Escuela Técnica Superior de Ingenieria

Departamento de Arquitectura de Computadores y Sistemas Operativos

CONTROL DE CONGESTIÓN ADAPTATIVO EN REDES INFINIBAND

Memoria del trabajo experimental realizado por

Diego Fernando Lugones

dentro del programa de Doctorado en Informática (Arquitectura de Computadores y Procesamiento Paralelo)

Barcelona (España), Julio de 2007

Control de Congestión Adaptativo en Redes InfiniBand

El Dr. Daniel Franco Puntes, profesor titular del Departamento de Arquitectura de Computadores y Sistemas Operativos de la Universidad Autónoma de Barcelona.

CERTIFICA: que la presente memoria “Control de congestión Adaptativo en Redes InfiniBand”, ha sido realizada bajo su dirección por Diego Fernando Lugones constituyendo el Trabajo experimental del programa de doctorado en Informática.

Bellaterra, Julio de 2006

Dr. Daniel Franco Puntes

A mis viejos y a mi hermano

A mi grupo

A la flaca … gracias

Contenido

Índice de Figuras

Capitulo 1 Introducción

1.1 Contexto

La evolución en el campo de las redes de interconexión ha sido constante en los últimos años. Los avances tecnológicos han permitido una mejora importante en la velocidad de transmisión, incrementando significativamente el ancho de banda de los enlaces. Dichos avances también han tenido gran impacto en la integración de puertos en los conmutadores aumentando la cantidad de conexiones y permitiendo obtener topologías mas complejas y flexibles.

En virtud de esta evolución, el interés de los investigadores en el estudio de técnicas de encaminamiento y control de flujo eficientes, que en conjunto permitan obtener una mejora significativa en las prestaciones de la red, ha crecido notoriamente.

La reciente aparición de redes de interconexión comerciales como InfiniBand, Myrinet, Quadrics,…, etc. con alta velocidad de transmisión de datos, permