el receptor enviará un paquete de asentimiento (ACK o NACK) notificando al remitente si el paquete se ha recibido correctamente.

Cuando ocurre un error en la secuencia es que se ha perdido algún paquete intermedio. En ese caso, el receptor descartará los paquetes siguientes que pudieran llegar hasta que llegue el paquete perdido. En el caso del servicio sin asentimiento el receptor detecta un paquete perdido, aborta la operación y descarta todos los paquetes siguientes hasta que comience otra operación nueva.

Dependiendo de la clase de servicio utilizada y de la operación a realizar, un paquete podría llevar varias cabeceras extra (*Extended Transport Headers*, ETH). Será el QP destino el que deberá analizar las ETHs presentes y realizar las acciones oportunas según la operación a realizar.

El nivel de transporte es el núcleo central de la arquitectura de InfiniBand. Los servicios del nivel de transporte se acceden desde los QPs, donde cada QP se configura para una clase de servicio específico.

2.3.5. Protocolos de los niveles superiores

InfiniBand soporta cualquier número de protocolos de nivel superior que podrán usar directamente los usuario. Estos usuarios son los consumidores finales de los mensajes transmitidos por la red. Esta situación se muestra gráficamente en la Figura 2.19.

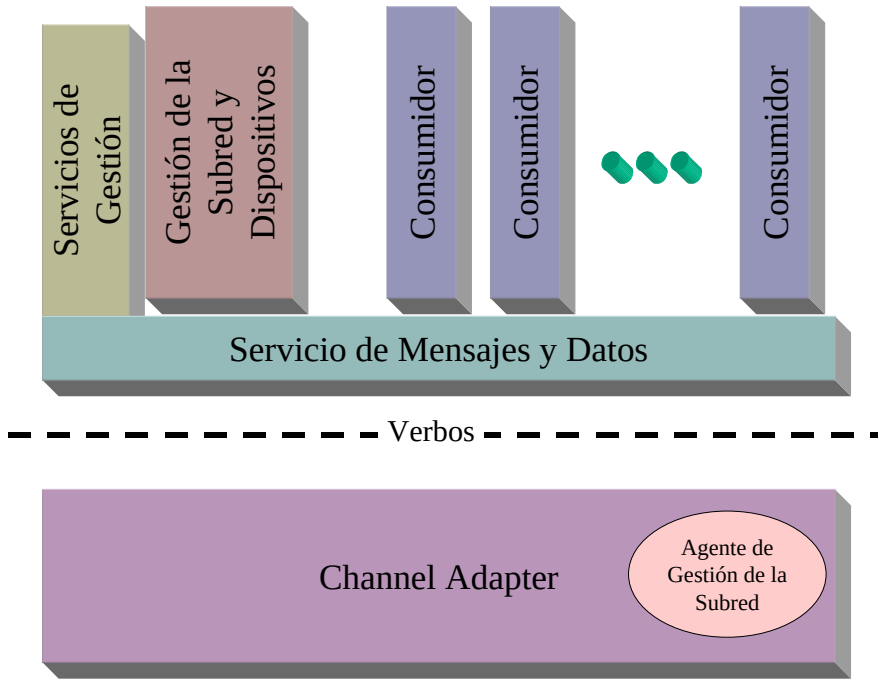


Figura 2.19: Niveles superiores en InfiniBand.

InfiniBand proporciona la plataforma para que las aplicaciones puedan montar encima sus protocolos. De esta forma, InfiniBand no es más que el mecanismo de transporte de los mensajes a intercambiar entre las aplicaciones. Para ellos InfiniBand proporciona unos tipos de servicios y una operaciones relativas a cada uno de ellos. Para qué utilicen las aplicaciones esos tipos de servicio y esas operaciones ya queda fuera de la cobertura de InfiniBand, que se limita a proporcionar la plataforma de comunicación.

2.4. Infraestructura para la gestión de InfiniBand

InfiniBand también define los mensajes y protocolos para muchas de las tareas de gestión de la red. Estos protocolos de gestión se pueden agrupar en gestiones de la subred y servicios de la subred, teniendo cada uno de esos grupos propiedades distintas.

En cada subred debe existir un **Subnet Manager** encargado de configurar la red y administrar la información necesaria para las operaciones de la subred. InfiniBand define la infraestructura básica para permitir diversas clases de gestión (Management Classes). Cada clase de gestión permite gestionar diversos aspectos de la red de forma independiente.

Para permitir la consistencia entre distintas clases, la infraestructura global de gestión define un conjunto común de acciones de gestión llamados métodos que pueden usar todas las clases. Cada clase puede definir los atributos a los que se les podrán

aplicar esos métodos. De hecho, una clase define qué atributos son apropiados para cada método.

Veamos en primer lugar las clases de gestión que se definen y sus características principales.

2.4.1. Management Classes

La arquitectura de InfiniBand define un número pequeño de clases de gestión y permite a los implementadores añadir otras nuevas. Las clases definidas en las especificaciones son:

Subnet Management. Especifica los procedimientos que permiten al Subnet Manager descubrir, configurar y gestionar la subred. Esto incluye configurar las tablas de encaminamiento de los conmutadores y fijar los parámetros operacionales de los conmutadores y adaptadores (LID, prefijo de GID, M_Keys, P_Keys, mapeo de SL a VL, etc.).

Subnet Administration (SubnAdm). Permiten a los nodos finales obtener información que necesitan para operar en la subred. Esta clase también es la responsable de permitir a los Subnet Managers redundantes conseguir la información necesaria por si el Subnet Manager que está funcionando dejara de hacerlo y tuviera que entrar a funcionar uno de ellos. El nodo que realiza las funciones de Subnet Manager habilita un Subnet Administration Agent (SA) que se encarga de habilitar la clase Subnet Administration.

Performance Management. Especifica los procedimientos que permiten a una aplicación de gestión obtener estadísticas de prestaciones e información de los nodos sobre los errores ocurridos.

Baseboard Management (BM). Permite hacer llegar mensajes a los elementos de gestión dentro del propio nodo. Estos elementos de gestión pueden estar distribuidos por la placa o la carcasa. Como ejemplo se puede citar el control de la temperatura de la placa. Todos los nodos deben tener un agente de gestión en la placa.

Device Management (DevMgt). Define los procedimientos para obtener los atributos de las unidades o controladores de E/S. Un ejemplo es el determinar la clase de controlador de E/S y los parámetros del protocolo a usar. No es obligatorio que los productos InfiniBand incorporen un agente para gestionar esta clase, siendo este tipo de agente opcional.

SNMP Tunneling. Especifica los procedimientos y formato de datos para proporcionar un túnel SNMP (Simple Network Management Protocol) encapsulando los

paquetes SNMP en los paquetes de gestión de InfiniBand. Es opcional que los productos InfiniBand incorporen un agente SNMP.

Communication Management (CommMgt). Define los procedimientos y atributos para establecer y terminar conexiones. También es tarea de esta clase fijar caminos alternativos cuando se requieran. Cada adaptador debe tener un agente para esta clase llamado Communication Manager (CM).

Configuration Management (CFM). Define los mecanismos y mensajes para que los nodos obtengan información sobre la E/S. Esa información incluye qué controladores de E/S tiene asignados, qué otros hosts comparten esos controladores, y qué dispositivo usar para arrancar. Esta clase de servicio es opcional y, aunque puede ser realizado por cualquier nodo, suele asociarse al nodo que actúa de Subnet Manager.

Como se ha visto, algunas de estas clases son obligatorias y otras opcionales, y depende de los implementadores el que aparezcan o no en los productos finales. También se ha comentado previamente que algunas de estas clases necesitan un agente especial para que proporcione ese servicio, mientras que otras las proporciona directamente el Subnet Manager.

2.4.2. Management Elements

La infraestructura de gestión de la red se basa en tres tipos de elementos de gestión: agentes de servicio, gestores de clase (*class managers*) y clientes y aplicaciones de gestión. La interacción entre ellos es la mostrada en la Figura 2.20.

Un agente de servicio es una entidad que reside en los nodos gestionados. Tiene una interfaz conocido para recibir y responder a las peticiones relacionadas con la gestión. La arquitectura de InfiniBand permite cualquier número de agentes de gestión en cada nodo. Lo usual es que en cada puerto haya un agente por cada clase de gestión. InfiniBand especifica cómo descubrir y configurar esos agentes de gestión.

Por otra parte, hay un gestor de clase en aquellos casos en que se necesite tener una entidad central de gestión que interactúe con los agentes de gestión. Estos gestores de clase son útiles por ejemplo para realizar una recolección eficiente de información de gestión distribuida entre los nodos, o para asegurar la atomicidad en el acceso a ciertos agentes de servicio.

Por último, los clientes y aplicaciones de gestión suelen estar ubicados normalmente en hosts. Estos clientes y aplicaciones de gestión acceden a los agentes de servicio y a los gestores de clase para obtener información relativa a la gestión y solicitar funciones de gestión.

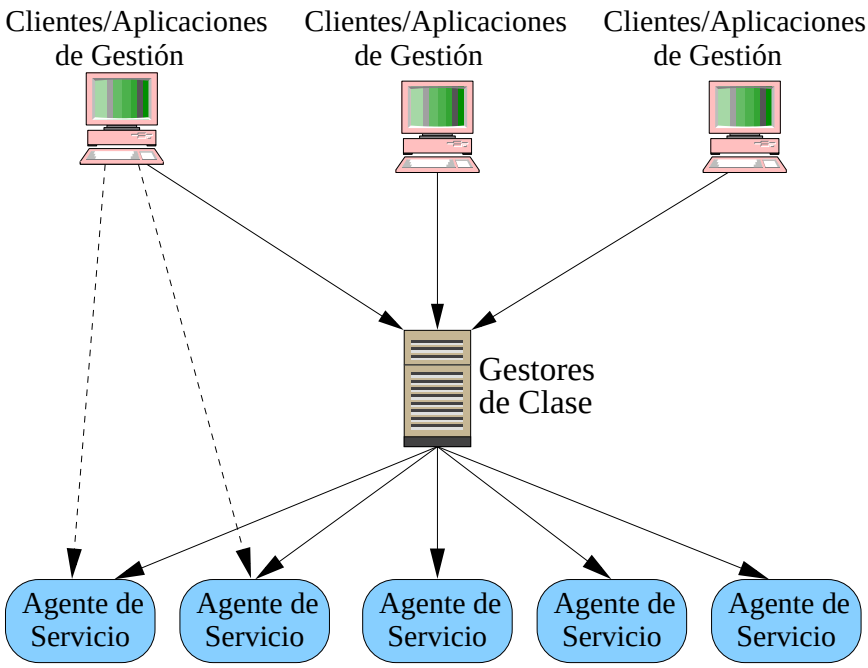


Figura 2.20: Elementos de gestión en InfiniBand.

La arquitectura de InfiniBand no especifica los requisitos de las aplicaciones y clientes de gestión, dejándolo a criterio de los implementadores. Algunos ejemplos de estas aplicaciones podrían ser:

- Un gestor de recursos de E/S en un host que quiere comunicar con una unidad de E/S y se pone en contacto con el agente del dispositivo.
- Un gestor de comunicaciones en un host que comunica con el agente de subred para obtener información relativa al camino a seguir para establecer una conexión con otro host. Una vez que tiene el camino a seguir, el gestor de conexiones local se comunica con el agente de clase de comunicaciones en el nodo final para establecer la conexión.
- Un gestor de prestaciones que comunica con los agentes de gestión de prestaciones ubicados en varios nodos.

Como se puede ver, estos clientes o aplicaciones de gestión podrán variar entre implementaciones, quedando a criterio de los fabricantes cuáles implementar según sus necesidades.

2.4.3. Mensajes y métodos para la gestión

La infraestructura para la gestión define la estructura y formato de los paquetes de gestión (*Management Datagram*, MAD). Estos paquetes de gestión son los que

intercambian los elementos de gestión. Estos mensajes se envían usando la clase de servicio no fiable y sin conexión (UD). El formato de los mensajes MAD es el mostrado en la Figura 2.21.

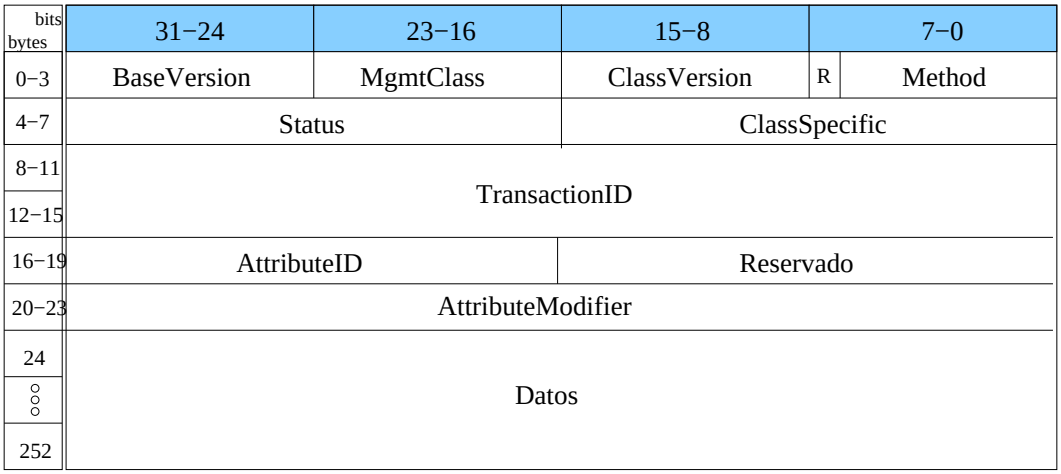


Figura 2.21: Formato básico de los paquetes de control.

Todos los paquetes MAD deben tener exactamente 256 bytes, y consisten en una cabecera MAD y los datos correspondientes. El contenido del campo de datos variará en función de la clase concreta para la que se esté usando. En general, el formato de un paquete MAD incluye los siguientes campos:

- BaseVersion (8 bits): Indica la versión del formato de MAD que se está usando. En la versión actual este campo es 1.
- MgmtClass (8 bits): Este campo especifica la clase para la que se está usando.
- ClassVersion (8 bits): Indica la versión del formato MAD que se está usando para los campos relativos a la clase usada.
- R (1 bit): Indica que el emisor del mensaje requiere de una repuesta del destinatario.
- Method (7 bits): Método a utilizar de los disponibles en la clase usada.
- Status (16 bits): Código que indica el estado de la operación.
- ClassSpecific (16 bits): Este campo está reservado, excepto para la clase Subnet Management.
- TransactionID (64 bits): Identificador de la transacción a realizar.
- AttributeID (16 bits): Define los objetos en los que actúa la clase.
- 16 bits reservados para uso futuro.

- AttributeModifier (32 bits): Especifica otras características adicionales de los atributos definidos.
- Datos (1856 bits): Los datos a enviar en el paquete, y cuyo contenido variará en función de la clase para la que se esté enviando el paquete.

Las especificaciones también definen los métodos que usarán los elementos de gestión para intercambiar información. Estos procedimientos incluyen:

- Gets y Sets: Leen y modifican los atributos de los objetos gestionados.
- Traps: Permiten obtener información de ciertos eventos de forma asíncrona por parte de un agente de servicio.
- Subscription: Es la forma en que las aplicaciones piden recibir traps.
- Reports: Es la forma en que el gestor de una clase puede distribuir traps y otras informaciones a las aplicaciones que lo hubieran solicitado.

2.4.4. Subnet Manager

El *Subnet Manager* (SM) se encarga de configurar y hacer el mantenimiento de la subred. El SM utiliza un formato especial de paquete llamado *Subnet Management Packet* (SMP) que tiene prioridad sobre el resto de paquetes. Un paquete SMP es un tipo especial de paquete MAD, con lo que la cabecera es la misma que la vista en la Sección 2.4.3, y especifica el contenido de la parte de datos.

Algunas de las funciones del SM son descubrir la topología, los nodos conectados a los conmutadores, configurar los conmutadores y los hosts con sus parámetros (LIDs, GIDs, P_Keys, etc.), calcular y distribuir a los conmutadores las tablas de encaminamiento en la subred, recibir peticiones de los agentes locales de gestión, etc.

Cada nodo contiene un Agente de Gestión de la Subred (*Subnet Management Agent*, SMA) que responde a los paquetes de gestión enviados por el SM. Cada puerto tiene un cola llamada QP0 dedicada a enviar y recibir paquetes de control. La interfaz que utiliza QP0 es el *Subnet Management Interface* (SMI). Esta estructura es la mostrada en la Figura 2.22.

Los paquetes de control sólo se envían desde y hacia un SMI, y no se pueden enviar otro tipo de paquetes a un SMI. Los paquetes de control nunca saldrán de la subred (los encaminadores no los reenvían) y siempre utilizan un canal virtual (el VL_{15}) que está totalmente dedicado a este tipo de tráfico y siempre tiene prioridad sobre el resto de canales virtuales. Este canal virtual del tráfico de control no está sujeto a control de flujo.

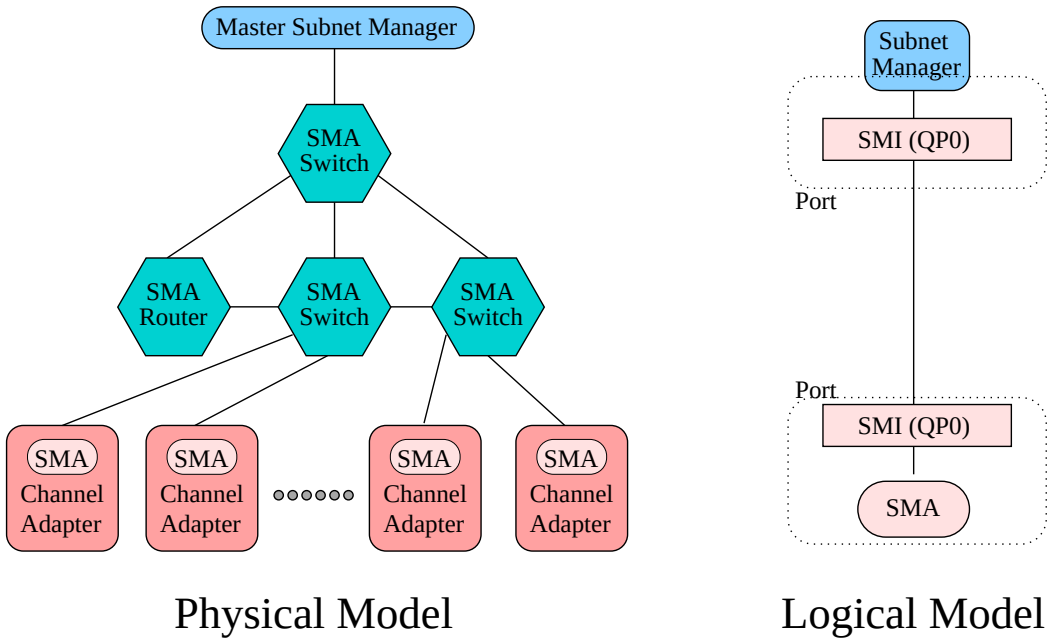


Figura 2.22: Modelo de gestión de la subred en InfiniBand.

Pueden existir varios SMs, pero sólo uno de ellos (llamado el *master*) puede estar activo en un momento dado. Si un SM descubre otro SM utiliza un protocolo específico para determinar cuál de ellos actuará como master, quedando el resto a la espera por si el master dejara de funcionar, pasar a actuar como tal. Solamente el master Subnet Manager es el que configura los nodos y conmutadores.

El master Subnet Manager se asegura el control exclusivo sobre los SMA al establecer una clave de control (M_Key) en cada puerto. Posteriormente, los paquetes de control que intercambie con ese puerto contendrán la clave que se les suministró. Sólo serán válidos los paquetes de control que contengan la clave correcta, ignorándose el resto de paquetes de control.

2.4.5. Subnet Administration

Sólo puede haber un *Subnet Administrator* (SA) en cada subred, y se supone que debe estar situado en el mismo nodo que el master subnet manager. De hecho, los nodos localizan el SA mediante el envío de paquetes de la clase SubnAdm al nodo SM. Si el SA no estuviera en el mismo nodo que el SM, éste reenviaría ese paquete al nodo donde el SA residiera.

El agente de administración de la subred actúa como gestor de clase para la gestión de la subred. Este SA proporciona a los clientes y aplicaciones de control los medios para obtener y enviar información de control. Este SA es el encargado de interactuar

con el master subnet manager. El SA no utiliza los típicos mensajes de control, sino que utiliza un formato genérico de paquete.

La clase de administración de la subred (SubnAdm) define los métodos para interactuar con el SA y obtener las características de la subred. Por ejemplo, el SA permite que un nodo encuentre otros nodos, averigüe el camino hasta ellos y determine las características de ese camino, para así poder establecer una conexión con ellos.

Otro servicio que proporciona el SA a los nodos es la posibilidad de que éstos le soliciten recibir notificación sobre los traps. Cuando el SM recibe un trap, el SA lo reenvía a todos los nodos que tenga registrados. Así pues, el SA es una parte importante de la infraestructura de gestión que proporciona InfiniBand, y debe implementarse en todos los nodos.

2.4.6. Servicios generales

Las especificaciones de InfiniBand describen una infraestructura de servicio general para las tareas de control llamadas no críticas. Como tareas no críticas se entiende todo lo necesario para el control de la subred excepto la configuración y su mantenimiento posterior. Todas las clases de servicios (excepto la SM) usan el modelo de servicio general mostrado en la Figura 2.23.

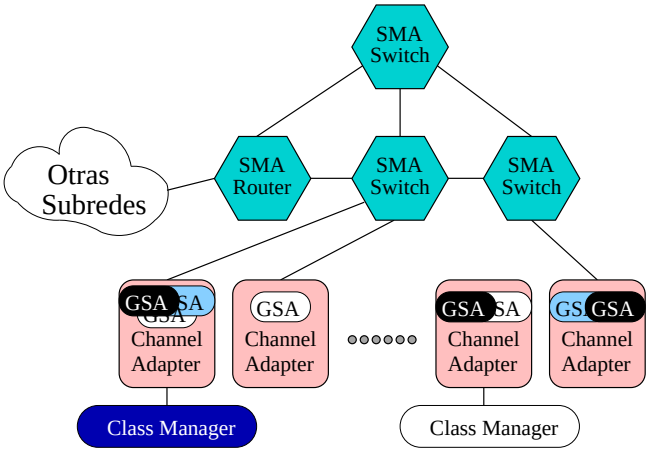


Figura 2.23: Modelo de servicios generales para la gestión en InfiniBand.

El modelo de servicios generales se basa en intercambiar información entre:

- Un gestor de clase y clientes/aplicaciones de control.
- Un gestor de clase y un agente de servicio general (*Global Service Agent*, GSA).
- Directamente entre un cliente/aplicación de control y un GSA.

Lo normal es que para cada clase haya varios clientes o aplicaciones de control, un gestor de la clase, y muchos GSAs (al menos uno por puerto). El GSA es la entidad de un puerto que habilita esa clase de gestión para ese puerto. Un GSA responde a peticiones relativas a su clase del gestor de la clase o de aplicaciones de control. Un ejemplo podría ser el agente de gestión de dispositivo que tienen todos los nodos de E/S para responder al gestor de recursos de E/S de cada host.

Las especificaciones de InfiniBand definen una interfaz para los servicios generales llamado *General Service Interface* (GSI), que viene representado por un QP reservado (el QP1). Cualquier petición, sea ésta de la clase que sea, se envía al QP1 del destino, y es el GSI quien se lo hace llegar al GSA correspondiente. El GSI permite redirigir la petición a otro QP y/o otro puerto.

Las aplicaciones de gestión, los gestores de clase y los GSAs pueden estar ubicados en cualquier nodo. La Figura 2.24 ilustra varios tipos de relaciones entre los elementos de los servicios generales.

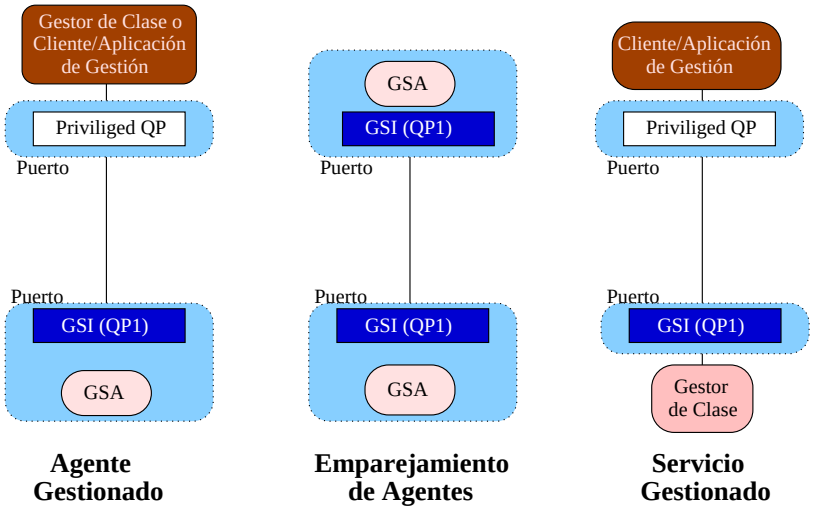


Figura 2.24: Modelo lógico de servicios generales en InfiniBand.

Como ejemplo de agente gestionado se podría citar un gestor de recursos de E/S en un host que accede al agente de gestión de dispositivo en una unidad de E/S. Como ejemplo de emparejamiento de agentes se podría citar el intercambio entre gestores de comunicación cuando se establece una conexión entre dos adaptadores. Por último, un ejemplo de un servicio gestionado podría ser el SA como un gestor de la clase de administración de la subred, que responde a los hosts sobre información de gestión.

2.4.7. Communication Management

La gestión de comunicaciones especifica los protocolos y mecanismos para establecer, mantener y liberar conexiones. También se encarga de la resolución de identificadores, que permite a los usuarios del servicio no fiable y sin conexión identificar el QP que soporta un servicio en particular.

En cada adaptador hay un *Communication Manager* (CM) accesible por cualquiera de sus puertos por medio del GSI de dicho puerto. Un CM interactúa con otros CMs por medio de los paquetes de la clase CommMgt. Las especificaciones de InfiniBand definen el comportamiento que deben tener los CM que interactúan. Sin embargo, la interfaz entre el QP cliente y el CM dependerá del sistema operativo y queda fuera de las especificaciones de InfiniBand.

Para los tipos de servicio con conexión (tanto fiable como no fiable), los CMs de ambos extremos configuran un QP en exclusividad en cada extremo para esa conexión. Los CMs usan la fase de establecimiento de la conexión para intercambiar información relativa a los requisitos de dicha conexión.

Por el contrario, para el tipo de servicio fiable pero sin conexión, un par de contextos extremo a extremo (EEC) definen un canal fiable que permite a múltiples QPs comunicar entre sí. Los CMs usan el protocolo de establecimiento del canal para establecer y gestionar el canal entre los EECs.

2.5. Calidad de servicio en InfiniBand

InfiniBand proporciona varios mecanismos que permiten al administrador de la subred gestionar distintas garantías de calidad de servicio, tanto para servicios con conexión como sin conexión. En los servicios con conexión se puede intentar garantizar calidad de servicio, como se indicará en los siguientes capítulos. Sin embargo, en los servicios sin conexión no es posible dar garantías, pues no se conoce a priori los requisitos necesitados por las aplicaciones. Para este tipo de servicio se puede configurar la subred con algún tipo de mecanismo similar a lo que hacen los servicios diferenciados.

Los mecanismos que proporciona InfiniBand para conseguir proporcionar calidad de servicio son básicamente los niveles de servicio, los canales virtuales, el arbitraje de dichos canales virtuales y las particiones. Veamos cada uno de ellos con un poco más de detalle.

2.5.1. Niveles de servicio

InfiniBand define el atributo nivel de servicio que deben incorporar todos los paquetes. Se permiten 16 niveles de servicio distintos, aunque el propósito de cada uno de ellos no está especificado. De esta forma, dependerá de las implementaciones o del administrador de la red cómo distribuir los tipos de tráficos existentes entre los niveles de servicio disponibles.

Los niveles de servicio permiten segregar los distintos tipos de tráfico existentes de forma que se pueda señalar cada tipo de tráfico distinto, para luego ser capaz de proporcionar a cada tipo de tráfico un trato distinto, en función de sus necesidades.

2.5.2. Correspondencia SL a VL

Tal y como se indicó en la Sección 2.2.3.7 los puertos en InfiniBand tienen canales virtuales. Un canal virtual representa un conjunto de buffers para transmisión y recepción en un puerto. Los puertos de InfiniBand deben soportar un mínimo de dos y un máximo de 16 canales virtuales (VL_0 ... VL_{15}). Todos los puertos deben soportar el VL_{15} que se reserva en exclusividad para el tráfico de control. Este canal VL_{15} siempre tiene prioridad sobre el resto de canales virtuales.

Tal y como puede apreciarse en la Figura 2.25, cada puerto puede tener un número de canales virtuales diferentes. Durante la fase de configuración ambos extremos del enlace deben ponrse de acuerdo para funcionar con el número mínimo de enlaces de los dos.

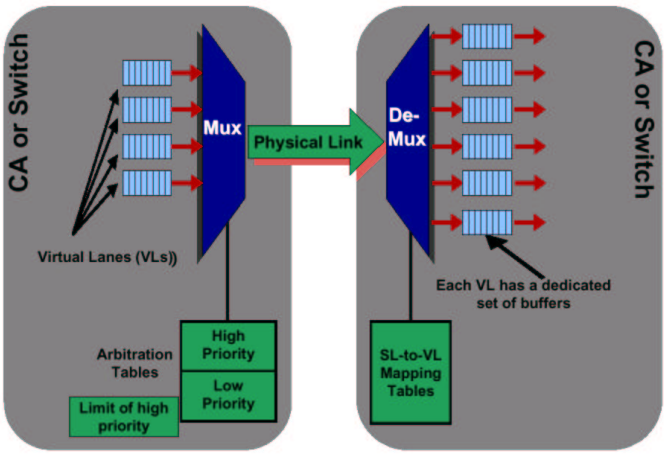


Figura 2.25: Funcionamiento de los canales virtuales en un canal físico.

Por otra parte, cada paquete contiene un SL y cuando viaja por la red es ese SL el que determina qué VL se usará en el siguiente enlace. De esta forma, es posible que

puedan funcionar en la misma subred dispositivos que implementan distinto número de canales virtuales.

Cada puerto (en conmutadores, encaminadores y nodos terminales) tiene una tabla de correspondencia entre los SLs y los VLs (*SLtoVLMappingTable*) configurada por las entidades de gestión de la red. Una correcta configuración de esta tabla hará que cada tipo de tráfico, previamente segregado en distintos niveles de servicio, use canales virtuales diferentes.

2.5.3. Arbitraje de los puertos de salida

Los puertos de salida (tanto en los conmutadores, como en los encaminadores y en los nodos terminales) incorporan una tabla de arbitraje que permite especificar la prioridad que tendrá cada canal virtual a la hora de enviar paquetes a través de ese puerto de salida.

Es obligatorio la implementación de esta tabla si el puerto tiene más de dos canales virtuales. El arbitraje se realiza sólo entre los canales virtuales de datos, pues el canal virtual del tráfico de control siempre tiene preferencia.

La estructura de la tabla de arbitraje se muestra en la Figura 2.26. Cada tabla de arbitraje está formada por dos tablas, una para gestionar el tráfico de los canales virtuales de alta prioridad y otra para los canales virtuales de baja prioridad. Sin embargo, InfiniBand no especifica qué es alta y baja prioridad, quedando este aspecto a criterio del administrador de la subred.

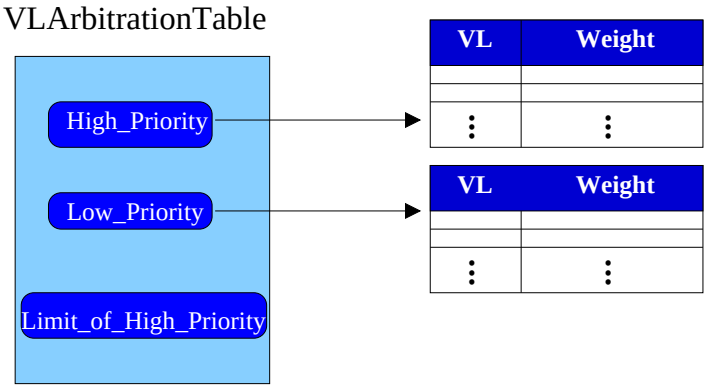


Figura 2.26: Estructura de la tabla de arbitraje.

Cada una de esas tablas tiene un funcionamiento cíclico y ponderado. Tanto la tabla de alta prioridad como la de baja prioridad tienen un máximo de 64 entradas. Cada entrada especifica un canal virtual y un peso. El peso indica la cantidad de unidades de 64 bytes a enviar por ese canal virtual. El peso debe estar entre 0 y 255, y siempre se redondea a un paquete completo.

El valor *LimitOfHighPriority* determina la cantidad máxima de información que puede sacarse de los canales virtuales de alta prioridad, antes de pasar a enviar un paquete de un canal virtual de los considerados de baja prioridad. En concreto, los canales virtuales situados en la tabla de alta prioridad pueden enviar $LimitOfHighPriority \times 4096$ bytes antes de enviar un paquete de baja prioridad. Una vez enviado el paquete de un canal virtual de baja prioridad (si había alguno listo para enviarse), se vuelve a la tabla de alta prioridad y se continúa por la entrada donde se quedó y con el peso que le restaba.

Una característica importante del funcionamiento de la tabla de arbitraje es que si no hay paquetes de alta prioridad para ser transmitidos, se puede pasar a transmitir paquetes de baja prioridad. De esta forma no se desperdicia ancho de banda, y cuando no haya tráfico de alta prioridad se puede pasar a transmitir otro tipo de tráfico de menos prioridad aprovechando el ancho de banda disponible.

2.5.4. Particiones

Tal y como se ha explicado en la Sección 2.2.3.6 (página 42), InfiniBand permite establecer particiones en la subred. De esta manera se puede conseguir aislar determinadas zonas de la subred del tráfico destinado a otras zonas.

Para utilizar las particiones como mecanismo para mejorar la calidad de servicio, se pueden dedicar algunos SLs a ciertas particiones. Esto, junto con lo anteriormente expuesto, permite aislar el tráfico de las distintas particiones en distintos canales virtuales, y garantizar a cada partición una calidad de servicio distinta.

2.6. Resumen

En este capítulo se han resumido algunas de las principales características de InfiniBand. Este nuevo estándar surgió de la unión de las principales empresas del sector de cara a solucionar muchos de los problemas que plantean los sistemas que se están usando actualmente. Desde este punto de vista, InfiniBand puede considerarse una apuesta revolucionaria más que una evolución natural de las tecnologías que se han venido utilizando.

Al igual que cualquier sistema de interconexión, InfiniBand plantea un sistema de niveles donde cada nivel proporciona unos servicios al nivel inmediatamente superior, utilizando para ello la funcionalidad que le proporciona el nivel sobre el que está situado. Se ha indicado cómo puede entenderse una comunicación virtual extremo a extremo entre ambas entidades comunicantes.

Se ha hecho un recorrido por la infraestructura que plantea InfiniBand para la gestión, lo cual constituye uno de los puntos más novedosos, pues las propias especificaciones de InfiniBand describen a un nivel muy detallado las operaciones de gestión, configuración y mantenimiento, y el funcionamiento que deben tener las entidades encargadas de esas tareas.

En la parte final de este recorrido por InfiniBand, se ha indicado la forma de utilizar varias de las características que proporciona InfiniBand para proporcionar calidad de servicio. Estas características son los niveles de servicio, la correspondencia entre niveles de servicio y canales virtuales, el arbitraje de dichos canales virtuales y las particiones. Puesto que esta tesis plantea cómo proporcionar QoS en InfiniBand, serán fundamentalmente estas características las que a partir de ahora sean consideradas. Concretamente las tres primeras, y de forma especial el arbitraje de los canales virtuales, pues será la tabla de arbitraje la principal protagonista de las propuestas realizadas.

Capítulo 3

Garantía de QoS en InfiniBand

Una vez que en capítulos anteriores se han mostrado cuáles son los mecanismos o soporte que proporciona InfiniBand en lo que se refiere a la QoS, y qué tipo de requisitos tienen las aplicaciones actuales, es el momento de indicar cuál es la vía que proponemos para garantizar dichos requisitos apoyándonos en lo que señalan las especificaciones de InfiniBand. Es decir, a lo largo de este capítulo se va a estudiar detenidamente cómo proporcionar QoS a las aplicaciones en InfiniBand. Para ello usaremos adecuadamente los mecanismos que proporciona InfiniBand. En concreto, se usarán los niveles de servicio, los canales virtuales y el arbitraje de dichos canales virtuales.

Comenzaremos planteando el modelo global que se va a usar para garantizar QoS en InfiniBand. Básicamente, este modelo está basado en garantizar ancho de banda y latencia máxima, mediante una reserva de recursos con control distribuido. A continuación, haremos un estudio en profundidad de cómo garantizar ancho de banda, y/o latencia máxima extremo a extremo. Se verá que para ello hay que buscar una secuencia de entradas en las tablas de arbitraje cuyas entradas tengan una determinada distancia máxima entre dos de ellas consecutivas en la secuencia. Se analizarán distintas opciones para hacer el tratamiento de las distancias máximas, estudiando las ventajas e inconvenientes de cada una de ellas.

3.1. Modelo para garantizar QoS en InfiniBand

En esta sección se va a plantear el modelo global que se va a utilizar para conseguir que la red proporcione QoS a las aplicaciones. El modelo que aquí se propone se basa principalmente en los siguientes aspectos:

a) Proporcionar garantía de ancho de banda y latencia. Cuando un host desea establecer una nueva conexión deberá decidir qué características quiere solicitar de

la red en cuanto a qué ancho de banda necesita que le sea garantizado y/o qué latencia máxima extremo a extremo necesita tener también garantizada.

b) Reserva de recursos. Se va a garantizar QoS a las aplicaciones mediante una reserva previa de recursos. Este esquema es similar al que se hace para los servicios integrados mediante el protocolo RSVP [BZB97, Wro97a]. Para proporcionar garantía absoluta necesitamos que las aplicaciones usen el servicio de transporte de InfiniBand conocido como Conexión Fiable (Reliable Connection).

c) Control distribuido del establecimiento de las conexiones. El host envía un mensaje de solicitud de conexión indicando los requisitos que necesita para dicha conexión. Dependiendo del modelo de red utilizado, ese mensaje de solicitud de conexión tendrá un destino u otro. Aquí se asume un modelo de control distribuido donde los conmutadores tienen capacidad para hacer esa gestión. En este caso, el mensaje de solicitud se envía hacia el host destino, y en cada conmutador intermedio de la ruta se analiza si se pueden cumplir los requisitos solicitados para el establecimiento de la conexión.

Así pues, el esquema seguido para proporcionar garantía de ancho de banda y/o latencia máxima a una aplicación dada, se basa en una reserva previa de recursos mediante un control distribuido entre los nodos que intervienen en el desarrollo de esa aplicación.

En este modelo, el mensaje de solicitud de conexión, que contiene los requisitos necesarios para poder establecer dicha conexión, viaja por la red hasta el host destino o hasta que se denieguen los requisitos en un punto intermedio de la ruta. Cuando esa solicitud de conexión llega a un conmutador intermedio o al host final de la ruta se analiza si se pueden satisfacer sus requisitos.

Si esa sonda de establecimiento de conexión llega hasta el host destino, y allí también se puede aceptar, este host destino envía al host origen de la conexión un mensaje de confirmación de establecimiento siguiendo la ruta establecida por la sonda de establecimiento. De esta forma, en cada nodo intermedio se confirma que se ha aceptado la conexión y cuando llega al host origen de dicha conexión, éste puede comenzar el envío de datos.

En algunos sistemas, este mensaje de confirmación, en vez de seguir la ruta establecida por la sonda, puede que siga una ruta distinta (por ejemplo, si la ruta inversa a la establecida no es una ruta válida). En este caso, el host que solicitó la conexión, antes de enviar datos por ella, deberá enviar otro mensaje de confirmación para que los conmutadores intermedios habiliten el tráfico de datos para esta conexión.

Si por el contrario, en algún conmutador intermedio, o en el host destino de la conexión, no se puede aceptar ésta, se envía un mensaje de denegación de conexión al host origen siguiendo la ruta inversa a la que siguió la sonda de establecimiento.

Nuevamente, aquí también existe la posibilidad de que ese mensaje de denegación de conexión siga una ruta distinta y que sea el host que solicitó la conexión el que envíe luego un mensaje de liberación de recursos a los conmutadores por donde ya había pasado la sonda de establecimiento antes de que fuera denegado su establecimiento.

Cuando un conmutador, o el host que fue origen de la solicitud, recibe un mensaje de denegación de conexión, debe liberar los recursos que le había asignado a ésta. En el caso de un conmutador, debe además reenviar el mensaje de denegación: a su antecesor en la ruta si dicho mensaje se generó donde se denegó la conexión, o a su sucesor, en el caso de que se esté enviando la liberación de recursos desde el origen.

En algunos sistemas, esta fase de establecimiento permite un proceso de negociación es decir, si el host hace una petición de conexión con unos determinados requisitos que no pueden ser aceptados, se le puede hacer una contraoferta con unas determinadas garantías para ver si pueden ser aceptadas. Por ejemplo, supongamos que un determinado usuario (o una aplicación) hace una solicitud de videoconferencia a 4 Mbps y que este requisito no puede ser aceptado en algún punto de la ruta que debe seguir. La red puede indicarle a ese usuario que no le puede garantizar 4 Mbps, pero que si le interesa podría llegar a ofrecerle 64 KBps. En este caso, será el usuario (o la aplicación origen) el que tenga que decidir si con esa garantía es suficiente, en cuyo caso aceptaría, para realizar su misión con unas garantías mínimas.

Dependiendo del protocolo concreto a usar en el establecimiento de conexiones en InfiniBand, se permitirá esta fase de negociación o no. En cualquier caso, esto no es relevante para el modelo que se está proponiendo, quedando este tipo de detalles concretos abiertos a la implementación final del protocolo de establecimiento.

Otro de los aspectos que caracteriza el modelo aquí propuesto es el hecho de que el establecimiento de las conexiones se realiza de forma distribuida, es decir, cada conmutador intermedio administra sus propios recursos (ancho de banda, canales virtuales, esquema de prioridades para minimizar los retrasos máximos, etc.), y es capaz de tomar la decisión de si acepta o no una conexión en función de la información local de la que dispone. Otra alternativa utilizada por algunos sistemas es que este establecimiento de las conexiones se realice de forma centralizada en un único nodo. Este nodo sería el administrador de la red, encargado de ir recogiendo la información de todos los nodos. En este caso, cuando una aplicación quiere establecer una conexión debe solicitarlo a este nodo administrador. Este administrador será el que acepte o deniegue la conexión, o incluso el que haga una contraoferta al host solicitante. Esta segunda alternativa tiene como gran problema el gran tráfico que puede producirse hacia y desde ese nodo administrador, que lo puede convertir en un cuello de botella.

Hay que señalar que la estrategia de establecimiento de conexiones no afecta al resto del trabajo desarrollado, y el tratamiento de la tabla de arbitraje puede hacerse

igual, tanto si la modifica de forma centralizada el Subnet Manager, como si lo hacen de forma distribuida cada uno de los agentes locales.

En cualquier caso, sea cual sea el esquema de establecimiento de conexión utilizado, en algún momento habrá que estudiar si es posible cumplir los requisitos de las aplicaciones. El tercero de los aspectos principales que caracteriza a nuestro modelo tiene que ver con estos requisitos. Como se ha indicado anteriormente, el host que desea establecer una conexión deberá indicar sus requisitos de ancho de banda y/o latencia máxima extremo a extremo. Para ello, vamos a comenzar analizando por separado cómo conseguir garantizar ambos objetivos de forma independiente, para finalizar estudiando la forma de tratar ambos simultáneamente.

3.2. Garantía de ancho de banda

En esta sección se va a asumir que sólo hay tráfico con necesidades de garantía de ancho de banda, y ningún tipo de requisito sobre latencia máxima. En la siguiente sección se estudiará cómo garantizar sólo latencia máxima, y posteriormente como ofrecer ambos tipos de garantías de forma conjunta.

Como se ha comentado en la sección anterior, la admisión de nuevas conexiones en InfiniBand puede implementarse de forma centralizada por el Subnet Manager, o de forma distribuida en cada conmutador por los agentes locales. En cualquier caso, las tablas de arbitraje deben modificarse para que los conmutadores puedan satisfacer las necesidades de las conexiones aceptadas. Que estas modificaciones las haga el Subnet Manager o el agente local es indiferente para nuestro estudio.

Por otra parte, tanto Pelissier como nosotros en trabajos previos, proponíamos dedicar la tabla de arbitraje de alta prioridad para el tráfico DBTS, y la tabla de baja prioridad para el resto de categorías. Sin embargo, en [ASD02] ya apuntábamos que esta propuesta tiene un problema. Si las fuentes que están enviando tráfico de tipo DBTS envían información superando el ancho de banda que previamente reservaron, todo el tráfico que utilice la tabla de baja prioridad se verá afectado. En concreto, si no se puede garantizar que las fuentes van a tener un comportamiento acorde con lo que se han comprometido, no puede dársele garantía al tráfico que utilice únicamente la tabla de baja prioridad. Esto es debido a que el $LimitOfHighPriority$ no permite hacer un reparto "fino" del ancho de banda. Como ya se ha comentado anteriormente, este límite puede tener un valor entre 0 y 255. En concreto, los canales virtuales de la tabla de alta prioridad pueden transmitir $LimitOfHighPriority \times 4096$ bytes antes de tener que enviar un paquete de baja prioridad. En el mejor de los casos, para el mayor tamaño de paquete permitido en IBA (4096 bytes), esto significa que para un valor de 1 de $LimitOfHighPriority$ podremos usar un 50 % del ancho de banda para

los canales virtuales de la tabla de alta prioridad y otro 50 % para los canales virtuales de la tabla de baja prioridad.

Así pues, parece evidente que para darle garantías a cualquier tipo de tráfico necesitamos poner los canales virtuales que utilice en la tabla de alta prioridad. En nuestro caso, situaremos los canales virtuales utilizados tanto por el tráfico DBTS como por el DB en dicha tabla. De esta forma, dejamos la tabla de baja prioridad para los tráficos sin necesidades explícitas de garantía, es decir, PBE, BE y CH. Así pues, de momento no vamos a considerar la necesidad de garantía de latencia del tráfico DBTS y vamos a estudiar como llenar la tabla de alta prioridad para garantizar ancho de banda tanto al tráfico DBTS como DB. Posteriormente se indicará cómo ofrecer ambos tipos de garantías.

En lo que respecta al tráfico sin necesidades de ningún tipo de garantía (PBE, BE y CH), el administrador de la red habrá fijado un ancho de banda mínimo disponible para ellos, estableciendo el valor adecuado en el $LimitOfHighPriority$. El reparto de ese ancho de banda entre cada uno de los tipos vendrá fijado por una adecuada configuración de las entradas correspondientes a ese porcentaje en la tabla de baja prioridad.

De acuerdo con las especificaciones de InfiniBand, cada tabla de arbitraje puede tener un máximo de 64 entradas, cada una con un peso máximo de 255. La tabla se recorre de forma cíclica, con lo que podemos suponer que cada vuelta de la tabla se va repitiendo de forma sucesiva. Vamos a llamar $frame$ al resultado de una vuelta completa a la tabla donde cada canal virtual habría enviado la información para la que está habilitado. Evidentemente, ese frame se va a ir repitiendo a lo largo del tiempo, con lo que las aplicaciones deberán tener reservado un porcentaje en dicho frame en función de sus necesidades. Así, a cada conexión con necesidades de garantía de ancho de banda (tráfico DBTS o DB) se le asignará un determinado número de slots en cada frame de forma que se puedan cumplir sus necesidades de ancho de banda.

Hay que recordar que un peso de uno en una entrada posibilita la transmisión de 64 bytes, o un paquete completo, del canal virtual correspondiente a esa entrada, en cada recorrido a la tabla de arbitraje. De esta forma, el máximo número de slots por frame es de 64×255, siendo cada slot de una anchura de 64 bytes. Si calculamos el peso necesario en cada entrada con respecto a este valor máximo por frame se consigue simplificar de una forma significativa los cálculos a realizar cuando se vayan aceptando nuevas conexiones.

Al tener un máximo fijo de 64 entradas en la tabla de arbitraje de alta prioridad, podemos plantearnos cómo repartir las entradas entre las conexiones. Como sabemos, cada entrada de la tabla de arbitraje especifica mediante un peso la cantidad de datos que se pueden transmitir por un determinado canal virtual. En función de las conexiones establecidas, y de qué niveles de servicio sean cada una de ellas, tendremos una relación

entre cada conexión establecida y el canal virtual que esté utilizando. De esta forma, podríamos dedicar una entrada (o varias) a cada conexión que se aceptara. Esto plantea el problema de que limitaríamos el número de conexiones a aceptar en función del número de entradas de la tabla de arbitraje, y no del ancho de banda disponible. Parece más adecuado agrupar en cada entrada de la tabla varias conexiones del mismo nivel de servicio hasta alcanzar el peso máximo de 255. De esta forma, la única limitación que tenemos para aceptar nuevas conexiones va a ser el ancho de banda disponible. Sin embargo, esto no es del todo cierto, pues podemos tener todas las entradas de la tabla de arbitraje ocupadas y llenas, menos unas cuantas de ellas que no estén completamente llenas, y si se recibe una nueva solicitud de conexión de otro nivel de servicio, no tener entradas para poder ubicar esta nueva conexión. Así, en este caso tendríamos ancho de banda disponible y no podríamos aceptar esa conexión. De todas formas, este es un caso límite y que surge cuando ya se tienen todas las entradas de la tabla asignadas, y por tanto estamos cercanos a un reparto del ancho de banda total.

Este planteamiento de que varias conexiones que usen el mismo canal virtual estén usando las mismas entradas de la tabla de arbitraje, obliga a almacenar para cada entrada de la tabla de arbitraje de alta prioridad el ancho de banda que está atendiendo. Podemos asumir que ese ancho de banda requiere un valor en punto flotante, y que éste ocupa 8 bytes. Como tenemos 64 entradas en cada tabla de arbitraje de alta prioridad, necesitamos $64 \times 8 = 512$ bytes por puerto.

Hay que señalar que sólo es necesario mantener esa información para la tabla de alta prioridad, pues la tabla de baja prioridad no va a ser modificada en función del ancho de banda. Por el contrario, la idea es rellenar la tabla de baja prioridad con unos valores asignados en función del ratio de prioridades asignado entre los distintos tipos de tráfico que van a usar esta tabla. Para nuestras pruebas hemos usado la siguiente proporción: una entrada con un peso de uno para el tráfico CH, otra entrada pero con un peso de 10 para el tráfico BE y 4 entradas a 255 cada una para el tráfico PBE.

Por otra parte, hay que señalar que, tanto para la tabla de alta prioridad como para la de baja prioridad, lo importante no es el valor concreto para una entrada, sino el ratio entre los pesos que contiene la tabla. Por ejemplo, si la tabla sólo tiene una entrada, tenga ésta el peso que tenga, podrá absorber el 100 % del tráfico. Si por ejemplo esa única entrada de la tabla tuviera un peso de 10 y luego añadimos otra entrada, el reparto de ancho de banda nos lo va a dar la relación entre los pesos de estas dos entradas. Si a la nueva entrada le ponemos también un peso de 10 le estaremos asignado la mitad del ancho de banda disponible a cada uno de los canales virtuales presentes en la tabla de arbitraje, tal y como se indica en la Figura 3.1(a). Si por el contrario el canal virtual de una entrada tiene un peso de 20, mientras que el otro canal virtual tiene un peso de 10 en su entrada, se está haciendo un reparto de 66 % y 33 %, respectivamente, tal y como se muestra en la Figura 3.1(b). En la Figura 3.1(c) se muestra otra situación donde uno de los canales tiene un peso de 10 y el otro de

30, teniendo cada uno de ellos un total de 25 % y 75 %, respectivamente, del ancho de banda total.

VL	Weight
1	10
2	10

(a)

VL	Weight
1	20
2	10

(b)

VL	Weight
1	10
2	30

(c)

Figura 3.1: Tres posibles situaciones de reparto de ancho de banda entre los canales virtuales VL1 y VL2: (a) 50 % para cada uno, (b) 66 % VL1 y 33 % VL2, (c) 25 % VL1 y 75 % VL2.

Así pues, asignando el peso de las entradas respecto al máximo del frame se simplifica todo el proceso. De esta forma, se le está dando a cada conexión el ancho de banda garantizado en caso de que se llegara a asignar todo el ancho de banda del enlace, o aún más si no se llega a cubrir todo, pues las conexiones presentes podrán utilizar el ancho de banda sobrante. De esta manera se tiene garantizado el buen comportamiento en el caso peor, y además se permite que se use el ancho de banda disponible cuando no se haya asignado en su totalidad.

Vamos por tanto a calcular el peso que debe asignarse a una entrada de la tabla de arbitraje de acuerdo al ancho de banda medio acumulado por las conexiones que comparten dicha entrada. En primer lugar, cuando se acepta una nueva conexión se intentará ubicarla en una entrada que ya estuviera en uso por otras conexiones del mismo canal virtual. Si con ella se sobrepasa el peso máximo por entrada de 255 se les asignará otra entrada (o varias) con el peso excedente de dicha cantidad. Si no hubiera ninguna entrada para ese canal virtual, se seleccionará otra entrada sin usar para dedicarla a ese canal virtual.

Esta metodología tiene la importante ventaja de no requerir recalcular las entradas ya utilizadas de la tabla de arbitraje por el resto de conexiones cada vez que se acepte una nueva conexión. Si la asignación del peso se hiciera en función de las entradas ya establecidas se deberían modificar todas las entradas de la tabla cada vez que se ubicara una nueva petición en la tabla de arbitraje. De esta forma, asignando el peso con respecto al máximo posible, se simplifica el proceso de aceptación de nuevas conexiones, y se minimizan las operaciones de gestión de la tabla de arbitraje.

Debido a las especificaciones de la estructura de la tabla de arbitraje y su funcionamiento, cada frame consiste en $64 \times 255 = 16320$ slots de 64 bytes. Se puede calcular el tiempo que se va a tardar en transmitir un frame completo asumiendo que todas las entradas tuvieran el peso máximo. Vamos a llamar a este tiempo T_{frame}. En la Tabla 3.1 se ha incluido el valor concreto para cada una de las tres posibles velocidades en InfiniBand.

Velocidad	Velocidad del Enlace (Gbps)	T_{frame} (mseg)
1x	2,5	3,342336
4x	10	0,835584
12x	30	0,278528

Tabla 3.1: Tiempo necesario para enviar un frame completo (16320 slots de 64 bytes) para las tres posibles velocidades de InfiniBand.

El ancho de banda B asignado a una conexión puede calcularse como el número de slots asignados a esa conexión dividido por el número total de slots en el frame y multiplicado por el ancho de banda total del enlace. Es decir, el porcentaje del frame asignado a esa conexión multiplicado por el ancho de banda total del enlace.

$$B = \frac{N_{Slots}}{N_{Slots\ por\ frame}} \times B_{enlace}$$

Así pues, el número de slots para una conexión que necesitara un ancho de banda medio de B bps es

$$N_{Slots} = \left\lceil \frac{B}{B_{enlace}} \times 64 \times 255 \right\rceil$$

Por otra parte, podemos ver el T_{Frame} como

$$T_{frame} = \frac{64\ entradas \times 255\ \frac{pesos}{entrada} \times 64\ \frac{bytes}{peso} \times 8\ \frac{bits}{byte}}{B_{enlace}}$$

A partir de las expresiones anteriores, el número de slots necesarios para satisfacer una nueva conexión con ancho de banda medio B se obtiene a partir de la siguiente expresión:

$$N_{Slots} = \left\lceil B \times \frac{T_{frame}}{512} \right\rceil$$

Como vemos, el número de slots del frame que debemos asignar a una conexión con un determinado requisito de ancho de banda medio, está en función de dicho ancho de banda y del tiempo que se tarda en enviar el frame completo, que a su vez depende de la velocidad concreta del enlace que se esté utilizando.

De esta forma se está fijando una cantidad de ancho de banda mínimo, que coincide con lo que la aplicación ha solicitado. En el caso de que no se utilice el total de ancho de banda, las aplicaciones podrán utilizar más ancho de banda del que tienen garantizado.

Ese exceso de ancho de banda será repartido también según la proporción de pesos establecida en la tabla de arbitraje. Así pues, las aplicaciones podrán repartirse el exceso de ancho de banda que pudiera existir, pero cada una de ellas tendrá garantizada la cantidad que haya solicitado en el momento de su establecimiento.

De igual forma, el porcentaje de ancho de banda que se fija para el tráfico PBE, BE y CH, es sólo una cota mínima para el caso peor donde se agote el ancho de banda total del enlace. En situaciones donde no haya tráfico suficiente de otras categorías para cubrir todo el ancho de banda, el tráfico sin garantías podrá sobrepasar esa cota mínima. Si no se le fijara una cota mínima a este tipo de tráfico, cuando hubiera tráfico de mayor prioridad no se dejaría ancho de banda a estos tipos de baja prioridad. El hecho de que sean tráficos de baja prioridad sin necesidad de una garantía de latencia máxima o de un ancho de banda mínimo garantizado para las aplicaciones, no implica que se pueda eliminar este tráfico de la red. Los paquetes de estos tráficos deben poder llegar a su destino, aunque por supuesto sin ningún tipo de garantía en cuanto al tiempo de llegada.

Es importante fijarse en que, según la definición de tráfico realizada, el tráfico CH no requiere ningún tipo de garantía, y se va a limitar a usar el ancho de banda disponible cuando no haya ningún otro tipo de tráfico en la red. Este tipo de tráfico puede ser interesante para realizar copias de seguridad o cualquier transferencia de información que requiere tiempo y ancho de banda, pero no tiene unos requisitos sobre cuándo debe hacerse o cuánto debe tardar. De esta forma, las copias de seguridad podrían, por ejemplo, hacerse por la noche, cuando nadie más está utilizando la red.

Ahora bien, por el funcionamiento propio de la tabla de arbitraje, el comportamiento pensado para el tráfico CH no es posible. Si no se le reserva a este tráfico ningún peso en la tabla de arbitraje nunca podrá utilizar la red, aunque no haya otro tipo de tráfico. Si se le reservan slots se le está reservando ancho de banda en cada frame y se le está quitando al resto de aplicaciones. Así pues, no hay otra alternativa que asignarle al tráfico CH una entrada con un peso mínimo de uno en la tabla de baja prioridad. De esta forma, cuando no haya otro tipo de tráfico, el tráfico CH podrá utilizar todo el ancho de banda de la red. Sin embargo, cuando haya más tráfico, se le estará dedicando al tráfico CH un porcentaje tan pequeño que se puede considerar que no tiene importancia el ancho de banda que le resta a los otros tipos de tráfico, y que sólo va a funcionar cuando no haya ningún otro tipo de tráfico disponible.

De esta forma, con una correcta configuración de las tablas de arbitraje, junto con un proceso de dimensionado de los recursos disponibles y un proceso de solicitud de requisitos, se consigue que la red de InfiniBand proporcione garantía de ancho de banda.

3.3. Garantía de latencia

Abordamos ahora cómo garantizar la latencia máxima que van a sufrir los paquetes de una conexión. Dada una conexión entre un host A y un host B el objetivo es conseguir que el tiempo invertido por los paquetes en alcanzar el destino sea menor que un cierto deadline, es decir:

$$Latencia\,Máxima\,AB \leq Deadline\,Fijado$$

Esos paquetes van a tener que cruzar por diversos conmutadores y usar varios enlaces para llegar al host B, y por tanto:

$$\begin{aligned}\text{Latencia Máxima AB} \quad \leq \quad & \text{Tiempo Enlaces} \; + \\ & + \sum_{Sws\ de\ AB} (\text{ Tiempo Máx por Switch }) \leq \text{ Deadline Fijado}\end{aligned}$$

o bien

$$\text{Latencia Máxima AB} - \text{Tiempo Enlaces} \leq \sum_{Sws\ de\ AB} (\text{Tiempo Máximo por Switch})$$

donde el tiempo de los enlaces es una cantidad conocida.

El host destino de la conexión deberá comprobar si el tiempo máximo acumulado entre todos los conmutadores, junto con el tiempo de los enlaces, es menor o igual a la latencia máxima admitida por dicha conexión. Esta información de la latencia máxima admitida viajará en la cabecera del paquete de establecimiento de conexión. También se puede ir comprobando en cada conmutador si el valor acumulado ya supera al máximo admisible, en cuyo caso ya no sería necesario seguir avanzando en la ruta. Hay que señalar que ésta es una de las estrategias posibles, y que sin pérdida de generalidad, podría hacerse lo mismo en un entorno centralizado. En este caso sería el Subnet Manager el encargado de hacer esta comprobación.

En las pruebas realizadas, de cara a simplificar el tratamiento en cuanto al cálculo de la latencia, se va a repartir la latencia máxima total de forma equitativa entre todos los nodos de la ruta, teniendo también en cuenta los tiempos de vuelo por los enlaces. De esta manera se simplifica mucho el estudio en los nodos intermedios y se garantiza un tratamiento idéntico en todos ellos, lo cual también es muy deseable. En cualquier caso, esta decisión no afecta al resto del trabajo desarrollado. Las tablas de arbitraje deberán adecuarse en cada nodo para poder satisfacer la garantía exigida, independientemente de que la latencia máxima se reparta equitativamente o no.

De esta forma, dada la especificación que realice una aplicación durante su fase de establecimiento, cada conmutador intermedio deberá estudiar si es capaz de cumplir

con su parte correspondiente. Si no se puede cumplir informará de ello a los nodos precedentes y al host origen. Si se puede satisfacer la petición, se hará avanzar el mensaje de solicitud de conexión al nodo siguiente, hasta que alcance el final de la ruta. Al igual que en el caso de garantía de ancho de banda, este esquema distribuido no afecta al planteamiento global de cómo configurar la tabla de arbitraje para conseguir satisfacer una latencia máxima, tan sólo a quién debe hacer dicha configuración, el agente local o el Subnet Manager.

El cálculo del tiempo máximo que un paquete de una determinada conexión va a tardar en salir de un conmutador va a depender de la arquitectura de dicho conmutador. Actualmente no hay un modelo concreto en cuanto a la arquitectura que van a tener los conmutadores de InfiniBand. En sus últimos documentos técnicos, IBM parecía que iba a utilizar un modelo de conmutador con buffers centrales con reserva dinámica del espacio de dichos buffers [IBM01]. Sin embargo, otras posibles arquitecturas tienen buffers tanto en los puertos de entrada como en los de salida. Además, en este caso la entrada al crossbar puede estar multiplexada o no. Es decir, podemos tener una entrada y salida al crossbar por cada canal virtual o una única entrada y salida al crossbar por puerto. Así pues, vamos a considerar tres posibles arquitecturas del conmutador: con buffer central, con buffers a la entrada y a la salida con el crossbar multiplexado y con crossbar no multiplexado.

En cualquier caso, cuando se reciba la solicitud de establecimiento de una nueva conexión con un requisito de una determinada latencia máxima, en cada conmutador habrá que calcular el tiempo máximo que se puede tardar en sacar un paquete de esa conexión, si es que se llegara a establecer. Para ello hay que calcular el peor caso para un paquete que acabara de llegar al conmutador y que lo hiciera por el canal virtual que se le asignaría en caso de que la conexión fuera aceptada. Veamos los cálculos para cada una de las arquitecturas consideradas.

3.3.1. Conmutador con crossbar multiplexado

En este caso, por cada puerto de entrada (salida) del conmutador hay una única entrada (salida) al crossbar. Así, no es posible que dos canales virtuales del mismo puerto de entrada estén transmitiendo al mismo tiempo por el crossbar, aunque vayan a puertos de salida distintos, pues ambos deben usar la misma entrada al crossbar. Este esquema simplifica el diseño del crossbar, y con conmutadores grandes es la única alternativa viable. La estructura de este conmutador se muestra en la Figura 3.2.

Podemos obtener el tiempo máximo que un paquete correspondiente a un determinado canal virtual puede tardar en salir de un conmutador a partir del número máximo de paquetes que lo harán antes que él (en el peor de los casos posibles), y el tiempo que éstos tardan en conseguirlo. Analicemos, paso a paso, el cálculo de ambos valores.

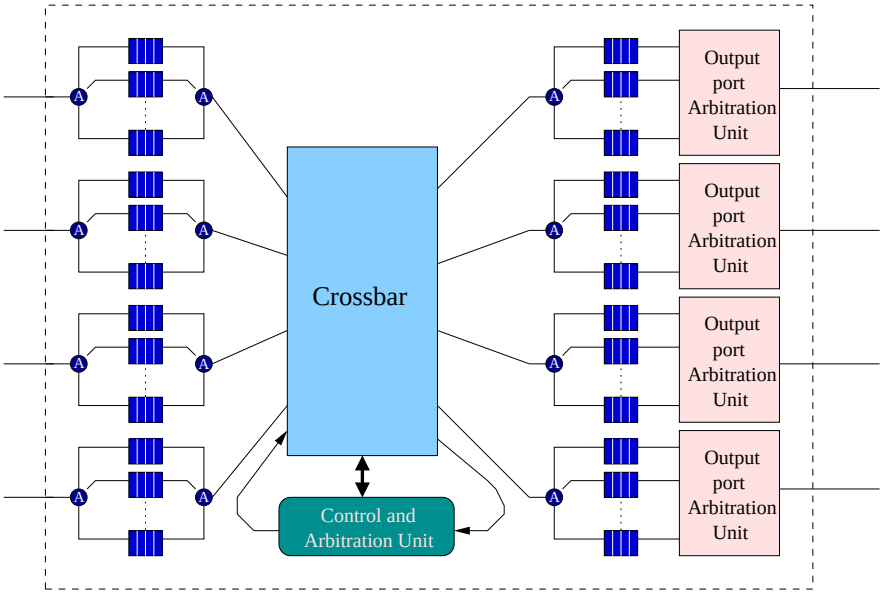


Figura 3.2: Estructura para un conmutador de InfiniBand con el crossbar multiplexado.

- Lo primero que debemos considerar es el número de paquetes que puede haber delante de él en su buffer de entrada. Suponiendo que los buffers tienen un comportamiento FIFO, todos los paquetes que hubieran llegado antes que él al buffer deben abandonarlo también antes que él. Este número máximo de paquetes en el buffer de entrada viene dado por el tamaño del buffer y el tamaño máximo de un paquete (MTU), según la expresión $\left\lceil \frac{\text{Tamaño Buffer}}{\text{MTU}} \right\rceil$.

- Como el crossbar está multiplexado, hay que considerar todos los paquetes de la entrada que pueden ocupar la salida del crossbar que él debe utilizar. Como no se indica nada en las especificaciones de InfiniBand al respecto de cómo gestionar la asignación del crossbar a los puertos, vamos a suponer que la asignación del crossbar se realiza de forma cíclica (round-robin) entre los puertos de entrada. De todas formas, cuando debido al comportamiento de la tabla de arbitraje un canal virtual de un puerto de salida se llene, ningún otro paquete podrá cruzar desde la entrada hacia ese canal virtual mientras no se desaloje alguno de los actuales. Así pues, el comportamiento de la tabla de arbitraje afecta al comportamiento del algoritmo round-robin que se aplica al crossbar, adaptándolo, en caso de mucho tráfico, a la política de prioridades que se esté aplicando en la salida del puerto.

  Por tanto, para calcular el número de paquetes que pueden cruzar el crossbar antes que un determinado paquete pueda hacerlo, hay que tener en cuenta cualquier paquete que esté en cualquier buffer de cualquier puerto de entrada y que quiera cruzar hacia el mismo puerto de salida. No habría que incluir en esta cifra al puerto interno ni a los canales virtuales dedicados al tráfico de control de los puertos externos, ya que por ellos sólo pueden ir mensajes de control que siempre

van a tener prioridad. Esto ya incluye el valor calculado en el punto anterior para los paquetes que están delante de él en su buffer. Así pues, habría que considerar:

$$\Big(PuertosE \times CVDatos\Big) \times \left\lceil \frac{\text{Tamaño Buffer}}{\text{MTU}} \right\rceil$$

donde se ha considerado el número máximo de paquetes que pueden tener almacenados los canales virtuales de datos de usuario de los puertos externos.

- Una vez que el paquete consiga cruzar el crossbar se debe considerar qué le puede ocurrir antes de salir del conmutador. Cuando un paquete consiga cruzar el crossbar todos los paquetes que estén delante de él en el buffer del puerto de salida deberán salir antes que él, pues se está suponiendo un comportamiento FIFO de esos buffers. Así pues, a la expresión anterior debe sumársele un paquete que pudiera estar cruzando en ese momento (sea del puerto y canal virtual que sea) y tantos paquetes como pueda ya haber en el buffer del canal virtual del puerto de salida. El número máximo de paquetes que puede haber en el puerto de salida viene dado por la expresión $\left\lceil \frac{\text{Tamaño Buffer}}{\text{MTU}} \right\rceil$. De esta forma, el número de paquetes a considerar ahora sería:

$$\left[\Big(PuertosE \times CVDatos\Big) \times \left\lceil \frac{\text{Tamaño Buffer}}{MTU} \right\rceil\right] + 1 + \left\lceil \frac{\text{Tamaño Buffer}}{MTU} \right\rceil$$

- El tiempo que va a tardar en abandonar el puerto de salida lo va a marcar el barrido de la tabla de arbitraje. Sea *Barrido* el número de paquetes que pueden salir por el canal físico antes de que un canal virtual dado pueda utilizar el canal físico. Como en la tabla de arbitraje el turno es rotatorio, este número de paquetes dependerá de la separación máxima entre dos entradas consecutivas del canal virtual que está usando la conexión a la que pertenece el paquete. En caso de que ese canal virtual sólo tenga una entrada en la tabla de arbitraje, ese número se corresponderá con una vuelta completa a la tabla.

  De esta forma, la expresión calculada en el punto anterior hay que multiplicarla por ese número *Barrido*, ya que todos los paquetes que tengan que salir delante de un determinado paquete procedentes del mismo canal virtual de salida, van a tener el mismo tratamiento que él en el arbitraje. Así pues, el número máximo de paquetes que puede encontrar delante un paquete será ahora:

$$\left[\left[\Big(PuertosE \times CVDatos\Big) \times \left\lceil \frac{Buffer\ Size}{MTU} \right\rceil\right] + 1 + \left\lceil \frac{Buffer\ Size}{MTU} \right\rceil\right] \times Barrido$$

- Finalmente se debe considerar el efecto del LimitOfHighPriority (LHP). Este límite permite enviar $LHP \times 4$ Kbytes de paquetes de alta prioridad antes de enviar un paquete de baja prioridad. Así pues, el número de paquetes de baja prioridad que pueden enviarse antes de un paquete de alta prioridad va a depender del tamaño del buffer y del tamaño máximo del paquete (MTU). Al añadir este valor a la expresión anterior, se obtiene:

$$P = \left[\left[\left(PuertosE \times CVDatos\right) \times \left\lceil \frac{\text{Tamaño Buffer}}{MTU} \right\rceil\right] + 1 + \left\lceil \frac{\text{Tamaño Buffer}}{MTU} \right\rceil\right] \times Barrido + 1 + \left\lceil \frac{\text{Tamaño Buffer}}{4096 \times LHP} \right\rceil \quad (3.1)$$

donde P es el número máximo de paquetes que pueden enviarse antes de que se envíe un paquete de alta prioridad que acabara de llegar al conmutador.

A partir de este número máximo de paquetes calcular el tiempo máximo para un conmutador es inmediato, puesto que se conoce el tamaño máximo del paquete (MTU) y la velocidad del enlace:

$$T_{MaxSw} = \frac{P \times MTU}{Velocidad\ Enlace}$$

Así pues, este T_{MaxSw} sería el tiempo máximo que un paquete que llegue al conmutador puede tardar en salir de él. Evidentemente, este tiempo es una cota máxima suponiendo las peores circunstancias, pero para dar garantía absoluta hay que considerar el caso más desfavorable.

3.3.2. Conmutador con crossbar no multiplexado

En este caso, el modelo de conmutador sería el mostrado en la Figura 3.3. Ahora cada canal virtual tiene una entrada en el crossbar. De esta forma, dos paquetes situados en dos canales virtuales del mismo puerto de entrada pueden estar cruzando hacia dos canales virtuales distintos (del mismo puerto o no) de la salida. Este tipo de arquitectura puede usarse solamente si el número de canales virtuales es pequeño. Para un número grande de canales virtuales la tecnología actual no permite tener crossbar no multiplexados.

Para este modelo de conmutador los cálculos anteriores se simplifican, pues los paquetes que llegan a un determinado buffer del conmutador no tienen competencia con otros canales virtuales para entrar en el crossbar, aunque sí para salir de él. Los paquetes de un determinado nivel de servicio que llegan al crossbar van a querer cruzar a un buffer de ese tipo del puerto de salida correspondiente. De esta forma, la competencia

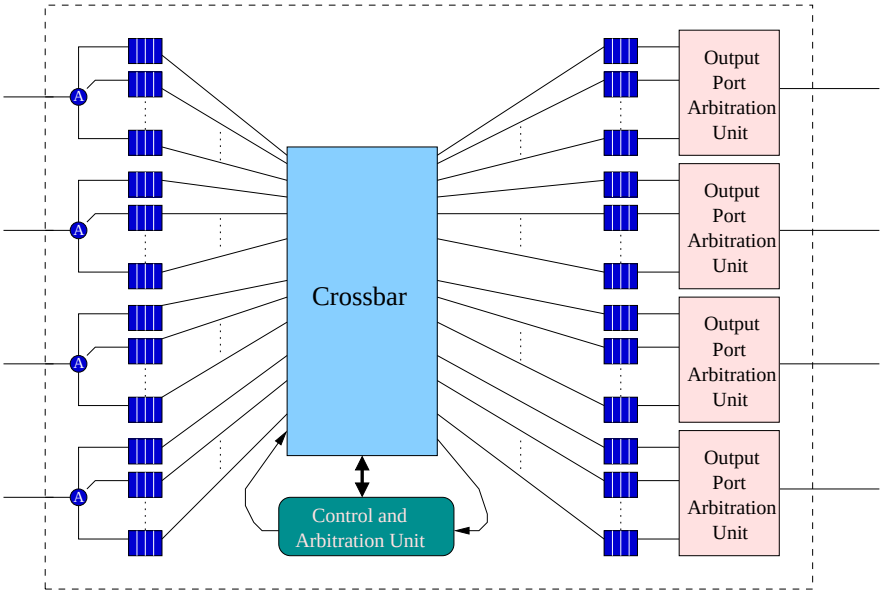


Figura 3.3: Estructura para un conmutador de InfiniBand con el crossbar no multiplexado.

por el buffer de salida se limita a los canales virtuales de los puertos de entrada que compartan el mismo canal virtual en el puerto de salida. Esto es así porque la tabla $SLtoVLMappingTable$ permite agrupar varios niveles de servicio en un único canal virtual, pero no proporciona ninguna forma para distribuir el tráfico de un mismo nivel de servicio entre varios canales virtuales. Entre estos canales virtuales del mismo nivel de servicio se aplica round-robin. Así pues, para el cálculo del número máximo de paquetes que pueden cruzar delante de un paquete de alta prioridad, habrá que considerar el número máximo de paquetes por cada uno de estos buffers.

Hay que señalar que sólo puede haber un canal virtual en cada puerto de entrada compitiendo por el mismo canal virtual del puerto de salida. De esta forma, el primer término de la Expresión (3.1) se simplifica, quedando sólo un canal virtual por puerto. El resto de los cálculos son los mismos, con lo que la expresión final para P quedaría:

$$P = \left[\left[PuertosE \times \left\lceil \frac{\text{Tamaño Buffer}}{MTU} \right\rceil\right] + 1 + \left\lceil \frac{\text{Tamaño Buffer}}{MTU} \right\rceil\right] \times Barrido + \\ +1 + \left\lceil \frac{\text{Tamaño Buffer}}{4096 \times LHP} \right\rceil \quad (3.2)$$

3.3.3. Conmutador con buffer central

En esta arquitectura los paquetes se almacenan en un buffer central compartido por todos los canales virtuales de todos los puertos. La estructura del conmutador podría ser como la mostrada en la Figura 3.4. En este caso, el espacio disponible para alojar los paquetes se comparte entre todos los canales virtuales, y se va asignando dinámicamente a los buffers en función de la demanda del tráfico. Esta es la estructura que en los últimos documentos técnicos parecía que iban a tener los conmutadores InfiniBlue de IBM [IBM01].

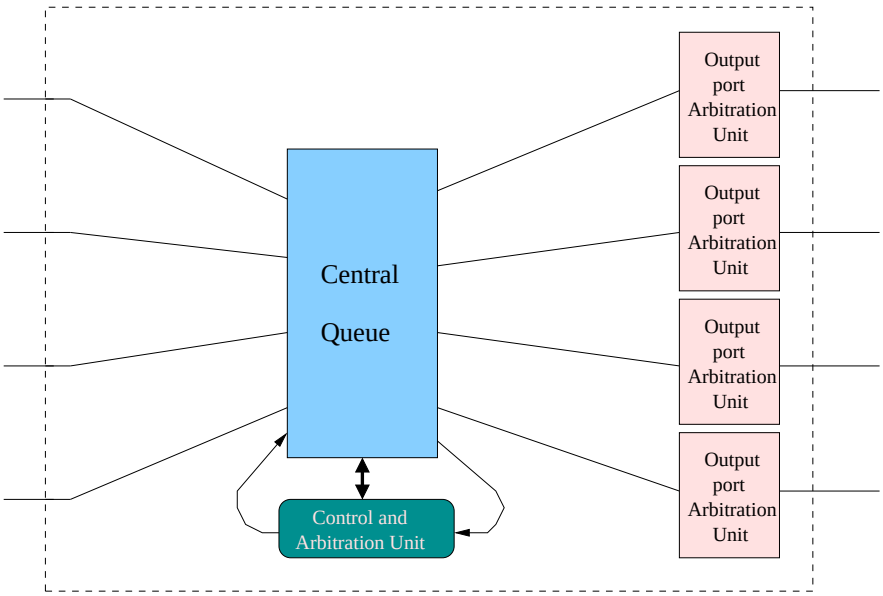


Figura 3.4: Estructura para un conmutador de InfiniBand con buffer central que dinámicamente asigna espacio a los buffers para ajustarse a las demandas del tráfico.

Igual que en los casos anteriores, el algoritmo de arbitraje se encarga de seleccionar los paquetes para ir saliendo directamente de este buffer. Como no hay que cruzar ningún crossbar, sólo hay que considerar los paquetes que pueden salir por el mismo puerto de salida. Estos paquetes son los que ya se consideraron en la sección anterior como almacenados en el buffer de salida. Así pues, la expresión final sería en este caso:

$$P = \left\lceil \frac{\text{Tamaño Buffer}}{MTU} \right\rceil \times Barrido + 1 + \left\lceil \frac{\text{Tamaño Buffer}}{4096 \times LHP} \right\rceil \tag{3.3}$$

3.3.4. Resumen

Como se ha visto en las secciones anteriores, sea cual sea el modelo de conmutador utilizado, se puede calcular el tiempo máximo que puede transcurrir desde que un

paquete llega al puerto de entrada de un conmutador hasta que sale de este último. En las secciones anteriores se ha comprobado que este cálculo depende principalmente del número de puertos, del número de canales virtuales por puerto y del tamaño de los buffers de entrada y salida. Otros aspectos que influyen en este cálculo son el tamaño máximo del paquete y el hecho de que el crossbar esté o no multiplexado. Además, hay que tener en cuenta el estado y comportamiento de la tabla de arbitraje del puerto de salida.

En las secciones anteriores se han estudiado algunos casos, más o menos comunes, de conmutadores comerciales, con una serie de suposiciones. Sin embargo, al no saber cuál va a ser el modelo seguido por los fabricantes de InfiniBand, no es posible hacer un estudio específico. En cualquier caso, sea cual sea el modelo de conmutador utilizado, será posible realizar un estudio similar para calcular el tiempo máximo que puede tardar un paquete en cruzar por ellos. También es evidente que el término *Barrido* aquí utilizado estará presente cualquiera que sea la expresión final.

El valor de *Barrido* será distinto para cada canal virtual, y está en función de la distancia máxima entre dos entradas consecutivas asignadas a dicho canal virtual en la tabla de arbitraje. Si ese canal virtual sólo tiene una entrada en la tabla de arbitraje, el término *Barrido* corresponderá a una vuelta completa a la tabla. Este término *Barrido* podrá modificarlo la entidad encargada de configurar la tabla de arbitraje (el agente local o el Subnet Manager), en función de los requisitos a cumplir por cada conexión.

Así pues, para cumplir un determinado requisito de latencia habrá que situar en la tabla de arbitraje una secuencia de entradas separadas entre sí a una distancia máxima. Esta distancia máxima vendrá dada por las características del conmutador a utilizar y del requisito de latencia a conseguir. De esta manera se asegura que cuando un paquete llegue al conmutador va a poder ser transmitido en un determinado espacio de tiempo que vendrá marcado por dicha separación.

Así pues, para estudiar si es posible satisfacer la petición de latencia máxima, cada nodo debe convertir esa latencia máxima en una distancia máxima entre dos entradas consecutivas en la tabla de arbitraje. En función de las características arquitectónicas del conmutador (número de puertos, tamaño de los buffers, tipo de crossbar, etc.), se puede calcular el número de paquetes que pueden salir delante de otro paquete que acabe de llegar al conmutador. Este número indica la espera máxima (contada como número de paquetes a transmitir), que puede tener que realizar un paquete de un determinado SL. Sabiendo el número máximo de paquetes que pueden transmitirse delante de él, se puede fácilmente calcular el tiempo máximo que pasará hasta que pueda salir del nodo. A partir de ese tiempo máximo se puede calcular la distancia máxima, mediante un proceso igual de sencillo. Una vez calculada esta distancia máxima, la conexión tendrá garantía de la cantidad de información de otros canales virtuales (tiempo máximo) que puede transmitirse antes que se deje salir a un paquete suyo.

3.4. Garantía conjunta de ancho de banda y latencia

Como se ha visto en las secciones anteriores, una petición de conexión de un determinado ancho de banda de B Mbps y una latencia máxima extremo a extremo de t mseg, acabará tratándose en cada nodo intermedio (o en el Subnet Manager) como una petición de un determinado peso w y una distancia máxima d entre dos entradas consecutivas de la tabla de arbitraje.

Así pues, el número de entradas que deben ser asignadas a una conexión en la tabla de arbitraje de alta prioridad del puerto de salida de los conmutadores por donde pasa, vendrá dado por el máximo entre las que necesite por su requisito de ancho de banda ($\frac{Peso}{255}$) y las que necesite por su necesidad en cuanto a distancia máxima entre dos entradas consecutivas de la tabla, debidas a su requisito de latencia máxima.

Si varias conexiones que requieren la misma distancia entre entradas comparten una secuencia de entradas en la tabla, es evidente que todas ellas utilizarán el mismo canal virtual. Como se ha dicho anteriormente, el peso que tendrá cada una de las entradas de la secuencia vendrá dado por el ancho de banda acumulado por todas las conexiones que comparten dicha secuencia de entradas.

Por otra parte, en trabajos previos se ha considerado categorizar el tráfico de las distintas aplicaciones en función de sus requisitos de ancho de banda medio, independientemente de las necesidades que tengan en cuanto a garantía de latencia máxima, y por tanto en cuanto a separación máxima entre dos entradas en la tabla de arbitraje. Esto obliga a juntar conexiones de distintos requisitos en cuanto a distancia máxima, pero similar ancho de banda medio, bajo el mismo nivel de servicio y por tanto en el mismo canal virtual.

Con esta agrupación del tráfico basada en el ancho de banda medio se le está dando a todo el tráfico de un mismo canal virtual el tratamiento que necesita la conexión más restrictiva. El problema es qué hacer cuando esa conexión más restrictiva termina y se deben eliminar sus requisitos de la tabla de arbitraje. Se podrían considerar, en principio, dos líneas de actuación:

- No hacer nada y mantener la situación actual para ese canal virtual. Esto acabará significando desperdiciar recursos de la tabla de arbitraje, pues se le estarían dedicando más entradas de las que realmente hacen falta para las conexiones que ahora quedan en ese canal virtual.

- Modificar las entradas de la tabla de arbitraje para ese canal virtual para que tenga ahora el tratamiento de la conexión más restrictiva entre las que aún quedan que usan ese canal virtual. Esta opción tiene el problema de que es necesario ir guardando la información de las conexiones que hay en cada puerto, ya que si no es así, no hay forma de saber cuál es la conexión más restrictiva entre las que

quedan. Esta opción tampoco es viable, ya que también es necesario guardar una información que podría ocupar un tamaño considerable.

Así pues, cualquiera de las dos alternativas tiene demasiados problemas y hace poco interesante esta forma de considerar el tráfico. Una solución a este problema consiste en usar distintos canales virtuales para los flujos de tráfico a los que se deba dar trato distinto.

Nuestra propuesta es segregar el tráfico de forma que todas las conexiones que comparten un mismo canal virtual tengan el mismo requisito de distancia, independientemente de su ancho de banda medio. De esta manera no es necesario mantener ningún tipo de información adicional, ya que todas las conexiones que comparten ese canal virtual tendrán los mismos requisitos en cuanto a latencia máxima. Cuando una conexión termina se descontarán de las entradas de la tabla que estaba utilizando, el peso correspondiente a su ancho de banda. Cuando el peso de las entradas sea cero significa que ya no quedan conexiones acumuladas utilizando esa secuencia de entradas, y por tanto podrán ser liberadas.

Así pues, una vez el nodo sabe cuál es el requisito de la petición en cuanto a distancia máxima debe comprobar si hay una secuencia de entradas en la tabla que pueda ser usada para atender esa petición. Se pueden dar estos tres casos:

- Se puede utilizar alguna otra secuencia de distancia máxima igual, para la que no se haya completado aún su peso máximo. Para una secuencia de entradas de distancia máxima d, se tienen $\frac{64}{d}$ entradas de la tabla, que a razón de un peso máximo de 255 por entrada arrojan un peso máximo de $\frac{64 \times 255}{d}$ que poder acumular en las entradas de esa secuencia.

  Así pues, cada nodo tendrá una tabla de requisitos con las secuencias de entradas de la tabla que ya tiene asignadas y el ancho de banda que tienen acumulado. Si ya hay en la tabla de arbitraje una petición de la misma distancia que la solicitada, se tratará de reutilizar sus entradas de la tabla. En concreto, si el peso necesario para el ancho de banda acumulado por todas las conexiones que comparten esa secuencia más el requerido por la nueva conexión no sobrepasa el límite, esa nueva petición podrá utilizar esa secuencia de entradas y tan sólo habrá que recalcular los pesos de las entradas de la secuencia.

- Habiendo una o varias secuencias en la tabla de arbitraje con la misma distancia máxima igual a la solicitada por esta petición, ninguna de ellas puede ser utilizada. La razón es que el peso acumulado en sus entradas es tal que no permite ubicar esta nueva petición, pues se sobrepasaría el límite en cuanto a peso por entrada.

- No hay en la tabla de arbitraje ninguna secuencia de entradas para la distancia solicitada por la petición.

El primero de los casos no tiene mayor dificultad, pero los otros dos requieren de algún tipo de procedimiento más elaborado, no sólo para encontrar la nueva secuencia, sino además para hacerlo de manera eficiente.

Así pues, en ciertas situaciones es necesario encontrar una secuencia de entradas aún sin usar, que permita garantizar las exigencias de una nueva petición de conexión. Además, sería conveniente que ese método a utilizar fuera eficiente y sencillo. Eficiente en el sentido de poder elegir, de entre las posibles, la secuencia más adecuada, y sencillo para que su realización no signifique un coste que lo haga inviable.

Por otra parte, hay que tener en cuenta que la gestión de las peticiones debe ser dinámica, es decir, puede haber continuamente nuevas peticiones a situar en la tabla y peticiones que finalicen, y por tanto asignación (reserva) y liberación de entradas en la tabla en cualquier momento. Esto último puede obligar a añadir hardware adicional o mantener algún tipo de información extra, que nos permita hacer esa gestión de las peticiones ya ubicadas.

Así pues, quizás no sea posible, por su complejidad, la realización de un método óptimo, y esto suele ser habitual en muchos otros ámbitos del área de la arquitectura e ingeniería de computadores, pero sí uno mucho más simple con una eficiencia cercana a la óptima.

En el estudio que nos ocupa, un aspecto que tiene gran importancia de cara a la complejidad del método de búsqueda, es el relacionado con la separación que deben tener las entradas consecutivas que forman la secuencia de entradas. Esto es así hasta el punto de llevarnos, según la elección hecha, a métodos de búsqueda muy distintos y con muy diferentes órdenes de complejidad. Además, según el método elegido será más o menos sencilla la gestión posterior de las peticiones, es decir, la eliminación de peticiones ya ubicadas en la tabla y la inserción de otras nuevas.

Si bien es cierto que la separación entre las entradas de la secuencia viene dada por el tipo de petición, esa separación es la máxima permitida por la aplicación, pero también serían posibles distancias más pequeñas, con lo que se cumplirían sobradamente las necesidades de la aplicación. Ahora bien, en ese caso sería necesario usar un mayor número de entradas de la tabla de las estrictamente necesarias.

Pues bien, buscar una secuencia con el número mínimo de entradas (máxima distancia permitida) o hacerlo considerando un mayor número de entradas (distancia menor de la exigida) se traduce finalmente en algoritmos de búsqueda muy diferentes. Así pues, antes de plantearnos el diseño del algoritmo de búsqueda, debemos fijar cuál va a ser el criterio seguido en cuanto a la separación de las entradas de la secuencia.

3.5. Categorización de las distancias

Se tiene una tabla de arbitraje con 64 entradas que deben ser asignadas a diferentes peticiones con unos determinados requisitos de latencia. Como ya sabemos, será esta latencia la que marcará el número de entradas y la separación entre ellas.

Llamamos petición de tipo d a aquella que exige una máxima distancia[1] de d unidades entre dos entradas consecutivas en dicha tabla. Así, una petición de tipo 15 exige que entre cada dos de sus entradas haya una separación máxima de 15 entradas y una de tipo 8 exige que haya un máximo de 8 entradas de separación. En tales circunstancias, y teniendo en cuenta el carácter cíclico de la tabla de arbitraje, se necesitan un mínimo de $\lceil 64/d \rceil$ entradas de la tabla para atender una petición de tipo d.

Hay que insistir en que esta distancia máxima entre dos entradas consecutivas no es la única que puede admitir la conexión, pues cualquier separación inferior sería aceptable. Sin embargo, separaciones inferiores a la máxima podrían consumir más entradas en la tabla de las estrictamente necesarias, lo cual, en principio, no es deseable.

Según las especificaciones de InfiniBand, todo el tráfico del mismo VL va a recibir el mismo tratamiento en la tabla de arbitraje, pues dicho arbitraje se hace en base al VL de salida del puerto o interfaz. Recordemos que la asignación de un paquete a un VL u otro se hace en función del SL que viene indicado en su cabecera y por medio de la tabla *SLtoVLMappingTable*. Así pues, en función de las características del tráfico generado se le va a marcar con un SL, y en función de éste se va a ir seleccionando el VL que deberá usarse en cada conmutador de la ruta.

Obviamente, en una tabla de 64 entradas se pueden tener 64 distancias distintas, y por tanto 64 tipos de peticiones. Como el número de SLs y VLs es limitado, como mucho podrán usarse tantos tipos de peticiones como indiquen estos valores. Se sabe que el número de SLs es 16, y el número de VLs puede ser 16, 8, 4, ó 2, según las implementaciones. Así pues, será el número de canales virtuales el que impondrá el número máximo de tipos distintos de peticiones, y como consecuencia, distancias que se pueden considerar. Además, hay que considerar que se debe dedicar algún SL y VL (o varios) para el tráfico sin necesidad de calidad de servicio.

Teniendo en cuenta lo anteriormente expuesto, vamos a determinar cuál va a ser el conjunto de distancias diferentes que van a ser contempladas. Para ello, se van a considerar dos alternativas en base a criterios distintos, para elegir finalmente una de ellas. La primera consiste en agrupar las distancias que requieren el mismo número de entradas. La segunda alternativa considera la simetría para establecer los distintos tipos de peticiones.

[1] Puesto que tipo y distancia tendrán siempre el mismo valor para una petición dada, a lo largo del texto usaremos indistintamente ambos términos para referirnos al mismo concepto.

3.5.1. Categorización en base al número de entradas

Ya se ha indicado que, por el tamaño de la tabla, se pueden tener 64 tipos de peticiones distintos, pero en cualquier caso, no se podrán considerar todos, y deben ser agrupados en base a algún criterio. Ahora bien, muchos de los 64 posibles tipos de distancias requieren el mismo número de entradas en la tabla, y por tanto podrían agruparse dándoles a todos ellos el mismo trato. Por otra parte, una petición de distancia 1 necesitaría todas las entradas de la tabla, lo cual no parece razonable. Pensemos que este tipo de petición reflejaría una necesidad de latencia muy baja, lo cual hemos visto en la Sección 1.3 que no es cierto. Así pues, se podrían considerar los siguientes 14 tipos distintos de distancias máximas:

- La peticiones con bajos requisitos de latencia sólo necesitarán una única entrada de la tabla, y por tanto la distancia solicitada será 64.
- Las peticiones de distancia en el rango $[32, 63]$ requieren todas ellas 2 entradas en la tabla, y por tanto todas pueden tratarse como si fueran de distancia 32.
- Las peticiones de distancia en el rango $[22, 31]$ requieren todas 3 entradas en la tabla, con lo que pueden ser tratadas como si fueran de distancia 22.
- Las peticiones en el rango $[16, 21]$ requieren todas 4 entradas y pueden tratarse como si fueran de distancia 16.
- Las peticiones de distancia en el rango $[13, 15]$ requieren 5 entradas, con lo que no se pierde generalidad si son tratadas todas como si fueran de distancia 13.
- Las peticiones en el rango $[11, 12]$ requieren ambas 6 entradas.
- Las peticiones de distancia 10 son las únicas que requieren 7 entradas, con lo que no podrían agruparse con otras peticiones distintas sin usar más entradas de las necesarias.
- Las peticiones en el rango $[8, 9]$ requieren ambas 8 entradas.
- Las peticiones de distancia 7 son las únicas que requieren 10 entradas, con lo que no podrían agruparse con otras peticiones distintas sin usar más entradas de las necesarias.
- Lo mismo ocurre con las peticiones de distancia 6 que son las únicas que requieren 11 entradas.
- Las peticiones de distancia 5 requieren 13 entradas.
- Las peticiones de distancia 4 requieren 16 entradas.

- Las peticiones de distancia 3 requieren 22 entradas.
- Finalmente, las peticiones de distancia 2 requieren 32 entradas.

En algunos de estos casos, al convertir una distancia en otra inferior, se consigue que la separación entre dos entradas consecutivas siempre pueda ser la misma, lo que en principio parece ser beneficioso, pues simplificará la gestión posterior. En concreto, eso ocurre cuando se agrupa en una distancia que es un divisor del número total de entradas de la tabla de arbitraje. Así, las agrupaciones donde se considera distancia máxima 64, 32, 16, 8, 4 y 2, permiten mantener la misma distancia entre dos entradas consecutivas. Sin embargo, en el resto de los casos no es posible hacer que cualquier pareja de entradas consecutivas estén separadas por la misma distancia.

3.5.1.1. Elección de la secuencia

En cualquier caso, hay que seleccionar qué entradas en concreto se utilizarán para atender una determinada petición, sea ésta de la distancia que sea. Para las distancias y agrupaciones vistas anteriormente, son posibles muchos algoritmos capaces de ubicar una petición en la tabla. Hay que señalar que, en general, la única condición necesaria para poder ubicar en la tabla una petición de distancia d es que no exista una secuencia de entradas consecutivas ocupadas en la tabla, de longitud igual o mayor que d.

Siguiendo el criterio de intentar utilizar el mínimo número de entradas, como ya se ha visto anteriormente, una petición de tipo 8 necesita $\lceil 64/8 \rceil = 8$ entradas (por ejemplo las entradas 2, 10, 18, 26, 34, 42, 50, 58), mientras que una de tipo 15 necesita $\lceil 64/15 \rceil = 5$ entradas (por ejemplo, las entradas 1, 16, 31, 46, 61).

Como ya se ha indicado, nuestro objetivo es conseguir una colocación eficiente de una secuencia de peticiones en la tabla. Para ello debemos tener en cuenta tres aspectos: el tipo de petición, la cantidad de entradas necesarias y el orden en el que llegan las peticiones. Efectivamente, el orden en que llegan las peticiones también podría ser importante, pues para ubicar una petición debe tomarse una decisión en el momento en que ésta se produce con la información que se tiene en ese instante. Dependiendo de la posición elegida, puede ser que más tarde sea posible o no ubicar otras peticiones.

El tipo de petición y el número de entradas están relacionados. Las entradas correspondientes al primero de los ejemplos antes mencionados (la petición de tipo 8) se pueden elegir de manera simétrica en progresión aritmética de razón (diferencia) 8, mientras que las entradas del segundo caso (la de tipo 15) pierden la simetría en la última entrada pues la distancia de 61 a 1 (teniendo presente el carácter cíclico de la tabla de arbitraje) no es de 15 sino de 4. Para este segundo caso se tiene otra posibilidad que es distribuir las 5 entradas necesarias a lo largo de toda la tabla con una distancia inferior a 15. Así por ejemplo, una secuencia válida podría ser 1, 14, 27, 40,

53. En este caso todas las entradas tienen una separación de 13, salvo la última que sólo tiene una separación de 12. Este caso se correspondería con la agrupación de distancias mostrada anteriormente, donde las peticiones con distancias en el rango [13, 15] se podrían tratar como si fueran de distancia 13, usando 5 entradas. Evidentemente, hay más posibilidades de situar una petición de tipo 15. Por ejemplo, se podría ir variando la distancia entre dos entradas consecutivas, por supuesto siempre que ésta sea menor de 15. En este caso, otra secuencia válida podría ser 1, 14, 24, 37, 50.

En resumen, hay múltiples posibilidades para situar esta petición. Lo ideal es obtener un método que optimice el número de colocaciones, es decir, al atender una petición se debe tener en cuenta que debe quedar el mayor número de huecos[2] posible y en las posiciones más convenientes para otras peticiones posteriores. Es decir, el algoritmo que optimiza las colocaciones es aquél que siempre permite colocar en la fase siguiente la petición más restrictiva posible, entendiéndose como más restrictiva aquella petición que exija menor distancia máxima, y como consecuencia mayor número de entradas.

Como es lógico, la más restrictiva es la de tipo 1 que necesita las 64 entradas, aunque ya hemos señalado que esta petición no es demasiado realista. A continuación la petición más restrictiva es la de tipo 2 que necesita 32 entradas a distancia 2, luego la de tipo 3 que necesita 22 entradas, y así sucesivamente. Por este motivo (y sin perder generalidad) para permitir, siempre que se pueda, colocar la petición más restrictiva considerada (la de tipo 2), sin usar más entradas de la tabla de las estrictamente necesarias, puede traducirse en separar las colocaciones en pares e impares. Es decir, mientras se pueda se colocará una petición en los lugares pares dejando libre los impares para una posible entrada de tipo 2. Cuando no sea posible, se usarán los impares. Es claro que este mismo criterio se puede aplicar con peticiones de orden superior.

Por supuesto, otra posibilidad es usar más entradas de las necesarias e ir modificando las distancias entre cada dos entradas en función de las necesidades. Por ejemplo, suponiendo ocupada la entrada número 20 por una petición anterior y queriendo ubicar una petición de distancia 2, se podría utilizar la secuencia ..., 16, 18, 19, 21, 23, 25, ..., si dichas entradas están libres. Evidentemente esto aumenta el número de entradas usadas, lo cual en principio no es deseable.

3.5.1.2. Gestión de las secuencias

Como ya se ha indicado, en este modelo se pueden agrupar las distancias posibles en 14 categorías distintas. Como todo el tráfico del mismo SL va a ir por el mismo VL y por tanto recibir el mismo servicio, se va a utilizar un SL distinto para cada una de esas 14 distancias consideradas. De esta forma, aún quedan otros 2 SLs más para el

[2]Conjunto de entradas libres consecutivas.

tráfico sin garantía de calidad de servicio. Por ejemplo, se podría dedicar uno de ellos para el tráfico PBE y BE, y el otro para el tráfico CH.

Con 16 VLs se podrá utilizar un VL distinto para cada SL. Sin embargo, si hay menos de 16 VLs se debe tomar algún criterio para que tráfico de distintos SLs usen un mismo VL, y por tanto reciba el tratamiento del SL más restrictivo de todos los que comparten el VL. Esa sería otra decisión en cuanto a la agrupación de distancias máximas, y deberían considerase menos de las 14 distancias máximas ahora consideradas.

Por otra parte, otra consideración a tener en cuenta a la hora de elegir el método para ubicar las peticiones en la tabla es la gestión posterior de dichas peticiones. Como es obvio, en primer lugar necesitamos un algoritmo capaz de ubicar una petición de una determinada distancia en la tabla. Ese algoritmo debe ser rápido y capaz de hacer un buen uso de las entradas disponibles. Para ello deberá ir colocando las peticiones lo más separadas que pueda para dejar hueco a posteriores peticiones. Como ya se dijo anteriormente, la única condición necesaria para poder ubicar en la tabla una petición de distancia d es que no exista una secuencia de entradas consecutivas ocupadas en la tabla de longitud igual o mayor que d. Así pues, debería intentarse dejar siempre los mayores huecos posibles para luego poder atender otras peticiones. En concreto, lo que se debe hacer es maximizar el menor de los huecos existentes.

Por otra parte, hay que mantener la información necesaria para, cuando esa conexión termine, poder retirar de la tabla de arbitraje los pesos y entradas que se estaban usando para ella. Hay que intentar que la información necesaria para esta tarea sea mínima, pues el número de peticiones atendidas podría llegar a ser muy alto. Si se permite que la separación entre dos entradas consecutivas de la secuencia pueda variar, sería necesario almacenar la lista completa de entradas que forman cada secuencia, para poder liberarlas cuando termine la conexión. Sin embargo, si la secuencia guarda siempre la misma distancia entre dos entradas consecutivas, la tarea se simplifica, pues sólo es necesario mantener la primera entrada y la separación. Es precisamente esta característica la que da lugar a otra forma de clasificar las distancias, y por tanto las peticiones.

3.5.2. Categorización en base a la simetría

Como se ha indicado al final de la sección anterior, considerar la misma distancia entre cualquier pareja de entradas consecutivas de una secuencia puede simplificar el proceso de selección y gestión posterior de las secuencias de entradas. Al hacerlo de esta forma las entradas de la tabla serán distribuidas en progresión aritmética tomando como razón de la progresión la distancia marcada por la petición.

Las únicas progresiones aritméticas que son simétricas en una tabla de 64 (es decir, 2^6) son las que tienen por diferencia de la progresión a los divisores de 64 y al tratarse de una potencia de 2, los únicos divisores son las potencias de 2 inferiores o igual a 2^6, es decir: $2^0, 2^1, 2^2, 2^3, 2^4, 2^5$ y 2^6, esto es, entradas de tipo 1, 2, 4, 8, 16, 32 y la propia de 64, cuando no se exige ningún límite de latencia.

Con esta nueva clasificación, cualquier petición será transformada en la correspondiente a la potencia de 2 inmediatamente inferior. Es decir, una petición de distancia 11 será tratada como si fuera de distancia 8, una de distancia 45 como si fuera de distancia 32, y así sucesivamente. Analicemos las situaciones posibles, indicando a qué tipo se reduce cada una de los 64 tipos de peticiones posibles, y cuáles son las consecuencias de esa transformación:

- Todas las peticiones de distancia 64 se dejan como tal, y por tanto ocuparán una única entrada de la tabla.
- Las peticiones con distancia en el intervalo [32, 63] serán consideradas de tipo 32. Cada una de ellas será distribuida de manera uniforme y simétrica en dos entradas separadas por 32 unidades (es decir, una progresión aritmética de diferencia 32). De esta forma, al situarlas de manera simétrica a distancia 32, estamos tratando la petición como si fuera de distancia 32, y por tanto dejando entre ellas el mayor hueco posible para poder ubicar posteriormente otras peticiones. Para las peticiones en este rango, el número de entradas usadas, tanto si se consideran las distancias indicadas por el tipo de petición, como si se considera la potencia de 2, usan el mismo número de entradas de la tabla.
- Peticiones con distancia en el intervalo [16, 31]: Las peticiones de tipos comprendidos entre 16 y 21 necesitan 4 entradas y, en consecuencia, al considerarlas como si fueran de tipo 16 ocurre como en el caso anterior y no hay ninguna situación desfavorable. Sin embargo, para las peticiones desde la 22 a la 31 bastaría con 3 entradas para satisfacer su demanda y, por tanto, al considerarlas como si fueran de tipo 16, se usaría una entrada más de las necesarias. Esto ocurre a partir de ahora con los restantes tipos de petición (situación que vamos a detallar a continuación).
- Para las peticiones con distancia en el intervalo [8, 15] se tiene que:
  - Las de tipo 8 y 9 necesitan 8 entradas.
  - La de 10 necesita 7 entradas (una menos que si se trata como si fuera de distancia 8).
  - Las de 11 y 12 necesitan 6 entradas (dos menos).
  - Las de 13, 14 y 15 necesitan 5 entradas (tres menos).

Como vemos, por ejemplo, las últimas requieren sólo cinco entradas mientras que al convertirlas a tipo 8 se están usando tres entradas más de las necesarias.

- Las del intervalo [4, 7] se convierten en tipo 4, con 16 entradas, y sin embargo las realmente necesarias son:
  - 13 para las de tipo 5 (3 menos).
  - 11 para las de tipo 6 (5 menos).
  - 10 para las de tipo 7 (6 menos).
- Por último se llega a la situación más restrictiva en el intervalo [2, 3] donde las de tipo 3 (que se colocarían con sólo 22 entradas) se convierten en tipo 2 y por tanto requieren 32 entradas, es decir diez más de las necesarias.

En la Tabla 3.2 se resumen los casos en los que esta propuesta utiliza más entradas de las necesarias.

Distancia máxima necesitada	**Tratada como**	**Entradas usadas de más**
2	2	0
3	2	10
4	4	0
5	4	3
6	4	5
7	4	6
8, 9	8	0
10	8	1
11, 12	8	2
13, 14, 15	8	3
16, ..., 21	16	0
22, ..., 31	16	1
32, ..., 63	32	0
64	64	0

Tabla 3.2: Cada una de las 64 posibles distancias, la categorización llevada a cabo en base a distancias potencias de dos, y las entradas usadas de más.

Por los requisitos de latencia de las aplicaciones actuales, serán precisamente esas distancias más restrictivas las que menos se van a demandar en la práctica. De hecho, según se ha visto en la Sección 1.3, en las aplicaciones actuales suelen ser admisibles latencias del orden de decenas o centenas de milisegundos, lo cual nos llevaría a distancias mayores de 32 para la mayoría de los casos, incluso para trayectos largos que signifiquen el cruce por muchos conmutadores [Cue98]. Así pues, parece que las

distancias más utilizadas en la práctica serán precisamente aquellas para las que esta propuesta desaprovecha menos entradas, con lo que se haría un mejor uso de la tabla.

3.5.2.1. Elección de la secuencia

Como ya se ha indicado anteriormente, el que la distancia entre cualquier pareja de entradas consecutivas de una secuencia sea la misma simplifica enormemente el proceso de ubicación de la petición actual y las posteriores a ella. El que esas distancias sean potencias de 2, además, hace que su manejo resulte muy sencillo. Con ello, el algoritmo que puede ser implementado tiene una complejidad muy reducida. Una versión de este algoritmo se puede encontrar en secciones posteriores y como se podrá ver, es capaz de ubicar cualquier petición en la tabla siempre y cuando haya entradas suficientes para ello. Esto es debido a que el algoritmo va dejando las entradas libres de forma que siempre se pueda atender posteriormente a la petición más restrictiva posible.

3.5.2.2. Gestión de las secuencias

Al tener un número reducido de tipos diferentes de peticiones, con este modelo se podría tener un nivel de servicio para cada distancia utilizada (1, 2, 4, 8, 16, 32 y 64) y a su vez subdividir los niveles de servicio que se estime que van a tener más conexiones (los de mayor distancia) en varios niveles de servicio en función ahora de su ancho de banda medio. Así por ejemplo, se pueden dividir las de separación 32 en dos niveles de servicio de poco y mucho ancho de banda, y las de separación 64 en cuatro niveles de servicio (bajo, bajo-medio, medio-alto y gran ancho de banda). De esta manera se tienen 10 niveles de servicio sólo para los tráficos DBTS y DB, quedando el resto para los tráficos sin calidad de servicio (PBE, BE y CH) y de control.

Con canales virtuales suficientes se puede dedicar uno a cada nivel de servicio de los que hemos propuesto ahora, pero si no hubiera suficiente número de canales virtuales se deberían agrupar los niveles de servicio de igual distancia máxima en la tabla, en un único canal virtual. Lo que no se puede hacer es agrupar niveles de servicio de distintas distancias máximas, pues esto obligaría a necesitar otra vez las características de cada conexión para cuando se cierre una conexión del tipo más restrictivo poder cambiar la tabla, si fuera necesario, opción que ya hemos descartado por sus requisitos.

Así pues, serían necesarios un mínimo de 8 canales virtuales: 6 para las 6 distancias de DBTS y DB, otro para los tráficos PBE, BE y CH y el último para el tráfico de control. Evidentemente, si disponemos de 16 canales virtuales por puerto se podrá asignar un canal virtual a cada uno de los 10 niveles de servicio definidos anteriormente, más otro para PBE, otro para BE, otro más para CH y el último para el tráfico de control. Esto permitiría incluso distinguir dentro del tráfico PBE o BE, de cara a utilizar la totalidad de los 16 canales virtuales.

Si por el contrario hay menos de 8 canales virtuales podría considerarse utilizar menos valores para la distancia. Por ejemplo, si sólo hay 4 canales virtuales, se dedicaría uno de ellos al tráfico de control y otro para el tráfico sin necesidad de QoS. Los otros 2 canales virtuales se podrían asignar sólo a dos distancias máximas. Por ejemplo, se podría utilizar la distancia de separación 64 (para el tráfico DB y el DBTS con muy bajo requisito de latencia máxima) y la distancia 16. Las distancias menores a 16 deberían ser rechazadas pues no son admisibles, y las de distancia 32 convertirlas en distancia 16.

Otra ventaja que presenta esta propuesta es que permite realizar una gestión posterior mucho más sencilla de las peticiones, tanto cuando sea para incorporar una nueva petición en la tabla como para retirar de la misma una petición ya ubicada en ella. Al mantener cualquier par de entradas consecutivas de una secuencia la misma separación, para su posterior liberación tan sólo será necesario saber cuál es la primera entrada de la secuencia y la separación entre ellas.

3.5.3. Comparación de los dos modelos

Expuestas las dos alternativas para la categorización de las distancias, se han puesto de manifiesto las importantes ventajas de la segunda de ellas, la basada en la simetría. Su principal inconveniente es que, para algunos tipos de petición, usa más entradas de las necesarias, y como consecuencia podría, en principio, admitir un número menor de peticiones. En esta sección se va a analizar cuál es el impacto real en este sentido, comparándolo además con la otra propuesta. Esta comparación servirá para elegir finalmente uno de ellos.

Así pues, como se ha indicado, con la propuesta de agrupar en base a las potencias de 2, en algunos casos se usan más entradas de la tabla de las que serían estrictamente necesarias. Sin embargo, el otro modelo también puede que use más entradas de las necesarias por las colisiones entre peticiones.

Para el primer modelo, se ha implementado una versión que es capaz de ubicar cualquier petición de distancia d siempre y cuando no haya en la tabla una secuencia de entradas ocupadas mayor que d. Esto puede suponer que en algunos casos se deba modificar la distancia entre dos entradas consecutivas para ir amoldándose a los huecos existentes, aunque por supuesto siempre respetando la distancia máxima. Los dos criterios que sigue este algoritmo a la hora de seleccionar las entradas son:

- La secuencia elegida es tal que deja en la tabla el menor grupo de entradas consecutivas ocupadas, permitiendo de esta manera que se pueda atender posteriormente a la petición más restrictiva posible.

- En caso de que haya varias secuencias que cumplan el criterio anterior, se elige aquella que maximiza el menor de los grupos de entradas libres consecutivas.

Evidentemente este algoritmo se acerca al óptimo, por lo menos en cuanto a capacidad para ubicar peticiones en la tabla. Siempre que haya entradas libres suficientes y no haya una secuencia de entradas ocupadas mayor que la distancia requerida, el algoritmo selecciona una secuencia válida. También es evidente que en algunos casos este algoritmo deberá utilizar más entradas de las estrictamente necesarias. Efectivamente, cuando a partir de la primera entrada libre se van seleccionando las demás, si la siguiente coincide con una entrada ya ocupada, se intenta usar la anterior. Si ésta también está ocupada lo intenta con la anterior, y así sucesivamente.

Se ha implementado también el algoritmo que reduce cualquier petición a la distancia potencia de 2 menor más próxima, y que es detalladamente descrito en la Sección 4.3 del capítulo siguiente.

Para ambos algoritmos se han generado secuencias de peticiones que se han intentado ubicar en la tabla, hasta que se ha conseguido llenar. Esas peticiones tienen un determinado requisito en cuanto a distancia máxima, que se ha generado aleatoriamente entre 2 y 64, en principio todas con la misma probabilidad. Si una petición no puede ubicarse en la tabla se desestima y se genera otra. En primer lugar estamos interesados en ver cuántas entradas se desperdician en ambos casos. Es decir, ambos algoritmos utilizan más entradas de la tabla de las estrictamente necesarias: el de potencias de 2 por los redondeos y el otro por usar a veces distancias menores a las necesarias por estar ocupada la entrada que se quería usar.

Según los resultados obtenidos, el método de redondear a las potencias de 2 desperdicia en media $8,78$ entradas, mientras que el algoritmo supuestamente óptimo desperdicia en media $1,96$ entradas. Esto hace una diferencia entre ambos métodos de casi 7 entradas, lo cual es bastante considerable.

Como ya se ha comentado anteriormente, se han considerado todas las peticiones igualmente probables, cuando sabemos que esto no es verdad. Es difícil calcular la probabilidad real de cada una de las distancias, pues eso dependerá de las aplicaciones, pero también de aspectos de la propia red (tamaño, diámetro, tamaño de paquete, etc.) y de los conmutadores (tipo de crossbar, número de puertos y canales virtuales, etc.). Como una sencilla aproximación se ha considerado una probabilidad para cada distancia proporcional a ella. De esta forma una petición de distancia 64 es el doble de probable que una de 32 y 32 veces más probable que una petición de distancia 2. Con estas condiciones se ha vuelto a realizar la misma prueba obteniendo en este caso que con el método de redondear a potencias de 2 se desperdician en media $5,68$ entradas. Por otra parte, considerando todas las distancias (agrupadas en las 14 categorías estudiadas anteriormente) ahora se desperdician $0,86$ entradas. De esta manera la diferencia relativa entre ellos es ahora de 5 entradas.

Es fácil ver que en los casos reales estas probabilidades serán todavía más desproporcionadas, dándose con mucha más frecuencia las distancias grandes que las pequeñas. Es más, para redes de tamaño pequeño o mediano, es bastante probable que no se lleguen a dar distancias menores de 16. De esta forma la propuesta de redondear a potencias de 2 apenas desperdiciaría entradas de la tabla, teniendo por contra las ventajas detalladas anteriormente.

Otro aspecto a tener en cuenta es la complejidad del algoritmo de reserva en la tabla. Obviamente, la decisión sobre la secuencia concreta a seleccionar debe basarse en la información que el algoritmo tiene cuando se produce la petición. En ese momento no es posible conocer cuál será la secuencia de peticiones que vendrán a continuación. Así, el primero de los modelos propuestos podrá atender a peticiones posteriores en función de las decisiones previas que se hayan tomado. Pero, evidentemente, no se puede saber qué peticiones se producirán a continuación.

A modo de ejemplo, se va a estudiar la secuencia de peticiones de distancia 45, 8, 53, 61, 60, 55, 24, 3 y 9. Cada petición se sitúa en la tabla de acuerdo a las reglas indicadas anteriormente para el primer método. Como punto de partida, la primera petición (la de distancia 45) se sitúa en las entradas 32 y 64, teniendo pues una separación de 32. Las otras peticiones se sitúan en la tabla con las mismas reglas de forma sucesiva. La tabla final en este caso se muestra en la Figura 3.5(a). Se puede observar como la última petición (la de distancia 9) no puede ubicarse en la tabla porque hay una secuencia de entradas consecutivas ocupadas mayor de 8. En concreto, para la situación mostrada en esa figura, se tienen las secuencias 28, 29, ..., 36 y 63, 64, 0, 1, ..., 8 que tienen una longitud igual o mayor que la distancia solicitada. Así pues, la última petición (la de distancia 9) no puede ubicarse en la tabla, aunque aún hay suficientes entradas libres para ello (quedan 17 entradas libres). Esto es debido a que las entradas libres están situadas con una separación incorrecta entre dos de ellas consecutivas. Una posible solución sería mover las peticiones previamente establecidas en la tabla de cara a hacer hueco a otras nuevas.

Sin embargo, si se supiera la secuencia completa de peticiones que se van a producir, quizás se podrían situar las peticiones en la tabla de manera adecuada. En este caso, una posible situación final, con todas las peticiones situadas en la tabla, sería la mostrada en la Figura 3.5(b).

Por otra parte, utilizando el algoritmo que convierte las peticiones en la correspondiente potencia de 2, la secuencia de peticiones anterior también puede ser ubicada en la tabla. En concreto, con el algoritmo de reserva en la tabla que se presentará en la Sección 4.3, el estado final de la tabla sería el mostrado en la Figura 3.5(c). En dicha figura se ha representado también la conversión realizada para cada distancia solicitada. En este caso, aún restan 2 entradas libres en la tabla (las entradas 19 y 51) con una distancia entre ellas de 32, y por tanto siendo posible utilizarlas para atender posteriormente una petición que sea convertida en esa distancia. Así pues, a pesar de

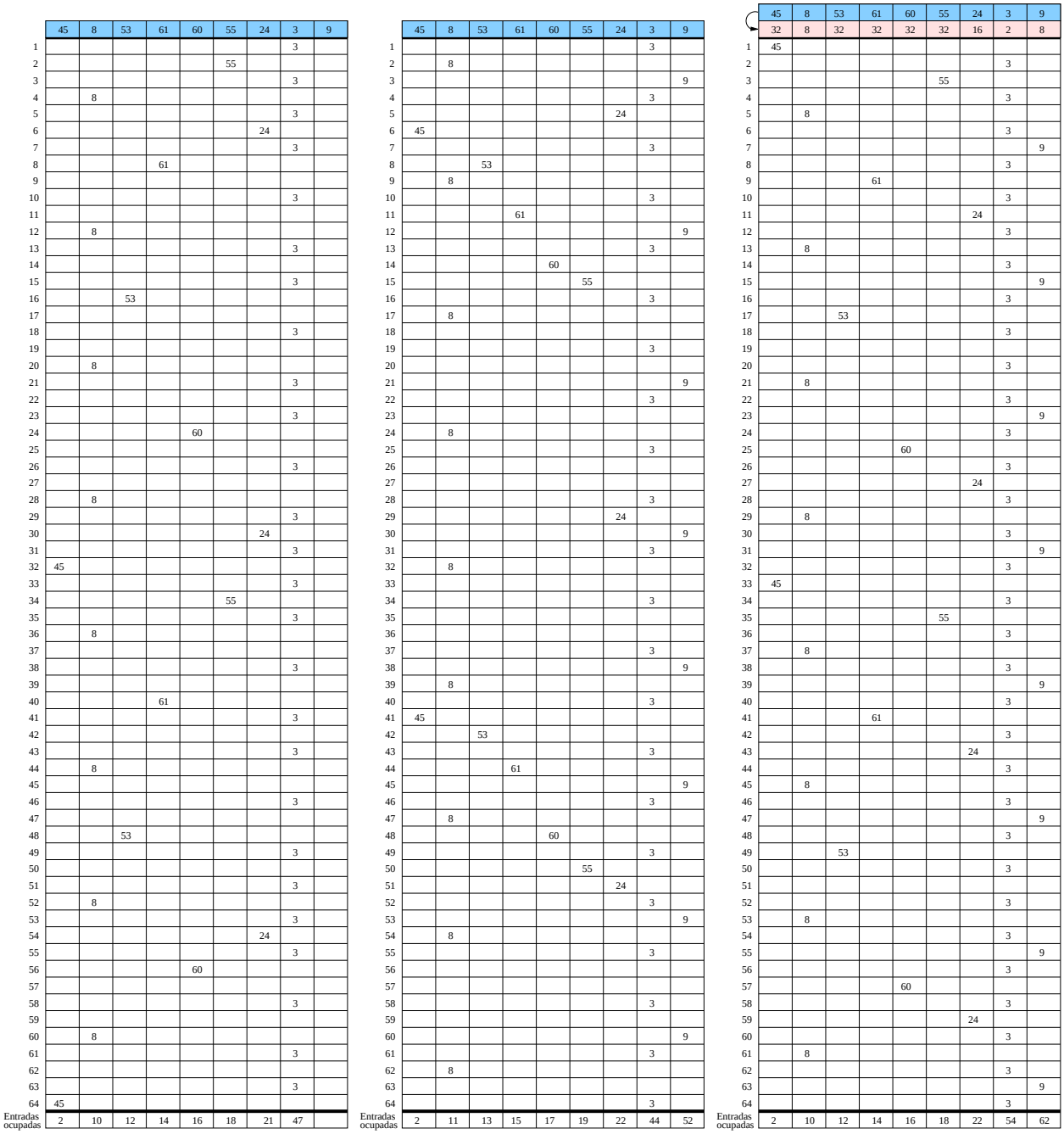


Figura 3.5: Posibles asignaciones de entradas para la secuencia 45, 8, 53, 61, 60, 55, 24, 3 y 9 usando (a) y (b) el primero de los métodos propuesto, y (c) el método de convertir a potencias de 2.

que este algoritmo usa más entradas de las estrictamente necesarias, va dejando siempre las entradas libres de forma que se pueda atender posteriormente la petición más

restrictiva posible. De esta forma, una petición podrá ubicarse en la tabla si y sólo si hay entradas suficientes para ello.

Por todo lo expuesto, nuestra propuesta en cuanto al tratamiento de la latencia máxima es convertirla en distancia máxima entre dos entradas consecutivas en la tabla de arbitraje del puerto de salida, y redondear esa distancia máxima a las potencias de dos. En lo que sigue vamos a considerar siempre este criterio a la hora de tratar las distancias.

3.6. Resumen

A lo largo de este capítulo se ha estudiado cómo configurar los mecanismos que proporciona InfiniBand para conseguir que las aplicaciones tengan garantía de QoS. En concreto, nos hemos centrado en cómo configurar la tabla de arbitraje para que se vaya haciendo una correcta planificación de los distintos flujos de datos, de forma que cada uno de ellos consiga la QoS que se le garantizó en el momento del establecimiento de la conexión.

Se ha abordado en profundidad cómo configurar las tablas de arbitraje de los puertos de salida para garantizar ancho de banda a las aplicaciones. A continuación se ha realizado un estudio similar pero para garantizar a cualquier conexión una latencia máxima extremo a extremo. Para ello, se han obtenido unas expresiones para cuantificar el tiempo máximo que un paquete puede tardar en cruzar un conmutador. Se ha visto que esa expresión depende principalmente del tipo de conmutador, del número de puertos y del número de canales virtuales por puerto. Se ha concluido que en cada conmutador la latencia a garantizar puede convertirse en una determinada distancia máxima entre dos entradas consecutivas de la secuencia de entradas de la tabla de arbitraje a utilizar por esa conexión.

Una vez visto por separado cómo garantizar ancho de banda y latencia, se ha abordado el problema de dar garantía conjunta de ambos tipos de requisitos. Se ha comprobado que el caso de garantizar solamente ancho de banda no deja ser un caso particular de garantizar ancho de banda y latencia, pero con un requisito de latencia suficientemente grande para no necesitar más de una entrada en la tabla de arbitraje. De esta forma, ambos tipos de requisitos pueden ser tratados de la misma manera.

En la última parte de este capítulo se ha estudiado la forma de categorizar la distancia que se acabará necesitando debido a la solicitud de una conexión. Se ha visto que no es posible considerar los 64 tipos posibles de peticiones y se han estudiado dos posibles alternativas. La primera de ellas está basada en agrupar peticiones en función del número de entradas que se necesitan en cada caso. La segunda alternativa consiste en categorizar la distancia en potencias de 2. Se han comparado los aspectos

generales de ambas propuestas, así como los relativos a la forma de elegir la secuencia concreta y los aspectos de gestión posterior de las peticiones. Analizadas las ventajas e inconvenientes de cada propuesta, cuantitativa y cualitativamente, se ha elegido como método para categorizar las distancias el segundo de ellos, esto es, el que convierte cualquier petición en una potencia de 2.

Capítulo 4

Modelo formal de la tabla de arbitraje. Inserciones

Como se ha indicado en capítulos anteriores, los agentes locales o el Subnet Manager, recibirán peticiones de conexión y también serán informados de la finalización de conexiones. Esto provocará que se deban hacer nuevas inserciones en varias tablas de arbitraje, así como eliminar peticiones previamente establecidas. A lo largo de este capítulo y del siguiente se va a desarrollar un modelo para el tratamiento de las tablas de arbitraje de InfiniBand.

Con el propósito de hacer más clara la descripción de este modelo, se han organizado todos los aspectos formales del mismo en dos capítulos. Así, en este capítulo se incluye todo lo que tiene que ver con las solicitudes de conexión, y por tanto la asignación o reserva de entradas de la tabla de arbitraje. Se considerará para ello que no hay eliminación de peticiones, es decir, en este capítulo no se contempla la posibilidad de que una petición ya ubicada en la tabla sea eliminada de ella. En el siguiente capítulo se realizará un tratamiento completo de la tabla de arbitraje, incluyendo las situaciones originadas por la liberación de entradas de la tabla, como consecuencia de la finalización de conexiones.

Hay que insistir en que, a lo largo de estos dos capítulos, se considerará que una petición de conexión se traducirá en la solicitud de un nuevo grupo de entradas con una separación dada entre dos entradas consecutivas, la cual viene dada por la propia petición. Ya se analizó en el capítulo anterior otra situación bien distinta en la que una petición puede ubicarse en una secuencia de entradas ya utilizadas por otras peticiones previas. Evidentemente, esta opción no presenta dificultad, y ya fue estudiada en el capítulo anterior.

En este capítulo se incluye un conjunto de definiciones e hipótesis de partida que establecen el marco en el que se plantea el modelo. Se incluye así mismo un algoritmo para buscar un grupo de entradas para una petición dada, y se verifica su correcto

funcionamiento a través de varios teoremas, que se apoyan en diversas propiedades también adecuadamente enunciadas y verificadas. Con el objeto de que los aspectos formales que rodean a toda esta parte del modelo sean mejor y más rápidamente comprendidos, se incluye en primer lugar una descripción informal del modelo en la que se hace uso de ejemplos y representaciones gráficas.

4.1. Descripción informal

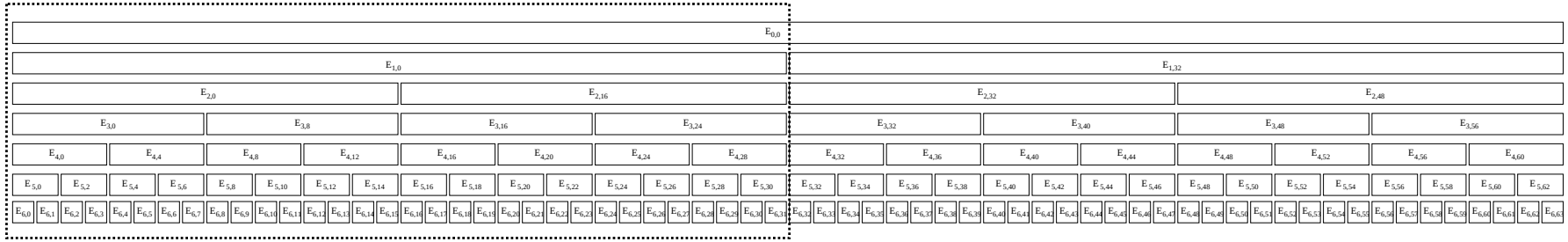
En las siguientes secciones se va a presentar una formalización de la gestión de la tabla de arbitraje de InfiniBand. Con ello se pretende demostrar una serie de propiedades de las entradas de la tabla de arbitraje y proponer un algoritmo para la reserva eficiente de dichas entradas. Antes de ello, en esta sección, y de una manera informal, se van a introducir los conceptos que más tarde serán formalizados. La idea es introducir de forma sencilla las ideas que acabarán dando forma al marco completo de trabajo.

La tabla de arbitraje está compuesta por una serie de entradas. De cara a simplificar el posterior tratamiento del modelo realizado, se va a utilizar una numeración de las entradas de la tabla un tanto peculiar. Como se verá más adelante, dicha numeración está basada en la aplicación de la permutación bit-reversal a la típica numeración correlativa. Como veremos, esta nueva numeración simplificará mucho los algoritmos a utilizar y todo su tratamiento posterior.

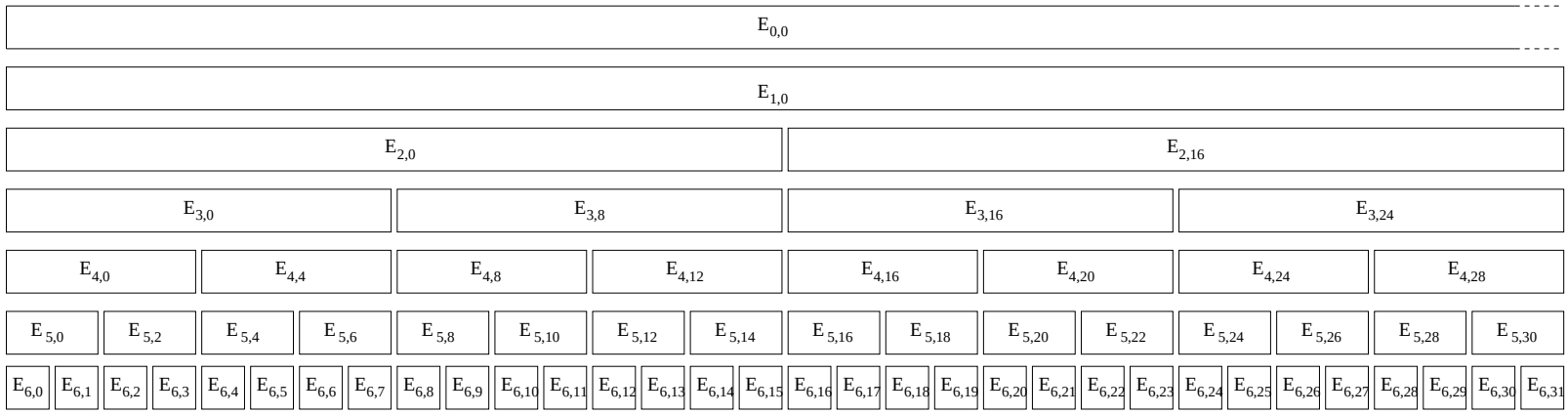
En la tabla de arbitraje se pueden distinguir diferentes secuencias de entradas en función de la distancia entre dos entradas consecutivas de la secuencia. Cada una de estas secuencias, con las que se puede atender una determinada petición, constituirá un conjunto. Como veremos, hay conjuntos de distinto tamaño (cardinal), y en función de dicho tamaño se tiene un número distinto de conjuntos. Esta situación se plantea gráficamente en la Figura 4.1, en la que se puede apreciar que todos los conjuntos forman una estructura de árbol binario con siete niveles. Como se verá más adelante, la numeración utilizada para identificar a cada uno de los conjuntos depende de las entradas que contiene dicho conjunto y del nivel del árbol en que está situado.

Un cuadrado en el nivel inferior de la Figura 4.1 representa un conjunto con una única entrada de la tabla. Un rectángulo en un nivel cualquiera representa un conjunto con varias entradas, que son precisamente aquellas contenidas en los cuadrados situados debajo de dicho rectángulo. Las entradas contenidas en cada rectángulo están equiespaciadas en la tabla de arbitraje.

Cada una de las secuencias puede dividirse en dos secuencias, cada una de ellas con la mitad de entradas. Esto significa que si con una determinada secuencia se puede atender una petición que requiera n entradas de la tabla, con las dos secuencias en que se puede dividir ésta se pueden atender dos peticiones que requieran $\frac{n}{2}$ entradas.



(a)



(b)

Figura 4.1: Descomposición en forma de árbol binario de todos los conjuntos posibles de entradas de la tabla, donde cada nivel se corresponde con un tipo de petición distinto: (a) el árbol completo, (b) ampliación de la mitad marcada en (a).

En términos de conjuntos, podemos dividir un conjunto en dos subconjuntos disjuntos (cada una de las secuencias) con la mitad de entradas cada uno. De esta manera, y con todos los conjuntos posibles, se establece, como antes indicábamos, un árbol binario con raíz el conjunto formado por el total de entradas, donde en cada nivel cada conjunto se divide en dos, pertenecientes al nivel inferior. Por simplicidad, diremos que un conjunto es el padre de los dos conjuntos que tiene inmediatamente debajo de él en dicha figura, los cuales diremos que son hijos del padre y hermanos entre sí.

Por otra parte, se dice que un conjunto está libre si todas sus entradas están sin usar. Además, decimos que un conjunto es singular si está libre, pero su hermano (hijo del mismo padre) está ocupado, pues sus entradas se están usando para atender peticiones. En la Figura 4.2 se ha representado una parte del árbol de la Figura 4.1. Se han sombreado los conjuntos que están libres, estando el resto ocupados. En esta figura hay cuatro conjuntos singulares que son $E_{4,4}$, $E_{3,16}$, $E_{3,32}$ y $E_{2,48}$, los cuales se han sombreado en tono más oscuro. Hay que señalar que los restantes conjuntos libres, sombreados con un color más claro, están incluidos en aquellos. Por ejemplo, por estar libre el conjunto $E_{2,48}$, lo están también los conjuntos $E_{3,48}$ y $E_{3,56}$, y también los hijos de éstos en los niveles inferiores ($E_{4,48}$, $E_{4,52}$, $E_{4,56}$ y $E_{4,60}$). Sin embargo, ninguno de ellos es singular pues sus respectivos hermanos están libres.

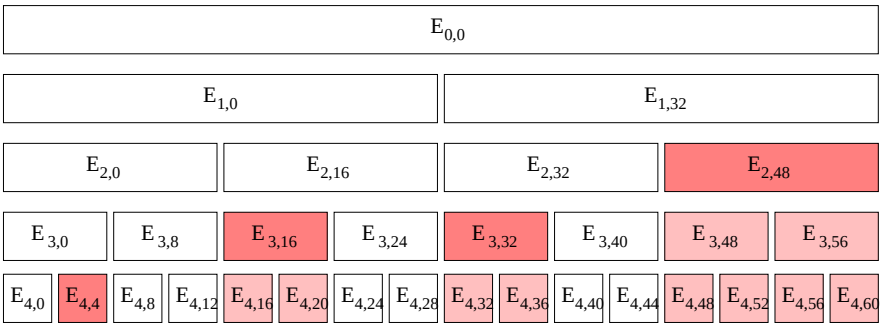


Figura 4.2: Tabla con cuatro conjuntos singulares.

En esta misma línea, se dirá que una tabla está normalizada si a lo sumo hay un conjunto singular por nivel. Es decir, en cada nivel del árbol puede que no haya ningún conjunto singular, pero si hay alguno será único. Así por ejemplo, la tabla representada por el árbol de la Figura 4.2 no está normalizada pues en el nivel 3 hay dos conjuntos singulares. Un ejemplo de tabla normalizada se muestra en la Figura 4.3. En esa figura, son conjuntos singulares $E_{4,4}$, $E_{2,16}$ y $E_{3,32}$, y como se ve, no hay ningún nivel en el que haya más de un conjunto singular.

Otro concepto que se va a utilizar es el de tabla ordenada. Se dirá que una tabla está ordenada si los conjuntos singulares están ordenados de menor a mayor tamaño de izquierda a derecha. Así por ejemplo, la tabla representada en el árbol de la Figura 4.3 no está ordenada, pues aunque el conjunto $E_{4,4}$ sí que está situado más a la izquierda

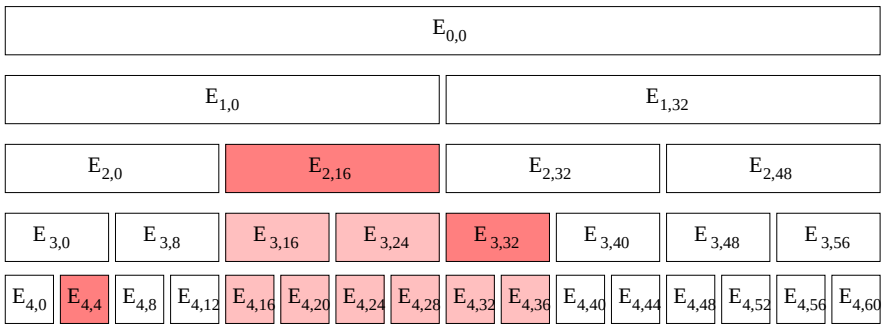


Figura 4.3: Tabla normalizada con tres conjuntos singulares.

que el conjunto $E_{2,16}$, sin embargo, éste está a la izquierda de $E_{3,32}$, siendo este último de menor tamaño.

Sin embargo, la tabla representada en el árbol de la Figura 4.2 sí está ordenada, pues los conjuntos singulares más pequeños siempre están situados a la izquierda de los conjuntos singulares de mayor tamaño. Como se puede ver, los conceptos de tabla normalizada y tabla ordenada son independientes, pudiendo una tabla estar normalizada y sin embargo no estar ordenada, y viceversa. En la Figura 4.4 se incluye la representación de una tabla ordenada y normalizada.

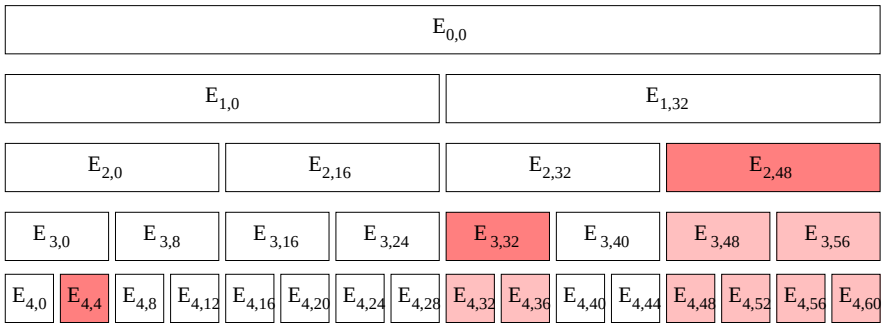


Figura 4.4: Tabla ordenada y normalizada con tres conjuntos singulares.

En este marco, y haciendo uso de las definiciones previas, en esta tesis se propone un algoritmo para encontrar una nueva secuencia de entradas libres para ubicar una petición dada. Este algoritmo, en los términos en los que se plantea, hace un buen uso de la tabla. El algoritmo siempre es capaz de atender una petición, es decir, de encontrar la secuencia de entradas necesarias si hay un número suficiente de ellas libres. Para ello, este algoritmo deja la tabla siempre en condiciones adecuadas para poder atender la petición más restrictiva, es decir, la de menor distancia posible según el número de entradas libres que hay en la tabla. Este hecho es debidamente demostrado, así como otros que son consecuencia del funcionamiento del algoritmo. Por ejemplo, se demostrará que después de aplicar el algoritmo de reserva la tabla siempre está normalizada y ordenada.

Con todas estas ideas, se pasa ahora a plantear el modelo formal de la tabla de arbitraje. Para ello se parte de unas definiciones e hipótesis previas, para exponer posteriormente el modelo a utilizar para la tabla de arbitraje, y algunas propiedades inmediatas que pueden extraerse. A continuación se enunciará el algoritmo de reserva de la tabla y se probarán sus características a través de diversos teoremas.

4.2. Modelo formal de la tabla de arbitraje

En base a lo expuesto en capítulos anteriores, en las siguientes secciones se presenta un algoritmo para encontrar una nueva secuencia de entradas libres en la tabla de arbitraje capaz de atender una determinada petición de conexión. El número de entradas a reservar en la tabla viene dado por la separación entre ellas, la cual depende a su vez de las características de las conexiones solicitadas.

El tratamiento del problema que a continuación se hace consiste, básicamente, en plantear un algoritmo que seleccione un conjunto de entradas libres de la tabla de arbitraje ante una solicitud o petición de conexión, la cual indicará la separación máxima que deben tener dichas entradas. Para su desarrollo se partirá de una serie de hipótesis y definiciones previas, con la intención de establecer el marco adecuado para ello.

Hay que indicar que aunque el algoritmo en particular, y todo el tratamiento en general, se centran en un marco concreto como es InfiniBand, las características del problema abordado lo pueden hacer aplicable a muchos otros entornos, en la medida en que éste se reduzca a buscar, y en su caso seleccionar, una secuencia de entradas con una determinada separación entre ellas en una tabla de un cierto tamaño. Este carácter general se consigue sin más que no mencionar las palabras *arbitraje*, al referirnos a la tabla, y *conexión*, para referirnos al origen de las peticiones de las entradas. Esto, unido al deseo de no ser excesivamente prolijos en las explicaciones, nos ha llevado a elegir esa opción, sin olvidar, eso sí, el entorno en el que nos movemos. Bastará, para volver a él, tener en mente que las peticiones son originadas por conexiones, con el fin de que sean garantizados unos determinados requisitos, y que el grupo de entradas que acabarán asegurando esos requisitos pertenecen a la tabla de arbitraje asociada a los puertos de salida de los conmutadores propuestos por InfiniBand.

4.2.1. Definiciones

Para el posterior tratamiento se definen los siguientes conceptos:

- *Tabla*: lista circular de 64 entradas.
- *Entrada*: cada una de las 64 partes en las que puede dividirse una tabla.

- *Peso*: valor numérico que tiene asignado cada una de las entradas de una tabla. Puede variar entre 0 y 255.
- *Estado de una entrada*: situación de una entrada de la tabla. Los estados posibles de una entrada son libre (peso=0) u ocupada (peso≠0).
- *Petición*: solicitud de un número determinado de entradas.
- *Distancia*: separación máxima que puede existir entre dos entradas consecutivas de la tabla asignadas a una petición.
- *Tipo de petición*: cada una de las diferentes clases en las que se pueden agrupar las peticiones, y que vienen dadas por las distancias solicitadas.
- *Grupo o secuencia de entradas*: conjunto de entradas de una tabla separadas entre sí por una determinada distancia. Para caracterizar una secuencia de entradas será suficiente con indicar la primera de ellas y la distancia que las separa.

4.2.2. Hipótesis de partida

En lo que sigue, y mientras no se indique lo contrario, se van a considerar las siguientes hipótesis de partida:

1. No hay eliminación de peticiones, es decir, las entradas de la tabla se van reservando según se van recibiendo peticiones, y éstas no desaparecen nunca. Dicho de otra forma, las entradas pueden pasar del estado libre al estado ocupado, pero no de ocupado a libre. Esta hipótesis se eliminará en el capítulo siguiente.
2. Pueden ser necesarios más de un grupo de entradas para un conjunto de peticiones del mismo tipo.
3. El peso total asociado a una petición es repartido entre las entradas de la secuencia seleccionada de tal modo que el peso de la primera entrada de dicha secuencia siempre es mayor o igual que los pesos del resto de entradas de la secuencia.
4. La distancia d asociada a una petición siempre será potencia de 2 y cumplirá que $1 \leq d \leq 64$. (Estos mismos serán los diferentes tipos de peticiones que se van a considerar, es decir, en lo que sigue *distancia* y *tipo de petición* serán términos equivalentes). De otra forma, la distancia máxima d solicitada por una petición será $d = 2^i$ con $i = 0, 1, 2, \ldots, 6$. Hay que indicar en este punto que, aunque en el tratamiento del problema se van a considerar esas distancias, en las pruebas posteriores se desestimarán algunas de ellas, en base al estudio de los requisitos de las aplicaciones actuales con necesidades de calidad de servicio. Es el caso, por ejemplo, de la distancia $d = 1$, la cual necesitaría toda la tabla.

4.2.3. Modelo

Dada una tabla T, cada una de sus entradas será identificada mediante un identificador t_i, de tal forma que se tienen 64 identificadores distintos t_0, t_1, $\cdots$, t_{62}, t_{63}. De acuerdo con las definiciones anteriores, cada t_i tiene un peso asociado w_i cuyo valor puede variar entre 0 y 255. Diremos que una entrada t_i está libre si $w_i = 0$, y que está ocupada en caso contrario.

Por conveniencia, y sin que ello suponga ninguna pérdida de generalidad, entradas consecutivas de la tabla no tendrán identificadores con índices consecutivos. La asignación de identificadores a las entradas está basada en la aplicación de la permutación bit-reversal a la que sería la numeración habitual, y que asignaría identificadores con índices consecutivos a entradas consecutivas. Recordemos que la permutación bit-reversal es tal que siendo $m = b_{n-1}b_{n-2}\ldots b_1b_0$, $\Re(m) = b_0b_1\ldots b_{n-2}b_{n-1}$.

Para el caso de una tabla de 8 entradas, la Figura 4.5(a) muestra una numeración correlativa, mientras que la Figura 4.5(b) muestra la nueva numeración de las entradas.

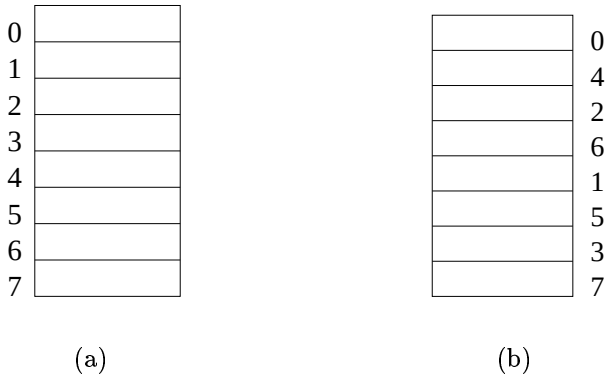


Figura 4.5: Tabla de 8 entradas, (a) numeración correlativa de las entradas, (b) numeración basada en la aplicación de la permutación bit-reversal.

Se puede observar en la Figura 4.5(b) que las dos únicas secuencias de entradas con las que se puede atender una petición de distancia máxima 2 son $\{0, 2, 1, 3\}$ y $\{4, 6, 5, 7\}$. Puesto que entre los elementos de un conjunto no hay ningún orden establecido[1] los dos conjuntos anteriores son los mismos que $\{0, 1, 2, 3\}$ y $\{4, 5, 6, 7\}$, respectivamente. Como se ve, las entradas de un conjunto tienen una numeración consecutiva. Este es el motivo por el que se utiliza la numeración propuesta. Algo similar sucede para otras distancias.

La siguiente definición de conjuntos de entradas usa la nueva numeración. Cada conjunto contiene las entradas necesarias para poder atender una petición de una distancia dada.

[1]Un conjunto es un grupo de elementos no repetidos ni ordenados.

Definición 1 *Dada una tabla T, para una petición cualquiera de tipo $d = 2^i$, con $0 \leq i \leq 6$, se definen los conjuntos $E_{i,j}$ con $j = k \times 2^{6-i}$ y $0 \leq k < 2^i$, como*

$$E_{i,j} = \{t_n \mid \quad j \leq n < j + 2^{6-i}\}$$

Por conveniencia, se define el conjunto $I_i = \{j \mid j = k \times 2^{6-i} \text{ con } 0 \leq k < 2^i\}$ que está formado por los posibles valores de j para un i dado. Así, cuando se tenga que hacer referencia a este índice j se indicará de la siguiente forma: $j \in I_i$.

Para la tabla de 8 entradas de la Figura 4.5(b), se tendrían los siguientes conjuntos:

- Con entradas de la tabla a distancia 1: $E_{0,0} = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$.
- Con entradas de la tabla a distancia 2: $E_{1,0} = \{t_0, t_1, t_2, t_3\}$ y $E_{1,4} = \{t_4, t_5, t_6, t_7\}$.
- Con entradas de la tabla a distancia 4: $E_{2,0} = \{t_0, t_1\}$, $E_{2,2} = \{t_2, t_3\}$, $E_{2,4} = \{t_4, t_5\}$ y $E_{2,6} = \{t_6, t_7\}$.
- Con entradas de la tabla a distancia 8: $E_{3,0} = \{t_0\}$, $E_{3,1} = \{t_1\}$, $E_{3,2} = \{t_2\}$, $E_{3,3} = \{t_3\}$, $E_{3,4} = \{t_4\}$, $E_{3,5} = \{t_5\}$, $E_{3,6} = \{t_6\}$ y $E_{3,7} = \{t_7\}$.

Hay que señalar que los conjuntos para una misma distancia son disjuntos entre sí, teniendo cada uno de ellos entradas suficientes para atender una petición de la distancia requerida. Con todos los conjuntos puede establecerse una estructura de árbol binario, donde un conjunto de un determinado nivel i es la unión de dos conjuntos disjuntos pertenecientes al nivel $i + 1$. Para la tabla de 8 entradas de la Figura 4.5(b), este árbol es el mostrado en la Figura 4.6.

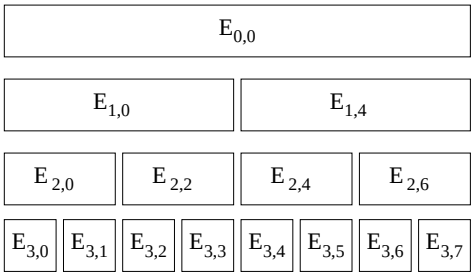


Figura 4.6: Árbol de los conjuntos para la tabla de 8 entradas de la Figura 4.5(b).

En general, para una tabla de 64 entradas, cada $E_{i,j}$ contiene las entradas, separadas entre sí una distancia $d = 2^i$, con las que se puede atender una petición de tipo d. En este caso, el árbol binario con todos los conjuntos es el mostrado en la Figura 4.1. La raíz del árbol es el conjunto $E_{0,0}$. Dicho conjunto es la unión de los conjuntos $E_{1,0}$ y $E_{1,32}$, que son los conjuntos raíz de sus dos subárboles. Estos a su vez se dividen en dos conjuntos, y así de forma sucesiva.

Aunque la terminología básica utilizada en el resto de este trabajo será la basada en conjuntos, en numerosos casos, y con el propósito de clarificar aún más, si cabe, los conceptos introducidos, también se hará uso de términos relacionados con árboles binarios.

Los conjuntos $E_{1,0}$ y $E_{1,32}$ son los únicos disponibles para una distancia de tipo $d = 2$. Estos conjuntos tienen las siguientes entradas:

$$E_{1,0} = \{t_0 t_1 t_2 t_3 \ldots t_{30} t_{31}\} \quad \text{y} \quad E_{1,32} = \{t_{32} t_{33} t_{34} t_{35} \ldots t_{62} t_{63}\}$$

esto es, un conjunto con la primera mitad de las entradas y otro con la segunda mitad. Hay que señalar que esta distribución corresponde a la numeración de las entradas usada, que se correspondería con dividir las entradas entre las entradas pares y las impares con la numeración correlativa. Así pues, cada uno de estos conjuntos tiene una separación de 2 entre dos entradas consecutivas, y por tanto cada uno de ellos contiene las entradas suficientes y con la separación adecuada para atender una petición de tipo 2. Para el resto de tipos de peticiones se pueden igualmente obtener los correspondientes conjuntos $E_{i,j}$ (Figura 4.1).

Definición 2 *Un conjunto $E_{i,j}$, con $0 \leq i \leq 6$ y $j \in I_i$, se dice que está libre si $w_k = 0$ $\forall\ t_k \in E_{i,j}$.*

Esto, unido a la propia definición de $E_{i,j}$ (Definición 1), implica que para atender a una petición de tipo $d = 2^i$ se necesita un $E_{i,j}$ libre[2].

A partir de las definiciones anteriores, aparecen las siguientes propiedades:

Propiedad 1 *Para un i y j dados con $0 \leq i < 6$ y $j \in I_i$*

$$E_{i,j} = E_{i+1,j} \cup E_{i+1,j+2^{6-(i+1)}}$$

En términos de árboles binarios, esta propiedad podría enunciarse como *"la unión de los hijos es el padre"*.

Demostración: Por la Definición 1,

[2]Hay que recordar que esto es así pues en este estudio, como se ha indicado en la Sección 3, se va a considerar sólo el caso en que una nueva petición necesita una nueva secuencia de entradas libres.

$$\begin{aligned}
E_{i,j} &= \left\{t_n \mid \quad j \leq n < j + 2^{6-i}\right\} = \\
&= \left\{t_j, t_{j+1}, \ldots, t_{j+2^{6-(i+1)}-1}, t_{j+2^{6-(i+1)}}, t_{j+2^{6-(i+1)}+1}, \ldots, t_{j+2^{6-i}-1}\right\} = \\
&= \left\{t_j, t_{j+1}, \ldots, t_{j+2^{6-(i+1)}-1}\right\} \cup \left\{t_{j+2^{6-(i+1)}}, t_{j+2^{6-(i+1)}+1}, \ldots, t_{j+2^{6-i}-1}\right\} = \\
&= \left\{t_n \mid \quad j \leq n < j + 2^{6-(i+1)}\right\} \cup \left\{t_m \mid \quad j + 2^{6-(i+1)} \leq m < j + 2^{6-i}\right\} = \\
&= E_{i+1,j} \cup E_{i+1,j+2^{6-(i+1)}}
\end{aligned}$$

□

Como consecuencia de esta propiedad, con las entradas que permiten atender una petición de tipo $d = 2^i$ se pueden atender dos peticiones de tipo $d' = 2^{i+1}$.

De la Definición 1, dado un nivel i, los $E_{i,j}$, con $0 < i \leq 6$ y $j \in I_i$, son aquellos que se obtienen con $j = k \times 2^{6-i}$, y $0 \leq k < 2^i$. Es decir, los valores de j son:

$$0, 1 \times 2^{6-i}, 2 \times 2^{6-i}, 3 \times 2^{6-i}, \ldots, (2^i - 2) \times 2^{6-i}, (2^i - 1) \times 2^{6-i}$$

o lo que es lo mismo

$$\begin{aligned}
0, 2 \times 2^{6-i}, 4 \times 2^{6-i}, \ldots, (2^i - 2) \times 2^{6-i} \quad &\text{y} \\
1 \times 2^{6-i}, 3 \times 2^{6-i}, \ldots, (2^i - 1) \times 2^{6-i} &
\end{aligned}$$

es decir

$$\left.\begin{array}{r} 2k \times 2^{6-i} \\ (2k+1) \times 2^{6-i} \end{array}\right\} \; 0 \leq k < 2^{i-1}$$

Para un k dado, las dos expresiones permiten obtener el índice que identifica a dos conjuntos hermanos. La primera de ellas, al hermano *situado a la izquierda*, y la segunda, al hermano *situado a la derecha*.

La siguiente función permite relacionar los identificadores de dos conjuntos hermanos. Concretamente, aplicada sobre el índice que identifica a uno cualquiera de ellos devuelve el índice del otro.

$$Brother(p, i) = \begin{cases} p + 2^{6-i} & \text{si} \quad (p \bmod 2^{6-i+1}) = 0 \\ p - 2^{6-i} & \text{si} \quad (p \bmod 2^{6-i+1}) \neq 0 \end{cases}$$

Como dos hermanos siempre están situados en el mismo nivel, vamos a simplificar la expresión anterior omitiendo el nivel, quedando entonces como $Brother(p)$. Comprobemos la validez de la función aplicándola a una pareja de hermanos cualquiera. Dado el nivel i $(0 < i \leq 6)$ los índices de los dos conjuntos hermanos son j y $j + 2^{6-i}$.

Para $2k \times 2^{6-i}$: $(2k \times 2^{6-i}) \bmod 2^{6-i+1} = (k \times 2^{6-i+1}) \bmod 2^{6-i+1} = 0$

Por tanto,

$Brother(2k \times 2^{6-i}) = (2k \times 2^{6-i}) + 2^{6-i} = (2k+1) \times 2^{6-i}$

Para $(2k+1) \times 2^{6-i}$: $((2k+1) \times 2^{6-i}) \bmod 2^{6-i+1} =$

$(k \times 2^{6-i+1} + 2^{6-i}) \bmod 2^{6-i+1} = 2^{6-i} \neq 0$

Por tanto,

$Brother((2k+1) \times 2^{6-i}) = ((2k+1) \times 2^{6-i}) - 2^{6-i} = 2k \times 2^{6-i}$

Al igual que para los hermanos, también se puede obtener una expresión que relaciona el identificador de un conjunto con el de cualquiera de sus ancestros. Así, para cualquier conjunto $E_{k,l}$, $0 < k \leq 6$ y $l \in I_k$, su índice l está relacionado con el de cualquiera de sus ancestros en el nivel i, $0 \leq i < 6$, mediante la expresión $(l \text{ div } 2^{6-i}) \times 2^{6-i}$.

De cara a simplificar el uso posterior de esta expresión, vamos a definir también la función $Ancestor(l,i) = (l \text{ div } 2^{6-i}) \times 2^{6-i}$, como el ancestro de l en el nivel i.

La generalización de la Propiedad 1, surgida al ser aplicada sucesivamente a cada uno de los descendientes de un $E_{i,j}$, viene a indicar que con las entradas de dicho conjunto, que atienden a una petición de tipo $d = 2^i$, se pueden atender 2 de tipo 2^{i+1}, 4 de tipo 2^{i+2}, etc. Esto queda reflejado en la siguiente propiedad.

Propiedad 2 *Para un i y j dados con $0 \leq i < 6$ y $j \in I_i$*

$$E_{i,j} = \bigcup_{l=0}^{2^k-1} E_{i+k,j+l\times 2^{6-(i+k)}} \qquad k = 1, 2, \ldots, 6-i$$

Demostración: Hacemos la demostración por inducción sobre k. Para $k = 1$ se tiene $E_{i,j} = E_{i+1,j} \cup E_{i+1,j+2^{6-(i+1)}}$, que es cierto por la Propiedad 1. Supongamos ahora que es cierto para $k-1$, y veamos que también se cumple para k. Como se cumple para $k-1$, se tiene que:

$$E_{i,j} = \bigcup_{l=0}^{2^{(k-1)}-1} E_{i+(k-1),j+l\times 2^{6-(i+k-1)}} \tag{4.1}$$

Al aplicar la Propiedad 1 a cada uno de esos conjuntos se obtiene:

$$\begin{aligned} E_{i+(k-1),j+l\times 2^{6-(i+k-1)}} &= E_{i+k,j+l\times 2^{6-(i+k-1)}} \cup E_{i+k,j+l\times 2^{6-(i+k-1)}+2^{6-(i+k)}} = \\ &= E_{i+k,j+l\times 2\times 2^{6-(i+k)}} \cup E_{i+k,j+(l\times 2+1)\times 2^{6-(i+k)}} \end{aligned}$$

y sustituyendo en la Expresión 4.1 se obtiene:

$$E_{i,j} = \bigcup_{l=0}^{2^{(k-1)}-1} E_{i+k,j+l\times 2\times 2^{6-(i+k)}} \cup E_{i+k,j+(l\times 2+1)\times 2^{6-(i+k)}}$$

Para el rango de valores de l, $\left[0\,..\,2^{(k-1)}-1\right]$, las expresiones $l\times 2$ y $(l\times 2+1)$ incluyen a todos los valores en el intervalo $\left[0\,..\,2^k-1\right]$ pues:

$$\begin{aligned} l\times 2 \quad &\text{incluye} \quad 0,2,4,\ldots,2(2^{(k-1)}-1) = 0,2,4,\ldots,2^k-2 \\ l\times 2+1 \quad &\text{incluye} \quad 1,3,5,\ldots,2(2^{(k-1)}-1)+1 = 1,3,5,\ldots,2^k-1 \end{aligned}$$

Por tanto,

$$\begin{aligned} E_{i,j} &= \bigcup_{l=0}^{2^{(k-1)}-1} E_{i+k,j+l\times 2\times 2^{6-(i+k)}} \cup E_{i+k,j+(l\times 2+1)\times 2^{6-(i+k)}} = \\ &= \bigcup_{l=0}^{2^k-1} E_{i+k,j+l\times 2^{6-(i+k)}} \end{aligned}$$

tal y como queríamos demostrar.

□

A partir de esta última propiedad es fácil comprobar que si $E_{i,j} = \bigcup E_{i+k,m}$, entonces se cumple que $m \geq j$, es decir, el índice de cualquiera de los descendientes, en cualquier nivel, de un conjunto dado es siempre mayor o igual que el índice de éste.

Propiedad 3 *Las secuencias de entradas disponibles para una distancia $d = 2^i$ son independientes entre sí. De manera más formal:*

$$E_{i,j} \cap E_{i,l} = \phi \quad \forall\ l \neq j \ \ con \ \ 0 < i \leq 6 \ y \ j,l \in I_i$$

En términos de árboles binarios esta propiedad podría enunciarse como *"los nodos del árbol situados en el mismo nivel son disjuntos entre sí"*.

Demostración: Las entradas correspondientes a estos conjuntos son:

$$\begin{array}{l} E_{i,j} = \{t_n \mid \quad j \leq n < j + 2^{6-i}\} \\ E_{i,l} = \{t_m \mid \quad l \leq m < l + 2^{6-i}\} \end{array} \left\{ \begin{array}{l} j = k \times 2^{6-i} \\ l = t \times 2^{6-i} \\ 0 \leq k,t < 2^i \end{array} \right.$$

Pero

$$\left.\begin{array}{l} j \leq n < j + 2^{6-i} \\ l \leq m < l + 2^{6-i} \end{array}\right\} \Longrightarrow \left\{ \begin{array}{l} k \times 2^{6-i} \leq n < (k+1) \times 2^{6-i} \\ t \times 2^{6-i} \leq m < (t+1) \times 2^{6-i} \end{array} \right.$$

Como $j \neq l$ entonces $k \neq t$ y por tanto $n \neq m$, con lo que

$$E_{i,j} \cap E_{i,l} = \phi$$

□

Propiedad 4 *Cada conjunto de entradas $E_{i,j}$ disponible para una distancia $d = 2^i$ es disjunto respecto a todos los conjuntos de entradas disponibles para una distancia $d' = 2^{i+1}$ que no cumplan la Propiedad 1 respecto de $E_{i,j}$. Más formalmente:*

$$E_{i,j} \cap E_{i+1,l} = \phi \qquad \forall\ l\ tal\ que\ (l \neq j)\ y\ (l \neq j + 2^{6-(i+1)}),\ \begin{cases} 0 < i < 6 \\ j \in I_i \\ l \in I_{(i+1)} \end{cases}$$

En términos de árboles binarios, esta propiedad puede enunciarse como *"un nodo es disjunto con todos aquellos de una profundidad mayor excepto con sus descendientes"*.

Demostración: Por la Propiedad 3 se sabe que $E_{i,j} \cap E_{i,l} = \phi$, $\forall j \neq l$. Por otra parte, por la Propiedad 1 se tiene que $E_{i,l} = E_{i+1,l} \cup E_{i+1,l+2^{6-(i+1)}}$. Combinando ambas expresiones se tiene:

$$\begin{aligned} E_{i,j} \cap E_{i,l} = \phi &\iff E_{i,j} \cap (E_{i+1,l} \cup E_{i+1,l+2^{6-(i+1)}}) = \phi \\ &\iff (E_{i,j} \cap E_{i+1,l}) \cup (E_{i,j} \cap E_{i+1,l+2^{6-(i+1)}}) = \phi \\ &\implies \begin{cases} (E_{i,j} \cap E_{i+1,l}) = \phi \\ (E_{i,j} \cap E_{i+1,l+2^{6-(i+1)}}) = \phi \end{cases} \end{aligned}$$

tal y como se quería demostrar.

□

Propiedad 5 *Dados $E_{i,l}$ libre y $E_{i,m}$ no libre, $l \neq m$, cuyas entradas ocupadas se usan para atender completamente peticiones de tipo $d \geq 2^i$, entonces todas esas peticiones pueden ser totalmente atendidas con entradas pertenecientes a $E_{i,l}$, dejando así libre a $E_{i,m}$, con $0 < i \leq 6$ y $l, m \in I_i$.*

Demostración: Si todas las entradas de $E_{i,m}$ atienden completamente a una petición de tipo $d = 2^i$, ésta también podrá ser atendida con las entradas de cualquier $E_{i,j}$ libre, $j \neq m$, y en particular con las de $E_{i,l}$.

Si todas o parte de las entradas de $E_{i,m}$ atienden completamente a peticiones de tipos $d > 2^i$, significa que se están usando para ello algunos de los $E_{i+k,m+n\times 2^{6-(i+k)}}$, $n = 0, \ldots, 2^k - 1$ y $k = 1, \ldots, 6 - i$, tales que $E_{i,m} = \bigcup_{n=0}^{2^k-1} E_{i+k,m+n\times 2^{6-(i+k)}}$. Para

atender a esas peticiones también se pueden usar otros $E_{i+k,j+n\times 2^{6-(i+k)}}$ libres, $j \neq m$, en particular los $E_{i+k,l+n\times 2^{6-(i+k)}}$, $n = 0, \ldots, 2^k - 1$ y $k = 1, \ldots, 6-i$, pues son conjuntos libres por el hecho de que $E_{i,l}$ está libre y se cumple la Propiedad 2 para este conjunto.

□

Definición 3 *Se dice que un conjunto $E_{i,j}$ es* **singular** *si dicho conjunto es libre pero el conjunto $E_{i,n}$ no lo es, con $0 < i \leq 6$, $n = Brother(j)$ y $j, n \in I_i$. Es decir, un conjunto $E_{i,j}$ es singular si es libre pero su conjunto hermano no lo es.*

Como consecuencia de esto, y aplicando la terminología de árboles binarios, ninguno de los ancestros de un conjunto singular es libre.

Definición 4 *Una tabla se dice que está normalizada si $\forall i, 0 \leq i \leq 6$, a lo sumo existe un $E_{i,j}$ singular, con $j \in I_i$.*

De acuerdo con esta definición, una tabla vacía y una tabla llena están normalizadas pues en ningún nivel existe un conjunto singular.

Entre los conjuntos definidos anteriormente puede definirse una relación de orden parcial, basada en su posición en el árbol binario.

Definición 5 *Sea $\mathcal{E}$ el conjunto formado por todos los conjuntos $E_{i,j}$, con $0 < i \leq 6$ y $j \in I_i$. Se define sobre $\mathcal{E} \times \mathcal{E}$ la siguiente relación binaria:*

$$E_{k,l} \blacktriangleleft E_{i,j} \Longleftrightarrow \begin{cases} i \leq k \\ l < j \end{cases}$$

$E_{k,l} \blacktriangleleft E_{i,j}$ se lee como: $E_{k,l}$ está situado más a la izquierda que el conjunto $E_{i,j}$, en el mismo nivel o en niveles inferiores en el árbol binario de la Figura 4.1.

Como no todos los $E_{i,j}$ son comparables (por ejemplo, $E_{3,8} \not\blacktriangleleft E_{4,12}$ y $E_{4,12} \not\blacktriangleleft E_{3,8}$) esta relación establece un orden parcial estricto, y así, $\mathcal{E}$ es un conjunto parcialmente ordenado.

A partir de la Definición 5 surgen nuevas propiedades que se enuncian a continuación.

Propiedad 6 *Dados $E_{k,l}$ y $E_{k,m}$, con $0 < k \leq 6$ y $l, m \in I_k$ tales que $E_{k,l} \blacktriangleleft E_{k,m}$, si $p \neq q$, con $p = Ancestor(l, i)$, $q = Ancestor(m, i)$ y $0 < i < k$, entonces se cumple que $E_{i,p} \blacktriangleleft E_{i,q}$.*

Es decir, si dos conjuntos están relacionados, sus respectivos ancestros hasta el nivel inmediatamente anterior al que pertenece el primer ancestro común de ellos, están también relacionados entre sí.

Demostración: Para que $E_{i,p} \blacktriangleleft E_{i,q}$ deben cumplirse las dos condiciones de la Definición 5, es decir, $i \leq i$ y $p < q$. La primera de ellas es trivial.

Por ser $E_{k,l} \blacktriangleleft E_{k,m}$, entonces se cumple que $l < m$, y por tanto $(l \text{ div } 2^{6-i}) \leq (m \text{ div } 2^{6-i})$, y $(l \text{ div } 2^{6-i}) \times 2^{6-i} \leq (m \text{ div } 2^{6-i}) \times 2^{6-i}$, es decir $p \leq q$.

Ahora bien, por la definición, $p \neq q$, así que $p < q$, lo que significa que $E_{i,p} \blacktriangleleft E_{i,q}$.

□

Propiedad 7 *Dados $E_{k,l}$ y $E_{k,m}$, con $0 < k \leq 6$ y $l, m \in I_k$, tales que $E_{k,l} \blacktriangleleft E_{k,m}$, si $i < k$ y $n \neq j$, con $n = Ancestor(l, i)$, $j = Ancestor(m, i)$, entonces se cumple que $E_{k,l} \blacktriangleleft E_{i,j}$.*

Es decir, si dos conjuntos están relacionados, el ancestro del segundo de ellos en cualquier nivel, hasta aquel al que pertenece el primer ancestro común, también está relacionado con el primero.

Demostración: Como $E_{k,l} \blacktriangleleft E_{k,m}$, se tiene que $l < m$. Además, como $i < k$, y $n \neq j$, por la Propiedad 6 se tiene que $E_{i,n} \blacktriangleleft E_{i,j}$.

Como $n \neq j$, por la Propiedad 3, se tiene que $E_{i,n} \cap E_{i,j} = \phi$. Pero como $E_{k,l} \subset E_{i,n}$ entonces también se cumple que $E_{k,l} \cap E_{i,j} = \phi$.

Supongamos que $E_{k,l} \blacktriangleleft E_{i,j}$, y por tanto $l \geq j$. También se tiene que $l < m$, y por tanto $j \leq l < m$. Como $E_{k,m} \subset E_{i,j}$, y por la consecuencia de la Propiedad 2, que relaciona los índices de un conjunto y sus descendientes, se tendría que $E_{k,l} \subset E_{i,j}$, lo cual es una contradicción, pues $E_{k,l} \cap E_{i,j} = \phi$. Así pues, se cumple que $E_{k,l} \blacktriangleleft E_{i,j}$.

□

Propiedad 8 *Dados $E_{k,l}$ y $E_{i,j}$, con $0 < i < k \leq 6$, $l \in I_k$ y $j \in I_i$, tales que $E_{k,l} \blacktriangleleft E_{i,j}$, entonces se cumple que $E_{t,m} \blacktriangleleft E_{i,j}$, con $i \leq t < k$ y $m = Ancestor(l, t)$.*

Es decir, si dos conjuntos de niveles k e i, con $i < k$ están relacionados, entonces todos los ancestros hasta el nivel i del conjunto del nivel k, están relacionados con el conjunto del nivel i.

Demostración: Para que se cumpla $E_{t,m} \blacktriangleleft E_{i,j}$, según la Definición 5, debe cumplirse $i \leq t$ y $m < j$. La primera condición se cumple por el enunciado. Veamos pues la

segunda condición. Por cumplirse $E_{k,l} \blacktriangleleft E_{i,j}$ entonces $l < j$, y como consecuencia, se cumple que

$$m = (l \text{ div } 2^{6-t}) \times 2^{6-t} \leq (j \text{ div } 2^{6-t}) \times 2^{6-t} \tag{4.2}$$

Como $j \in I_i$, $j = p \times 2^{6-i}$, con $0 \leq p < 2^i$. Como $i \leq t$, entonces $t = i + x$, $x \geq 0$. De esta forma:

$$\begin{aligned} j \bmod 2^{6-t} &= (p \times 2^{6-i}) \bmod 2^{6-t} = (p \times 2^{6-t+x}) \bmod 2^{6-t} = \\ &= (p \times 2^x \times 2^{6-t}) \bmod 2^{6-t} = 0 \end{aligned}$$

Como $(j \bmod 2^{6-t}) = 0$, entonces $(j \text{ div } 2^{6-t}) \times 2^{6-t} = j$, y por tanto, de (4.2) se cumple que $m \leq j$.

Además, como $m = Ancestor(l, t)$, por la consecuencia de la Propiedad 2, que relaciona los índices de un conjunto y sus descendientes, $m \leq l$. Por tanto, se tiene que $m \leq j$ y $m \leq l$, además de que $l < j$. Si $m = j$ se tendría que $l < j = m$, lo cual es una contradicción. Por tanto, $m \neq j$, con lo que debe cumplirse que $m < j$.

Así pues, $i \leq t$ y $m < j$, con lo que se cumple que $E_{t,m} \blacktriangleleft E_{i,j}$, con $i \leq t < k$.

□

Definición 6 *Una tabla se dice que está ordenada si* $\forall E_{i,j}$ *y* $\forall E_{k,l}$*, ambos conjuntos singulares, con* $0 < i < k \leq 6$*,* $j \in I_i$ *y* $l \in I_k$*, entonces se tiene* $E_{k,l} \blacktriangleleft E_{i,j}$*.*

Una vez visto el modelo a utilizar, algunas definiciones al respecto, así como algunas propiedades derivadas, se presenta a continuación el algoritmo propuesto para buscar una nueva secuencia de entradas en la tabla situadas con una separación determinada entre sí. Primero se enuncia el algoritmo y después algunos teoremas para probar sus propiedades.

4.3. Algoritmo de reserva en la tabla

Como ya se ha comentado en secciones anteriores, dada una nueva petición de distancia máxima $d = 2^i$, se hace necesario encontrar un grupo de $\frac{64}{d}$ entradas libres en la tabla, de forma que entre cualesquiera dos de esas entradas consecutivas haya como máximo una distancia d. Evidentemente, esto es lo mismo que encontrar un conjunto $E_{i,j}$ que esté libre. Para buscar ese conjunto se va a aplicar un algoritmo del que damos a continuación una descripción informal para inmediatamente después enunciar formalmente y validar mediante las correspondientes demostraciones.

Dada una petición de distancia $d = 2^i$, el algoritmo examina todos los conjuntos $E_{i,j}$ posibles para ese tipo de petición, en un orden determinado, y selecciona el primero que cumple que todas sus entradas están libres.

El orden en el que se recorren los conjuntos es uno concreto y su objetivo es maximizar la distancia entre dos entradas libres consecutivas de las que quedarían en la tabla después de realizar la selección. De esta forma, la tabla queda en las mejores condiciones posibles para poder atender posteriormente a la petición más restrictiva posible.

El orden en que el algoritmo examina los conjuntos libres es el que resulta de recorrer los conjuntos $E_{i,j}$ de izquierda a derecha según la disposición de la Figura 4.1. De manera formal, el algoritmo podría enunciarse como:

> **Dada una nueva petición de distancia máxima $d = 2^i$, el algoritmo selecciona el primer $E_{i,j}$ libre de la secuencia**
>
> $$E_{i,0}, E_{i,1\times 2^{6-i}}, E_{i,2\times 2^{6-i}}, E_{i,3\times 2^{6-i}}, \ldots, E_{i,(2^i-1)\times 2^{6-i}}$$

Hay que destacar que la aparente sencillez de este algoritmo (barrer los conjuntos $E_{i,j}$ de izquierda a derecha) se ha conseguido gracias a la numeración de las entradas de la tabla mediante la permutación bit-reversal en lugar de usar una numeración correlativa.

Una vez enunciado el algoritmo para ubicar una petición en la tabla, en la siguiente sección se presentan una serie de características que pueden derivarse de él, adecuadamente enunciadas y demostradas.

4.4. Propiedades del algoritmo de reserva

Este algoritmo tiene una serie de características que lo hacen muy eficiente para la reserva de las entradas en una tabla a partir de una serie de peticiones con unos requisitos de distancia máxima, y que se van a mostrar a partir de la demostración de una serie de teoremas.

Teorema 1 *Si hay un grupo de entradas libres en la tabla a una distancia solicitada $d = 2^i$, $0 \leq i \leq 6$, el algoritmo las encuentra.*

Demostración: Si hay un grupo de entradas libres en la tabla de forma que cualesquiera dos consecutivas estén situadas a la distancia solicitada $d = 2^i$, eso quiere decir que hay un conjunto $E_{i,k}$ libre, con $k \in I_i$. Por la propia definición del algoritmo de

reserva, que analiza consecutivamente todos los conjuntos $E_{i,j}$ hasta que encuentra uno libre, está garantizado que ese conjunto $E_{i,k}$ será finalmente encontrado.

□

Teorema 2 *Tras aplicar el algoritmo de reserva, la tabla queda normalizada. Es decir, el algoritmo de reserva deja en cualquier nivel del árbol de la Figura 4.1, a lo sumo, un conjunto singular.*

Demostración: Supongamos que no es así, es decir, $\exists i$, $1 < i \leq 6$, en el cual hay más de un conjunto singular. Supongamos que hay dos, concretamente $E_{i,k}$ y $E_{i,l}$, con $k, l \in I_i$ y $k < l$. Por ser $E_{i,l}$ singular, $E_{i,n}$ no está libre, con $n = Brother(l)$. Como $k < l$, y $E_{i,k}$ y $E_{i,l}$ son singulares, entonces $Ancestor(k, i-1) \neq Ancestor(l, i-1)$, y por tanto también se cumple que $k < n$. Es decir, el conjunto $E_{i,k}$ está más a la izquierda que el conjunto $E_{i,n}$.

Sin embargo esta situación no es posible, pues el algoritmo de reserva hubiese seleccionado el conjunto $E_{i,k}$, o cualquiera de sus hijos en el árbol según la Propiedad 2, antes que el $E_{i,n}$, o cualquiera de sus hijos en el árbol según la Propiedad 2, pues con los primeros se pueden atender las peticiones atendidas con los últimos.

Cualquier situación que considere más de dos conjuntos que no cumplan las hipótesis de partida incluye al caso de dos, y como consecuencia ninguna de ellas será posible. En definitiva, $\forall i, 0 < i \leq 6$, si hay más de un $E_{i,j}$ libre, a lo sumo uno de ellos es un conjunto singular, y por tanto la tabla está normalizada.

□

Teorema 3 *Si tras aplicar varias veces el algoritmo de reserva quedan en la tabla n entradas libres, es posible atender la petición de tipo $d = 2^i$ más restrictiva posible, con $\frac{64}{d} \leq n < \frac{64}{\frac{d}{2}}$, es decir, existe un $E_{i,m}$ libre, con $m \in I_i$.*

Demostración: Si de las n entradas libres $\frac{64}{d}$ pertenecen a un mismo conjunto $E_{i,m}$, la demostración es trivial.

Supongamos ahora que no hay ningún $E_{i,j}$ libre, es decir, las n entradas libres están repartidas entre una serie de conjuntos libres $E_{k,l}$, con $i < k \leq 6$ y $l \in I_k$. Por el Teorema 2, tras aplicar varias veces el algoritmo de reserva, la tabla queda normalizada. Esto significa que en el nivel $i+1$ a lo sumo hay un $E_{i+1,l}$ singular. Pero esto se cumple $\forall k$, con $i < k \leq 6$. Ahora bien, cada uno de esos $E_{k,l}$ tendría $\frac{64}{2^k} = 2^{6-k}$ entradas, y por tanto, el número total de entradas libres que se obtiene con esos conjuntos libres vendría dado por

$$\sum_{k=i+1}^{6} 2^{6-k} = \sum_{k=0}^{6-i-1} 2^k = 2^0 + 2^1 + 2^2 + ... + 2^{6-i-1}$$

Se trata de la suma de una progresión geométrica de razón 2, cuyo resultado es $2^{6-i} - 1$, y como consecuencia

$$\sum_{k=i+1}^{6} 2^{6-k} = \sum_{k=0}^{6-i-1} 2^k = 2^{6-i} - 1 < 2^{6-i} = \frac{64}{2^i} = \frac{64}{d}$$

lo cual es una contradicción ya que partíamos de la existencia de $n \geq \frac{64}{d}$ entradas libres. Así pues, debe existir un $E_{i,m}$ libre.

Por tanto, si tras aplicar varias veces el algoritmo de reserva quedan n entradas libres, de ellas $\frac{64}{2^i}$ ($\frac{64}{2^i} \leq n < \frac{64}{\frac{d}{2}}$) pertenecen a un mismo conjunto $E_{i,m}$, y de esta forma es posible atender la petición, de tipo $d = 2^i$, más restrictiva posible.

□

Teorema 4 *Dada una tabla con n entradas libres, el algoritmo es capaz de ubicar cualquier conjunto de peticiones que no requiera más de n entradas.*

Demostración: Sean $d_1, d_2, ..., d_m$ las distancias de las peticiones a ubicar en la tabla que tiene n entradas libres. Dichas peticiones se realizan en el orden indicado por la secuencia y cumplen que

$$\sum_{i=1}^{m} \frac{64}{d_i} \leq n$$

Sea n_i el número de entradas libres que hay en la tabla en el momento de ser realizada la petición d_i. Se cumple que $n_1 = n$. Pues bien, analicemos lo que ocurre cuando es realizada cada una de las peticiones d_i, $1 \leq i \leq m$. En el momento de ser realizada, una petición d_i puede ser la más restrictiva o puede no serlo. Hay pues dos casos posibles:

- La petición d_i es la más restrictiva. Por el Teorema 3 puede ser atendida. Tras ser atendida, quedarán $n_{i+1} = n_i - \frac{64}{d_i}$ entradas libres.

- La petición d_i no es la más restrictiva. Por el Teorema 3, con las n_i entradas libres se puede atender una petición, la más restrictiva, d_k, tal que $\frac{64}{d_k} \leq n_i < \frac{64}{\frac{d_k}{2}}$. Como d_i no es la petición más restrictiva, $d_i > d_k$, y por la Propiedad 2, con las entradas libres capaces de atender a d_k se podrán atender $\dfrac{d_i}{d_k}$ peticiones de tipo d_i, donde $\dfrac{d_i}{d_k} \geq 2$. Por tanto, existe el conjunto libre requerido para atender la petición d_i.

Así pues, si hay suficientes entradas libres para atender un cierto conjunto de peticiones, el algoritmo de reserva es capaz de ubicarlas en la tabla.

□

Teorema 5 *Tras aplicar el algoritmo de reserva la tabla queda ordenada.*

Demostración: Por el Teorema 2, el algoritmo de reserva siempre deja la tabla normalizada. Por tanto $E_{i,j}$ y $E_{k,l}$ son los únicos conjuntos singulares de los niveles i y k, respectivamente, $\forall i$ y $\forall k$ tal que $0 < i < k \leq 6$, con $j \in I_i$ y $l \in I_k$.

Como $E_{k,l}$ es singular, y debido al funcionamiento del algoritmo de reserva, se cumple que $E_{k,m}$ no es libre $\forall m$ tal que $m < l$. Por la Propiedad 2, estos conjuntos $E_{k,m}$ están incluidos en sus ancestros del nivel i, es decir, $E_{k,m} \subset E_{i,n}$, con $n = Ancestor(m, i)$, lo que significa que los $E_{i,n}$ no están libres.

Además, como tras aplicar el algoritmo de reserva el conjunto $E_{k,l}$ es singular, entonces se cumple que su ancestro en el nivel i, no es libre. Es decir, todos los ancestros en el nivel i de los conjuntos del nivel k situados a la izquierda del conjunto $E_{k,l}$, y el propio ancestro del conjunto $E_{k,l}$ en el nivel i, no están libres. Por tanto, los conjuntos libres en el nivel i deben estar más a la derecha del ancestro de l en el nivel i, es decir, $Ancestor(l, i) < j$. Como $E_{k,l} \subset E_{i,Ancestor(l,i)}$ y por la Propiedad 3, $E_{k,l} \not\subset E_{i,j}$, entonces $l < j$. Esto unido a que $i < k$ hace que $E_{k,l} \blacktriangleleft E_{i,j}$. Como esto se cumple $\forall i$ y $\forall k$ tal que $0 < i < k \leq 6$, entonces según la Definición 6, la tabla está ordenada.

□

Este último teorema será utilizado en el siguiente capítulo para probar que la situación alcanzada en la tabla cuando hay eliminaciones es similar a la alcanzada si sólo se hubieran hecho las inserciones que restan en la tabla tras las eliminaciones. De esta forma será posible aplicar el Teorema 4 y garantizar que siempre que haya entradas suficientes para atender una secuencia de peticiones, éstas podrán ubicarse en la tabla pues las entradas están siempre colocadas en la mejor disposición posible para atender cualquier combinación de distancias máximas.

4.5. Resumen

En este capítulo se ha comenzado el tratamiento formal de la gestión de la tabla, considerando inicialmente sólo el proceso de reserva en la misma. Se ha supuesto, por tanto, que las peticiones no se eliminan de la tabla. De esta forma, las peticiones se van ubicando en la tabla, pero nunca se liberan las entradas que están ocupando. En el siguiente capítulo se eliminará esta restricción y se estudiarán los problemas que esto conlleva y cómo solucionarlos.

Inicialmente se ha introducido una descripción informal de lo que se ha desarrollado posteriormente. En esa sección se han dado sencillos ejemplos para ayudar a comprender los conceptos aquí presentados. En la Sección 4.2 se ha enunciado el modelo formal para tratar la tabla. Dicho modelo está basado en agrupar las entradas de la tabla en conjuntos de forma que cada conjunto contiene las entradas necesarias para atender una petición de una determinada distancia. Se han propuesto y demostrado una serie de propiedades derivadas del modelo, así como otras definiciones adicionales.

En la Sección 4.3 se ha propuesto el algoritmo para ubicar una secuencia en la tabla, y posteriormente se han visto las características que este algoritmo presenta por medio de una serie de teoremas. Se ha demostrado que el algoritmo de reserva, cuando no hay eliminaciones, deja siempre la tabla normalizada y ordenada. Este hecho será utilizado en el siguiente capítulo.

Uno de los resultados más importantes de los presentados en este capítulo es el Teorema 4. Este teorema prueba que el algoritmo de reserva propuesto siempre es capaz de ubicar en la tabla cualquier secuencia de peticiones que requiera un número de entradas menor o igual que las disponibles en la tabla. Esto es debido a que el algoritmo de reserva siempre deja la tabla en la situación propicia para poder atender a la petición más restrictiva posible, o a cualquier combinación de peticiones de distancias variadas, pero siendo el total de entradas demandadas menor o igual que el número de entradas libres disponibles. Esta es una característica importante que presenta el algoritmo propuesto que lo hace eficiente para ser utilizado en la reserva de entradas de la tabla.

Capítulo 5

Modelo formal de la tabla de arbitraje. Inserciones y eliminaciones

En este capítulo se completa el modelo formal para el tratamiento de la tabla. Si bien en el capítulo anterior se han presentado las características del proceso de reserva de entradas en la tabla, sin embargo todo el desarrollo se apoyó en una de las hipótesis de partida: las peticiones no se eliminaban nunca de la tabla. Evidentemente esta suposición es completamente irreal, pues la situación normal será que las peticiones se vayan ubicando y retirando de la tabla.

En este capítulo se verán detenidamente los problemas que genera el hecho de que las peticiones se retiren de la tabla. Se verá como este hecho puede llevar a situaciones donde el algoritmo de reserva propuesto en el capítulo anterior no funciona correctamente. Para solucionar estos problemas se propone un determinado tratamiento que se apoya en otros dos algoritmos. Uno de ellos es el algoritmo de desfragmentación, cuya idea básica es unir secuencias pequeñas de entradas para formar una mayor, y por tanto capaz de ubicar una petición de mayor tamaño que las anteriores. El otro es un algoritmo de reordenación, donde la idea básica consiste en ordenar los conjuntos singulares de manera correcta para que cuando se produzca una nueva petición se elija la secuencia más apropiada a las características de la conexión. Una vez enunciados los nuevos algoritmos se verán una serie de propiedades asociadas mediante la demostración de una serie de teoremas.

En la parte final, se presentará la forma de gestionar globalmente la tabla, es decir, cuando se tienen peticiones para ubicar una nueva secuencia en la tabla, pero también eliminación de peticiones ya ubicadas en la tabla. Por medio de una serie de teoremas se mostrará que la situación alcanzada por una tabla en la que se producen inserciones y eliminaciones es totalmente equivalente a la que se hubiera alcanzado si sólo se hubieran producido las inserciones de las peticiones que finalmente quedan en la tabla tras las inserciones y eliminaciones. Este hecho permitirá ligar ambas partes de la teoría

desarrollada. En concreto, se verá como el Teorema 4 es aplicable en cualquier caso, y por tanto nuestro algoritmo de reserva será capaz de ubicar cualquier secuencia de peticiones en la tabla, siempre que el número de entradas requeridas sea menor o igual al número de entradas disponibles.

5.1. Descripción informal

El algoritmo presentado en la Sección 4.3 estaba planteado en base a unas determinadas hipótesis de partida. La primera de ellas impone que sólo se den inserciones en la tabla. Como evidentemente ésta no será la situación real, se debe contemplar el hecho de que también haya eliminaciones, y de esta forma ir al comportamiento general de la tabla, en el que de manera totalmente aleatoria podrá haber inserción y eliminación de peticiones.

Eliminamos pues la restricción impuesta por esa primera hipótesis de partida, es decir, ahora puede haber eliminación de peticiones y como consecuencia liberación de entradas ocupadas.

Considerando el algoritmo de reserva de la Sección 4.3 y este nuevo hecho, se puede llegar a estados de la tabla tales que habiendo entradas suficientes para atender a una petición dada, ésta no pueda serlo, por no mantener todas ellas una separación adecuada. Veamos este hecho con el siguiente ejemplo.

Ejemplo 1 *Con la tabla llena*[1]*, se eliminan dos peticiones de tipo* $d = 8$ *que estaban usando las entradas de los conjuntos* $E_{3,16}$ *y* $E_{3,32}$*. Eso quiere decir que ahora hay 16 entradas libres y por tanto se podría atender a una petición de tipo* $d' = 4$*. Sin embargo, no hay ningún conjunto* $E_{2,j}$ *libre (ver Figura 5.1).*

Puede observarse en el ejemplo anterior como la tabla deja de estar normalizada al producirse esa eliminación. Para solucionar el problema planteado en el Ejemplo 1, hay que conseguir que las 16 entradas tengan una separación de 4 unidades entre dos consecutivas de ellas, o lo que es lo mismo, que haya un conjunto $E_{2,j}$ libre. Es decir, debemos conseguir que la tabla vuelva a estar normalizada. Esto se puede conseguir como se indica en el Ejemplo 2, en el que se sigue un procedimiento basado en dejar libres entradas que, estando ocupadas, son necesarias para obtener un conjunto de entradas libres con las que se pueda atender a la petición más restrictiva.

[1]Se ha elegido este punto de partida para hacer el desarrollo del ejemplo más sencillo y fácil de entender. Sin embargo, la situación puesta de manifiesto en este y los siguientes ejemplos se puede alcanzar partiendo de estados de la tabla muy diversos.

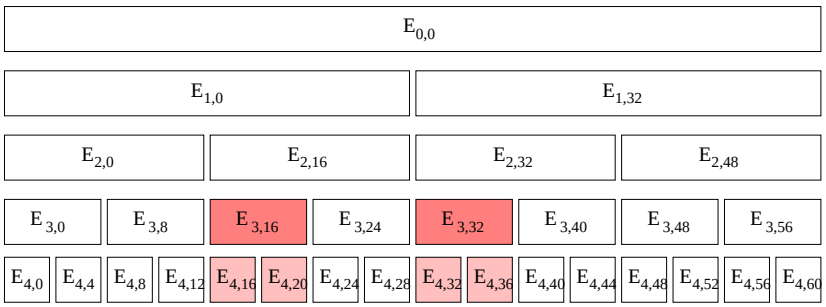


Figura 5.1: Situación de la tabla tras la liberación de los conjuntos $E_{3,16}$ y $E_{3,32}$, partiendo de la tabla completamente llena.

Ejemplo 2 *Para la situación del Ejemplo 1, se trataría, por ejemplo, de dejar el conjunto $E_{3,40}$ libre, y de esta forma, junto al conjunto $E_{3,32}$, también libre, obtener $E_{2,32}$ libre (por la Propiedad 1), permitiendo así atender una petición de tipo $d' = 4$.*

Para hacer que el conjunto $E_{3,40}$ quede libre, se deben usar las entradas del conjunto libre $E_{3,16}$ para atender a las peticiones que están siendo atendidas con las entradas de $E_{3,40}$, lo cual es factible según la Propiedad 5. La situación final tras el intercambio sería la mostrada en la Figura 5.2, quedando ahora la tabla normalizada.

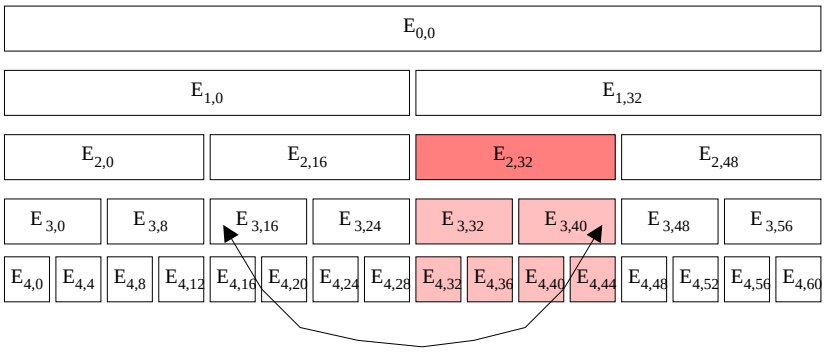


Figura 5.2: Situación de la tabla del Ejemplo 1 tras el intercambio de los conjuntos $E_{3,16}$ y $E_{3,40}$.

Así pues, este procedimiento, al que llamaremos *desfragmentación*, tiene como objetivo obtener el mayor conjunto libre posible con las entradas libres disponibles.

Aunque la necesidad de aplicar este proceso de desfragmentación surge como consecuencia de considerar eliminaciones de peticiones, también puede que deba ser aplicado tras atender peticiones. Veámoslo con un ejemplo.

Ejemplo 3 *A partir de la situación obtenida en el Ejemplo 2, es decir, con las entradas del conjunto $E_{2,32}$ libres y el resto ocupadas, se produce la liberación de las entradas correspondientes al conjunto $E_{3,56}$. Esta situación puede verse gráficamente en la Figura 5.3.*

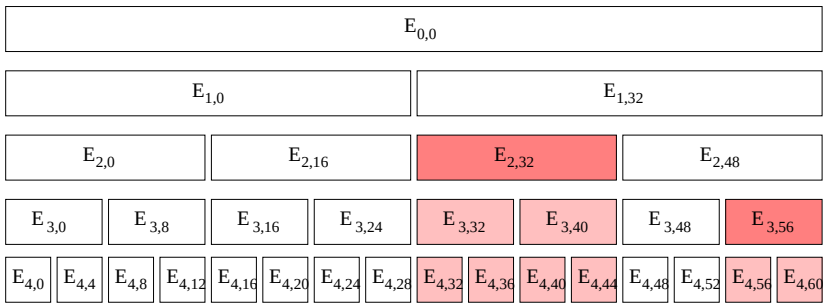


Figura 5.3: Situación de la tabla tras la liberación del conjunto $E_{3,56}$.

A continuación se realiza una petición de tipo $d = 8$, que puede ser atendida pues hay entradas libres suficientes y con la separación adecuada entre ellas. Al aplicar el algoritmo de reserva de entradas, será elegido el conjunto $E_{3,32}$ (el primer conjunto libre del nivel 3), creando así una situación similar a la presentada en el Ejemplo 1, y que podemos ver gráficamente en la Figura 5.4.

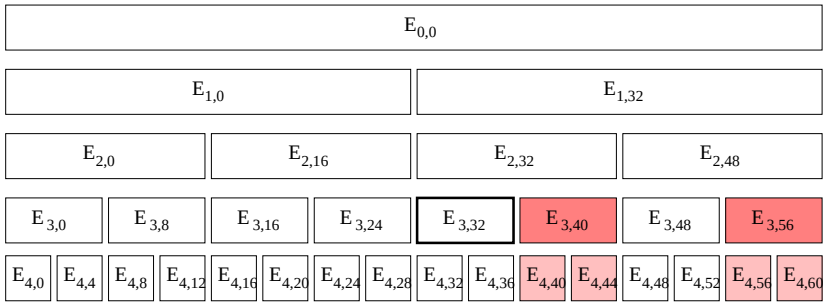


Figura 5.4: Situación final de la tabla del Ejemplo 3.

Como consecuencia, y al existir suficientes entradas libres para atender a una petición de tipo $d' = 4$, pero no pertenecer todas ellas al mismo conjunto, se hace necesaria una nueva desfragmentación. Ésta consistiría, por ejemplo, en conseguir que $E_{2,48}$ quede libre, haciendo que las entradas de $E_{3,40}$ atiendan a las peticiones que están siendo atendidas con las entradas del conjunto $E_{3,48}$.

En el Ejemplo 3 se puede observar como al producirse la liberación del conjunto $E_{3,56}$ la tabla deja de estar ordenada. Al aplicar posteriormente el algoritmo de reserva sobre una tabla no ordenada ya no obtenemos una tabla normalizada. Esto es consecuencia directa del funcionamiento del algoritmo de reserva, que como sabemos recorre los conjuntos para la distancia solicitada de izquierda a derecha y selecciona el primero de ellos que esté libre. La solución aplicada en el Ejemplo 3 ha consistido en aplicar desfragmentación después de una nueva inserción.

Una alternativa consistiría en ordenar los conjuntos singulares que se generan tras la liberación, de forma que la tabla vuelva a estar ordenada. De esta forma, el problema se corregiría si, en la situación mostrada en la Figura 5.3, el conjunto singular del nivel

3 (ahora $E_{3,56}$) estuviera *a la izquierda* del conjunto singular del nivel 2 (ahora $E_{2,32}$), y así al atender la petición de tipo $d = 8$, usando ese conjunto singular del nivel 3, y sin necesidad de aplicar desfragmentación, seguiría quedando un conjunto libre capaz de atender una petición de tipo $d' = 4$. De forma general, la solución alternativa consiste en conseguir que los conjuntos libres de menor tamaño, cuyas entradas tendrán una separación mayor, queden situados más a la izquierda que otros conjuntos libres de mayor tamaño. En definitiva, que la tabla esté ordenada.

La detección de este tipo de situaciones pasa por hacer un recorrido por la tabla y comprobar si existe algún conjunto libre de un determinado tamaño *más a la izquierda* que otro de menor tamaño. Para eliminar esa situación hay que aplicar la Propiedad 5 pero al nivel del mayor de los conjuntos libres, es decir, el intercambio debe realizarse entre el conjunto mayor y otro a su nivel al que pertenezca el más pequeño. En el Ejemplo 4 se muestra como se aplicaría este nuevo método.

Ejemplo 4 *Tomando como punto de partida la situación final del Ejemplo 2 (Figura 5.2), y producida la liberación de las entradas del conjunto $E_{3,56}$, se alcanza la situación mostrada en la Figura 5.3. En esa situación, se realiza una búsqueda por la tabla, implicando a todos los niveles del árbol, hasta encontrar la situación a corregir. Para eliminarla, se aplica la Propiedad 5, afectando el intercambio a los conjuntos $E_{2,32}$ y $E_{2,48}$. El nuevo estado de la tabla es tal que los conjuntos singulares son ahora el $E_{3,32}$ y el $E_{2,48}$, tal y como puede verse en la Figura 5.5. Si ahora se realiza una petición de tipo $d = 8$, será atendida con las entradas del conjunto $E_{3,32}$, quedando de esta forma un conjunto libre, $E_{2,48}$, para atender a una posible petición de tipo $d' = 4$.*

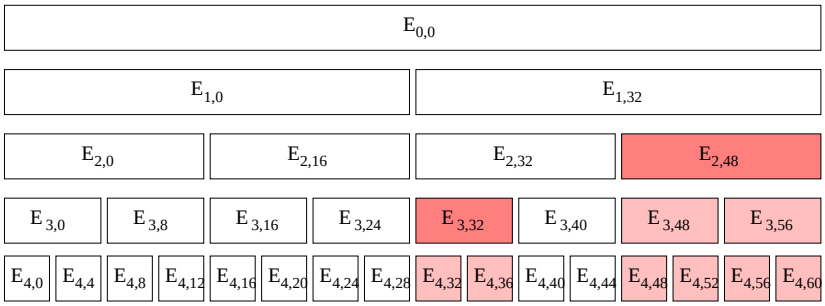


Figura 5.5: Situación de la tabla del Ejemplo 4 tras producirse la reordenación.

Así pues, con la combinación del algoritmo de reserva, un algoritmo de desfragmentación y otro de reordenación se tendrá cubierto el tratamiento general de la tabla. Se han mostrado dos alternativas. En un caso conseguimos mantener siempre la tabla ordenada y normalizada, mientras que en el otro sólo es necesario que la tabla esté normalizada, no siendo necesario que la tabla esté ordenada.

En resumen, acabamos de esbozar dos métodos para tratar aquellas situaciones que han surgido como consecuencia de haber eliminado la restricción impuesta por la

primera de las hipótesis de partida de la Sección 4.2.2. Estos dos métodos alternativos consisten básicamente en lo siguiente:

1. Tanto en el caso de que se atienda a una petición (usando para ello entradas libres) como en el caso de que una petición sea eliminada (liberándose entradas ocupadas) se debe examinar el estado resultante de la tabla para comprobar si habiendo entradas libres suficientes para atender a una petición dada, esto no sea posible por no existir ningún conjunto libre necesario para ello. Si esta situación se ha producido se debe aplicar un algoritmo de desfragmentación para corregir esa situación. Dicho de otra forma, si tras una inserción o eliminación la tabla no queda normalizada, hay que aplicar un algoritmo de desfragmentación para normalizarla.

2. Sólo en el caso de que se elimine una petición se realizarán las siguientes comprobaciones: se comprobará si hay conjuntos singulares de mayor tamaño situados más a la izquierda que conjuntos singulares de menor tamaño, en cuyo caso se realizará una reordenación; y se comprobará si habiendo entradas libres suficientes para atender a una petición dada, no se pueda hacer por no existir ningún conjunto libre necesario para ello. En ese caso, se debe aplicar el algoritmo de desfragmentación. El orden de las comprobaciones, en principio, no debe ser relevante.

   En definitiva, este segundo método debe comprobar tras una eliminación si la tabla ha quedado no ordenada y/o no normalizada, en cuyo caso se debe aplicar un algoritmo de reordenación y/o un algoritmo de desfragmentación para dejarla ordenada y normalizada.

Hay, por tanto, dos nuevos algoritmos a desarrollar: uno para realizar la desfragmentación y otro para realizar la reordenación. En las siguientes secciones se describen de forma detallada estos algoritmos y las situaciones genéricas de tratamiento de peticiones (reserva y eliminación) en los que se debe hacer uso de ellos. También veremos algunas características que pueden extraerse de la aplicación de estos nuevos algoritmos mediante una serie de teoremas relacionados.

5.2. Algoritmo de desfragmentación

La idea básica de este algoritmo es agrupar todas las entradas libres de la tabla en una serie de conjuntos que permitan atender a cualesquiera peticiones que requieran un número de entradas menor o igual que las disponibles en la tabla. Es decir, el objetivo del algoritmo es realizar un reagrupamiento de las entradas libres para que se pueda aplicar el Teorema 4. Este algoritmo repite, las veces que sea necesario, un proceso

que consiste en reunir en un único conjunto libre las entradas de dos conjuntos libres del mismo tamaño. Esta unión se producirá sólo si los dos conjuntos libres no forman ya parte de un mismo conjunto libre de mayor tamaño, es decir, el algoritmo se va a restringir a conjuntos singulares. El objetivo que se persigue es disponer del conjunto necesario para atender una petición de un tipo dado, para la cual se dispone de entradas libres suficientes pero, por estar estas entradas repartidas entre varios conjuntos libres, no pertenecen a un único conjunto que pueda ser seleccionado por el algoritmo de reserva visto en el capítulo anterior.

1: **Procedure Defragmentation** (set $E_{i,k}$)
2: **while** $i > 1$ **do**
3: found = $Find(E_{i,l})$ // $E_{i,l}$ singular
4: **if** found **then**
5: $swap(E_{i,k}, E_{i,m})$ // $E_{i,l}$ is the $E_{i,m}$'s brother
6: **end if**
7: $i = i - 1$
8: $k = Ancestor(m, i)$
9: **end while**
10: **End Defragmentation**

Figura 5.6: Algoritmo de desfragmentación

En la Figura 5.6 se muestra el código básico del algoritmo de desfragmentación. A partir de un $E_{i,k}$ libre se comprueba si existe otro $E_{i,l}$ libre que no sea hermano de $E_{i,k}$ (función *Find*, línea 3). Si existe, se reunen todas las entradas de ambos conjuntos en un conjunto de doble tamaño. Para ello se aplica la Propiedad 5 (procedimiento *swap*, línea 5). Como el nuevo conjunto libre puede que no sea el único en el nivel $i-1$ el proceso debe continuar, y es posible que se dé hasta el nivel 1 (el bucle de las líneas 2-9). El conjunto resultado de la unión se convierte en el $E_{i,k}$ de su nivel (líneas 7 y 8).

La acción básica del algoritmo de desfragmentación es la ejecución del procedimiento *swap()*, el cual realiza una transformación de la tabla mediante un intercambio de conjuntos. Este hecho fue mostrado de manera informal en el Ejemplo 1, donde ya se mostraba que el intercambio de los conjuntos adecuados dejaba un único conjunto singular en el nivel en cuestión, lo cual permitiría pasar de una situación en la que no se podía atender una petición del tipo más restrictivo a otra en la que ya era posible. Este cambio queda reflejado en el Teorema 6.

Teorema 6 *Dada una situación en la que* $\exists E_{i,k}$, $E_{i,l}$ *singulares, con* $1 < i \leq 6$ *y* $k \neq l$, *se puede pasar a otra en la que* $\exists E_{i-1,j}$ *libre tal que* $E_{i-1,j} = E_{i,m} \cup E_{i,l}$, *con* $k, l, m \in I_i$, $j \in I_{i-1}$.

Demostración: Sean $E_{i,k}$ y $E_{i,l}$ singulares y por tanto $\nexists E_{i-1,j}$, tal que $E_{i-1,j} = E_{i,k} \cup E_{i,l}$. La Propiedad 5 permite intercambiar el conjunto singular $E_{i,k}$ y el conjunto no libre $E_{i,m}$, siendo $m = Brother(l)$. Es decir, se intercambia el conjunto $E_{i,k}$, que está libre, con el conjunto hermano del $E_{i,l}$, que está ocupado. Este proceso crea un $E_{i-1,j}$ libre tal que $E_{i,m} \cup E_{i,l} = E_{i-1,j}$.

□

Pero puede haber situaciones donde el problema no se solucione aplicando un único intercambio. En estos casos el algoritmo de desfragmentación deberá hacer varias iteraciones, tal y como puede verse en el siguiente ejemplo.

Ejemplo 5 *Tomando como punto de partida la situación mostrada en la Figura 5.7(a), a la que se ha llegado por la liberación del conjunto $E_{3,0}$ o el $E_{3,16}$, se aplica el algoritmo de desfragmentación propuesto. En la primera iteración del algoritmo se intercambiarán los conjuntos $E_{3,0}$ y $E_{3,24}$, quedando así como conjunto libre ahora el $E_{2,16}$. Sin embargo, habiendo 32 entradas libres seguimos sin tener un conjunto de nivel 1 libre capaz de atender una posible petición de tipo $d = 2$. Evidentemente esto se debe a que la tabla no está normalizada y tenemos dos conjuntos singulares en el nivel 2. Así pues, el algoritmo de desfragmentación hará otra iteración. Al comienzo de esta segunda iteración tenemos como conjuntos singulares $E_{2,16}$ y $E_{2,48}$. El algoritmo intercambiará ahora $E_{2,16}$ y $E_{2,32}$, de forma que tanto $E_{2,32}$ como $E_{2,48}$ queden libres, y por tanto el conjunto singular sea ahora $E_{1,32}$. Esta es la situación mostrada en la Figura 5.7(b), donde se puede observar que ahora la tabla sí está normalizada. Si ahora se realizase una petición de tipo $d = 2$, podría ser atendida con las entradas del conjunto $E_{1,32}$.*

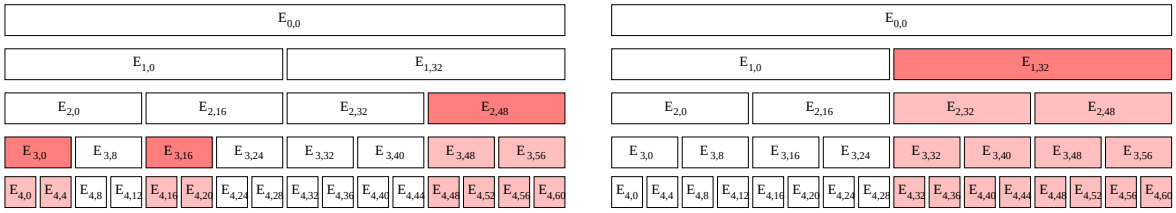


Figura 5.7: Situación antes (a) y después (b) de aplicar el algoritmo de desfragmentación de forma sucesiva en varios niveles.

Así pues, con el algoritmo de desfragmentación, que consiste en aplicar el Teorema 6 a todos los niveles donde haya más de un conjunto singular, se consigue obtener una tabla normalizada partiendo de una que no lo esté. Esto queda reflejado en el siguiente teorema.

Teorema 7 *El algoritmo de desfragmentación permite obtener una tabla normalizada a partir de una no normalizada, a la cual se ha llegado después de producirse una eliminación en una tabla normalizada.*

Demostración: El algoritmo de desfragmentación aplica el Teorema 6 sucesivamente en todos los niveles donde haya dos o más conjuntos singulares. Partiendo del nivel donde se ha producido la eliminación, recorre todos los niveles hasta el nivel 1, agrupando pares de conjuntos libres del mismo nivel. Dicha agrupación es posible en todos los casos. De esta forma, tras aplicar el algoritmo no puede existir más de un conjunto singular en cada nivel. Así pues, tras aplicar el algoritmo de desfragmentación, la tabla vuelve a estar normalizada.

□

Veamos ahora cómo influye la desfragmentación sobre la ordenación de la tabla.

Ejemplo 6 *Supongamos como situación de partida la mostrada en la Figura 5.8(a) a la que sólo se ha podido llegar tras la liberación del conjunto $E_{3,8}$ o el $E_{3,16}$. Vemos que la tabla está ordenada pues los conjuntos más pequeños siempre están a la izquierda de los mayores. Sin embargo, la tabla no está normalizada pues en el nivel 3 hay dos conjuntos singulares. Al aplicar el algoritmo de desfragmentación se obtendría la situación mostrada en la Figura 5.8(b), donde puede observarse que la tabla sigue ordenada, pero ahora también está normalizada.*

Si el algoritmo de desfragmentación se aplicara de forma sucesiva a varios niveles, tampoco se modificaría el orden en la tabla pues partiendo de una tabla ordenada los conjuntos mayores siempre estarán más a la derecha. Por muchos niveles que se recorran, uniendo el conjunto singular formado en la iteración anterior del algoritmo con el que existía en ese nivel, nunca se alterará el orden.

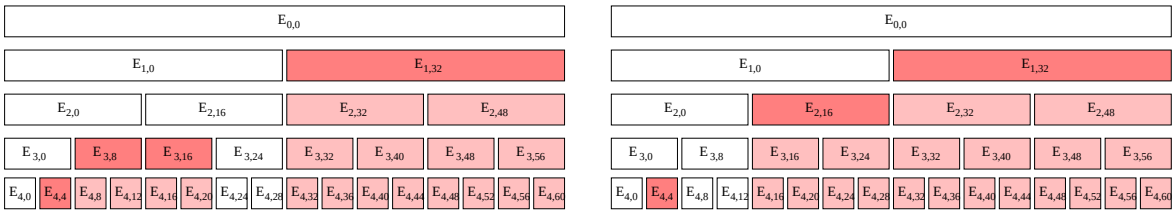


Figura 5.8: Situación antes (a) y después (b) de aplicar el algoritmo de desfragmentación. Puede observarse que partiendo de una tabla ordenada el algoritmo de desfragmentación deja la tabla ordenada y normalizada.

Hay que señalar que con esto no se está afirmando que el algoritmo de desfragmentación ordene la tabla. Tampoco, que tras producirse una eliminación la tabla vaya a quedar siempre ordenada, como se ha podido ver en el Ejemplo 5.5. Cuando interese restablecer el orden en la tabla se debe aplicar el algoritmo de ordenación que se estudiará en la siguiente sección. Lo que sí se quiere resaltar es que el algoritmo de desfragmentación no desordena una tabla ordenada.

Así pues, una vez visto un ejemplo donde se observa que aplicar el algoritmo de desfragmentación sobre una tabla ordenada, pero no normalizada, produce una tabla ordenada y normalizada, veámoslo de manera formal mediante el siguiente teorema.

Teorema 8 *Tras aplicar el algoritmo de desfragmentación sobre una tabla ordenada, en la que a lo sumo hay un nivel, i, con $1 < i \leq 6$, con dos conjuntos singulares, la tabla queda normalizada y continúa ordenada.*

Demostración: Sean $E_{i,k}$ y $E_{i,l}$ los dos conjuntos singulares del único nivel i que tiene dos conjuntos singulares, con $k < l$ y $k, l \in I_i$. Aplicando el Teorema 7 se obtiene $E_{t,u}$ libre, con $0 < t < i$ y $u \in I_t$.

Sean $E_{p,q}$ y $E_{r,s}$ dos conjuntos singulares cualesquiera tales que $0 < r < i < p \leq 6$, $q \in I_p$ y $s \in I_r$. Como la tabla estaba inicialmente ordenada, se cumple que $E_{p,q} \blacktriangleleft E_{i,k} \blacktriangleleft E_{i,l} \blacktriangleleft E_{r,s}$.

Con esta situación como punto de partida se tienen dos posibilidades:

- Si $r < t$, entonces lo que se debe probar es que $E_{p,q} \blacktriangleleft E_{t,u} \blacktriangleleft E_{r,s}$.

  Veamos primero el caso en que $t = i - 1$. En este caso $u = \mathit{Ancestor}(l, t)$. Como se cumple que $E_{i,k} \blacktriangleleft E_{i,l}$, entonces por la Propiedad 7 $E_{i,k} \blacktriangleleft E_{t,u}$. Además, como $E_{p,q} \blacktriangleleft E_{i,k}$ y $E_{i,k} \blacktriangleleft E_{t,u}$ por la propiedad transitiva de la relación de orden se cumple que $E_{p,q} \blacktriangleleft E_{t,u}$.

  Si $r < t < i - 1$ la demostración es similar pero aplicando de forma sucesiva la Propiedad 7 y la propiedad transitiva de la relación de orden. Luego, en cualquier caso, si $r < t$ entonces $E_{p,q} \blacktriangleleft E_{t,u}$.

  Por otra parte, también debe cumplirse que $E_{t,u} \blacktriangleleft E_{r,s}$, para lo cual debe cumplirse que $r \leq t$ y que $u < s$. La primera desigualdad está en las hipótesis de partida. Veamos la segunda también para el caso particular donde $t = i-1$. Como $E_{i,l} \blacktriangleleft E_{r,s}$ entonces se cumple que $l < s$. Además, como $u = \mathit{Ancestor}(l, t)$, por la consecuencia de la Propiedad 2, que relaciona los índices de un conjunto y sus descendientes, se cumple que $u \leq l$, con lo que $u < s$. Por tanto $E_{t,u} \blacktriangleleft E_{r,s}$.

  Nuevamente, para el caso general donde $r < t < i - 1$ la demostración es similar pero haciéndola de forma sucesiva. Luego, en cualquier caso, si $r < t$ entonces $E_{t,u} \blacktriangleleft E_{r,s}$.

De esta forma, $E_{p,q} \blacktriangleleft E_{t,u} \blacktriangleleft E_{r,s}$, y por tanto si $r < t$ el nuevo conjunto singular $E_{t,u}$ que ha generado el algoritmo de desfragmentación sigue manteniendo la tabla ordenada.

- Si $r = t$, entonces el algoritmo de desfragmentación, usando $E_{t,u}$ y $E_{r,s}$, generará un conjunto singular $E_{r-1,h}$, con $h \in I_{r-1}$, $h = Ancestor(s, r-1)$. En este caso, sólo se debe probar que $E_{p,q} \blacktriangleleft E_{r-1,h}$.

  Esta demostración es similar a la realizada en el apartado anterior donde se debe aplicar de forma sucesiva la Propiedad 7 y la propiedad transitiva de la relación de orden.

Así pues, en cualquier caso, tras aplicar el algoritmo de desfragmentación sobre una tabla ordenada, en la que a lo sumo hay un nivel, i, con $1 < i \leq 6$, con dos conjuntos singulares, la tabla queda normalizada y continúa ordenada.

□

Así pues, el algoritmo de desfragmentación consigue normalizar una tabla que no lo estaba, y si la tabla estaba ya ordenada, continúa ordenada tras aplicar dicho algoritmo. Estos resultados serán utilizados de nuevo en la Sección 5.4.

Se presenta ahora el algoritmo de reordenación, y al igual que se ha hecho con el algoritmo de desfragmentación, se incluye en primer lugar una descripción informal mediante sencillos ejemplos, para a continuación mostrar sus propiedades mediante el enunciado y demostración de los correspondientes teoremas.

5.3. Algoritmo de reordenación

Se trata básicamente de un algoritmo de ordenación, pero aplicado a nivel de conjuntos. Este algoritmo está diseñado para ser aplicado a una tabla no ordenada y como resultado la tabla queda ordenada de acuerdo a la Definición 6. El algoritmo de reordenación establece un orden de menor a mayor tamaño, en orden ascendente y de izquierda a derecha. Es decir, cuando se rompa este orden, como consecuencia de la aparición de nuevos conjuntos singulares mal colocados, actuará el algoritmo para restablecer dicho orden.

Cuando un conjunto queda libre, el algoritmo comprueba si está adecuadamente situado con respecto al resto de conjuntos singulares. Es decir, debe tener *a su izquierda* todos los conjuntos singulares más pequeños, y *a su derecha* todos los conjuntos singulares más grandes. Si su posición no es correcta se producirán los movimientos necesarios para restablecer el orden en la tabla. Esos movimientos se producirán realizando intercambios, como también ocurría con el algoritmo de desfragmentación.

Como se indicó en la Sección 5.1, el algoritmo de reordenación se usará cada vez que se produzca una liberación de entradas. A su vez, esta liberación se produce en una tabla que está ordenada. Por tanto, justo antes de aplicar el algoritmo de reordenación sólo habrá en la tabla un conjunto que no esté ordenado con respecto a los demás.

Hay que indicar que el funcionamiento del algoritmo de reordenación es totalmente similar al de uno de los algoritmos de ordenación bien conocidos como es el de *selección directa* [Knu73].

En la Figura 5.9 se muestra el código básico que constituye el algoritmo de reordenación. El algoritmo recorre todos los niveles excepto los dos últimos (líneas 2-11) empezando por el más bajo. En cada uno de ellos se tendrá a lo sumo un conjunto singular, $E_{k,l}$, excepto en quizás uno que puede haber dos, aunque sólo uno de ellos puede estar mal ordenado. Por tanto, en cada nivel para el $E_{k,l}$ singular correspondiente, se comprobará si existe en niveles superiores (líneas 4-9) un conjunto $E_{i,j}$ singular situado más a su izquierda. Si hay varios, se selecciona el más a la izquierda (líneas 5 y 6), y se ordenan entre sí. Para ello se aplica la Propiedad 5 sobre $E_{i,j}$ y el ancestro de $E_{k,l}$ en el nivel i (procedimiento *swap*, línea 7).

```
1:  Procedure Reordering
2:  for k = 6 downto 2 do
3:    for all  E_{k,l} singular do
4:      for i = k − 1 downto 2 do
5:        found = Find(E_{i,j})                    // E_{i,j} singular and E_{k,l} ◄ E_{i,j}
6:        if found then
7:          swap(E_{i,j}, E_{i,Ancestor(k,i)})
8:        end if
9:      end for
10:   end for
11: end for
12: End Reordering
```

Figura 5.9: Algoritmo de reordenación

Al igual que hemos hecho en la sección anterior con el algoritmo de desfragmentación, veamos ahora mediante un sencillo ejemplo el funcionamiento del algoritmo de reordenación.

Ejemplo 7 *Supongamos como situación de partida la mostrada en la Figura 5.10(a), a la que se ha llegado tras la liberación del conjunto $E_{2,16}$ o el $E_{3,40}$ en una tabla ordenada. El algoritmo de reordenación intercambiará los conjuntos $E_{2,16}$ y $E_{2,32}$. De esta forma*

la situación final será la mostrada en la Figura 5.10(b), donde los conjuntos singulares son ahora $E_{4,4}$, $E_{3,24}$ y $E_{2,32}$, estando la tabla de nuevo ordenada.

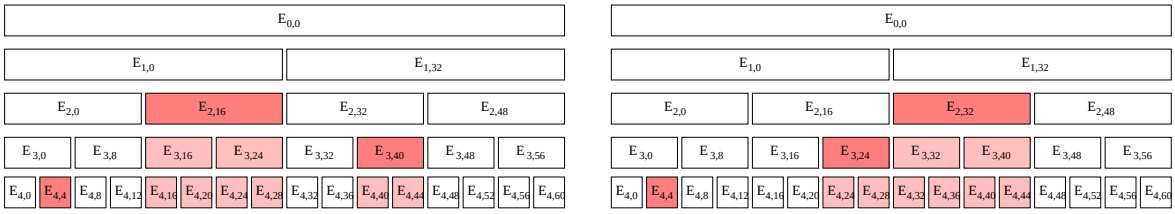


Figura 5.10: Situación antes (a) y después (b) de aplicar el algoritmo de reordenación.

Así pues, con el intercambio de los conjuntos adecuados, se puede pasar de una situación en la que algunos conjuntos singulares no mantienen la relación de orden de la Definición 5, a otra situación donde los nuevos conjuntos singulares sí que mantienen ese orden. Esto queda reflejado de manera formal en el siguiente teorema.

Teorema 9 *Sean $E_{k,l}$ y $E_{i,j}$ los únicos conjuntos singulares de los niveles k e i, respectivamente, con $0 < i < k \leq 6$, $l \in I_k$ y $j \in I_i$, tales que no mantienen una relación de orden entre sí, es decir $E_{k,l} \not\blacktriangleleft E_{i,j}$. Es posible pasar a otra situación con $E_{k,n} \blacktriangleleft E_{i,m}$, con $n \in I_k$ y $m \in I_i$, donde $E_{k,n}$ y $E_{i,m}$ son ahora los únicos conjuntos singulares de los niveles k e i, respectivamente, al tiempo que se mantienen todas las peticiones atendidas hasta ese momento.*

Demostración: La Propiedad 5 permite intercambiar los conjuntos $E_{i,j}$ y $E_{i,m}$, con $m = Ancestor(l, i)$. De esta forma, el conjunto singular en el nivel i es ahora $E_{i,m}$ mientras que $E_{i,j}$ ya no es libre. Lo mismo sucede con el conjunto singular en el nivel k, que es ahora $E_{k,n}$, con $n = l - m + j$. Lo que hay que probar es que $E_{k,n} \blacktriangleleft E_{i,m}$. Para ello bastaría con probar que $E_{k,n} \blacktriangleleft E_{k,l}$, pues a partir de ahí, y aplicando la Propiedad 7, se obtiene que $E_{k,n} \blacktriangleleft E_{i,m}$.

Supongamos que $E_{k,n} \not\blacktriangleleft E_{k,l}$, es decir, $n \geq l$. Pero como $n \neq l$, pues si no $E_{k,n}$ y $E_{k,l}$ serían el mismo conjunto, entonces $n > l$. Pero, entonces $E_{k,l} \blacktriangleleft E_{k,n}$, y por la Propiedad 7, entonces $E_{k,l} \blacktriangleleft E_{i,j}$, que no es cierto por las hipótesis del enunciado. Así pues, $E_{k,l} \not\blacktriangleleft E_{k,n}$, $n < l$, y $E_{k,n} \blacktriangleleft E_{k,l}$, y por la Propiedad 7, $E_{k,n} \blacktriangleleft E_{i,m}$.

□

Una vez que se puede conseguir que dos conjuntos singulares que no mantienen el orden deriven en otros dos conjuntos singulares, pero que ya mantienen el orden,

aplicando esto de forma sucesiva se puede llegar a una tabla totalmente ordenada. Este resultado lo proporciona el siguiente teorema.

Teorema 10 *El algoritmo de reordenación permite pasar de una tabla no ordenada a otra que sí lo está.*

Demostración: El algoritmo de reordenación aplica el Teorema 9 sucesivamente a cualquier pareja de conjuntos $E_{i,j}$ y $E_{k,l}$ singulares, con $0 < i < k \leq 6$, $j \in I_i$ y $l \in I_k$, tal que $E_{k,l} \blacktriangleleft E_{i,j}$. Después de aplicar dicho teorema se tiene que $E_{k,n} \blacktriangleleft E_{i,m}$, con $n = l - m + j$ y $m = Ancestor(l, i)$. De esta forma, tras aplicar el algoritmo de reordenación la tabla queda ordenada.

□

Como se ha indicado en la descripción del algoritmo de reordenación, su funcionamiento es similar al del algoritmo de ordenación de selección directa. En la medida en que esto es así, queda aún más garantizado que el algoritmo de reordenación aquí presentado funciona correctamente y obtiene una tabla ordenada.

Para concluir con el algoritmo de reordenación, veamos qué ocurre cuando dicho algoritmo se aplica a una tabla que está normalizada, y concretamente si ésta sigue estando o no normalizada. Para ello nos vamos a apoyar en que el algoritmo de reordenación no genera más conjuntos singulares de los que ya existían, y se limita solamente a cambiarlos de orden. Veámoslo en el siguiente teorema.

Teorema 11 *Tras aplicar el algoritmo de reordenación sobre una tabla normalizada la tabla sigue normalizada.*

Demostración: Si la tabla estaba normalizada es porque a lo sumo tenemos un conjunto singular por cada nivel. Pero, el algoritmo de reordenación siempre intercambia un conjunto $E_{i,j}$ singular con otro conjunto $E_{i,m}$ no libre, con lo que nunca genera más conjuntos libres de los que hubiera en cada nivel. De esta forma, a lo sumo podremos tener un conjunto singular por nivel, con lo que la tabla también queda normalizada.

□

Una vez presentados los algoritmos de reserva, desfragmentación y reordenación sólo queda enlazar todo lo visto hasta ahora y comprobar que el tratamiento de la tabla globalmente es adecuado y correcto.

5.4. Gestión global de la tabla

Para el tratamiento global de la tabla, en el que se deben contemplar tanto inserciones como eliminaciones, se ha mostrado la necesidad de hacer uso de una combinación del algoritmo de reserva y del algoritmo de desfragmentación (y en su caso del algoritmo de reordenación). Ahora bien, las propiedades del algoritmo de reserva han sido probadas en unas determinadas condiciones, las que surgen de no considerar eliminación de peticiones. Sin embargo, al considerar eliminaciones y hacer uso del algoritmo de desfragmentación (y en su caso del algoritmo de reordenación) se debe comprobar en qué situación queda la tabla y si el estado de ésta es o no el adecuado para que se sigan cumpliendo las propiedades del algoritmo de reserva, y se siga cumpliendo, por ejemplo, el Teorema 4. Vamos a demostrar que la situación final que se alcanza en la tabla tras producirse una serie de inserciones y eliminaciones, con sus desfragmentaciones y reordenaciones correspondientes, es equivalente a la que se hubiera alcanzado tan sólo con las inserciones de las peticiones que permanecen finalmente en dicha tabla. El término equivalente tiene que ver con la capacidad de atender las peticiones solicitadas, y se define más exactamente a continuación.

Definición 7 *Decimos que dos tablas T y T' son equivalentes si pueden atender las mismas peticiones.*

Es decir, independientemente de cuales sean en concreto los conjuntos libres que haya en esas tablas, ambas deben ser capaces de atender las mismas peticiones. Como se va a ver a continuación, dos tablas serán equivalentes si tienen el mismo número de conjuntos singulares y están distribuidos en los mismos niveles, ya que si no fuera así se podría atender una petición en una de ellas, y no en la otra por no disponer de un conjunto singular del nivel correspondiente.

A modo de ejemplo, las tablas mostradas en las Figuras 5.7(a) y 5.7(b) no son equivalentes a pesar de tener el mismo número de entradas libres, pues con la segunda de ellas se pueden atender peticiones de tipo $d \geq 2$, mientras que con la primera de ellas tan sólo podemos atender peticiones de tipo $d' \geq 4$. De igual forma, las tablas mostradas en las Figuras 5.8(a) y 5.8(b) tampoco son equivalentes. Veamos a continuación un ejemplo con dos tablas que sin ser estrictamente iguales sí que son equivalentes.

Ejemplo 8 *En la Figura 5.11 podemos ver dos tablas que son equivalentes. En la mostrada en la Figura 5.11(a) los conjuntos singulares son $E_{4,4}$, $E_{3,24}$ y $E_{2,32}$, mientras que en la Figura 5.11(b) los conjuntos singulares son $E_{4,8}$, $E_{3,16}$ y $E_{2,48}$, estando todos los demás conjuntos (no descendientes de esos) ocupados. Sin embargo, ambas tablas son capaces de atender exactamente a las mismas secuencias de peticiones.*

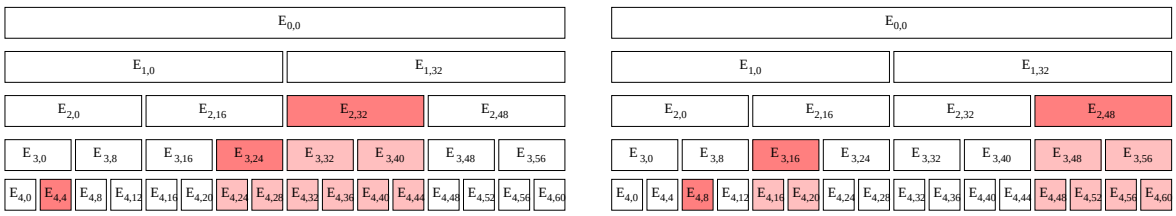


Figura 5.11: Dos tablas equivalentes.

A continuación vamos a ver un teorema para demostrar que dos tablas que estén normalizadas y que tengan el mismo número de conjuntos singulares, tienen el mismo número de entradas libres.

Teorema 12 *Dadas dos tablas T y T' normalizadas, tales que $\forall E_{i,j} \in T$, con $E_{i,j}$ singular, $\exists E_{i,k} \in T'$, con $E_{i,k}$ también singular, y $\forall E_{i,k} \in T'$, con $E_{i,k}$ singular, $\exists E_{i,j} \in T$, con $E_{i,j}$ singular, con $0 < i \leq 6$ y $j, k \in I_i$, entonces ambas tablas tienen el mismo número de entradas libres.*

Demostración: Vamos a hacer la demostración por reducción al absurdo. Supongamos que ambas tablas T y T' no tienen el mismo número de entradas. De esta forma, debe existir un nivel l, con $0 \leq l \leq 6$, a partir del cual (hacia niveles inferiores) las tablas no tengan el mismo número de entradas libres. Como el número de conjuntos singulares es el mismo en todos los niveles, entonces debe haber un número n de conjuntos libres en una de las tablas, que no sean singulares, que no estén en la otra. Para que esos n conjuntos libres no sean singulares deben ser tales que sus conjuntos hermanos estén también libres. Pero esto formaría en una tabla uno o más conjuntos singulares en algún nivel t, con $0 < t < i$, que no están en la otra, lo cual es una contradicción con la hipótesis de partida.

Así pues, si dos tablas T y T' están normalizadas, y $\forall E_{i,j} \in T$, con $E_{i,j}$ singular, $\exists E_{i,k} \in T'$, con $E_{i,k}$ también singular, y $\forall E_{i,k} \in T'$, con $E_{i,k}$ singular, $\exists E_{i,j} \in T$, con $E_{i,j}$ singular, con $0 < i \leq 6$ y $j, k \in I_i$, entonces ambas tablas tienen el mismo número de entradas libres.

□

Utilizando el teorema anterior, se va a demostrar que dos tablas que estén normalizadas son equivalentes si tienen el mismo número de conjuntos singulares y en los mismos niveles.

Teorema 13 *Dadas dos tablas T y T' normalizadas, tales que $\forall E_{i,j} \in T$, con $E_{i,j}$ singular, $\exists E_{i,k} \in T'$, con $E_{i,k}$ también singular, y $\forall E_{i,k} \in T'$, con $E_{i,k}$ singular, $\exists E_{i,j} \in T$, con $E_{i,j}$ singular, con $0 < i \leq 6$ y $j, k \in I_i$, entonces ambas tablas son equivalentes.*

Demostración: Si T y T' son normalizadas es porque tienen a lo sumo un conjunto singular por nivel. Como además $\forall E_{i,j} \in T$, siendo $E_{i,j}$ singular, $\exists E_{i,k} \in T'$, con $E_{i,k}$ también singular, $0 < i \leq 6$, y viceversa, por el Teorema 12 ambas tablas tienen el mismo número de entradas libres. Así pues, por el Teorema 4 las tablas T y T' son capaces de atender a las mismas peticiones, y como consecuencia son equivalentes.

□

Veamos ahora la demostración de que la situación resultante tras insertar y eliminar peticiones es equivalente a la situación que se obtendría si sólo se hubieran producido las peticiones que permanecen tras las eliminaciones. Así, todo lo desarrollado para el caso en que no había eliminaciones será también aplicable al caso de tener eliminación de peticiones (considerando los algoritmos de desfragmentación y reordenación), pues tendremos siempre una situación equivalente de cara a nuevas inserciones, y en ese caso será posible aplicar el Teorema 4. De esta forma conseguiremos relacionar ambas partes de la teoría desarrollada. Veamos el teorema para demostrar esa equivalencia.

Teorema 14 *Sean una serie de peticiones $d_1, d_2, \cdots, d_n$ seguidas de una serie de eliminaciones $d'_1, d'_2, \cdots, d'_m$, tras cada una de las cuales se aplica reordenación y/o desfragmentación, según sea necesario, para dejar la tabla normalizada y ordenada, donde $\exists d_i / d'_j = d_i$, $\forall j \in [1, m]$. La secuencia de inserciones d_i seguidas por las eliminaciones d'_j produce una tabla equivalente a la que se obtendría si sólo se hubieran producido las inserciones $\{d_1, d_2, \cdots, d_n\} - \{d'_1, d'_2, \cdots, d'_m\}$, que son las que permanecen tras las eliminaciones, una vez hechas todas las inserciones.*

Demostración: Sea T la tabla resultante después de hacer las inserciones de las peticiones $d_1, d_2, \cdots, d_n$, seguidas en cualquier orden de las eliminaciones $d'_1, d'_2, \cdots, d'_m$. Por otra parte, sea T' la tabla resultante si se hubieran hecho directamente las inserciones $\{d_1, d_2, \cdots, d_n\} - \{d'_1, d'_2, \cdots, d'_m\}$. Lo que queremos demostrar es que T y T' son equivalentes. Para ello vamos a ver que ambas tablas tienen el mismo número de conjuntos singulares y situados en los mismos niveles, y de esta forma (de acuerdo con el Teorema 13) ambas son capaces de atender exactamente las mismas peticiones.

Tal y como se planteaba en la Sección 5.1 en la página 130 tenemos dos opciones para plantear el funcionamiento dinámico de la gestión de la tabla:

1. Después de cada inserción d_i y de cada eliminación d'_j se aplica el algoritmo de desfragmentación, y por tanto, por el Teorema 7 la tabla T queda normalizada. Por otra parte, de acuerdo con el Teorema 2 la tabla T' también está normalizada.

2. Después de cada eliminación d'_j se aplica el algoritmo de reordenación y/o el algoritmo de desfragmentación, y por tanto, de acuerdo con los Teoremas 7, 8 y 11, la tabla T está normalizada. Igual que antes, por el Teorema 2 la tabla T' también está normalizada.

Luego en cualquier caso, ambas tablas T y T' están normalizadas, y por tanto a lo sumo tienen un único conjunto singular por nivel. Veamos que además, ambas tablas tienen el mismo número de conjuntos singulares y situados en los mismos niveles.

Si T y T' no tienen el mismo número de conjuntos singulares o, teniendo el mismo número, éstos no están situados en los mismos niveles, es porque para algún nivel i, con $0 < i \leq 6$, $\exists E_{i,j} \in T$, $j \in I_i$, con $E_{i,j}$ singular, y sin embargo $\nexists E_{i,k} \in T'$, $k \in I_i$, con $E_{i,k}$ singular. Los conjuntos $E_{i,l} \in T'$ no libres, $l \in I_i$, podrían ser debidos a peticiones de tipo $d = 2^i$, o por la Propiedad 2, a peticiones de tipo $d' = 2^{i+p}$, con $1 \leq p \leq 6-i$ de forma que $E_{i,l} = \bigcup_{m=0}^{2^p-1} E_{i+p,j+m\times 2^{6-(i+p)}}$. Esto supondría que la petición de tipo $d = 2^i$ (o de forma similar, la de tipo $d' = 2^{i+p}$) en la que difieren ambas tablas, estaría en la secuencia de peticiones d_1, d_2, $\cdots$, d_n y no en la secuencia $\{d_1, d_2, \cdots, d_n\} - \{d'_1, d'_2, \cdots, d'_m\}$, lo cual sabemos que no es posible.

Así pues, se ha demostrado que ambas tablas T y T' tienen el mismo número de conjuntos singulares y situados en los mismos niveles, y por el Teorema 13, son capaces de atender las mismas peticiones. Es decir, se ha demostrado que ambas tablas son equivalentes.

□

En definitiva, con la demostración de este teorema, se han relacionado ambas partes de la teoría desarrollada, y tenemos que en cualquier situación es posible aplicar el Teorema 4 de cara a nuevas inserciones. Es decir, cualquiera que sea la secuencia de peticiones de inserción, seguidas en cualquier orden por eliminaciones de algunas de esas peticiones ya ubicadas en la tabla, la combinación de los tres algoritmos desarrollados permite ubicar siempre en la tabla cualquier secuencia de peticiones que no requiera más entradas de las que haya libres, sean estas peticiones de los tipos que sean.

Consideramos que este resultado es muy importante pues se han propuesto y probado unos algoritmos para manejar la tabla de arbitraje de InfiniBand que consiguen garantizar QoS a las aplicaciones.

5.5. Resumen

En este capítulo se han abordado las distintas situaciones posibles cuando se van atendiendo y retirando peticiones de la tabla. Se han presentado las situaciones que

pueden ocurrir e identificado dos nuevos problemas que pueden llegar a suceder cuando se producen reservas y liberaciones de peticiones en la tabla.

Para solucionar estos nuevos problemas se han propuesto dos nuevos algoritmos. El primero de ellos es un algoritmo de desfragmentación que se ejecutará cada vez que haya una liberación de una secuencia de la tabla. La idea de este algoritmo es unir secuencias aisladas para formar otras secuencias capaces de atender peticiones más restrictivas. Para ello, puede que deba procederse al intercambio entre una secuencia de entradas libre y otra no libre, de cara a unir las dos que estaban libres y formar otra secuencia mayor con la que se pueda atender a peticiones más restrictivas.

El segundo algoritmo propuesto es de reordenación. La idea es que resulta más interesante que las secuencias libres queden en un determinado orden para que el algoritmo de reserva actúe de la manera más eficiente posible. Igual que en el caso anterior, para conseguir sus propósitos, el algoritmo procederá al intercambio de secuencias de entradas.

En la última parte de este capítulo se han relacionado ambas partes de la teoría desarrollada. Se ha probado que la combinación del algoritmo de reserva en la tabla, junto con los algoritmos de desfragmentación y reordenación cuando hay liberaciones en dicha tabla, permiten hacer un uso de forma dinámica de la tabla cuando hay peticiones y eliminaciones de establecimiento de conexión.

En resumen, en este capítulo y el anterior, se ha propuesto un modelo formal para la gestión de la tabla de arbitraje de InfiniBand proporcionando garantía de QoS a las aplicaciones. Las garantías pueden ser de ancho de banda y/o de latencia máxima. El modelo desarrollado se basa en la categorización realizada en capítulos anteriores de las posibles distancias máximas a solicitar por las aplicaciones, y de la segregación del tráfico con distintos requisitos en distintos canales virtuales. Utilizando los algoritmos propuestos en ambos capítulos, se ha probado que se consigue proporcionar a las aplicaciones el nivel que necesiten de QoS, y que ellas habrán previamente solicitado de cara a que las entidades encargadas configuren, entre otras cosas, las tablas de arbitraje de la manera adecuada. Así pues, las propuestas realizadas pueden ser utilizadas para configurar las tablas de arbitraje de cara a proporcionar esa QoS requerida por las aplicaciones.

Capítulo 6

Evaluación de prestaciones

El último de los objetivos de este trabajo ha sido la evaluación de las distintas propuestas presentadas en los capítulos anteriores. Como en otros estudios de similares características, se han empleado técnicas de simulación para este propósito. De esta forma, se verá de una manera experimental que las propuestas demostradas formalmente en los capítulos anteriores consiguen las prestaciones esperadas.

El proceso de evaluación consiste básicamente en dos etapas: en la primera de ellas se somete al modelo que incorpora nuestras propuestas a una amplia batería de pruebas que se diferencian unas de otras fundamentalmente en las condiciones de carga de la red. La segunda etapa consiste en recoger y analizar los resultados que diversos índices de prestaciones han arrojado tras completarse las simulaciones.

En este capítulo, por tanto, se presentan todos aquellos detalles que tienen que ver con el proceso de evaluación de prestaciones. Así, y después de justificar el método de evaluación utilizado, se realiza una descripción de la herramienta usada para ello. En la medida en que ha sido la simulación la técnica elegida para desarrollar la primera fase del proceso de evaluación, ha sido necesario desarrollar un simulador que modelara con un cierto detalle una subred InfiniBand, y con la flexibilidad adecuada para poder incluir las propuestas propias de esta tesis.

Por las características del estudio realizado se ha incluido un conjunto de índices de prestaciones específicos, al margen de los habitualmente considerados. Todos ellos se presentan y comentan en las siguientes secciones.

Finalmente, y ocupando gran parte de este capítulo, se presenta una amplia muestra de los resultados obtenidos. Estos se han agrupado atendiendo a diversos parámetros de diseño que han sido considerados, de tal forma que se podrá conocer además del comportamiento global de las propuestas aquí realizadas, la influencia particular que tienen esos parámetros en dicho comportamiento.

6.1. Método de evaluación

Existen diferentes métodos para la evaluación de las prestaciones de un sistema [Jai91]. Por un lado, se pueden efectuar mediciones reales sobre un sistema existente. En este caso es necesario disponer de al menos un prototipo del sistema. Los principales inconvenientes de este método son la necesidad de un esfuerzo previo de desarrollo, y la falta de versatilidad, pues para probar diferentes opciones de diseño hay que elaborar distintos prototipos.

Otra posibilidad a la hora de abordar una evaluación de prestaciones es elaborar un modelo analítico del sistema. Se trata de una alternativa económica, que proporciona resultados de forma inmediata. Sin embargo, su utilización depende del nivel de detalle requerido para los resultados. Además, para que el modelo sea susceptible de una evaluación analítica, se debe simplificar notablemente el modelo del sistema, con la consiguiente pérdida de precisión de los resultados.

La última alternativa posible en cuanto a evaluación de prestaciones es la simulación. Es un método utilizado ampliamente en muchos campos de investigación. Mediante la simulación por ordenador se puede evaluar con precisión cualquier sistema que no pueda ser evaluado analíticamente debido a su complejidad. Además, es posible considerar distintos niveles de detalle en el sistema. El principal inconveniente de este método es que, normalmente, suele requerir un elevado tiempo de ejecución para realizar una simulación del sistema a nivel detallado.

Una vez realizado el simulador, hay que introducir la carga de prueba para obtener resultados. En concreto, cuando se trata de evaluar elementos de interconexión, la carga consiste en el tráfico inyectado en la red. Existen diversas posibilidades para modelar la carga de un sistema. La aproximación más extendida es el empleo de *cargas sintéticas*, donde el tráfico inyectado se genera aleatoriamente, siguiendo determinados patrones, tanto de distribución de los destinos para los mensajes, como de tamaño y frecuencia de los mismos. De esta manera, la carga se genera de forma sencilla y rápida, aunque no siempre modela perfectamente el tráfico que circula por la red en el sistema real.

Otra posibilidad para modelar la carga del sistema es emplear trazas obtenidas a partir de sistemas reales. La idea es almacenar la información acerca del tráfico medido en un sistema real en los denominados *ficheros de traza*. En estos ficheros se almacenan datos tales como tamaño, origen, destino o tiempo de generación de los mensajes, aunque el formato concreto depende de la aplicación específica. El problema de este método es que el tráfico registrado en la traza depende de muchos factores del sistema donde se ha obtenido, y no siempre es extrapolable a otros sistemas donde varía alguno de dichos factores.

En concreto, para modelar tráfico de una subred, habría que capturar el tráfico presente en una subred dada e ir almacenando las características de los paquetes que

estén circulando y que más tarde se quieran reproducir (origen, destino, tamaño, etc.). Esto genera enormes ficheros que suelen ser difíciles de manejar. Otro problema de esta aproximación es el tema de la confidencialidad. El fichero de traza no puede reflejar hechos concretos que haya realizado un usuario, es decir, no se debe poder identificar *quién ha hecho qué*. Para ello se suelen alterar las direcciones origen y destino mediante algún tipo de alias, lo que la mayoría de las veces resta validez a este tipo de estrategia.

Otra posibilidad para modelar la carga del sistema consiste en emplear carga real de aplicaciones. Para ello se utilizan los denominados *simuladores dirigidos por ejecución*. Este tipo de simuladores ejecutan realmente una aplicación sobre el sistema simulado. Se suele simular la ejecución completa de una aplicación paralela de complejidad media-alta sobre el sistema simulado, y todo ello se ejecuta típicamente en un monoprocesador. Todo esto hace su ejecución especialmente costosa en recursos computacionales. Por tanto, cuando se trata de evaluar un conjunto de parámetros de tamaño moderado, esta aproximación resulta inviable.

Esta última aproximación resulta por contra especialmente útil cuando lo que se quiere medir es el comportamiento de un sistema (por ejemplo la red de interconexión) con una determinada aplicación. Sin embargo, es difícilmente utilizable cuando lo que se quiere modelar es el comportamiento habitual de un sistema con múltiples aplicaciones interactuando. Así pues, ésta no sería una buena opción para modelar el comportamiento de una subred donde se tienen múltiples aplicaciones enviando y recibiendo peticiones de distintas clases y protocolos, y por tanto con muy distintas características y requisitos.

En el caso del trabajo que nos ocupa hubiera sido deseable la medición real sobre un sistema existente, si se hubiera dispuesto de una subred InfiniBand. En estos momentos el grupo de investigación está en vías de adquirir equipos suficientes para poder montar una subred InfiniBand. Tan pronto como se disponga de los recursos necesarios se probarán las propuestas realizadas en este trabajo sobre un entorno real.

En los capítulos anteriores se ha propuesto un modelo formal bajo una serie de hipótesis de partida. En dichos capítulos se ha realizado una validación del modelo formal presentado, mediante la demostración de una serie de teoremas. Ha quedado probado allí que el modelo propuesto es capaz de proporcionar garantía de ancho de banda y/o latencia máxima a las aplicaciones que lo requieran.

Sin embargo, de cara a dar mayor robustez a las propuestas realizadas, se ha considerado oportuno proceder también a una evaluación mediante simulación, pues es una técnica capaz de ofrecer tanto el nivel de detalle como la versatilidad necesaria para este tipo de estudio. En la siguiente sección se describe la herramienta de simulación utilizada, así como la interacción de un usuario con la herramienta, el modelo de red y el modelo de tráfico que se utiliza.

6.2. Herramienta de simulación

Con el objeto de evaluar las propuestas realizadas, se ha desarrollado una herramienta de simulación que modela el funcionamiento de una subred InfiniBand con un elevado nivel de detalle. La herramienta se ha desarrollado en el lenguaje de programación C++. Partiendo de otros simuladores para entornos similares [Cas01, Fli01, Cam02], se han adaptado detalles de ellos para modelar las particularidades que presentan las especificaciones de InfiniBand. Los resultados ofrecidos por el simulador han sido exhaustivamente cotejados para asegurar que cumplen con las especificaciones de InfiniBand.

El objetivo fundamental de esta herramienta es validar las propuestas realizadas para proporcionar garantía de QoS por parte de la subred a las aplicaciones que lo requieran. Para ello, el simulador modela una de estas subredes, implementando con suficiente nivel de detalle los conmutadores, junto con las tarjetas de interfaz de red conectadas a cada puerto de entrada.

La simulación está conducida por eventos. La transmisión de datos se simula al nivel de flit, estando la base de tiempos a nivel de phit. Los tiempos de los distintos eventos modelados han sido fijados de acuerdo con los criterios utilizados en otros estudios similares, estando siempre de acuerdo con las especificaciones de InfiniBand.

La interacción con el simulador se realiza mediante un fichero de configuración donde se especifican las características de la simulación. Este planteamiento es similar al de otro tipo de trabajos parecidos [Cas01]. En este fichero de configuración se detalla por ejemplo la topología de la subred, indicando las conexiones entre conmutadores y entre hosts.

El encaminamiento se realiza usando las tablas de encaminamiento que hay en cada conmutador. Dichas tablas se han generado con herramientas al margen de este simulador [Fli01], pero respetando las especificaciones de InfiniBand. El algoritmo de encaminamiento utilizado es up*/down*, aunque adaptado a las características de InfiniBand [SRF01].

El simulador modela el proceso de establecimiento de conexión con una cierta demanda de requisitos. Para el establecimiento de conexiones se ha implementado un protocolo similar a RSVP [BZB97]. Los hosts generan aleatoriamente solicitudes de conexión con otros hosts, teniendo cada solicitud unos determinados requisitos en cuanto a ancho de banda y latencia. Estas solicitudes se estudian de forma distribuida en cada uno de los conmutadores por los que pasa, así como en los hosts origen y destino. Más adelante se explicará con más detalle cómo se realiza este proceso.

Pasemos ahora a describir el modelo de red utilizado por el simulador. Se describirá con detalle la arquitectura implementada, así como las decisiones que se han tomado

durante el desarrollo de este trabajo. A continuación, se describirá el modelo de red implementado en el simulador, cómo se modela el tráfico que se simulará, y qué otras características se han considerado.

6.2.1. Modelo de red

Para la evaluación de las propuestas realizadas se van a utilizar tanto redes con topología regular (hipercubo y malla), como irregular. Las redes irregulares se han generado aleatoriamente, pero teniendo siempre unas características comunes: todos los conmutadores tienen ocho puertos, de los cuales cuatro se utilizan para conexiones con otros conmutadores y los otros cuatro para conexiones con hosts. Cada host dispone sólo de una interfaz de red, con lo que estará conectado a un único conmutador. Así pues, cada conmutador tendrá cuatro hosts conectados a él, y ese conmutador es la única vía que van a tener esos hosts para comunicarse con otros hosts. Este modelo de red ha sido ampliamente utilizado en otros estudios similares [SD97, Fli01, Cas01].

InfiniBand no especifica cómo debe ser la arquitectura de los interfaces de los hosts ni de los conmutadores, tan sólo especifica cómo debe ser su funcionamiento. Tampoco hay un criterio claro entre los fabricantes que ya tienen productos InfiniBand comercializados, o por lo menos no es público el modelo que utilizan. Para la evaluación a realizar en este trabajo se ha supuesto un modelo de conmutador con buffers a la entrada y a la salida. Además, el crossbar está multiplexado entre los distintos canales virtuales. Por supuesto, este modelo cumple con las especificaciones de InfiniBand y permitiría implementar tanto el arbitraje de salida como la correspondencia programada entre niveles de servicio y canales virtuales. La arquitectura del conmutador modelado sería como la mostrada en la Figura 6.1.

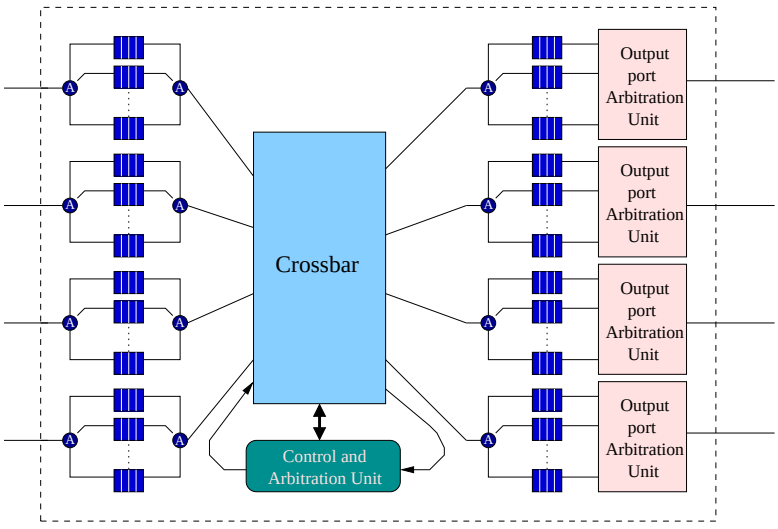


Figura 6.1: Modelo de conmutador utilizado para la simulación. Dispone tanto de buffers a la entrada como a la salida y el crossbar está multiplexado.

Tanto los buffers de la entrada del conmutador como los de la salida se han considerado del mismo tamaño. En todos los casos el tamaño del buffer es proporcional al tamaño de paquete. En esta evaluación se han utilizado varios tamaños de paquetes, con valores situados entre 256 y 4096 bytes. Estos valores corresponden al menor y mayor MTU, según las especificaciones de InfiniBand. En la mayoría de las pruebas realizadas, los buffers, tanto de entrada como de salida, tendrán un tamaño que permita almacenar cuatro paquetes. Sin embargo, en una de las pruebas realizadas se varía este aspecto probando otros tamaños de buffer.

Como se ha comentado anteriormente, en crossbar con muchos puertos, o con un elevado número de canales virtuales, no es viable utilizar crossbar no multiplexados. Así pues, en el modelo de conmutador que se va a utilizar en las simulaciones, el crossbar está multiplexado entre los distintos canales virtuales, con lo que cada crossbar tiene tantas entradas como puertos. De esta forma, no podrán cruzar el crossbar a la vez dos paquetes desde el mismo puerto de entrada, aunque salgan desde dos canales virtuales distintos, ni aunque vayan a dos puertos distintos de salida del conmutador. Para que el crossbar no se convierta en un cuello de botella de la arquitectura, se ha considerado que es capaz de funcionar al doble de velocidad que lo haga el enlace, lo que ya utilizan algunos conmutadores comerciales [Avi]. De esta forma, para una velocidad del enlace de 2,5 Gbps, el crossbar funcionaría a 5 Gbps. También se ha considerado que la anchura del crossbar es de 16 bits, que para las velocidades de InfiniBand supone que el ciclo de reloj del conmutador sea $3,2$ nseg. El valor del ciclo de reloj, para las tres velocidades de InfiniBand, es el mostrado en la Tabla 6.1.

Velocidad	Ancho de banda (Gbps)	Ciclo de Reloj (nseg)
1x	2,5	3,2
4x	10	0,8
12x	30	0,27

Tabla 6.1: Valor del ciclo de reloj para las tres velocidades de InfiniBand.

Hay que señalar que el hecho de que varíe el ciclo de reloj para cada una de las velocidades de InfiniBand es sólo a efectos de simulación. Según las especificaciones de InfiniBand, las distintas velocidades se consiguen utilizando 1, 4 ó 12 líneas en paralelo, pero todas ellas funcionando a 2,5 Gbps. De cara a la simulación, para modelar que se tarda n veces menos en transmitir un paquete porque se dispone de n líneas en paralelo, ha resultado más sencillo suponer que el ciclo de reloj se ha rebajado n veces. El resto de eventos del simulador siguen tardando el mismo tiempo absoluto, aunque distinto número de ciclos en función del valor establecido para el ciclo de reloj. Así pues, esta suposición no afecta a los resultados obtenidos, pues éstos son siempre relativos al ciclo de reloj utilizado.

En cuanto al modelo de host implementado, se ha modelado una arquitectura de host similar a la de otro tipo de estudios [Fli01, Cas01, Cam02], que cumple con las

especificaciones de InfiniBand. Los hosts disponen de buffers a la salida, entre los cuales se aplica arbitraje. Se ha supuesto que los paquetes llegan a esos buffers de salida directamente desde el espacio de memoria de usuario por medio de algún dispositivo de DMA. De igual forma, cuando los paquetes llegan al host no se almacenan en ningún tipo de buffer, y se llevan directamente al espacio de direcciones del usuario. Este esquema se muestra en la Figura 6.2.

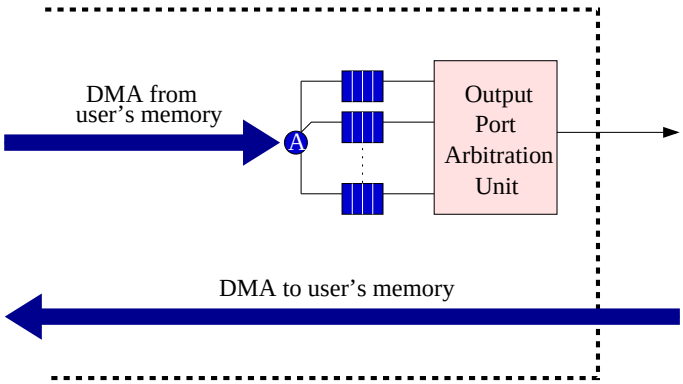


Figura 6.2: Modelo de host utilizado para la simulación. Dispone de buffers a la salida, mientras que a la entrada es directamente el DMA el que lleva los datos recibidos al espacio de memoria del usuario.

Como puede observarse, en la arquitectura modelada tanto del conmutador como del host, se han realizado muchas simplificaciones en aspectos que no eran fundamentales para este estudio. Esto ha permitido acelerar las simulaciones, centrando el estudio en los aspectos que resultaban fundamentales, pero sin pérdida de rigurosidad.

Por otra parte, respecto al tamaño de la red, se han evaluado redes desde 8 conmutadores hasta 64 conmutadores, conteniendo de esta forma cada red desde 32 hosts hasta 256 hosts, respectivamente. En la mayoría de las pruebas se van a utilizar redes de 16 conmutadores (con 64 hosts), aunque se harán pruebas con el resto de tamaños.

Por último, se ha considerado si el arbitraje del puerto de salida debería introducir o no un pequeño retraso. Hay que considerar que este arbitraje puede hacerse mientras está terminando de salir el paquete anterior. En este caso, no consumiría tiempo adicional. Sólo en el caso de que no hubiera un paquete saliendo anteriormente debería contársele un tiempo adicional. Pero esta situación es bastante particular, pues si no estaba saliendo ningún paquete anteriormente es que no hay ningún otro paquete disponible en los buffers de los canales virtuales del puerto de salida listo para abandonar el conmutador, pues si lo hubiera estaría saliendo y no estaría el puerto ocioso. De esta forma, en este caso sólo hay un paquete para ser seleccionado, con lo cual tampoco es necesario hacer tal selección y el paquete podría abandonar el conmutador de forma inmediata sin necesidad de ningún arbitraje. Así pues, el arbitraje del puerto de salida no tiene por qué introducir ningún tiempo adicional, pues en caso de que haya

competencia se puede solapar con la salida del paquete actual, y si no la hay no sería necesario tal arbitraje, con lo que tampoco se introduciría ningún retraso.

6.2.2. Modelo de tráfico

El objetivo de este trabajo es evaluar las propuestas realizadas para conseguir proporcionar QoS a las aplicaciones que lo demanden. Para ello, en el simulador se ha modelado tráfico con una gran variedad de requisitos en cuanto a ancho de banda y latencia máxima.

Se ha utilizado tráfico CBR pues para dar una garantía absoluta es necesario hacer un correcto dimensionado de los recursos de la red. Para ello, las aplicaciones deben tener un comportamiento constante, respetando los requisitos que solicitaron. En trabajos anteriores [ASO02] se han realizado pruebas con tráfico VBR, mostrando que las aplicaciones obtienen la QoS solicitada aunque, debido a la variabilidad, no es posible proporcionar garantía absoluta. No se van a incluir aquí resultados para tráfico VBR pues al no poderle garantizar QoS no se ha considerado directamente un objetivo. Así pues, en la evaluación que se va a realizar en este trabajo se va a utilizar sólo tráfico CBR con un amplio rango de requisitos de ancho de banda y latencia máxima.

Aunque el tráfico PBE, BE o CH no tiene requisitos de QoS en términos de garantía de ancho de banda o latencia máxima, en este trabajo también se les va a reservar un espacio en la red. En concreto, se va a suponer que el 20 % del ancho de banda disponible en cada enlace está reservado para estos tipos de tráfico. Cómo se haga el reparto de este ancho de banda entre estas clases de tráfico no es relevante, y dependerá de los intereses del administrador de la red.

Como se ha comentado anteriormente, se va a modelar el tráfico con necesidades de QoS mediante el establecimiento de conexiones con unos determinados requisitos. Los requisitos de cada conexión dependerán del nivel de servicio de esa conexión, y serán generados aleatoriamente dentro del rango establecido por dicho nivel de servicio.

Para el establecimiento de las conexiones se ha implementado un protocolo similar al estándar RSVP [BZB97]. El host origen solicita unos determinados requisitos que los nodos intermedios y el host destino deben ser capaces de satisfacer. Si es así, la conexión se establece y pueden comenzar a fluir datos a través de ella. Si en algún nodo intermedio, o en el host destino, no se pueden satisfacer los requisitos solicitados, se denegará la conexión. Ese nodo incapaz de satisfacer los requisitos solicitados informará de este hecho a los nodos precedentes en la ruta y al host origen, de cara a que liberen los recursos que ya tenían reservados para esa conexión.

Si la sonda de establecimiento de conexión llega hasta el host destino y éste también es capaz de satisfacer los requisitos solicitados, la conexión se va a establecer. En este

caso, el host destino envía un paquete de asentimiento informando al resto de nodos de la ruta de que la conexión se ha conseguido establecer, y al host origen para que pueda empezar a enviar información por esa conexión.

Así pues, el simulador implementa un modelo orientado a conexión donde las conexiones se aceptan o deniegan de forma distribuida con la información de que disponen los agentes locales. La otra alternativa hubiera sido un enfoque centralizado donde el Subnet Manager fuera el encargado de recopilar toda la información, de decidir las conexiones que se aceptan o se deniegan y de informar a los nodos de las modificaciones que deben hacer para ir adaptándose. Como se señaló en su momento, esta segunda alternativa tiene como gran problema el gran tráfico que puede suponer hacia y desde el Subnet Manager. De todas formas, la estrategia de establecimiento de conexiones no afecta al resto del trabajo desarrollado, y el tratamiento de la tabla de arbitraje puede hacerse igual, tanto si lo modifica de forma centralizada el Subnet Manager, como si lo hacen de forma distribuida cada uno de los agentes locales.

Por otra parte, para poder apreciar el efecto del arbitraje propuesto en este trabajo, debe hacerse la evaluación en condiciones donde la red esté muy cargada. A baja carga será difícil apreciar ningún tipo de planificación. Sin embargo, a cargas elevadas será la planificación realizada la encargada de repartir adecuadamente el turno entre los distintos candidatos a salir por un canal. Por tanto, se establecerán tantas conexiones como sea posible de cara a conseguir una elevada carga.

Para alcanzar una gran carga de la red se va a intentar establecer un determinado número de conexiones. Evidentemente, de éstas habrá unas que se aceptarán y otras que serán denegadas. Cada vez que se deniegue una conexión se intentará otra del mismo nivel de servicio, pero con origen y destino generados, de nuevo, aleatoriamente. Esta fase de establecimiento de conexiones continúa hasta que se alcance un número máximo de reintentos, en cuyo caso se ha alcanzado la carga máxima admisible por la red. Como se verá en las secciones siguientes, de esta forma se pueden alcanzar cargas cercanas al 80 %, que es el máximo posible.

Como se ha comentado previamente, para la evaluación se va a utilizar tráfico CBR. En concreto, para cada nivel de servicio, el ancho de banda que se va a solicitar será generado aleatoriamente en el rango asociado a cada nivel de servicio. En cuanto a la latencia máxima admitida, el simulador genera aleatoriamente para cada conexión una distancia máxima entre dos entradas de la tabla de arbitraje de cada puerto de salida de cada conmutador de la ruta. Esto es equivalente a que cada conexión tuviera una demanda de latencia máxima y en cada conmutador se calculara la distancia máxima a utilizar.

De acuerdo con lo estudiado en los capítulos precedentes, los requisitos de las conexiones se traducen en una distancia máxima entre dos entradas consecutivas. Estas

pueden ser: 2, 4, 8, 16, 32 ó 64. De esta forma, los niveles de servicio que se han considerado, así como sus requisitos son los siguientes:

- SL 0: conexiones con distancia máxima de 2 entre dos entradas consecutivas de la secuencia de entradas que tiene asignada. El rango posible a solicitar de ancho de banda varía entre 64 Kbps y 1,55 Mbps.
- SL 1: conexiones con distancia máxima de 4 entradas entre dos entradas consecutivas de la secuencia. El rango posible a solicitar de ancho de banda varía entre 64 Kbps y 1,55 Mbps.
- SL 2: conexiones con distancia máxima de 8 entradas entre dos entradas consecutivas de la secuencia. El rango posible a solicitar de ancho de banda varía entre 64 Kbps y 1,55 Mbps.
- SL 3: conexiones con distancia máxima de 16 entradas entre dos entradas consecutivas de la secuencia. El rango posible a solicitar de ancho de banda varía entre 64 Kbps y 1,55 Mbps.
- SL 4: conexiones con distancia máxima de 32 entradas entre dos entradas consecutivas de la secuencia. El rango posible a solicitar de ancho de banda varía entre 64 Kbps y 1,55 Mbps.
- SL 5: conexiones con distancia máxima de 32 entradas entre dos entradas consecutivas de la secuencia. El rango posible a solicitar de ancho de banda varía entre 1,55 Mbps y 64 Mbps.
- SL 6: conexiones con distancia máxima de 64 entradas entre dos entradas consecutivas de la secuencia. El rango posible a solicitar de ancho de banda varía entre 8 Kbps y 64 Kbps.
- SL 7: conexiones con distancia máxima de 64 entradas entre dos entradas consecutivas de la secuencia. El rango posible a solicitar de ancho de banda varía entre 64 Kbps y 1,55 Mbps.
- SL 8: conexiones con distancia máxima de 64 entradas entre dos entradas consecutivas de la secuencia. El rango posible a solicitar de ancho de banda varía entre 1,55 Mbps y 64 Mbps.
- SL 9: conexiones con distancia máxima de 64 entradas entre dos entradas consecutivas de la secuencia. El rango posible a solicitar de ancho de banda varía entre 64 Mbps y 255 Mbps.

Puede observarse que el rango de ancho de banda para cada nivel de servicio es amplio. Además hay niveles de servicio con rangos de ancho de banda distintos. De

esta forma vamos a tener, para un mismo nivel de servicio, paquetes de características bastante distintas, que deberán coexistir en los mismos niveles de servicio, y según nuestra propuesta, en los mismos buffers.

También puede observarse que para las distancias mayores, que como se indicó anteriormente serían las más demandadas, se han considerado varios niveles de servicio, cada uno con un rango de ancho de banda distinto. En concreto, para la distancia máxima 64 (conexiones tipo DB sin requisito de ancho de banda, o con un requisito lo suficientemente amplio para necesitar solamente una entrada en la tabla) se han considerado cuatro niveles de servicio con un rango de ancho de banda muy bajo, bajo-medio, medio-alto y muy alto. En la Tabla 6.2 se han incluido los niveles de servicio considerados, y sus características particulares.

SL	Distancia máxima	Rango de ancho de banda (Mbps)
0	2	0,064 - 1,55
1	4	0,064 - 1,55
2	8	0,064 - 1,55
3	16	0,064 - 1,55
4	32	0,064 - 1,55
5		1,55 - 64
6	64	0,008 - 0,064
7		0,064 - 1,55
8		1,55 - 64
9		64 - 255

Tabla 6.2: Características de los niveles de servicio utilizados.

Al tener más niveles de servicio para las distancias más solicitadas, y al requerir esos niveles de servicio menos entradas de la tabla, van a llegar a establecerse más conexiones correspondientes a esos niveles de servicio. De esta forma, en las simulaciones se está considerando más tráfico de los niveles de servicio que, tal como se analizó en la Sección 1.3, van a ser más frecuentes en entornos y aplicaciones reales.

Se ha mencionado anteriormente que InfiniBand admite que los paquetes tengan un tamaño entre 256 y 4096 bytes. En la evaluación realizada se van a considerar varios tamaños de paquete incluidos en esos márgenes. En estas cantidades está ya incluida la cabecera de los paquetes, que obviamente, es igual para ambos tamaños. Así pues, para los paquetes de tamaño 256 bytes, la cabecera va a representar una mayor sobrecarga para la red, pero también tienen la ventaja de que al ser más pequeños están menos tiempo ocupando el canal.

Una vez que se ha conseguido establecer el número de conexiones deseado, o el máximo alcanzable, termina el periodo de establecimiento de conexiones y comienza un periodo transitorio. Este periodo transitorio tiene como finalidad conseguir que la red

alcance un régimen permanente. Este periodo transitorio se prolongará hasta que se hayan recibido en los hosts un total de 10000 paquetes [Fli01, Cas01], comenzando la cuenta cuando comienza el periodo transitorio. Terminado éste comienza el periodo estacionario, en el cual se van a tomar las medidas de los distintos índices de prestaciones a utilizar.

Cuando comienza el periodo estacionario se inicializan todos los contadores para desechar la información recogida hasta entonces. Se ha considerado que este periodo estacionario se prolongue hasta que la conexión con menor ancho de banda haya recibido 100 paquetes. De esta forma, las conexiones con ancho de banda mayor habrán recibido muchos más paquetes, y en las de poco ancho de banda habrá datos suficientes para que éstos sean significativos. Al tener las conexiones un comportamiento constante (es tráfico CBR) no es necesario alargar más la simulación, pues el comportamiento de todas las conexiones va a ser periódico con un periodo marcado por su ancho de banda medio.

6.3. Métricas para la evaluación de prestaciones

Las medidas de prestaciones más importantes y comúnmente utilizadas en las redes de interconexión son la latencia y la productividad [NGM97]. Sin embargo, para tráfico multimedia hay otra serie de índices que deben ser considerados [Cue98].

Así pues, para realizar la evaluación de prestaciones, en este trabajo se han considerado los índices habituales de ambos campos. En concreto, los índices escogidos para realizar la evaluación en este trabajo son:

- Tráfico inyectado. Medido en bytes/ciclo/nodo, indica cómo de cerca nos hemos quedado del 80 % máximo alcanzable (el otro 20 % se dedica al tráfico sin garantía de QoS).
- Tráfico transmitido. Se mide en bytes/ciclo/nodo e indica si la red ha sido capaz de transmitir todo el tráfico que se le ha inyectado. El cociente con el tráfico inyectado proporciona la productividad alcanzada por la red.
- Utilización de los enlaces. Medido en porcentaje, este índice indica cómo de cargada está la subred y permitirá comprobar que casi todos los enlaces se encuentran cercanos a su utilización máxima. Este índice se va a desglosar en dos valores distintos: el correspondiente a los enlaces que unen los hosts con los conmutadores, y el correspondiente a los enlaces entre conmutadores.
- Reserva en los enlaces. Indica qué ancho de banda del disponible ha sido reservado por las aplicaciones. Esta es otra forma de ver el uso que se va a hacer del ancho

de banda disponible. Al igual que en el caso anterior, también se va a desglosar entre la reserva de los enlaces a hosts y de los enlaces entre conmutadores.

- Distribución del retraso de los paquetes. Este índice indica qué porcentaje de paquetes han llegado en cada uno de los umbrales que se fijen. Estos umbrales se establecen en función de valores proporcionales al tiempo máximo garantizado a la conexión. Este índice permite comprobar cuál es el retraso máximo que están sufriendo los paquetes de las conexiones.

- Retraso de la mejor y peor conexión. Al igual que en el caso anterior, representa la distribución de los retrasos de los paquetes, pero en vez de en media, ahora en las conexiones consideradas como la mejor y la peor en cuanto al retraso de sus paquetes. Esto permitirá comprobar si la que peor comportamiento tiene cumple o no sus expectativas. Además, también es importante que no haya mucha diferencia entre la mejor y la peor conexión.

- Jitter medio de los paquetes. Se define el jitter como la diferencia entre los retrasos en la llegada de dos paquetes consecutivos. Este valor permite obtener la variabilidad de los retrasos con que llegan los paquetes. Se va a visualizar agrupado por intervalos proporcionales al tiempo teórico entre llegadas (IAT) de cada conexión.

La forma de representar estos datos variará de unos casos a otros. Cuando sea un único número por simulación (como la utilización, la tasa de inyección, etc.) se representarán en una tabla. En los casos en que se considere interesante contrastar varios valores correspondientes a distintos niveles de servicio, configuraciones posibles, etc., se representarán de manera gráfica.

6.4. Resultados de las simulaciones

En esta sección se van a mostrar los resultados obtenidos en el proceso de evaluación de las propuestas realizadas en los capítulos anteriores. La forma de mostrar experimentalmente que las propuestas realizadas consiguen las prestaciones esperadas va a ser mediante la variación de varios parámetros considerados significativos. Estos parámetros que se han variado son algunos de los que se han considerado que podrían tener influencia en los resultados obtenidos. Así por ejemplo, se ha variado la topología de la red de interconexión, realizando pruebas tanto con topologías regulares (hipercubo y malla) como irregulares. Se ha variado también el tamaño de la red, probando redes de 8, 16, 32 y 64 conmutadores. Se han considerado también varios tamaños de paquete, desde el más pequeño al máximo posible según las especificaciones de InfiniBand. Se van a mostrar los resultados obtenidos para las tres velocidades del enlace posibles

de InfiniBand. Por último, también se han probado varios tamaños de los buffers, que permitirán almacenar uno, dos, tres o cuatro paquetes de datos completos.

Pasemos pues, a analizar en la siguiente sección la influencia de la topología en los resultados obtenidos. Para ello se van a probar distintas topologías, tanto regulares (hipercubo y malla) como irregulares.

6.4.1. Estudio de la influencia de la topología

Se analiza en esta sección si la topología utilizada en la red de interconexión tiene influencia en el comportamiento de las propuestas realizadas en este trabajo, y en la QoS que las aplicaciones reciben. Para ello se van a utilizar tres topologías diferentes: una irregular y dos regulares (hipercubo y malla).

Se van a realizar tres pruebas con diferentes topologías, fijando el resto de parámetros, que se variarán en secciones posteriores. Así, los resultados mostrados en esta sección serán en todos los casos para tamaño de paquete pequeño (256 bytes), la velocidad 1x de InfiniBand y buffers con capacidad para cuatro paquetes completos. De igual forma, en todos los casos las pruebas realizadas en esta sección serán con redes de 16 conmutadores, y por tanto con 64 hosts conectados.

En la Tabla 6.3 se recogen algunos de los resultados obtenidos en estas pruebas. Puede observarse que no hay grandes diferencias en los resultados obtenidos por las tres topologías. En concreto, respecto al tráfico inyectado, en todos los casos los valores obtenidos están en torno al 72 %, siendo un poco más alto para el caso de la malla. Hay que recordar que el máximo alcanzable es 80 %, pues el otro 20 % restante está reservado para el tráfico best-effort, que no ha sido inyectado en estas pruebas.

	Topología		
	Irregular	Hipercubo	Malla
Tráfico inyectado (Bytes/Ciclo/Nodo)	0,7258	0,7224	0,7357
Tráfico entregado (Bytes/Ciclo/Nodo)	0,7258	0,7224	0,7357
Utilización media de los interfaces de los hosts (%)	72,58	72,24	73,58
Utilización media de los puertos de los conmutadores (%)	73,48	74,48	75,66
Reserva media de los interfaces de los hosts (Mbps)	1848,67	1840,20	1874,04
Reserva media de los puertos de los conmutadores (Mbps)	1871,75	1897,09	1927,16
Número de conexiones establecidas	111813	97136	121187

Tabla 6.3: Tráfico y utilización para diferentes topologías.

En todos los casos puede comprobarse que el tráfico entregado en los hosts coincide con el tráfico inyectado, con lo que la red está funcionando correctamente, y no existe ningún tipo de congestión.

En cuanto a la utilización, tampoco hay diferencia entre las distintas topologías evaluadas, aunque la malla también presenta un valor ligeramente mayor. Puede comprobarse como la utilización de los puertos de los conmutadores es un poco mayor a la de los interfaces de los hosts. Esto es debido al comportamiento de los algoritmos de encaminamiento utilizados, que tienden a utilizar las mismas rutas.

También se ha medido la reserva realizada tanto en los interfaces de los hosts, como en los puertos de los conmutadores. Con ello podemos comprobar cómo de lejos se ha quedado la reserva realizada del máximo alcanzable, que para la velocidad 1x recordemos que es 2 Gbps. Puede comprobarse como la reserva realizada está bastante próxima al máximo alcanzable, estando en todos los casos por encima del 90 %. Tampoco en este caso hay mucha diferencia entre los resultados obtenidos por las distintas topologías, aunque para el caso de la malla, la reserva media de los puertos de los conmutadores es un poco más elevada, debido al uso que hace el algoritmo de encaminamiento que utiliza más unos enlaces que otros.

La utilización alcanzada está bastante próxima al máximo alcanzable, pero además por las características de los algoritmos de encaminamiento utilizados, hay unos enlaces más cargados que otros. Como ejemplo, se puede citar que la utilización de algunos enlaces es del 79 %, y la reserva realizada es del máximo, 2 Gbps. De esta forma, para conseguir mejorar la utilización o la reserva media habría que utilizar otro tipo de algoritmo de encaminamiento que distribuyera mejor el tráfico entre las rutas posibles.

Donde sí que hay diferencias notables es en el número de conexiones que se han conseguido establecer. Con la topología hipercubo se han establecido menos conexiones, mientras que el mayor número de conexiones establecidas se consigue para el caso de la malla. Esto sólo depende del algoritmo de encaminamiento utilizado y de que éste sea capaz de utilizar adecuadamente las rutas disponibles, balanceando el tráfico entre los distintos enlaces. Así, al conseguir establecerse más conexiones para la malla, se genera posteriormente algo más de tráfico, y eso hace que el resto de índices estudiados alcancen unos valores un poco mayores.

Otro índice estudiado es la distribución de los retrasos de los paquetes para cada uno de los niveles de servicio considerados. Esta distribución se ha medido como porcentaje de paquetes que han conseguido llegar a su host destino en cada uno de los umbrales considerados. Estos porcentajes se han obtenido para la carga máxima de tráfico admitida, con lo que cualquier situación donde la red no esté tan cargada obtendrá unos resultados todavía mejores. Hay que señalar que los resultados son relativos al *deadline* (que en las figuras es D) solicitado por cada conexión, tal y como se ha señalado en los capítulos anteriores. De esta forma, los distintos umbrales son diferentes para cada conexión, y por tanto también para cada nivel de servicio.

En todos los casos el retraso máximo admisible por una conexión coincide con su *deadline*. Sin embargo, sería deseable que la mayoría de los paquetes llegaran bastante

antes de dicho límite, para poder ser procesados, decodificados, etc. con suficiente margen de tiempo. Esto, aunque no será una condición a cumplir, sí que puede ser tenido en cuenta a la hora de calificar la bondad de los resultados obtenidos.

En la Figura 6.3 pueden observarse los resultados obtenidos para las tres topologías evaluadas. En todos los casos todos los paquetes llegan a su destino bastante antes de que expire su *deadline*. Puede comprobarse como para aquellas conexiones con requisitos más exigentes (los SLs más bajos) sus paquetes tardan un poco más en llegar, mientras que para los SLs con límites poco exigentes, el 100 % de los paquetes han llegado incluso antes del primer umbral.

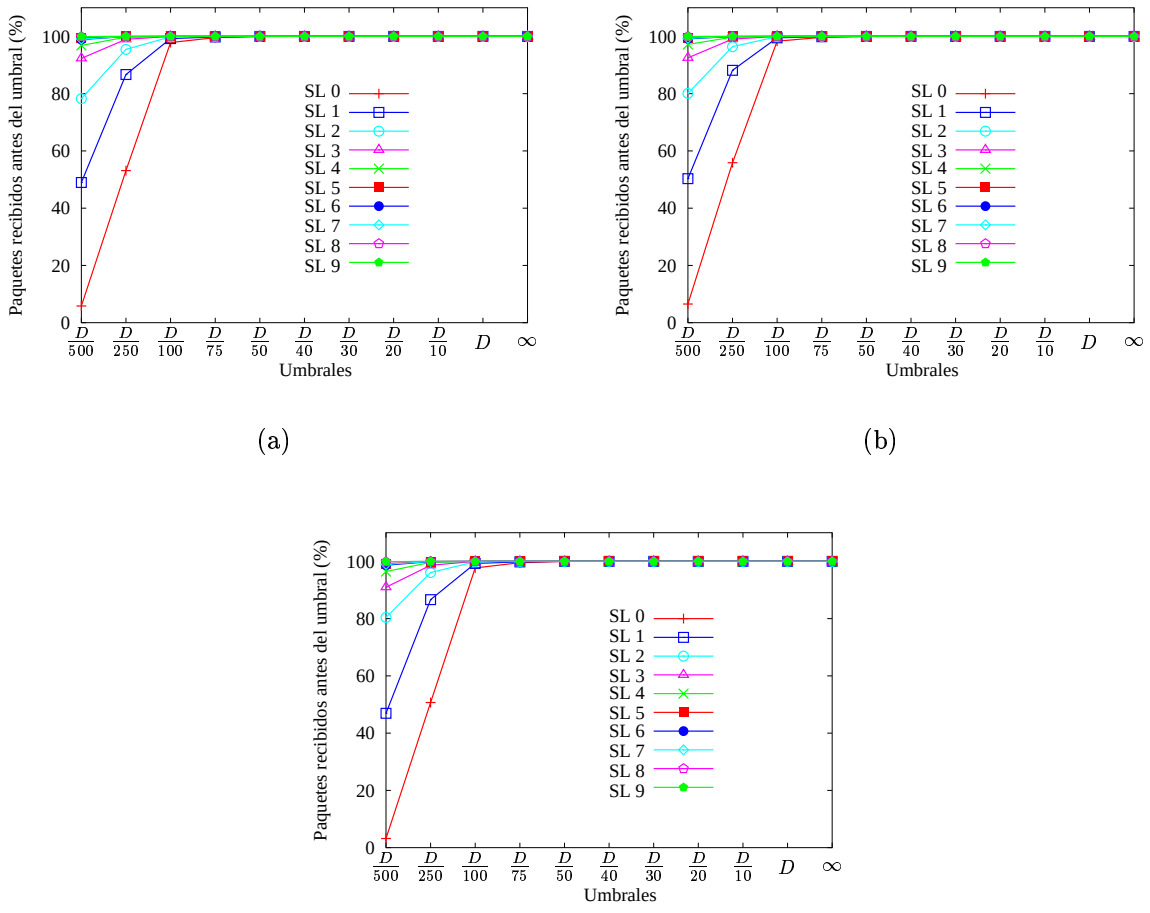


Figura 6.3: Distribución del retraso de los paquetes para (a) una topología irregular, (b) un hipercubo y (c) una malla.

Hay que señalar que los resultados obtenidos no varían para las tres topologías estudiadas, y que en todos los casos todos los paquetes llegan a su destino antes que expire su límite. De esta forma, con la distribución de las entradas de la tabla de

arbitraje realizada, se ha conseguido proporcionar a las aplicaciones los distintos niveles de exigencia en cuanto a garantía de latencia. Podría pensarse qué pasaría si la red estuviera un poco más cargada, pero esto no es fácil de obtener con los algoritmos de encaminamiento utilizados, pues, como ya se ha visto, éstos tienden a utilizar a menudo las mismas rutas, con lo que en estas redes hay muchas rutas que están en su máxima utilización. Como ejemplo, puede citarse que, para las tres topologías, la máxima utilización de algunos de sus enlaces está próxima al 79 % (siendo el máximo el 80 %), y hay varios enlaces que tienen una reserva de 2 Gbps, que es la máxima posible. Así pues, incluso teniendo esos enlaces a su máxima actividad, las aplicaciones consiguen los resultados que necesitan.

También se ha medido el jitter, o variabilidad en el retraso de la llegada de los paquetes. En concreto, se ha medido cómo varían los retrasos de los paquetes agrupándolos en varios intervalos. Estos intervalos se establecen en función del tiempo teórico entre llegadas (IAT) de la conexión, con lo que estos intervalos son distintos para cada conexión.

En la Figura 6.4 puede verse el jitter medio de los paquetes para las tres topologías estudiadas, y para los 10 SLs considerados. En todos los casos, para los SLs 0, 1, 2, 3 y 4 todos los paquetes están situados en el intervalo central. Esto es debido tanto a que son los SLs más prioritarios, y por lo tanto tendrán que esperar menos para ser tratados, como a que los anchos de banda considerados para estos SLs son más bajos y por tanto su IAT es más elevado. Para los SLs 5, 6. 7, 8 y 9 el jitter medio se distribuye entre los distintos intervalos considerados, teniendo forma de campana de Gauss.

Por último, también se ha observado para un determinado umbral, las conexiones de cada SL que han entregado el mayor y el menor porcentaje de paquetes. En las figuras, estas conexiones se identifican como la mejor y la peor, respectivamente. Se ha elegido un umbral suficientemente estricto para que el porcentaje de paquetes que han llegado a su destino antes de ese umbral sea inferior al 100 %. En concreto, se ha seleccionado el umbral igual a $\frac{Deadline}{100}$. Hay que señalar de nuevo, que ese umbral es diferente para cada conexión, y que se obtiene en base a la latencia máxima admisible por la propia conexión.

En la Figura 6.5 pueden observarse los resultados de la mejor y peor conexión para las tres topologías estudiadas y para los niveles de servicio 0, 1, 2 y 3. Hay que recordar que estos niveles de servicio son los que presentan exigencias más estrictas en cuanto a latencia.

En todos los casos puede observarse como no hay muchas diferencias entre las tres topologías estudiadas. Además, tampoco hay grandes diferencias en el comportamiento obtenido por las conexiones consideradas como la mejor y la peor de cada SL. Esto prueba que todas las conexiones obtienen unos resultados similares, y que el arbitraje realizado consigue unas prestaciones similares en todos los casos.

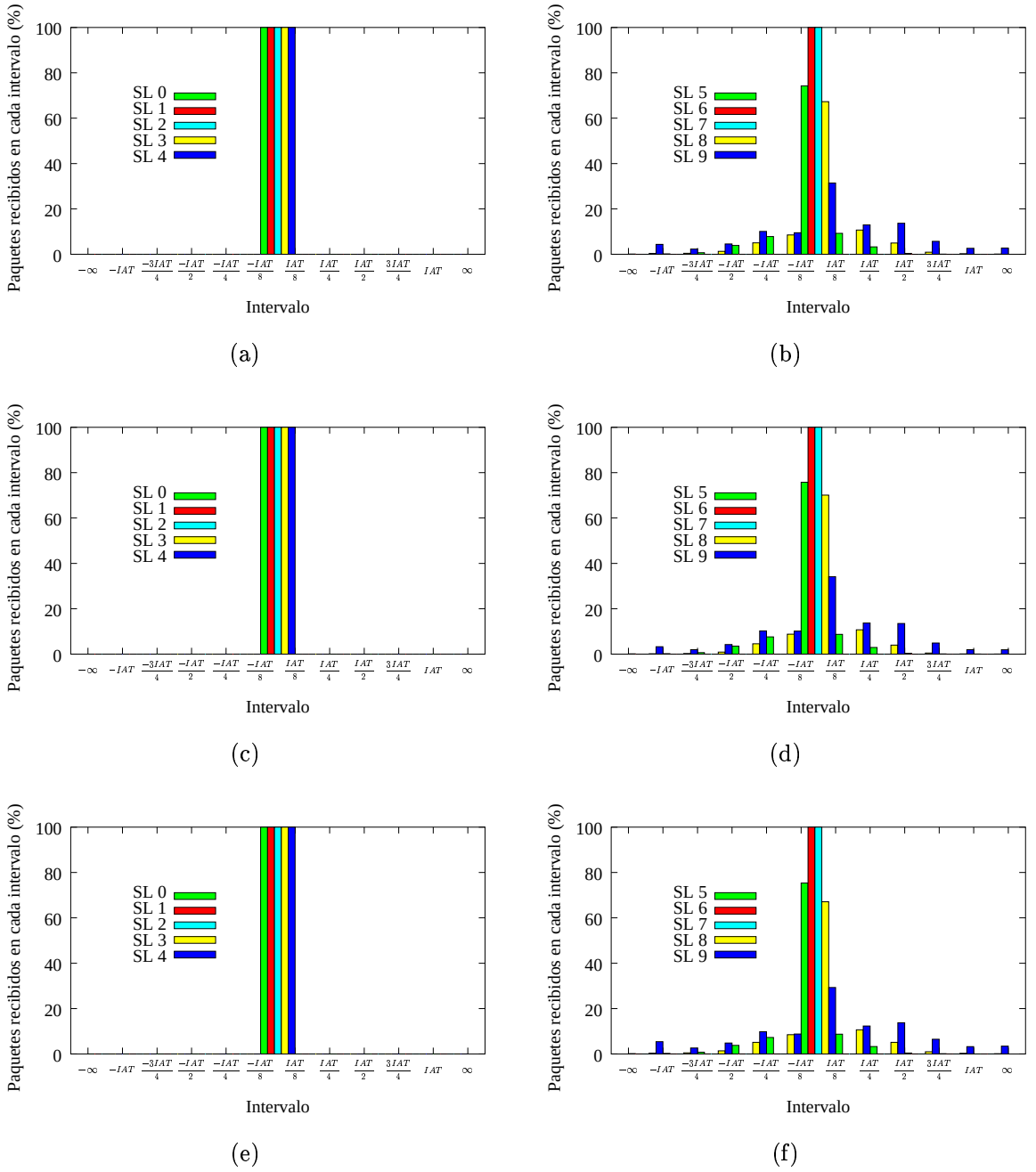


Figura 6.4: Jitter medio de los paquetes para (a) y (b) topología irregular, (c) y (d) para un hipercubo, y (e) y (f) para una malla. Las figuras (a), (c) y (e) muestran los resultados para los SLs 0, 1, 2, 3 y 4, mientras que las figuras (b), (d) y (f) lo hacen para los SLs 5, 6, 7, 8 y 9.

Así pues, una vez estudiados los distintos índices a considerar, puede concluirse que la calidad de servicio obtenida por las aplicaciones no depende de la topología utilizada en la red de interconexión. En todos los casos, al variar la topología de interconexión los resultados obtenidos presentan escasas diferencias.

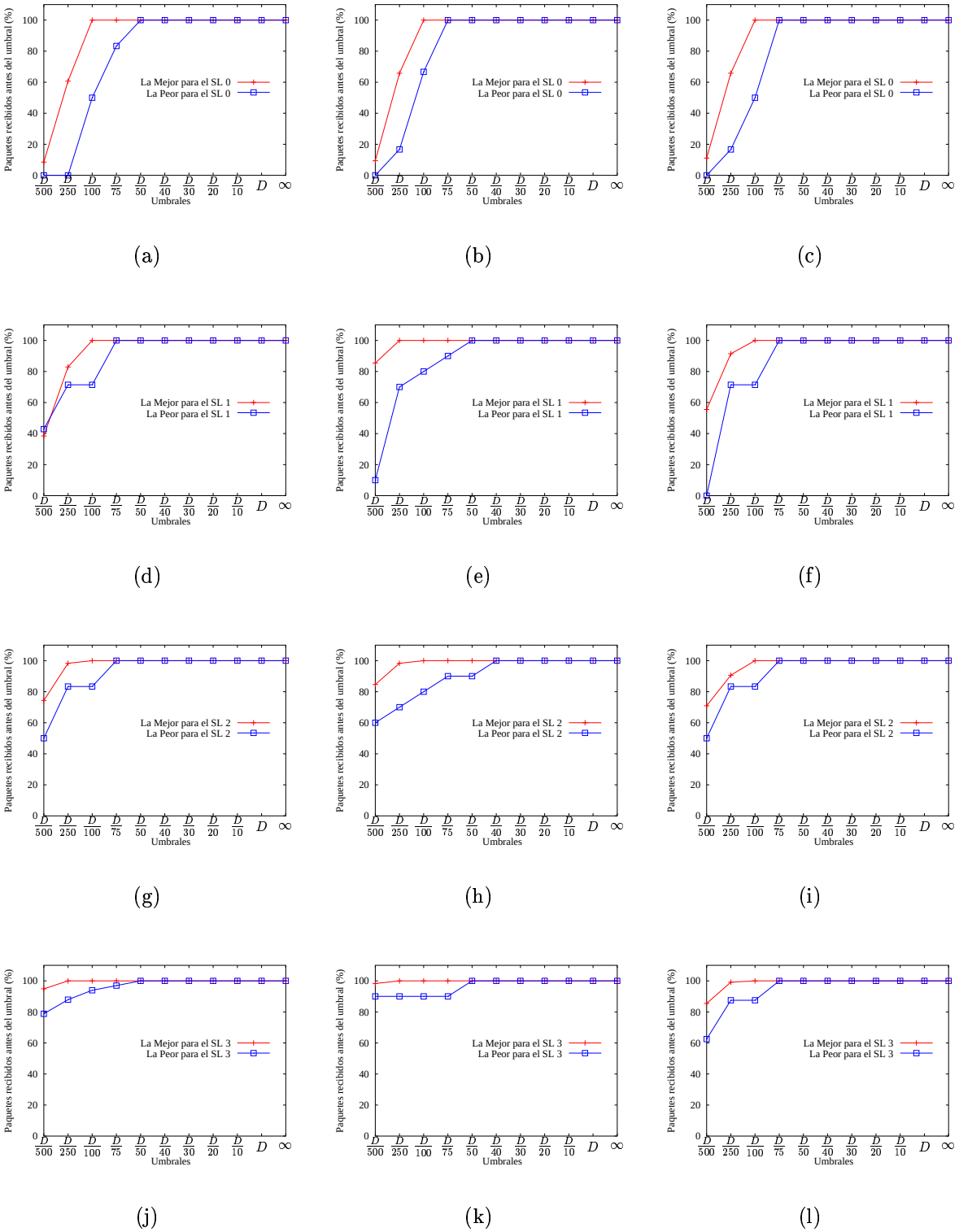


Figura 6.5: La mejor y la peor conexión de los niveles de servicios con requisitos más restrictivos, para (a), (d), (g) y (j) topología irregular, (b), (e), (h) y (k) topología hipercubo, (c), (f), (i) y (l) topología malla.

La única influencia de la topología tiene que ver con el uso que se hace de los enlaces disponibles. Así, el número de conexiones establecidas varía considerablemente al variar la topología, lo que afecta a lo cargada que puede llegar a estar la red. Sin embargo, más que de la topología, esto depende principalmente del algoritmo de encaminamiento utilizado, y de que éste sea capaz de repartir el tráfico entre las distintas rutas disponibles. Posiblemente, si se hubieran utilizado otros algoritmos de encaminamiento, se hubieran alcanzado grados de utilización y tasas de tráfico mayores para todas las topologías. Sin embargo, esto no variaría el comportamiento del arbitraje pues, como se ha visto, es capaz, mediante el correcto dimensionado de los recursos disponibles, de proporcionar la garantía de QoS requerida por cada aplicación. Recordemos que, para las tres topologías evaluadas, había varios enlaces funcionando a su máxima utilización y con la reserva máxima de recursos, y aún así las aplicaciones consiguen la QoS solicitada.

Una vez visto que la topología utilizada no tiene una influencia directa en los resultados obtenidos, en lo que sigue se van a considerar siempre redes con topología irregular.

6.4.2. Estudio de la influencia del tamaño de red

En esta sección se va a estudiar la influencia del tamaño de la red en el comportamiento de las propuestas realizadas para garantizar QoS a las aplicaciones. Para ello, se van a considerar redes de diversos tamaños, aunque siempre con topología irregular. Al igual que en la sección anterior, se va a considerar solamente el caso de paquetes de tamaño pequeño, la velocidad básica de InfiniBand (1x) y buffers con capacidad para almacenar cuatro paquetes completos.

En la evaluación realizada se han considerado redes con 8, 16, 32 y 64 conmutadores, cada uno de ellos con ocho puertos, cuatro de ellos dedicados a conexión con hosts, y los otros cuatro para la interconexión con otros conmutadores. De esta forma, en las simulaciones consideradas, el número de hosts será 32, 64, 128 y 256.

En la Tabla 6.4 pueden observarse algunos de los resultados obtenidos para cada uno de los tamaños de red considerados. Puede observarse como la tasa de tráfico alcanzada es similar en todos los casos, pues este índice se mide en bytes/ciclo/nodo, con lo que se hace independiente del tamaño de la red considerada. Cabe destacar que para la red de 8 nodos la tasa de tráfico es un poco mayor. Esto es debido a que se han alcanzado utilizaciones y reservas un poco mayores.

	Número de conmutadores			
	8	16	32	64
Tráfico inyectado (Bytes/Ciclo/Nodo)	0,7557	0,7258	0,7248	0,7227
Tráfico entregado (Bytes/Ciclo/Nodo)	0,7557	0,7258	0,7248	0,7227
Utilización media de los interfaces de los hosts (%)	75,57	72,58	72,48	72,27
Utilización media de los puertos de los conm. (%)	72,93	73,48	72,66	71,94
Reserva media de los interfaces de los hosts (Mbps)	1926,85	1848,67	1834,04	1822,41
Reserva media de los puertos de los conm. (Mbps)	1859,59	1871,75	1827,16	1837,62
Número de conexiones establecidas	46495	111813	232854	284365

Tabla 6.4: Tráfico y utilización para varios tamaños de red.

En cuanto a la utilización y la reserva, tampoco pueden observarse diferencias entre los distintos tamaños de red considerados, aunque para el caso de la red de 8 nodos, éstas son un poco mayores. Esto es debido a que en el caso de esta red se ha balanceado mejor el tráfico entre los enlaces disponibles.

Evidentemente, para cada tamaño se ha conseguido establecer un número de conexiones muy distinto, tal y como puede verse en la Tabla 6.4. De esta forma, al aumentar el tamaño de la red se establecen más conexiones, pero la utilización media alcanzada es bastante similar en todos los casos.

Al igual que en el estudio anterior, el algoritmo de encaminamiento no ha balanceado bien la carga entre las distintas rutas disponibles. Así, en todos los casos hay varios enlaces con una utilización superior al 79 % y una reserva de 2 Gbps.

En la Figura 6.6 puede verse la distribución de los retrasos de los paquetes para cada uno de los niveles de servicio y para los cuatro tamaños de red considerados. Nuevamente, esta distribución se ha medido como porcentaje de paquetes que han conseguido llegar a su destino en cada uno de los umbrales considerados, siendo estos umbrales diferentes para cada conexión y relativos a su requisito máximo de latencia.

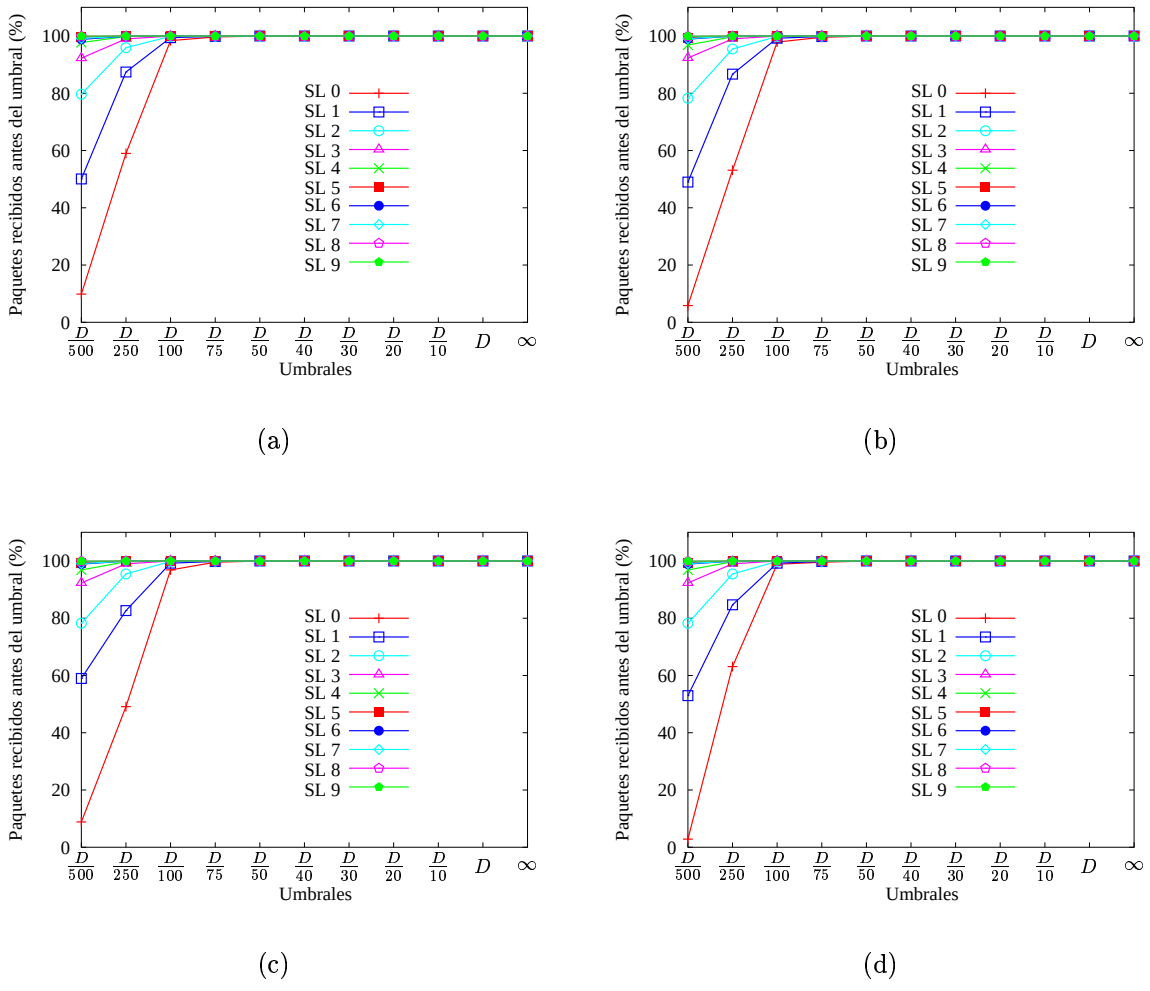


Figura 6.6: Distribución del retraso de los paquetes para redes de distintos tamaños: (a) 8 conmutadores, (b) 16 conmutadores, (c) 32 conmutadores y (d) 64 conmutadores.

En los resultados obtenidos no se aprecian diferencias entre los distintos tamaños de red considerados. En todos los casos los paquetes llegan con suficiente antelación y, más o menos, en los mismos márgenes. En todos los casos, los paquetes de las conexiones con requisitos más restrictivos (los de SLs menores), llegan a su destino con menos

margen de tiempo. Por el contrario, los SLs mayores, correspondientes a conexiones con requisitos menos exigentes, consiguen que todos sus paquetes lleguen a su destino antes del primer umbral.

En las Figuras 6.7 y 6.8 pueden observarse los resultados del jitter para los cuatro tamaños de red considerados. En la Figura 6.7 se muestran los resultados para los tamaños de red de 8 y 16 conmutadores, mientras que en la Figura 6.8 los resultados mostrados corresponden a los tamaños de red de 32 y 64 conmutadores.

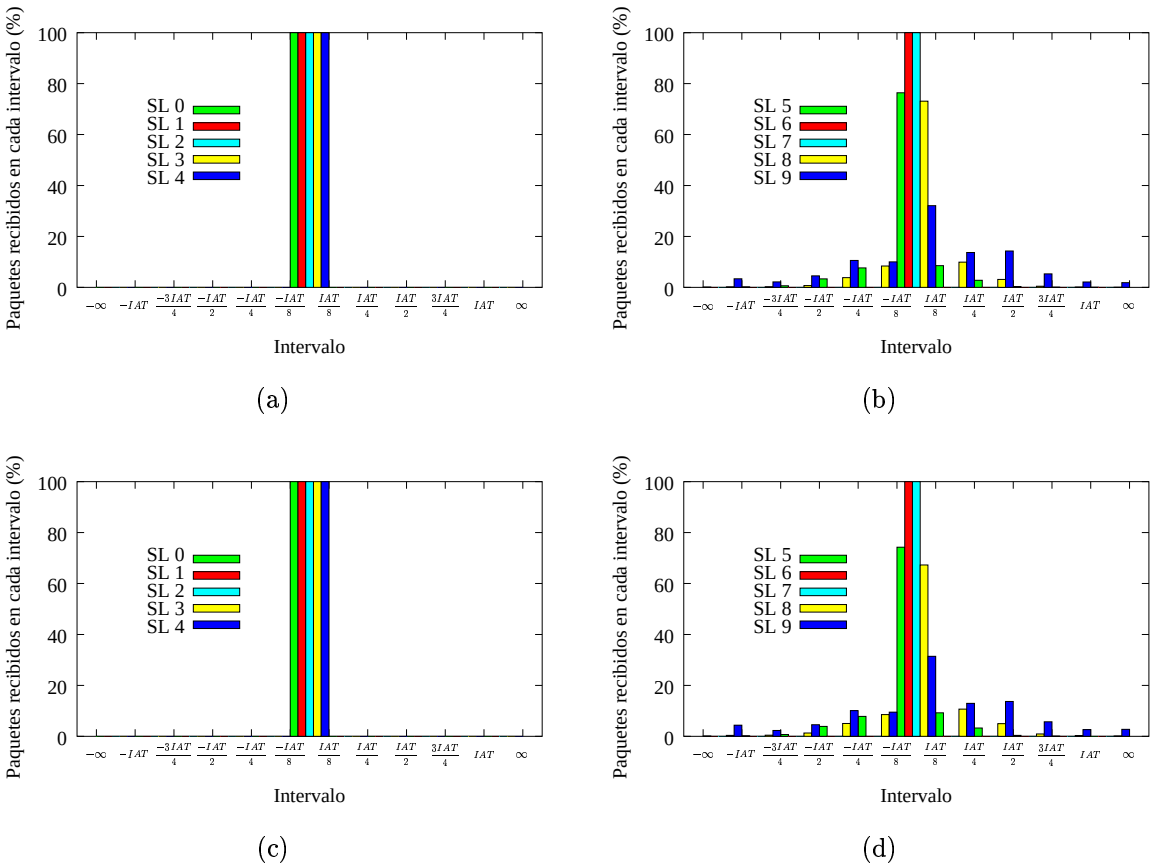


Figura 6.7: Jitter medio de los paquetes para (a) y (b) red de 8 conmutadores, y (c) y (d) red de 16 conmutadores. Las figuras (a) y (c) muestran los resultados para los SLs 0, 1, 2, 3 y 4, mientras que las figuras (b) y (d) lo hacen para los SLs 5, 6, 7, 8 y 9.

Recordemos que los resultados del jitter se han representado agrupando el retraso de los paquetes en diversos intervalos. Estos intervalos se establecen en base al tiempo teórico entre llegadas (IAT) de cada conexión, siendo pues distintos para cada conexión, y relativos a sus propias características.

Los resultados obtenidos son similares a los mostrados en la sección anterior. Para todos los tamaños de red considerados, los SLs menores tienen todos los paquetes en el intervalo central. Eso significa que el retraso con que llegan sus paquetes presenta escasa variabilidad. La razón es la misma que la apuntada en la sección anterior: estos

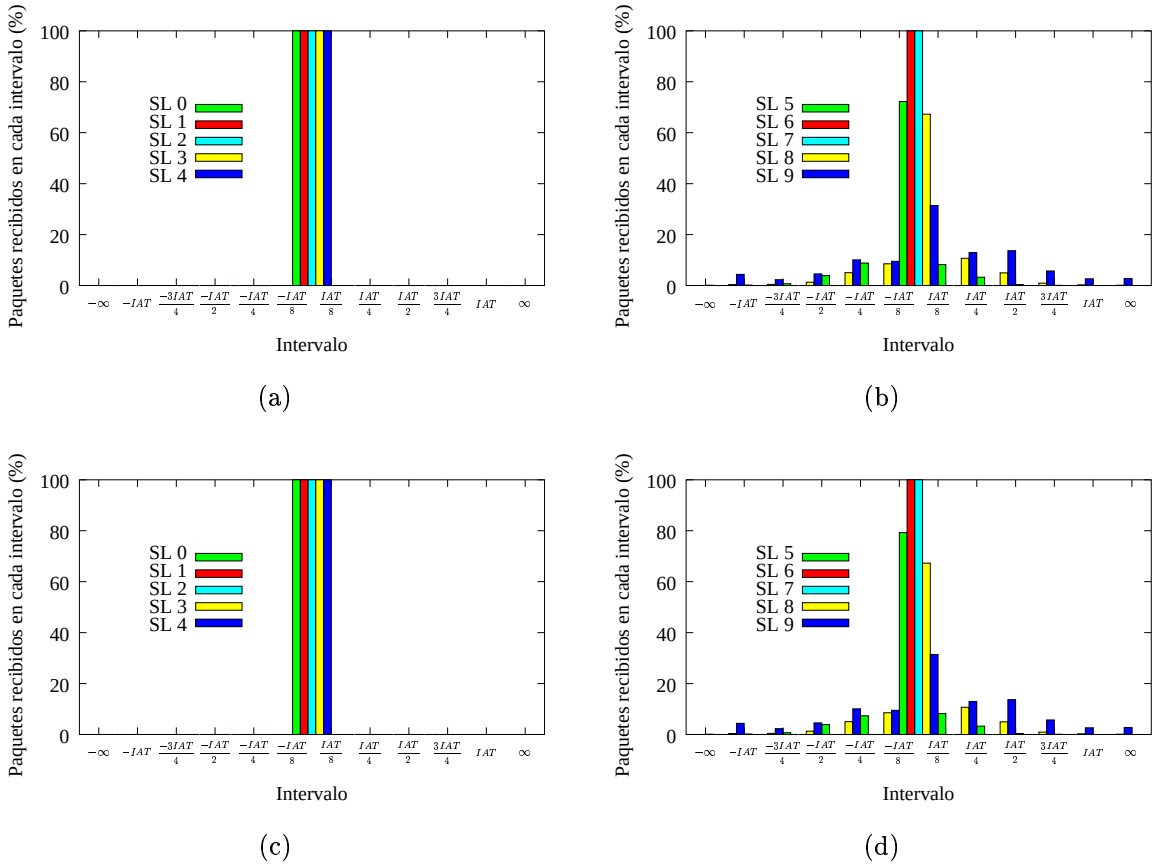


Figura 6.8: Jitter medio de los paquetes para (a) y (b) red de 32 conmutadores, y (c) y (d) red de 64 conmutadores. Las figuras (a) y (c) muestran los resultados para los SLs 0, 1, 2, 3 y 4, mientras que las figuras (b) y (d) lo hacen para los SLs 5, 6, 7, 8 y 9.

SLs tienen más entradas en la tabla para que sus paquetes puedan salir, con lo que es más fácil darles salida en cuanto lleguen al conmutador. De esta forma, la variabilidad en los retrasos dependerá menos del momento en que hayan llegado al conmutador y de por donde se esté recorriendo la tabla de arbitraje en ese momento.

Otra razón importante para explicar el comportamiento de los SLs inferiores es que el ancho de banda de sus conexiones es relativamente bajo, con lo que su tiempo teórico entre llegadas es relativamente grande para que los paquetes de esas conexiones puedan llegar en el primer intervalo considerado.

Para los SLs mayores, correspondientes a las conexiones con requisitos de latencia menos restrictivos, los resultados mostrados se distribuyen en forma de campana de Gauss, estando la mayor parte de los paquetes en el intervalo central.

Por último, en la Figura 6.9 se muestran los resultados en cuanto a la latencia media obtenida por los paquetes de las consideradas mejor y peor conexión de los SLs más restrictivos para los cuatro tamaños de red considerados. Al igual que en los

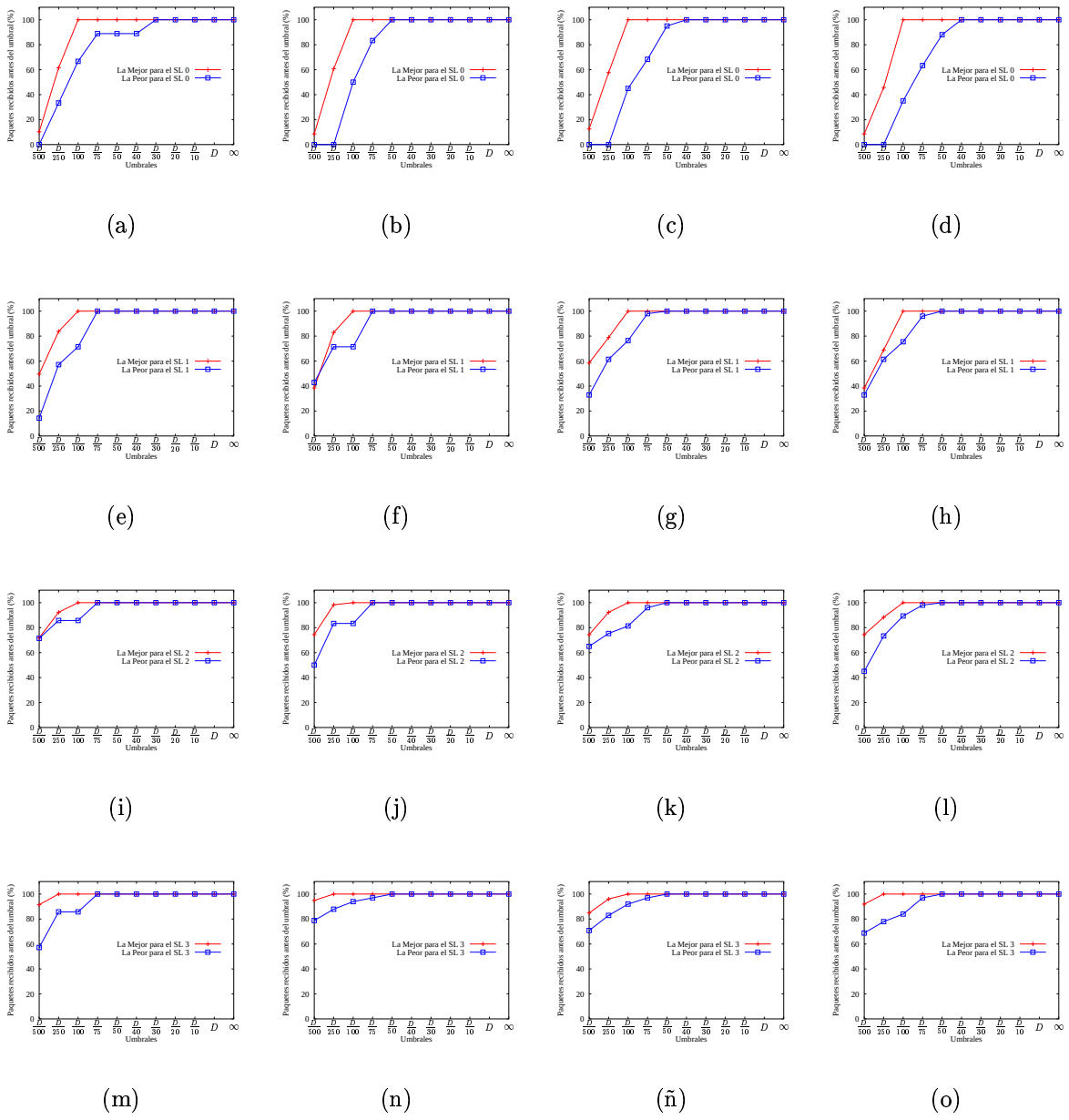


Figura 6.9: La mejor y la peor conexión de los niveles de servicio con requisitos más restrictivos, para (a), (e), (i) y (m) red de 8 conmutadores, (b), (f), (j) y (n) red de 16 conmutadores, (c), (g), (k) y (ñ) red de 32 conmutadores, (d), (h), (l) y (o) red de 64 conmutadores

casos anteriores, no se aprecian diferencias en los resultados obtenidos por los distintos tamaños de red considerados. Además, para todos ellos, los resultados obtenidos por la considerada mejor conexión no difieren mucho de los obtenidos por la considerada peor conexión.

De esta forma, hemos visto de forma experimental que el tamaño de red no tiene influencia en la QoS que las aplicaciones consiguen. El correcto dimensionado de los recursos, así como la reserva realizada, permite que el arbitraje proporcione a cada conexión el servicio que necesita.

6.4.3. Estudio de la influencia del tamaño de paquete

En esta sección se va a estudiar la influencia del tamaño de paquete utilizado por las aplicaciones. InfiniBand especifica que el tamaño de paquete debe estar comprendido entre 256 y 4096 bytes. En esta sección vamos a utilizar cuatro valores comprendidos entre el mínimo y el máximo: 256, 1024, 2038 y 4096 bytes. En todos los casos estas cantidades corresponden al MTU (maximum transfer unit) sin incluir la cabecera del paquete.

Para obtener los resultados que se mostrarán a lo largo de esta sección, se ha fijado el resto de parámetros estudiados. En concreto, se ha utilizado una red irregular de 16 conmutadores, con los enlaces funcionando a velocidad 1x, y con tamaño de buffer capaz de almacenar cuatro paquetes completos. Hay que señalar que al ser el buffer proporcional al tamaño de paquete, el tamaño de buffer utilizado en cada prueba será distinto para posibilitar que pueda almacenar cuatro paquetes completos.

En la Tabla 6.5 pueden observarse los resultados obtenidos para los tamaños de paquete considerados. Hay que señalar que los resultados son muy similares en todos los casos.

	Tamaño de paquete (Bytes)			
	256	1024	2048	4096
Tráfico inyectado (Bytes/Ciclo/Nodo)	0,7258	0,7432	0,7507	0,7607
Tráfico entregado (Bytes/Ciclo/Nodo)	0,7258	0,7432	0,7507	0,7607
Utilización media de los interfaces de los hosts (%)	72,58	74,32	75,07	76,07
Utilización media de los puertos de los conm. (%)	73,48	75,50	75,04	75,13
Reserva media de los interfaces de los hosts (Mbps)	1848,67	1867,80	1882,44	1905,32
Reserva media de los puertos de los conm. (Mbps)	1871,75	1897,58	1881,69	1881,63
Número de conexiones establecidas	111813	113233	115779	118938

Tabla 6.5: Tráfico y utilización para los tamaños de paquete considerados.

La utilización alcanzada crece un poco conforme se aumenta el tamaño de paquete. Esto es debido a que se han conseguido aceptar más conexiones. Conforme se aumenta el tamaño de los paquetes disminuye la sobrecarga que supone la cabecera de dichos paquetes. Para tamaño de paquete pequeño la sobrecarga que supone la cabecera es más importante que para tamaños de paquete grandes. De esta forma, para tamaños de paquete grandes se consiguen establecer más conexiones, y por tanto la red consigue

utilizaciones y cargas un poco mayores. Lo mismo sucede para el resto de índices evaluados. Hay que destacar que en todos los casos la red consigue entregar todo el tráfico que se le inyecta.

En la Figura 6.10 puede observarse la latencia sufrida por los paquetes de los distintos niveles de servicio, para los distintos tamaños de paquete considerados. Conforme va aumentando el tamaño, los paquetes tardan más en llegar. Esto se debe a que, como se vio en la tabla anterior, la red está más cargada. Hay más tráfico en la red y por tanto hay algo más de contención. Sin embargo, en todos los casos, todos los paquetes consiguen llegar a su destino antes de que expire su deadline.

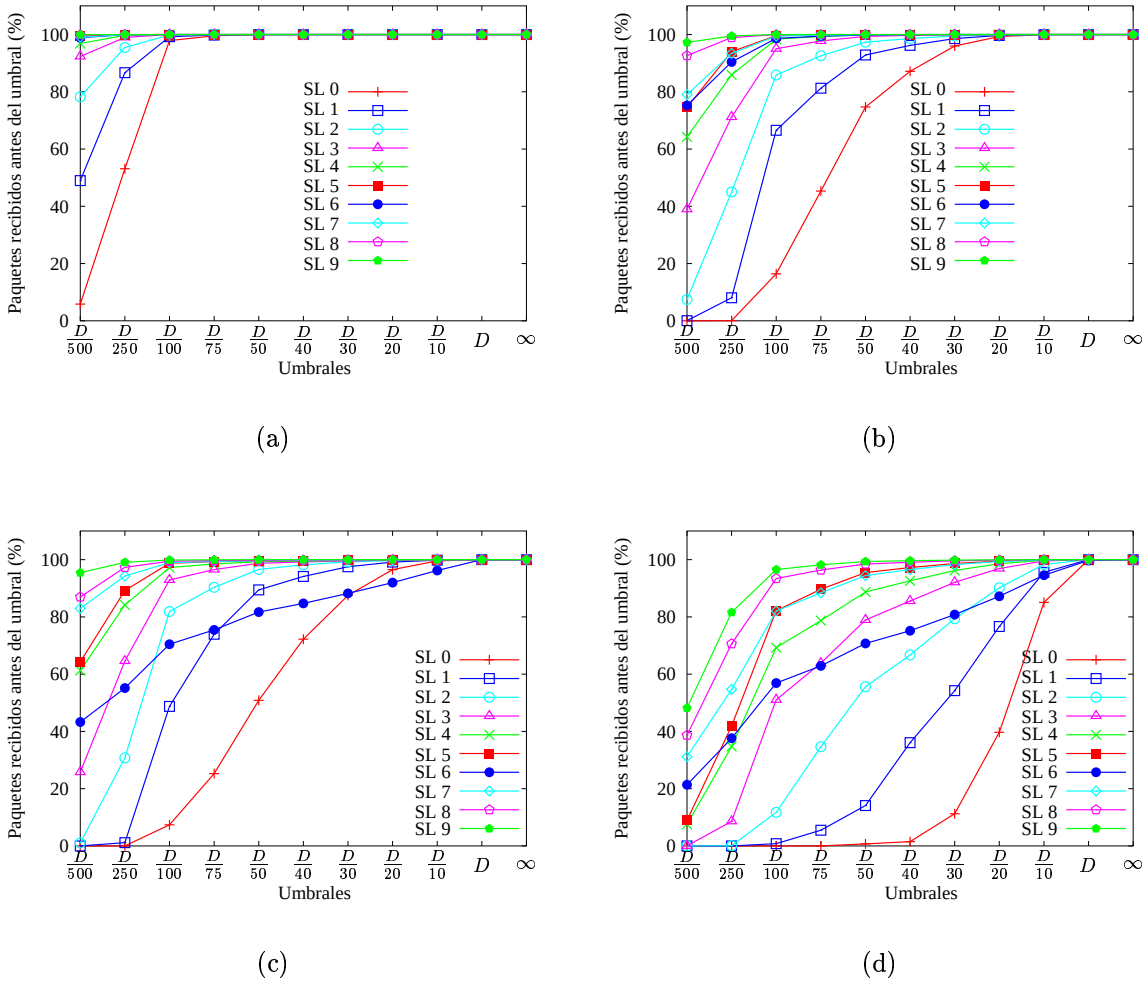


Figura 6.10: Distribución del retraso de los paquetes para los tamaños de paquete considerados: (a) 256 bytes, (b) 1024 bytes, (c) 2048 bytes y (d) 4096 bytes.

Como los umbrales son distintos para cada conexión (relativos a su propio deadline), y por tanto distintos para cada nivel de servicio, los niveles de servicio más restrictivos

llegan más cerca de que expire su deadline, pues éste es mucho más pequeño que para el resto de niveles de servicio.

En la Figura 6.11 puede verse el jitter medio de los paquetes para los tamaños de paquete de 256 y 1024 bytes, y en la Figura 6.12 para los tamaños de paquete de 2048 y 4096 bytes. Para todos los tamaños de paquete considerados, para los niveles de servicio más restrictivos (niveles de servicio 0, 1, 2, 3 y 4), todos los paquetes llegan a su destino en su intervalo central. Como ya se vio en secciones anteriores, esto es debido a que estos niveles de servicio son más prioritarios, pero también a que tienen un rango de ancho de banda más pequeño que algunos otros niveles de servicio, con lo que su IAT es algo mayor, y por tanto su intervalo central será mayor.

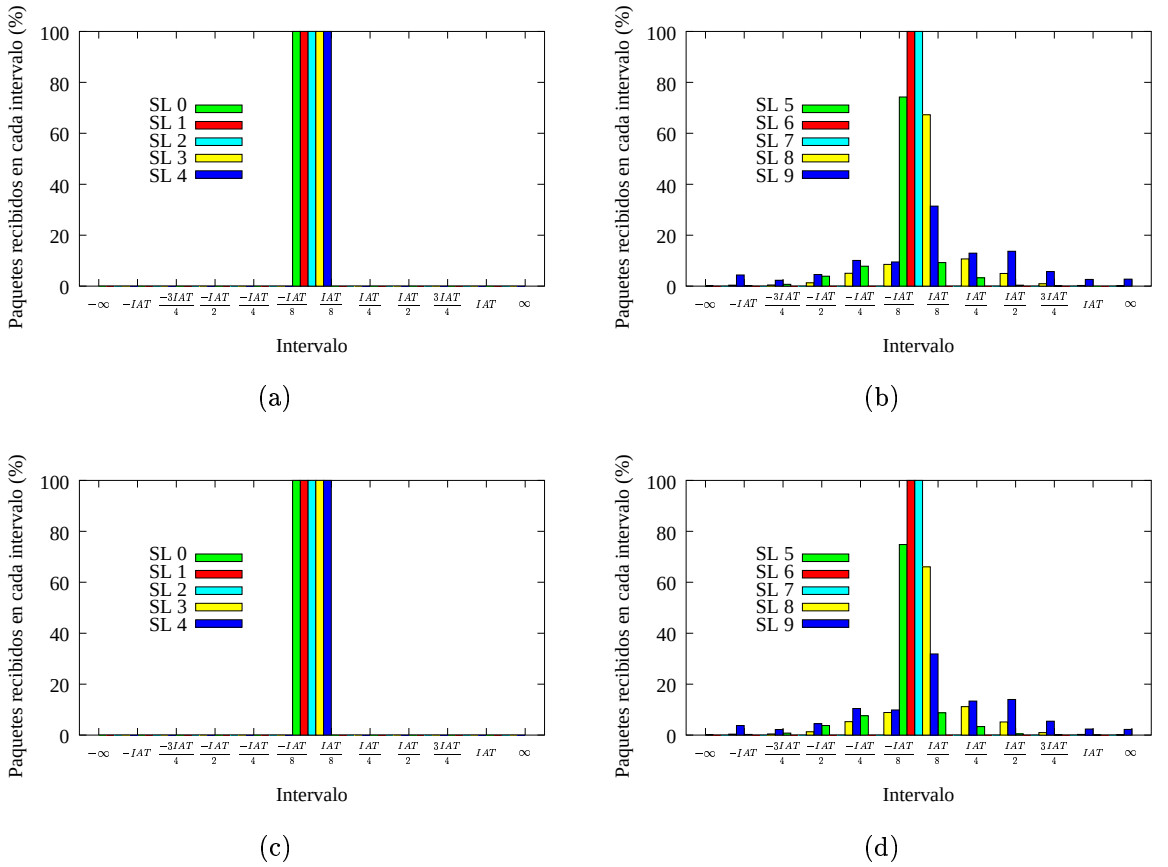


Figura 6.11: Jitter medio de los paquetes para (a) y (b) paquete de 256 bytes, y (c) y (d) paquete de 1024 bytes. Las figuras (a) y (c) muestran los resultados para los SLs 0, 1, 2, 3 y 4, mientras que las figuras (b) y (d) lo hacen para los SLs 5, 6, 7, 8 y 9.

Para los niveles de servicio menos restrictivos (niveles de servicio 5, 6, 7, 8 y 9) la gráfica del jitter tiene forma de campana de Gauss. En estos casos, la mayoría de los paquetes llegan en los intervalos centrales, disminuyendo el número conforme nos alejamos de dicho intervalo central. Igual que antes, la razón de este hecho es que estos niveles de servicio son menos prioritarios, pero también que algunos de ellos

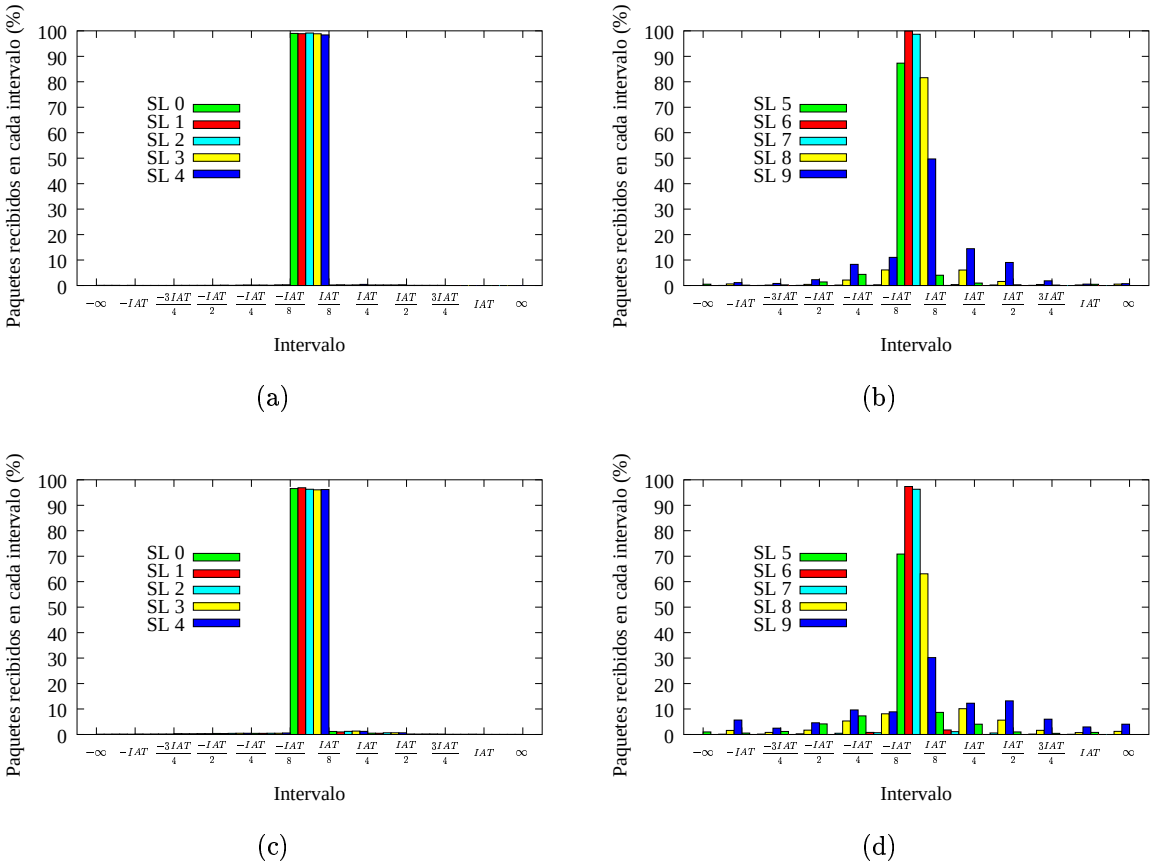


Figura 6.12: Jitter medio de los paquetes para (a) y (b) paquete de 2048 bytes, y (c) y (d) paquete de 4096 bytes. Las figuras (a) y (c) muestran los resultados para los SLs 0, 1, 2, 3 y 4, mientras que las figuras (b) y (d) lo hacen para los SLs 5, 6, 7, 8 y 9.

tienen rangos de ancho de banda mayores, y por tanto su IAT será menor, con lo que sus intervalos son más pequeños. Por ejemplo, podríamos señalar el caso del nivel de servicio 9, donde las conexiones tienen un rango de ancho de banda entre 64 Mbps y 255 Mbps, y para el cual su IAT es mucho menor que el de otros niveles de servicio con anchos de banda en torno a 1 Mbps.

Sin embargo, hay que señalar que para el jitter no se aprecian diferencias entre los distintos tamaños de paquete considerados. Según los resultados obtenidos hay otros factores que influyen en este índice (ancho de banda, prioridad del nivel de servicio, etc.), pero todos los tamaños de paquete obtienen unos resultados similares.

Por último, en la Figura 6.13 pueden observarse las conexiones que han entregado el mayor y el menor porcentaje de paquetes para el umbral $\frac{Deadline}{100}$. De nuevo, hay que señalar que este umbral es distinto para cada conexión, y que está en función de la latencia máxima admisible por la propia conexión.

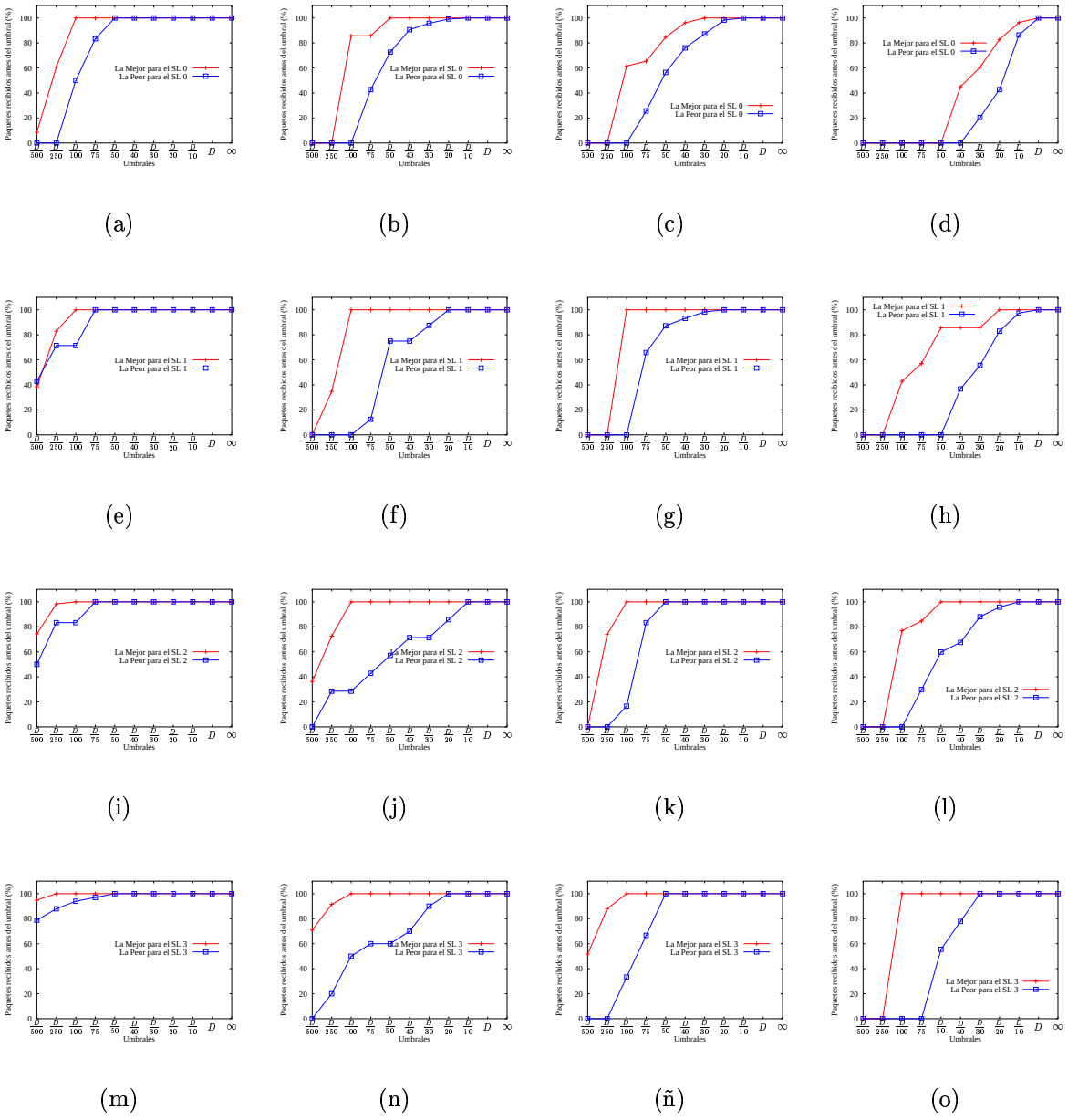


Figura 6.13: La mejor y la peor conexión para los niveles de servicios con requisitos más restrictivos, para (a), (e), (i) y (m) paquete de 256 bytes, (b), (f), (j) y (n) paquete de 1024 bytes, (c), (g), (k) y (ñ) paquete de 2048 bytes, y (d), (h), (l) y (o) paquete de 4096 bytes.

Al igual que sucedía con la latencia media mostrada en la Figura 6.10, conforme se aumenta el tamaño de paquete las consideradas mejores y peores conexiones entregan la mayoría de sus paquetes más tarde. Sin embargo, en todos los casos, incluso las consideradas peores conexiones consiguen entregar todos sus paquetes antes de que expire su deadline.

También hay que señalar que para todos los tamaños de paquete considerados, la diferencia entre la considerada mejor y la peor conexión es muy similar, no habiendo diferencias notables en este sentido. Como ya se ha apuntado anteriormente, este hecho es importante pues supone que la planificación propuesta consigue proporcionar un trato similar a todas las conexiones de un mismo nivel de servicio.

En resumen, el tamaño de paquete tiene una influencia colateral en la QoS que se proporciona a las aplicaciones. La metodología propuesta consigue garantizar los requisitos demandados por todas las conexiones aceptadas, independientemente del tamaño de paquete considerado. Sin embargo, para tamaños de paquete mayores, donde la cabecera de los paquetes supone una menor sobrecarga para la red, se consiguen establecer más conexiones. Este hecho resulta fundamental para los resultados obtenidos pues, al conseguir establecer más conexiones, se alcanza una utilización mayor en la red, con lo que ésta presenta algo más de contención. Esto afecta a la latencia media que consiguen las aplicaciones, que crece ligeramente al aumentar el tamaño de paquete.

Sin embargo, se puede concluir que el tamaño de paquete no tiene una influencia directa en la metodología propuesta en este trabajo, ya que en todos los casos se consiguen alcanzar los requisitos demandados por las aplicaciones. Simplemente cuesta más o menos, en función de la carga de la red, pero el dimensionado de los recursos y la planificación realizada es correcta.

6.4.4. Estudio de la influencia de la velocidad de los enlaces

En esta sección se va a estudiar el comportamiento de las propuestas realizadas, variando ahora la velocidad de los enlaces de la red, y dejando fijos el resto de parámetros. Así pues, en las pruebas realizadas para esta sección se ha considerado una red con topología irregular de 16 conmutadores (64 hosts conectados), tamaño de paquete pequeño (256 bytes) y buffers con capacidad para almacenar cuatro paquetes.

En la Tabla 6.6 pueden observarse los resultados obtenidos para las tres velocidades posibles de InfiniBand. Hay que señalar que los resultados obtenidos son muy similares en todos los casos, salvo para el caso del número de conexiones establecidas, que lógicamente aumenta cuando crece el ancho de banda del enlace.

	Velocidad		
	1x	4x	12x
Tráfico inyectado (Bytes/Ciclo/Nodo)	0,7258	0,7296	0,7192
Tráfico entregado (Bytes/Ciclo/Nodo)	0,7258	0,7296	0,7192
Utilización media de los interfaces de los hosts (%)	72,58	72,96	71,92
Utilización media de los puertos de los conmutadores (%)	73,48	73,62	72,87
Reserva media de los interfaces de los hosts (Mbps)	1848,67	7635,14	22719,68
Reserva media de los puertos de los conmutadores (Mbps)	1871,75	7699,73	23298,27
Número de conexiones establecidas	111813	313233	941938

Tabla 6.6: Tráfico y utilización para las tres velocidades posibles de los enlaces.

El aumento del número de conexiones no es proporcional al incremento en la anchura de los enlaces, pues como ya se ha señalado anteriormente, el algoritmo de encaminamiento utilizado no hace un correcto balanceado del tráfico, con lo que hay ciertas rutas que están más cargadas que otras. De esta forma, la reserva y utilización para ciertos enlaces está en el máximo, mientras que en otros se queda a media carga.

En todos los casos, todas las aplicaciones consiguen satisfacer los requisitos que habían demandado en el momento del establecimiento de la conexión, independientemente de la velocidad de los enlaces.

En la Figura 6.14 puede observarse la distribución del retraso de los paquetes para cada nivel de servicio. Como siempre, los niveles de servicio más restrictivos tardan algo más en entregar la totalidad de sus paquetes, aunque en todos los casos lo hacen bastante antes de que expire su deadline.

Los resultados obtenidos para las tres velocidades de InfiniBand son muy similares, no presentando excesivas diferencias entre ellos. Al aumentar el ancho de banda de los enlaces se establecen más conexiones, pero la planificación realizada tiene un comportamiento similar.

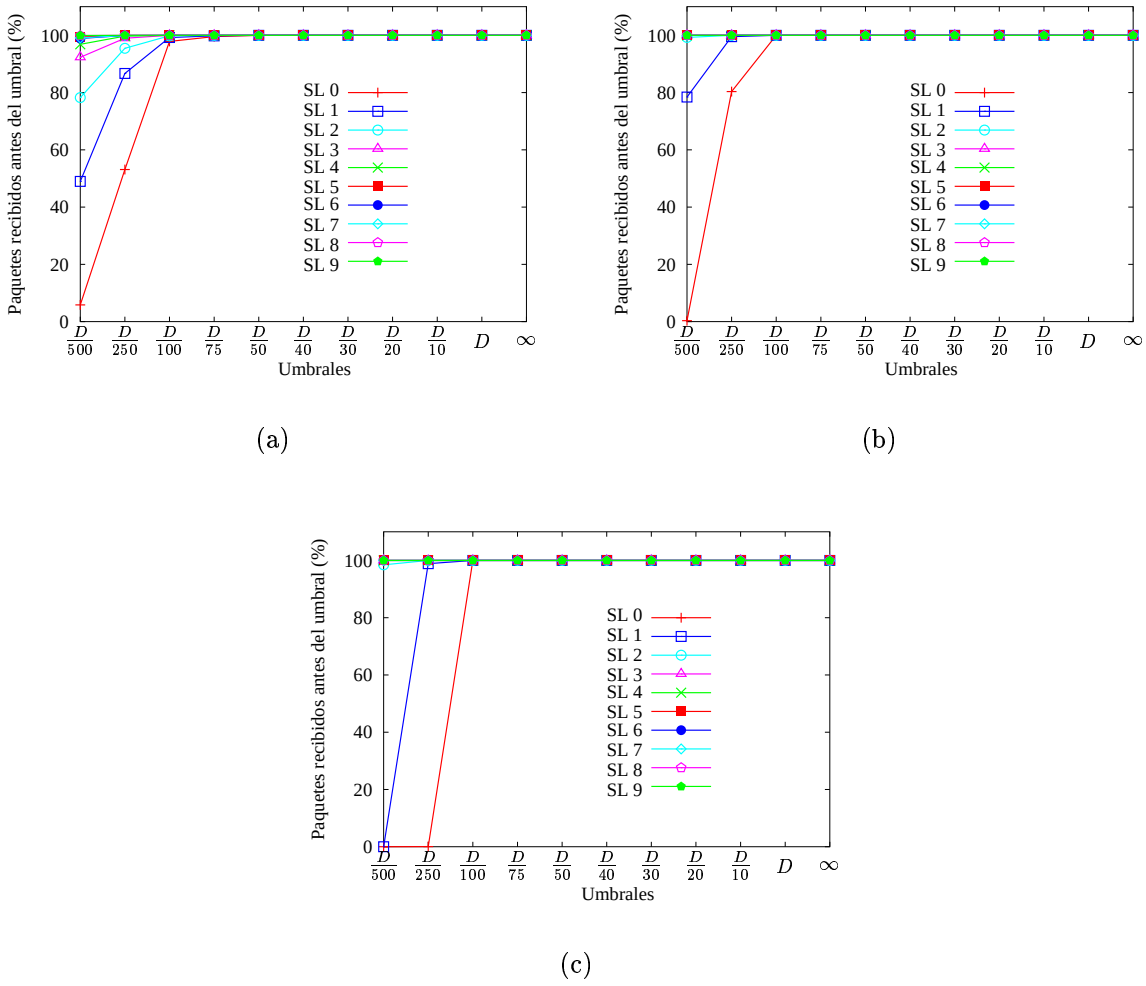


Figura 6.14: Distribución del retraso de los paquetes para las tres velocidades posibles del enlace: (a) 1x, (b) 4x y (c) 12x.

En la Figura 6.15 puede observarse la variabilidad en el retraso de la llegada de los paquetes para las tres velocidades y los distintos niveles de servicio. En todos los casos, los niveles de servicio más prioritarios entregan sus paquetes durante el intervalo central. Para el resto de niveles de servicio el jitter tiene forma de campana, estando la mayoría de los paquetes en el intervalo central.

Nuevamente, tampoco se aprecian diferencias en el jitter obtenido por las distintas velocidades de InfiniBand. Parece que la planificación realizada consigue adaptarse al incremento de conexiones que se da entre las tres velocidades debido al aumento de ancho de banda.

Por último, en la Figura 6.16 se muestran los resultados obtenidos para la considerada mejor y peor conexión para un determinado umbral, para los niveles de servicio más restrictivos.

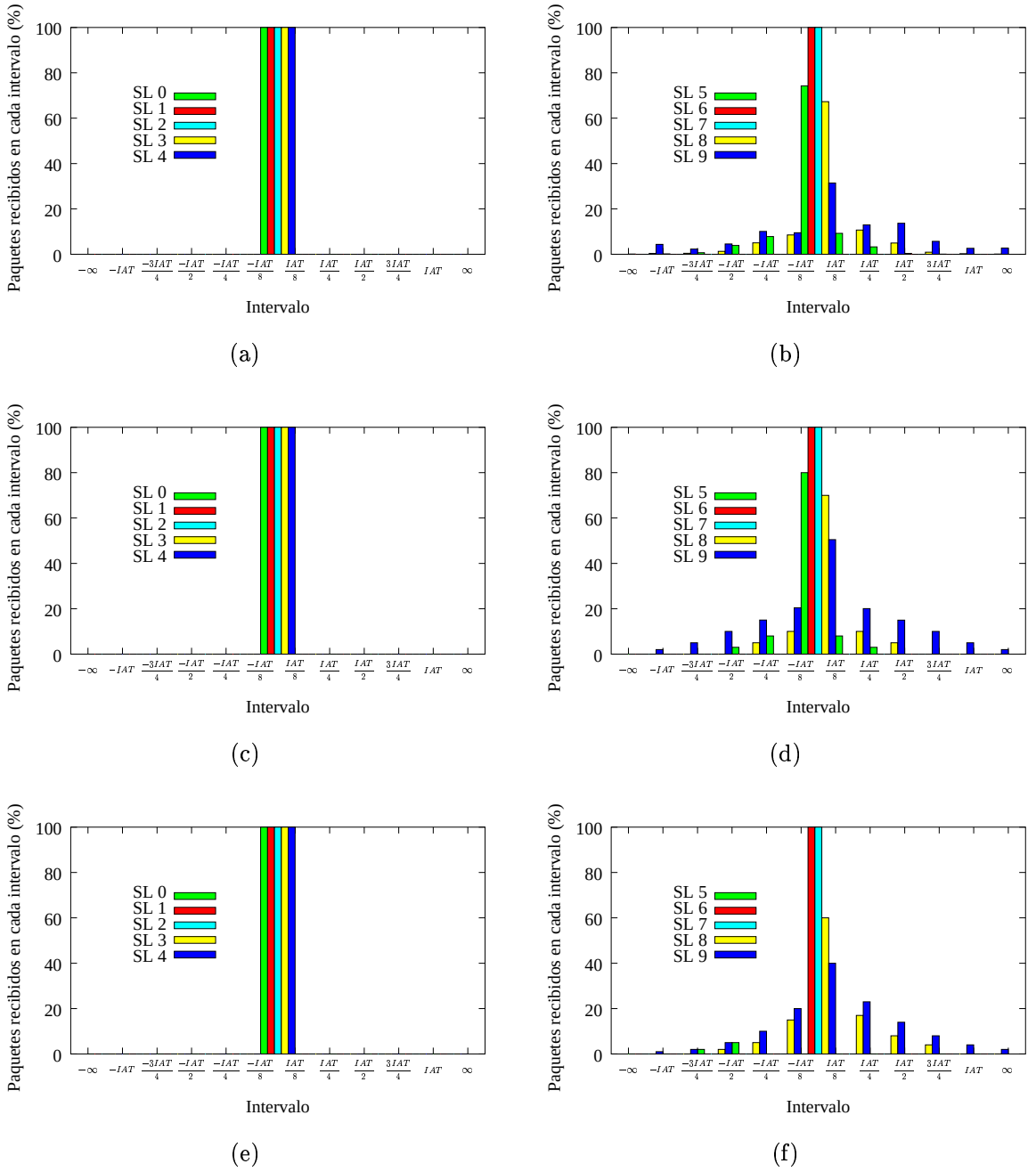


Figura 6.15: Jitter medio de los paquetes para (a) y (b) velocidad 1x, (c) y (d) velocidad 4x y (e) y (f) velocidad 12x. Las figuras (a), (c) y (e) muestran los resultados para los SLs 0, 1, 2, 3 y 4, mientras que las figuras (b), (d) y (f) lo hacen para los SLs 5, 6, 7, 8 y 9.

En todos los casos, incluso las consideradas peores conexiones, entregan todos sus paquetes en el destino con suficiente antelación. Además, no hay muchas diferencias entre la considerada mejor y peor conexión. Por otra parte, tampoco se aprecian diferencias en los resultados mostrados al variar la velocidad de los enlaces.

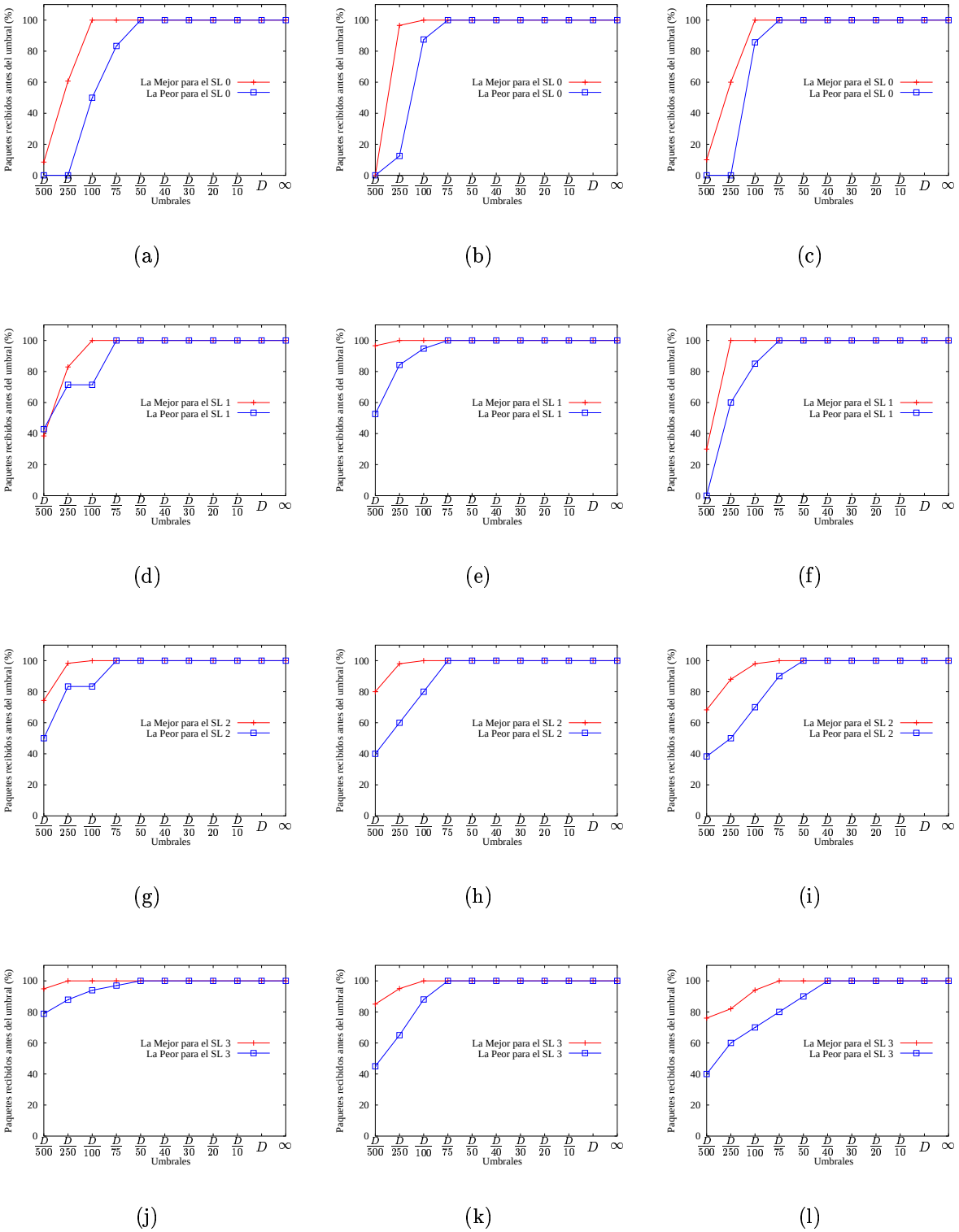


Figura 6.16: La mejor y la peor conexión para los niveles de servicios con requisitos más restrictivos, para (a), (d), (g) y (j) velocidad 1x, (b), (e), (h) y (k) velocidad 4x y (c), (f), (i) y (l) velocidad 12x.

Así pues, no se ha apreciado influencia de la velocidad de los enlaces en la QoS que se proporciona a las aplicaciones. La planificación realizada funciona adecuadamente independientemente de que se establezcan más conexiones pues, al disponer de mayor ancho de banda, el tener más conexiones no hace que aumente la congestión.

6.4.5. Estudio de la influencia del tamaño de los buffers

En esta sección se va a estudiar la influencia del tamaño de los buffers de los puertos e interfaces en la QoS proporcionada a las aplicaciones. Para ello se van a fijar el resto de parámetros y se variará el tamaño de los buffers. De esta forma, para las simulaciones realizadas para esta sección se ha considerado una red irregular con 16 conmutadores (64 hosts), tamaño de paquete pequeño (256 bytes), y enlaces funcionando a la velocidad básica de 1x.

Se han considerado cuatro tamaños de buffer proporcionales al tamaño de paquete. En concreto, los tamaños de buffer considerados permiten almacenar 1, 2, 3 ó 4 paquetes completos. Como se ha comentado anteriormente, para estas pruebas se han utilizado paquetes de tamaño pequeño (256 bytes). Como el tamaño de paquete no incluye la cabecera LRH, que son otros 8 bytes, a la cantidad correspondiente al número de paquetes deseados se le ha añadido la cantidad correspondiente a las cabeceras de esos paquetes. De esta forma, los tamaños de buffer considerados son 264, 528, 792 y 1056 bytes, respectivamente.

En la Tabla 6.7 pueden observarse los resultados obtenidos para los tamaños de buffer considerados. Hay que señalar que en todos los casos los resultados obtenidos son muy similares, independientemente del tamaño de buffer considerado. La red alcanza una utilización similar, y la reserva realizada en los distintos puertos e interfaces es independiente del tamaño de buffer usado.

	Tamaño de buffer (en paquetes)			
	1	2	3	4
Tráfico inyectado (Bytes/Ciclo/Nodo)	0,7262	0,7262	0,7258	0,7258
Tráfico entregado (Bytes/Ciclo/Nodo)	0,6788	0,7262	0,7258	0,7258
Utilización media de los interfaces de los hosts (%)	67,88	72,63	72,58	72,58
Utilización media de los puertos de los conmu. (%)	68,06	73,49	73,48	73,48
Reserva media de los interfaces de los hosts (Mbps)	1849,81	1849,81	1848,67	1848,67
Reserva media de los puertos de los conmu. (Mbps)	1871,84	1877,84	1871,75	1871,75

Tabla 6.7: Tráfico y utilización para los tamaños de buffer considerados.

Sin embargo, sí que hay una pequeña diferencia para el caso de tamaño de buffer de un único paquete. En este caso, la red no es capaz de entregar todo el tráfico que se

le inyecta, estando un poco por debajo de éste. En el resto de los casos la red consigue entregar todo el tráfico que se le inyecta.

En la Figura 6.17 puede observarse la cantidad de paquetes que se entregan para cada tamaño de buffer considerado y cada nivel de servicio en cada uno de los umbrales considerados. Como puede apreciarse, no hay diferencias en las prestaciones conseguidas por los distintos niveles de servicio al variar el tamaño del buffer.

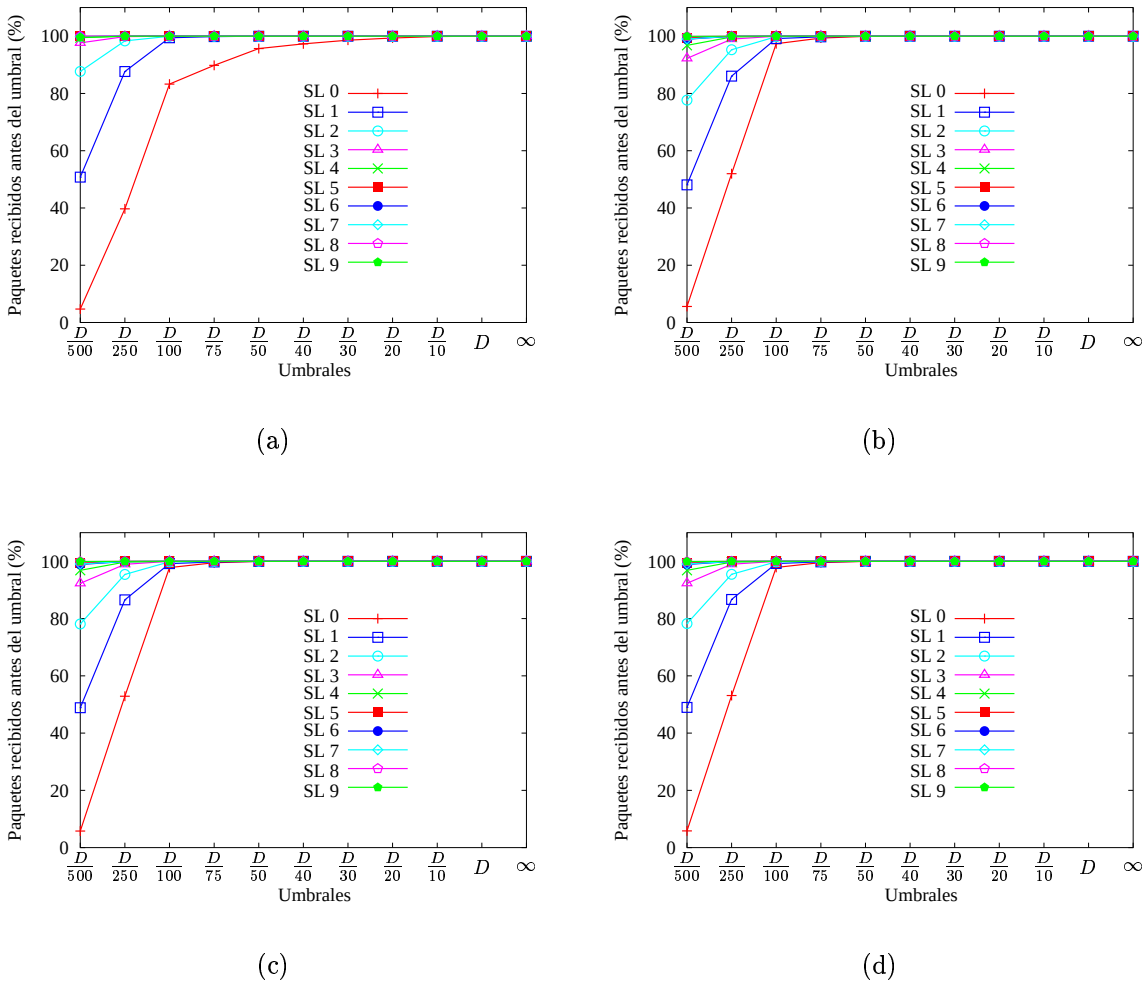


Figura 6.17: Distribución del retraso de los paquetes para los tamaños de buffer considerados: (a) 1 paquete, (b) 2 paquetes, (c) 3 paquetes y (d) 4 paquetes.

Como en los casos anteriores, los umbrales considerados son distintos para cada conexión y relativos a su propio deadline. Así, los niveles de servicio más restrictivos consiguen entregar todos sus paquetes con menos margen, pero eso también es debido a que su deadline es mucho más restrictivo y por tanto también lo son sus umbrales. Sin embargo, en todos los casos, los paquetes se entregan con suficiente antelación como para que puedan ser tratados a tiempo.

Para el caso de tamaño de buffer de un único paquete, y en el caso del nivel de servicio más restrictivo (SL 0), los paquetes llegan un poco más cerca de su deadline, y distribuidos entre muchos de los umbrales. Sin embargo, todos los paquetes de este nivel de servicio y esta configuración llegan con suficiente anticipación para poder ser tratados sin problemas.

En la Figura 6.18 se muestra el jitter de cada nivel de servicio para los tamaños de buffer correspondientes a 1 y 2 paquetes, mientras que en la Figura 6.19 se hace para el tamaño de buffer de 3 y 4 paquetes. Nuevamente, los umbrales son distintos para cada conexión y relativos a su propio IAT.

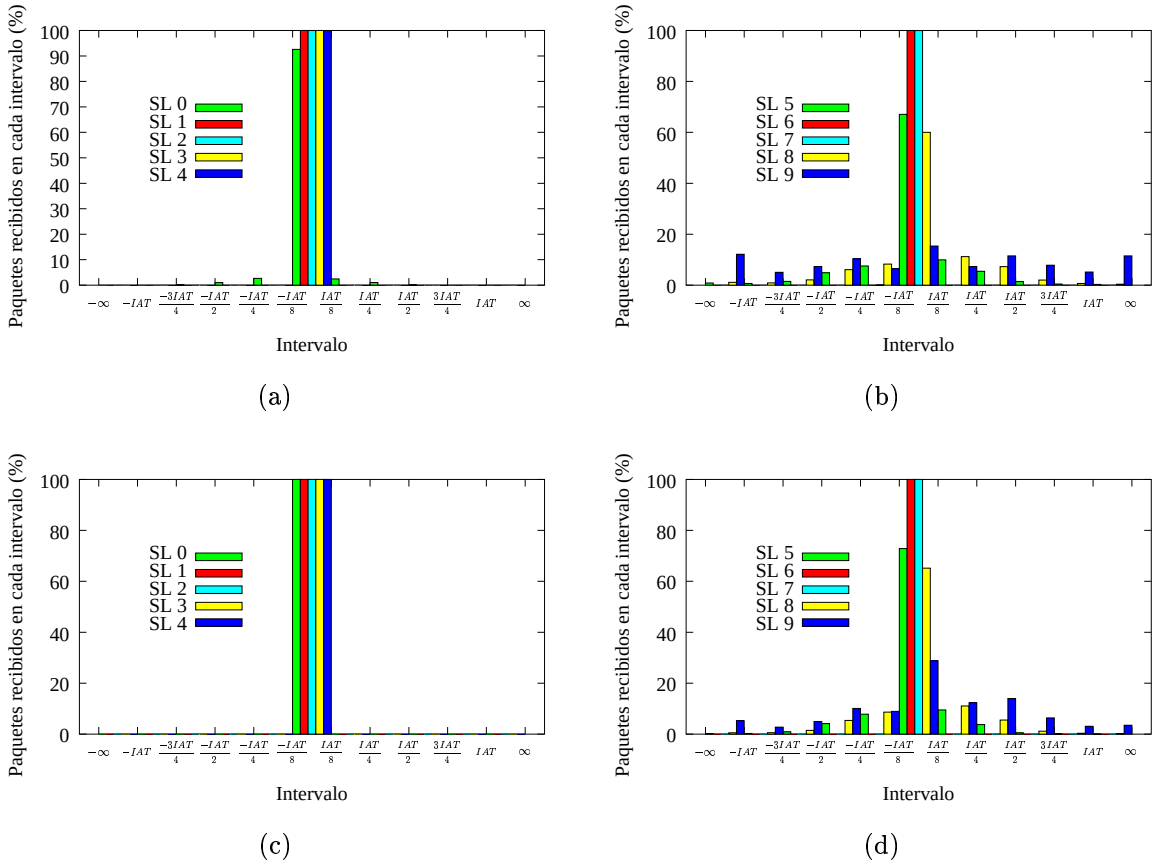


Figura 6.18: Jitter medio de los paquetes para (a) y (b) buffer de 1 paquete y (c) y (d) buffer de 2 paquetes. Las figuras (a) y (c) muestran los resultados para los SLs 0, 1, 2, 3 y 4, mientras que las figuras (b) y (d) lo hacen para los SLs 5, 6, 7, 8 y 9.

Hay que señalar que para el tamaño de buffer correspondiente a un único paquete, el jitter presenta unos resultados ligeramente peores. Para los niveles de servicio más prioritarios, son muy similares, aunque el SL 0 no consigue que lleguen todos sus paquetes en el intervalo central.

Sin embargo, para los niveles de servicio menos prioritarios la forma tradicional de campana se expande un poco más. En concreto, para el nivel de servicio 9, se pierde la

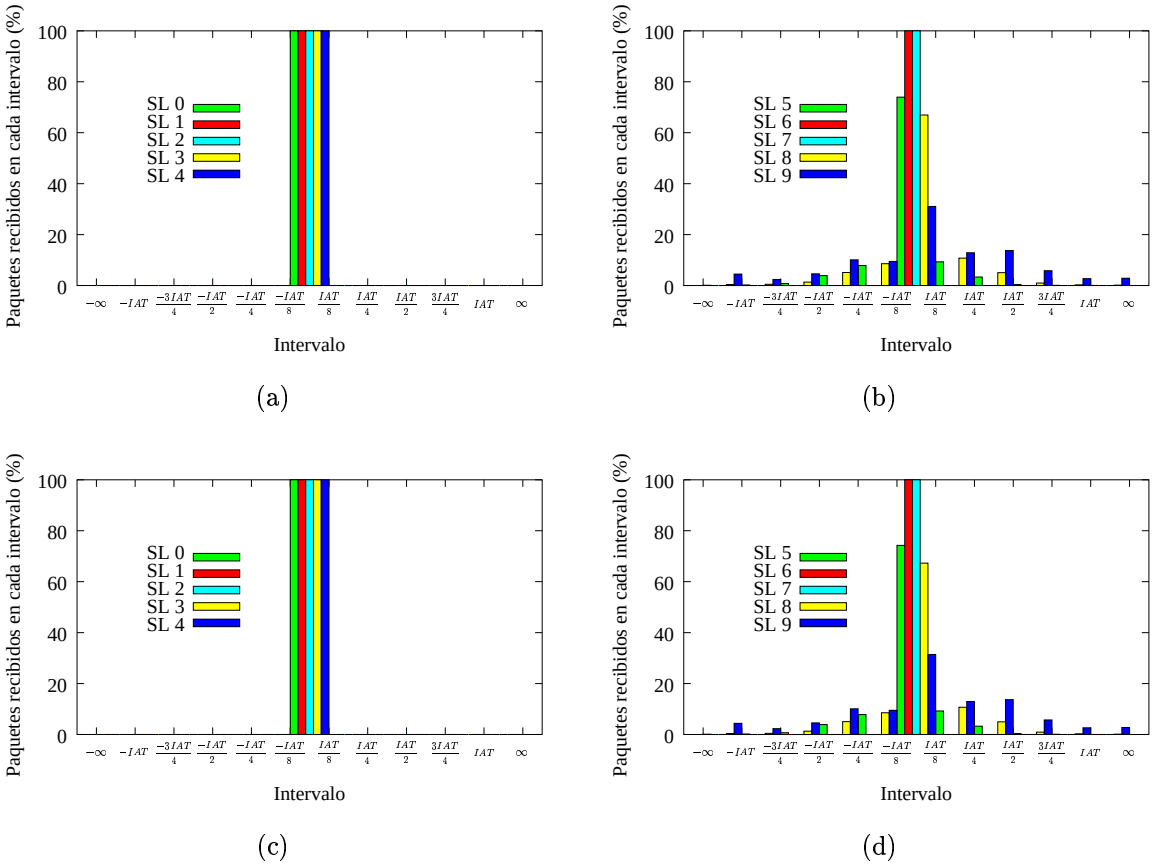


Figura 6.19: Jitter medio de los paquetes para (a) y (b) buffer de 3 paquetes y (c) y (d) buffer de 4 paquetes. Las figuras (a) y (c) muestran los resultados para los SLs 0, 1, 2, 3 y 4, mientras que las figuras (b) y (d) lo hacen para los SLs 5, 6, 7, 8 y 9.

forma de campana, estando la variabilidad de los retrasos distribuida casi igualmente entre todos los intervalos considerados. Esto es debido a que al haber menos espacio de buffer es más fácil que se produzca variabilidad en los retrasos, pues a veces los paquetes deberán esperar a que haya espacio en el buffer destino antes de cruzar.

Hay que recordar que para la configuración con buffers con capacidad para un único paquete, la red no es capaz de entregar todo el tráfico que se le inyecta. Esto hace que su productividad baje un poco, y aparezca un poco de congestión. En esta situación, es bastante probable que los retrasos de los paquetes varíen bastante, sobre todo para los niveles de servicio menos prioritarios.

Sin embargo, a pesar de que para este tamaño de buffer los paquetes llegan a su destino con mucha variabilidad, todos ellos lo hacen dentro de su margen máximo de latencia, tal y como se ha mostrado en la Figura 6.17. De esta forma, los paquetes pueden sufrir variabilidad en sus retrasos, pero la planificación realizada consigue cumplir la garantía que se le aseguró a la aplicación cuando ésta fue establecida.

Por último, en la Figura 6.20 se muestran las conexiones consideradas mejor y peor para un cierto umbral, para los niveles de servicio más prioritarios. Nuevamente, hay que recordar que estos umbrales son distintos para cada conexión y relativos a su propio requisito en cuanto a latencia máxima.

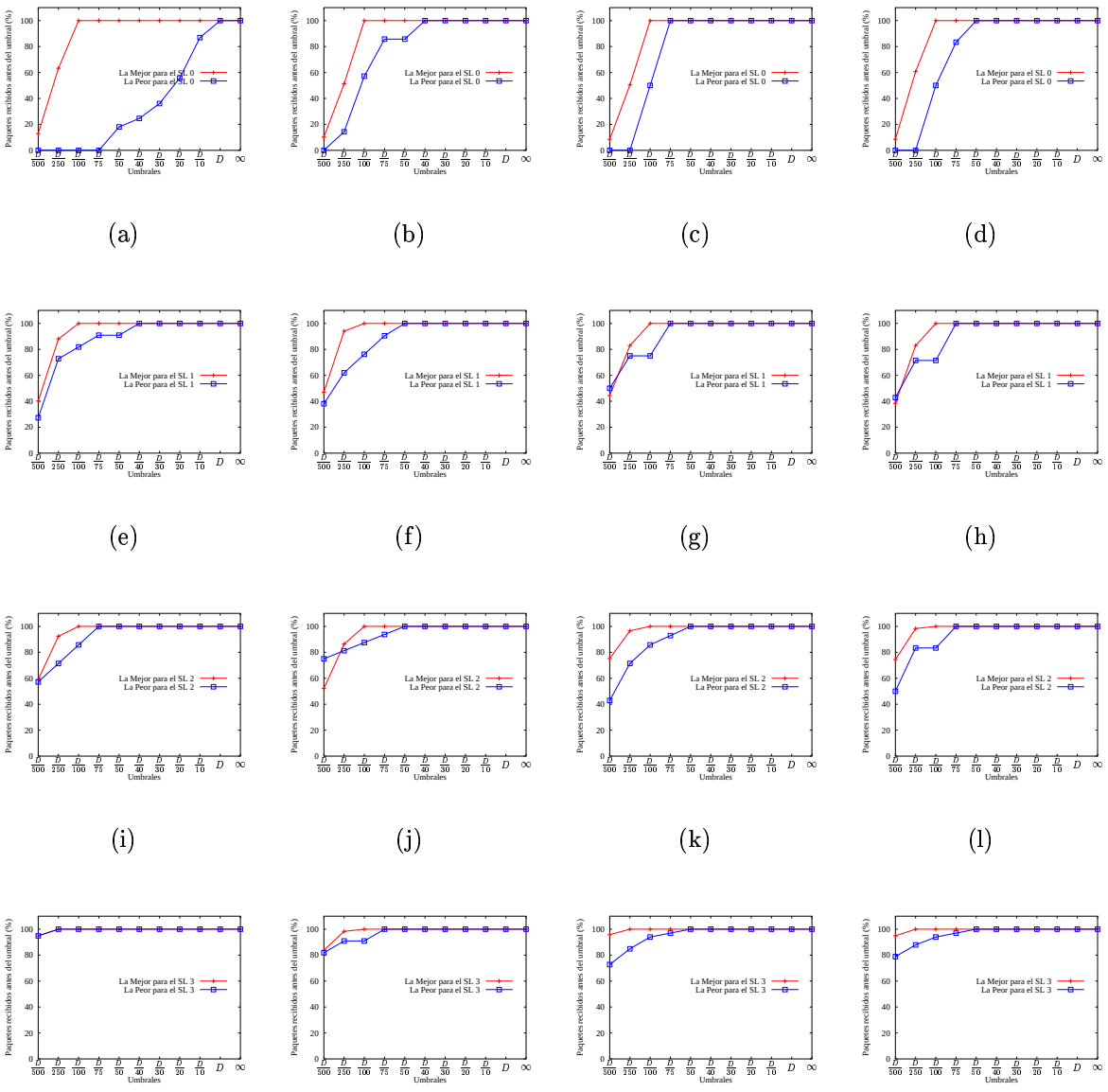


Figura 6.20: La mejor y la peor conexión para los niveles de servicios con requisitos más restrictivos, para (a), (e), (i) y (m) buffer de 1 paquete, (b), (f), (j) y (n) buffer de 2 paquetes, (c), (g), (k) y (ñ) buffer de 3 paquetes, y (d), (h), (l) y (o) buffer de 4 paquetes.

Hay que señalar que para el nivel de servicio más prioritario, el SL 0, en el caso de tamaño de buffer de un único paquete, hay mucha diferencia entre la considerada

mejor y peor conexión. Esto se debe a la influencia de las esperas que tienen que sufrir en algunas ocasiones los paquetes al no haber espacio disponible en el buffer destino.

Así pues, se ha visto que el tamaño de buffer tiene una influencia relativa en la QoS proporcionada. Aunque en todos los casos las aplicaciones consiguen holgadamente los requisitos que habían demandado durante el establecimiento, es más difícil de conseguir en los casos donde se dispone de menos buffer. En concreto, para el caso de buffer de un único paquete, aparece congestión y por tanto la red no consigue entregar a tiempo todo lo que se le inyecta. Para el resto de tamaños considerados la red tiene un buen comportamiento.

En cualquier caso, la metodología propuesta consigue garantizar las necesidades planteadas por las aplicaciones independientemente del tamaño de buffer considerado. Incluso para los tamaños de buffer más pequeños, todas las conexiones alcanzan sus requisitos.

6.5. Resumen

En este capítulo se ha presentado la evaluación experimental de las propuestas realizadas en los capítulos anteriores. Para ello se ha utilizado un simulador de InfiniBand desarrollado durante el transcurso de esta tesis. Este simulador modela las principales características de las especificaciones de InfiniBand, haciendo especial hincapié en todos aquellos aspectos relacionados con la provisión de QoS a las aplicaciones.

Los índices de prestaciones utilizados para la evaluación han sido tanto los habituales en los estudios de redes (tráfico, utilización de los enlaces, latencia, etc.), como los que permiten evaluar la QoS recibida por las aplicaciones (reserva de los enlaces, jitter, mejor/peor conexión, etc.). El conjunto de estos índices ha permitido hacer una evaluación de la metodología propuesta mediante la variación de algunos parámetros que se ha considerado podrían tener influencia. En concreto, se ha variado la topología de la red de interconexión, el tamaño de red, el tamaño del paquete de datos, la velocidad de los enlaces y el tamaño de los buffers de los conmutadores y hosts.

En todos los casos estudiados se ha comprobado que al variar los distintos aspectos considerados, las propuestas realizadas consiguen proporcionar la QoS previamente requerida por las aplicaciones. Se ha visto que ninguno de los parámetros variados tiene una influencia directa, aunque algunos de ellos sí que tienen una influencia colateral pues permiten alcanzar tasas de tráfico un poco mayores, lo que hace que la red esté algo más congestionada y cueste un poco más conseguir la QoS solicitada. Sin embargo, aún en estos casos donde la red está más cargada, la planificación realizada mediante la correcta configuración de las tablas de arbitraje consigue que todas las aplicaciones alcancen la QoS que habían solicitado.

Capítulo 7

Conclusiones

Una vez se ha expuesto exhaustivamente todo el trabajo realizado, este capítulo está dedicado a resumir las principales conclusiones obtenidas a partir de él. Asimismo, se indican las contribuciones a diversos congresos y revistas a las que ha dado lugar. Por último, se plantearán algunas posibles líneas de investigación que se pueden llevar a cabo como continuación del trabajo presentado en esta memoria.

7.1. Conclusiones y aportaciones

En la Sección 1.6.3 se planteaban una serie de objetivos a cubrir al principio de este trabajo. Es momento ahora de recapitular y comprobar si se han cubierto total o parcialmente los objetivos planteados al inicio del trabajo. Repasemos pues esos objetivos, e indiquemos en qué grado se han cumplido:

1. Estudio de las especificaciones de InfiniBand. Este objetivo ha sido ampliamente cubierto durante el desarrollo de este trabajo. Para ello se han utilizado las propias especificaciones de InfiniBand [Inf00], los libros que hasta la fecha hay publicados sobre el tema [Fut01, Sha02], así como los informes técnicos y el resto de material que hay disponible en la página web de InfiniBand [IBA99]. En el Capítulo 2 se han resumido los aspectos más importantes de las especificaciones de InfiniBand. En concreto, en la Sección 2.5 se han detallado ampliamente las principales características que proporciona InfiniBand para proporcionar QoS a las aplicaciones, lo que es el principal objetivo de este trabajo.

2. Realizar un estudio de las necesidades de QoS de las aplicaciones. En la Sección 1.3 se ha realizado un estudio sobre los distintos tipos de aplicaciones que se pueden tener, en función de sus necesidades de QoS. Se ha concluido que los requisitos en cuanto a ancho de banda van a depender de la calidad de la señal y

de las características de la misma, y que los requisitos en cuanto a latencia van a depender del grado de interactividad que presente la aplicación con el usuario. Partiendo de la clasificación realizada por Pelissier [Pel00], se han propuesto en la Sección 1.4 varias modificaciones de cara a priorizar determinados tipos de tráfico. En concreto, con las modificaciones propuestas aquí es posible proporcionar mejor servicio a determinado tipo de aplicaciones (por ejemplo, acceso a un WEB, servicios de bases de datos que se consideren prioritarios, etc.) a los que no pueden dársele garantía por no tener un servicio de conexión.

3. Configurar InfiniBand para proporcionar garantía de ancho de banda. En la Sección 3.2 se ha abordado este tema. Allí se ha estudiado cómo configurar las entradas de la tabla de arbitraje para conseguir garantizar un determinado ancho de banda a una conexión. Se ha visto cómo puede hacerse un reparto del ancho de banda máximo alcanzable, y si las aplicaciones demandan menos, el resto disponible que se reparta entre las aplicaciones que usen más ancho de banda del demandado y por el tráfico sin garantías de ancho de banda. De esta forma no se desperdicia ancho de banda, proporcionándole a cada aplicación lo que tenga garantizado, pero aprovechándose el resto para otras aplicaciones.

4. Configurar InfiniBand para proporcionar garantía de latencia máxima. En la Sección 3.3 se ha estudiado esta cuestión. En concreto, se ha estudiado este tema para varios tipos de conmutadores existentes, y se ha visto el grado de influencia de cada uno de sus componentes. También se ha visto la influencia de la separación máxima entre dos entradas consecutivas en la tabla de arbitraje, de la secuencia asignada al canal virtual que utiliza la conexión. Así pues, para poder garantizar una determinada latencia habrá que buscar una secuencia de entradas en la tabla de arbitraje con una determinada separación máxima entre dos entradas consecutivas de esa secuencia.

5. Plantear un marco global para proporcionar garantía tanto de ancho de banda como de latencia máxima. En la Sección 3.4 se ha indicado cómo proporcionar de manera conjunta garantía de ancho de banda y de latencia. Para ello se ha planteado un marco global para proporcionar QoS. Se ha visto que todas las conexiones que necesiten garantía deben tener su canal virtual ubicado en la tabla de arbitraje de alta prioridad. Lo que se ha propuesto es unificar ambos requisitos y proporcionar un mínimo de garantía de latencia (una entrada de la tabla) incluso a las conexiones que no lo necesiten. De esta forma, la latencia a garantizar determina la separación máxima entre dos entradas consecutivas de la secuencia, y la garantía de ancho de banda determina el peso total que debe tener esa conexión entre las entradas que utilice en la tabla.

6. Desarrollar un modelo formal para el tratamiento de la tabla de arbitraje. En los Capítulos 4 y 5 se ha abordado de forma exhaustiva esta cuestión. Se ha planteado

un modelo formal para el tratamiento de la tabla de arbitraje y se ha probado, por medio de los correspondientes teoremas, el correcto funcionamiento de las propuestas realizadas. Esta tarea se ha abordado en dos partes. En el Capítulo 4 se han considerado solamente las inserciones en la tabla, suponiendo que no hubiera eliminaciones, es decir, que las peticiones nunca se retiran de la tabla de arbitraje. En el Capítulo 5 se ha considerado el caso real con inserciones y eliminaciones, y se han estudiado las particularidades de este hecho. Finalmente, en la Sección 5.4 se ha probado que tener una serie de peticiones y una serie de eliminaciones de esas peticiones es equivalente a tener solamente las peticiones que restan en la tabla tras las eliminaciones, y por tanto todo lo desarrollado en la primera parte es aplicable también a la segunda. En concreto, queda probado en el Teorema 4 que, con la propuesta realizada, se es capaz de ubicar una petición en la tabla siempre que haya entradas suficientes para ello.

7. Realizar un estudio experimental de las propuestas realizadas. Para ello se ha desarrollado un simulador de una subred InfiniBand, que ha sido ampliamente descrito en la Sección 6.2. Este simulador modela las características más importantes de las especificaciones de InfiniBand, y sobre él se han implementado las propuestas realizadas en este trabajo para garantizar QoS a las aplicaciones. El simulador genera tráfico con unos determinados requisitos en cuanto a ancho de banda y latencia máxima, y la red debe estudiar si se pueden satisfacer esos requisitos o debe rechazarse la solicitud de conexión. El tráfico generado cubre un amplio rango, habiendo conexiones que tienen muy poco y conexiones con gran ancho de banda, y de muy poca tolerancia en cuanto a latencia máxima, hasta ser insensibles a dicha latencia máxima. El simulador desarrollado permite medir los índices que se han considerado interesantes para este tipo de estudio. Estos índices han sido descritos y justificados en la Sección 6.3. Por último, en la Sección 6.4 se ha realizado la evaluación de las propuestas realizadas, utilizando este simulador, para un amplio rango de configuraciones posibles. Se ha estudiado la influencia de diversos factores (topología, tamaño de la red, tamaño del paquete, velocidad de los enlaces y tamaño de los buffers), en la QoS obtenida por las aplicaciones.

   En todos los casos se ha comprobado, como al variar los distintos aspectos considerados, las propuestas realizadas consiguen proporcionar la QoS requerida por las aplicaciones. Se ha visto que factores como la topología implementada, el tamaño de la red, el tamaño de paquete y la velocidad de los enlaces, no tienen influencia en las prestaciones que las aplicaciones consiguen. Por otra parte, se ha comprobado como para el caso del tamaño de los buffers es suficiente con que éstos tengan capacidad suficiente para almacenar dos paquetes. Con menos de esa capacidad la red está congestionada. En este caso, las aplicaciones consiguen latencia bastante buena, aunque ésta tiene mucha variabilidad, siendo por tanto el jitter muy alto. Sin embargo, es a partir de un tamaño de dos paquetes

donde se observa que todas las aplicaciones consiguen los requisitos que habían demandado, además de que la red tiene un comportamiento adecuado.

Así pues, se han cumplido totalmente todos los objetivos que se planteaban al comienzo de este trabajo. Se ha propuesto una metodología para conseguir proporcionar garantía de QoS a las aplicaciones en InfiniBand, y se ha probado su correcto funcionamiento, tanto formalmente como experimentalmente.

A modo de resumen, las aportaciones más importantes a destacar de este trabajo son:

- Una nueva propuesta de clasificación del tráfico en función de sus necesidades en cuanto a QoS.
- Una sencilla propuesta para configurar la tabla de arbitraje que consigue proporcionar garantía de ancho de banda y de latencia máxima a las aplicaciones.
- Un modelo formal para la gestión de la tabla de arbitraje, que permite demostrar la corrección de las propuestas realizadas.

Así pues, se puede concluir que las propuestas realizadas permiten configurar InfiniBand de manera adecuada para que sea posible garantizar las necesidades de QoS de las aplicaciones.

7.2. Trabajos publicados

Durante el desarrollo de este trabajo se han presentado, en diversos congresos, partes y versiones del mismo, y se tienen pendientes de aceptación varios artículos en revistas especializadas.

Las publicaciones surgidas hasta el momento, ordenadas cronológicamente, son:

- *A Strategy to Compute the InfiniBand Arbitration Tables*. F.J. Alfaro, J.L. Sánchez, J. Duato. Technical report del Departamento de Informática de la Universidad de Castilla-La Mancha. Número DIAB-01-02-20. Octubre de 2001. `http://raap.info-ab.uclm.es/diab-01-02-20.pdf`.

  En este trabajo se estudian los mecanismos que presenta InfiniBand para proporcionar QoS y se presenta la primera propuesta en cuanto a garantía de ancho de banda. Esta propuesta se evalúa para gran tipo de configuraciones y tipos de tráfico.

- *A Strategy to Compute the InfiniBand Arbitration Tables*. Francisco J. Alfaro, José L. Sánchez, José Duato, Chita R. Das. International Parallel and Distributed Processing Symposium (IPDPS'02). Fort Lauderdale, Florida. Abril de 2002. Proceedings publicados por IEEE Computer Society con I.S.B.N. 0-7695-1573-8. `http://raap.info-ab.uclm.es/pubs/pub61.pdf`.

  Este trabajo es un resumen del technical report anterior, donde se presenta la propuesta realizada para proporcionar garantía de ancho de banda en InfiniBand. En este artículo la evaluación se limita a un par de configuraciones, remitiendo al lector interesado al technical report.

- *A Strategy to Manage Time Sensitive Traffic in InfiniBand*. Francisco J. Alfaro, José L. Sánchez, José Duato. Workshop on Communication Architecture for Clusters (CAC'02), celebrado junto al IPDPS'02, Fort Lauderdale, Florida. Abril 2002. Proceedings publicados por IEEE Computer Society con I.S.B.N. 0-7695-1573-8. `http://raap.info-ab.uclm.es/pubs/pub62.pdf`.

  En este artículo se hace un exhaustivo análisis de cómo proporcionar garantía de ancho de banda en InfiniBand. Se analizan las configuraciones de conmutador más habituales y se calculan las expresiones correspondientes a cada una de ellas. Por último se presenta una primera aproximación muy sencilla para proporcionar garantía en cuanto a latencia máxima. En la evaluación correspondiente se comprueba que las aplicaciones consiguen la garantía que solicitan. Sin embargo, ya en este trabajo se apuntan algunos de los problemas que planteaban las primeras propuestas, y que serán resueltas en trabajos posteriores.

- *Performance Evaluation of VBR Traffic in InfiniBand*. Francisco J. Alfaro, José L. Sánchez, Luis Orozco, José Duato. IEEE Canadian Conference on Electrical & Computer Engineering (CCECE'02). Winnipeg, Manitoba, Canada. Mayo de 2002. Proceedings publicados por IEEE Computer Society con I.S.B.N. 0-7803-7514-9. `http://raap.info-ab.uclm.es/pubs/pub67.pdf`.

  En este trabajo se aborda el tema de proporcionar QoS a las aplicaciones de caudal variable (VBR). Se plantea cómo usar los mecanismos de InfiniBand para conseguir proporcionar QoS a las aplicaciones, pero se concluye que, debido a la variabilidad inherente a este tipo de aplicaciones, no es posible proporcionar garantía absoluta. En la evaluación correspondiente se observa que, en las situaciones habituales, las aplicaciones consiguen la QoS demanda, a pesar de no poder tener garantía absoluta.

- *QoS en Subredes InfiniBand*. Francisco J. Alfaro Cortés, José L. Sánchez García, José Duato Marín. XIII Jornadas de Paralelismo. Lleida. Septiembre 2002. Actas publicadas con I.S.B.N. 84-8409-159-7. `http://raap.info-ab.uclm.es/pubs/pub70.pdf`.

Este trabajo presenta una visión global del trabajo realizado hasta esos momentos. Se presentan las dos propuestas para proporcionar garantía de ancho de banda y latencia, y se plantean los problemas que presentan.

- *Formalizing the Fill-In of the InfiniBand Arbitration Table*, F.J. Alfaro, J.L. Sánchez, M. Menduiña, J. Duato. Technical report del Departamento de Informática de la Universidad de Castilla-La Mancha. Número DIAB-03-02-35. Marzo 2003. `http://raap.info-ab.uclm.es/diab-03-02-35.pdf`.

  Este trabajo analiza las diferentes posibilidades de configurar la tabla de arbitraje, y se indica cómo unificar los dos tipos de requisitos posibles (ancho de banda y latencia). Se concluye que para proporcionar garantía de ancho de banda y/o latencia a una conexión, hay que dedicarle al canal virtual que ésta utiliza una secuencia de entradas separadas entre sí por una cierta distancia máxima.

  En este informe también se plantea el modelo formal para el tratamiento de la tabla de arbitraje de InfiniBand. Mediante una serie de definiciones y teoremas se demuestra que la propuesta realizada es capaz de proporcionar QoS a las aplicaciones que lo demanden.

- *Providing QoS in InfiniBand for Regular and Irregular Topologies.* Francisco J. Alfaro, José L. Sánchez, Luis Orozco, José Duato. IEEE Canadian Conference on Electrical & Computer Engineering (CCECE'03). Montreal, Canada. Mayo de 2003. Proceedings publicados por IEEE Computer Society con I.S.B.N. 0-7803-7514-9. `http://raap.info-ab.uclm.es/pubs/pub77.pdf`.

  En este trabajo se presenta el nuevo planteamiento que se hizo en el technical report anterior, y se evalúa la influencia de diversas topologías. Se concluye que no hay influencia de la topología utilizada. Como era de prever, lo importante para conseguir mejor utilización de los recursos es que el algoritmo de encaminamiento utilizado sea capaz de balancear la carga entre los distintos enlaces. De esta forma podrían llegar a establecerse más conexiones. En cualquier caso, las aplicaciones consiguen siempre la QoS que habían solicitado.

- *Una Nueva Propuesta para Proporcionar QoS en InfiniBand.* Francisco J. Alfaro Cortés, José L. Sánchez García, José Duato Marín. XIV Jornadas de Paralelismo. Leganés (Madrid). Septiembre 2003. `http://raap.info-ab.uclm.es/pubs/pub75.pdf`.

  En este trabajo se presentan los primeros resultados usando el nuevo modelo de tratamiento global de las garantías de ancho de banda y/o latencia. Se presenta el modelo global y los resultados usando simulación.

- *A New Proposal to Fill in the InfiniBand Arbitration Tables.* Francisco J. Alfaro, José L. Sánchez, José Duato. International Conference on Parallel Computing

(ICPP'03). Octubre 2003. Kaohsiung, Taiwan. `http://raap.info-ab.uclm.es/pubs/pub78.pdf`.

Este artículo comienza planteando el nuevo tratamiento para conseguir garantizar ancho de banda y/o latencia y se presenta el nuevo algoritmo para ubicar una petición en la tabla de arbitraje. En la evaluación se presentan una serie de resultados que muestran que incluso en situaciones de carga elevada todas las aplicaciones consiguen lo que tenían solicitado. Además, se argumenta que, con el nuevo planteamiento, un mal comportamiento de alguna de las fuentes sólo afectará a las conexiones que compartan canal virtual, pero el resto seguirá obteniendo lo que tenía garantizado. Esto es una novedad con respecto a las anteriores propuestas, donde un mal comportamiento de una conexión podía afectar al resto de aplicaciones.

Los artículos enviados a revistas especializadas, y que en los momentos de presentar este trabajo están pendientes de aceptación, son:

- *QoS in InfiniBand*. Enviado para revisión a la revista IEEE Transactions on Parallel and Distributed Systems (TPDS).

  Este artículo plantea los mecanismos que proporciona InfiniBand para proporcionar QoS. Se estudia como se puede proporcionar garantía de ancho de banda y garantía de latencia máxima. Se presenta la primera propuesta para configurar la tabla de arbitraje de forma que las aplicaciones obtengan la QoS requerida. En la parte de evaluación se muestran resultados comprobando que la metodología propuesta es capaz de proporcionar garantía de ancho de banda y/o latencia.

- *A New Strategic to Manage the InfiniBand Arbitration Tables Providing QoS*. Enviado para revisión a la revista IEEE Transactions on Parallel and Distributed Systems (TPDS).

  En este artículo se presenta la nueva estrategia para tratar la garantía de ancho de banda y de latencia de forma conjunta. Se estudian de manera exhaustiva las posibles formas de configurar la tabla de arbitraje. Con la opción seleccionada se hace una evaluación de prestaciones de cara a comprobar que las propuestas realizadas son capaces de proporcionar la QoS demandada por las aplicaciones.

- *A Formal Model to Manage the InfiniBand Arbitration Tables Providing QoS*. Enviado para revisión a la revista IEEE Transactions on Parallel and Distributed Systems (TPDS).

  En este artículo se presenta el modelo formal para el tratamiento de la tabla de arbitraje. Se hacen las definiciones oportunas, y se enuncian los teoremas necesarios para probar que cualquier secuencia de peticiones, que requiera menor o igual número de entradas que las disponibles, se puede ubicar en la tabla con el algoritmo de reserva propuesto.

7.3. Financiación disfrutada

Hay que señalar que el trabajo aquí presentado se ha desarrollado en el marco de los siguientes proyectos de investigación:

PBC-02-008 *"Diseño de estrategias avanzadas para encaminamiento, calidad de servicio y reconfiguración en redes InfiniBand"*. Proyecto concedido por la Junta de Comunidades de Castilla-La Mancha y coordinado entre la Universidad Politécnica de Valencia y la Universidad de Castilla-La Mancha. Su periodo de vigencia está comprendido entre enero de 2002 y diciembre de 2004, siendo el investigador principal Dr. D. José Luis Sánchez García.

TIC2000-1151-C07-02 *"Mejora de las prestaciones y servicios ofrecidos por las redes de computadores personales. Desarrollo de aplicaciones multimedia distribuidas"*. Proyecto concedido por la Comisión Interministerial de Ciencia y Tecnología (C.I.C.Y.T.), y coordinado entre la Universidad de Castilla-La Mancha, la Universidad Politécnica de Valencia, la Universidad de Murcia, la Universidad de Valencia y la Universidad Jaime I. Su periodo de vigencia es desde enero de 2001 hasta diciembre de 2003, siendo el investigador principal del subproyecto Dr. D. Antonio J. Garrido del Solo.

TIC97-0897-C04-02 *"Desarrollo de una Red de Estaciones de Trabajo de Altas Prestaciones y Bajo Coste"*. Proyecto concedido por la Comisión Interministerial de Ciencia y Tecnología (C.I.C.Y.T.), y coordinado entre la Universidad de Castilla-La Mancha, la Universidad Politécnica de Valencia y la Universidad de Murcia. Su periodo de vigencia está comprendido entre junio de 1997 y diciembre de 2000, siendo el investigador principal del subproyecto Dr. D. Francisco José Quiles Flor.

Asimismo, se ha contado con diversas ayudas de la Universidad de Castilla-La Mancha.

7.4. Trabajo futuro

Una vez alcanzados los objetivos de este trabajo, quedan todavía abiertos mucho temas directamente relacionados con él. En concreto, como trabajo futuro se plantean las siguientes actividades:

- Probar las estrategias aquí planteadas sobre una subred real de InfiniBand. Esta tarea siempre es conveniente para cotejar los resultados obtenidos en las simulaciones. De esta manera se podrá comprobar que los supuestos realizados, así como

las propuestas desarrolladas, son totalmente correctos y aplicables en entornos reales.

- Estudiar si es posible mejorar los algoritmos propuestos en los Capítulos 4 y 5 para reducir su orden de complejidad y evaluar sus tiempos de actuación. Por ejemplo, quizás podría ser interesante complicar el algoritmo de reserva para que haga un barrido completo de todos los conjuntos disponibles de un determinado nivel, en vez de tener que reordenar los conjuntos cuando son liberados.

- En este trabajo se han obviado las prestaciones del tráfico best-effort. De esta forma, sería interesante cuantificar cómo afectan las propuestas aquí realizadas a las prestaciones recibidas por el tráfico best-effort.

- En este trabajo se ha supuesto un encaminamiento similar al up*/down*. Sin embargo, es bien conocido que este algoritmo de encaminamiento no es el que mejores prestaciones ofrece. Así pues, podría probarse en qué factor puede mejorarse la utilización de la red con otros algoritmos de encaminamiento compatibles con InfiniBand, y si eso tiene influencia con la QoS recibida por las aplicaciones.

- Estudiar cómo configurar los mecanismos que proporciona InfiniBand para conseguir proporcionar QoS a las aplicaciones que necesiten otro tipo de garantía, como por ejemplo garantía de jitter, de ruta mínima, etc.

- Estudiar las estrategias propuestas en este trabajo en entornos mayores formados por la unión de varias subredes. Para ello hay que considerar otro tipo de entorno, con distancias mayores, donde se usarán las direcciones globales de InfiniBand (GID).

- Comprobar el efecto de una reconfiguración de la red de interconexión sobre la QoS que reciben las aplicaciones. Para ello, los agentes locales, o el Subnet Manager, cuando sea posible, debe rehacer las conexiones caídas por caminos alternativos.

- Probar estrategias distintas, donde no se trate de proporcionar garantía absoluta, pero se ofrezca mayor flexibilidad a las aplicaciones. Se trataría de implementar en el entorno que ofrece InfiniBand un esquema similar al de los Servicios Diferenciados [BBC98].

Bibliografía

[ACP95] T. E. Anderson, D. E. Culler, D. A. Patterson. *A case for NOW (Networks of Workstations).* IEEE Micro, Vol. 15, No. 1, pp. 54–64, February 1995.

[ASD02] Francisco J. Alfaro, José L. Sánchez, José Duato. *A Strategy to Manage Time Sensitive Traffic in InfiniBand.* In Proceedings of Workshop on Communication Architecture for Clusters (CAC'02), April 2002. Held in conjunction with IPDPS'02, Fort Lauderdale, Florida.

[ASO02] Francisco J. Alfaro, José L. Sánchez, Luis Orozco, José Duato. *Performance Evaluation of VBR Traffic in InfiniBand.* In Proceedings of IEEE Canadian Conference on Electrical & Computer Engineering (CCECE'02), pp. 1532 – 1537, May 2002.

[Avi] Avici Terabit Switch Router. Avici Systems. `http://www.avici.com`.

[BBC98] S. Blake, D. Back, M. Carlson, E. Davies, Z. Wang, W. Weiss. *An Architecture for Differentiated Services.* Internet Request for Comment RFC 2475, Internet Engineering Task Force, December 1998.

[BCF95] N.J. Boden, D. Cohen, R.E. Felderman. *Myrinet – A Gigabit per Second Local Area Network.* IEEE Micro, pp. 29–36, February 1995.

[BCQ03] A. Bermúdez, R. Casado, F. J. Quiles, T. M. Pinkston, J. Duato. *Modeling InfiniBand with OPNET.* In 2nd Annual Workshop on Novel Uses of System Area Networks (SAN-2), February 2003.

[BCS94] R. Braden, D. Clark, S. Shenker. *Integrated Services in the Internet Architecture: an Overview.* Internet Request for Comment RFC 1633, Internet Engineering Task Force, June 1994.

[Ber98] Yoram Bernet. *A Framework for Differentiated Services.* Internet draft 2275, Internet Engineering Task Force, May 1998.

[Bha02] Ajay V. Bhatt. *Creating a Third Generation I/O Interconnect.* White paper, Technology and Research Labs, Intel Corporation, 2002. `http://developer.intel.com/technology/pciexpress/index.htm`.

[Bla00] Uyless Black. *QoS in Wide Area Networks.* Prentice Hall Series in Advanced Communications Technologies. Prentice Hall, 2000.

[BZB97] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin. *Resource ReSerVation Protocol (RVP) – version 1 functional specification.* Internet Request for Comment RFC 2205, Internet Engineering Task Force, September 1997.

[Cad] Cadena100. Página web de la Cadena 100, en `http://www.cadena100.es`.

[Cam02] María Blanca Caminero. *Diseño de un Encaminador Orientado a Tráfico Multimedia en Entornos LAN.* PhD thesis, Departamento de Informática de la Universidad de Castilla-La Mancha, July 2002.

[Cas01] Rafael Casado. *Mecanismo de reconfiguración eficiente en redes de interconexión con topología irregular y encaminamiento up*/down*.* PhD thesis, Departamento de Informática de la Universidad de Castilla - La Mancha, 2001.

[CJ97] A. Chien, Kim J.H. *Approaches to Quality of Service in High Performance Networks.* In Lectures Notes in Computer Science, editor, *Workshop on Parallel Computer Routing and Communications*, pp. 1–19. Springer-Verlag, June 1997.

[CL95] T. M. Chen, S. S. Liu. *ATM Switching Systems.* Artech House Publishers, 1995.

[Cue98] Pedro Cuenca. *Codificación y transmisión robusta de señales de vídeo MPEG-2 de caudal variable sobre redes de transmisión asíncrona ATM.* PhD thesis, Universidad Politécnica de Valencia, 1998.

[Dal96] I. Dalgic. *Performance of Ethernet and ATM Networks Carrying Video Traffic Based on Accurate Characteristics of Video Sources.* PhD thesis, Stanford University, August 1996.

[Fli01] José Flich. *Mejora de las prestaciones de las redes de estaciones de trabajo con encaminamiento fuente.* PhD thesis, Departamento de Informática de Sistemas y Computación de la Universidad Politécnica de Valencia, 2001.

[FLS02] J. Flich, P. López, J. C. Sancho, A. Robles, J. Duato. *Improving InfiniBand Routing through Multiple Virtual Networks.* Lecture Notes in Computer Science, Vol. 2327, pp. 49–58, 2002.

[For] ATM Forum. Página web del ATM Forum, en `http://www.atmforum.com/`.

[Fut01] William T. Futral. *InfiniBand Architecture: Development and Deployment–A Strategic Guide to Server I/O Solutions.* Intel Press, 2001.

[Gar93] M. W. Garret. *Contributions Toward Real-Time Services on Packet-Switched Networks.* PhD thesis, CU/CTR/TR 340-93-20, Universidad de Columbia, Mayo 1993.

[GBH97] R. Guerin, S. Blake, S. Herzog. *Aggregating RSVP-based QoS Requests.* Internet draft, Internet Engineering Task Force, November 1997.

[GG99] N. Giroux, S. Ganti. *Quality of Service in ATM Networks.* Prentice Hall, 1999.

[Hal01] Fred Halsall. *Multimedia Communications: Applications, Networks, Protocols and Standard.* Addison-Wesley, 2001.

[Hor95] R.W. Horst. *TNet: A Reliable System Area Network.* IEEE Micro, pp. 36–44, February 1995.

[IBA99] *InfiniBand^{TM} Trade Association*, 1999. `http://infinibandta.com`.

[IBM01] IBM Corporation. *IBM InfiniBand product advance summary datasheet.* `http://www.chips.ibm.com/products/infiniband`, August 2001.

[IF] Inter-Fone. Página web de Inter-Fone, en `http://www.inter-fone.com/`.

[Inf00] InfiniBand Trade Association. *InfiniBand Architecture Specification Volume 1. Release 1.0*, October 2000.

[Jai91] R. Jain. *The art of computer system performance analysis: techniques for experimental design, measurement, simulation and modeling.* John Wiley and Sons, Inc., 1991.

[Kar96] G. Karlsson. *Asynchronous transfer of video.* IEEE communication Magazine, Vol. 24, No. 8, pp. 118–126, August 1996.

[KK79] P. Kermani, L. Kleinrock. *Virtual cut-through: A new computer communication switching technique.* Computer Networks, Vol. 3, pp. 267–286, 1979.

[KLC98] J. Kim, Z. Liu, A. Chien. *Compressionless routing: a framework for adaptive and fault-tolerant routing.* IEEE Transactions on Parallel and Distributed Systems, Vol. 8, No. 3, pp. 229–244, March 1998.

[KLS00] F. Kuhns, D. Levine, D.C. Schmidt, C. O'Ryan. *Supporting High-Performance I/O in QoS-enabled ORB middleware.* Cluster Computing, Vol. 3, No. 3, pp. 151–173, 2000.

[Knu73] D. E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching.* Addison-Wesley, 1973.

[KR00] J. F. Kurose, K. W. Ross. *Computer Networking: A top-down approach featuring the Internet.* Addison-Wesley, 2000.

[Med] Microsoft Windows Media. Página web de Microsoft Windows Media, en `http://www.microsoft.com/windows/windowsmedia/`.

[Mel01a] *Comparative I/O Positioning. InfiniBand Compared with PCI-X, Fiber Channel, Gigabit Ethernet, Storage over IP, HyperTransport, and RapidIO.* White Paper WP062001102, Mellanox Technologies Inc., 2001. `http://www.mellanox.com/products/whitepaper.html`.

[Mel01b] *Understanding PCI Bus, 3GIO and InfiniBand Architecture.* White Paper WP120501100, Mellanox Technologies Inc., 2001. `http://www.mellanox.com/products/whitepaper.html`.

[Net] Microsoft NetMeeting. Página web de Microsoft NetMeeting, en `http://www.microsoft.com/windows/netmeeting/`.

[NGM97] L.M. Ni, Y. Gui, S. Moore. *Performance evaluation of switch-based wormhole networks.* IEEE Transactions on Parallel and Distributed Systems, Vol. 8, No. 5, pp. 462–474, May 1997.

[Pat01] David Patterson. *High Performance Mass Storage and Parallel I/O*, chapter Foreword. IEEE Press and Wiley Press, 2001.

[PE00] E. Pendery, J. Eunice. InfiniBand Architecture: Bridge over troubled waters, 2000. `http://www.infinibandta.org/press/`.

[Pel00] Joe Pelissier. *Providing Quality of Service over Infiniband Architecture Fabrics.* In Proceedings of the 8th Symposium on Hot Interconnects, August 2000.

[Pfi01] G.F. Pfister. *High Performance Mass Storage and Parallel I/O*, chapter 42: An Introduction to the InfiniBand Architecture, pp. 617–632. IEEE Press and Wiley Press, 2001.

[Pry91] M. Prycker. *Asynchronous Transfer Mode: solution for broadband ISDN.* Ellis Horwood Limited, 1991.

[Rea] RealNetworks. Página web de RealNetworks. `http://www.realnetworks.com`.

[San02] José Carlos Sancho. *Contribución al diseño de algoritmos de encaminamiento en redes de estaciones de trabajo.* PhD thesis, Departamento de Informática de Sistemas y Computación de la Universidad Politécnica de Valencia, 2002.

[SBB90] 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.* Technical Report 59, Systems Research Center of Digital Equipment Corporation, 1990.

[Sch00] M. Schroeder. Página web "SRC Research: Networks", en `http://www.research.compaq.com/SRC/org/networks.html`, 2000.

[SD97] F. Silla, J. Duato. *On the use of virtual channels in networks of workstations with irregular topology.* In Proceedings of the 1997 Parallel Computing. Routing and Communication Workshop, June 1997.

[Ser] Cadena Ser. Página web de la Cadena Ser, en `http://www.cadenaser.es`.

[Sha02] Tom Shanley. *InfiniBand Network Architecture.* Addison Wesley Professional, 2002.

[SPG97] S. Shenker, C. Partridge, R. Guerin. RFC 2212: Specification of Guaranteed Quality of Service, September 1997. Status: proposed standard.

[SRF01] J. C. Sancho, A. Robles, J. Flich, P. López, J. Duato. *Effective Methodology for Deadlock-Free Minimal Routing in InfiniBand Networks.* In Proceedings of 2001 International Conference on Parallel Processing (ICPP'01), September 2001.

[SW97] R. Steinmetz, L.C. Wolf. *Quality of Service: Where are we?* In Proceedings of IFIP Fifth International Workshop on Quality of Service (IWQoS'97), May 1997.

[Tan88] Andrew S. Tanenbaum. *Computer networks.* Prentice Hall, 2ª Edición, 1988.

[Tow93] D. Towsley. *Providing Quality of Service in Packet Switched Networks.* Lecture Notes in Computer Science, Vol. 729, pp. 560–588, 1993.

[WGA01] J. Wu, A. Gulati, B. Abali, D.K. Panda. *Design of an InfiniBand Emulator over Myrinet: Challenges, Implementation, and Performance Evaluation.* Technical Report OSU-CISRC-2/01-TR03, Dept. of Computer and Information Science, The Ohio State University, 2001.

[Wro97a] J. Wroclawski. RFC 2210: The Use of RSVP with IETF Integrated Services, September 1997. Status: proposed standard.

[Wro97b] J. Wroclawski. RFC 2211: Specification of the Controlled-Load Network Element Service, September 1997. Status: proposed standard.

[XN99] X. Xiao, L.M. Ni. *Internet QoS: A Big Picture.* IEEE Network Magazine, pp. 8–18, March 1999.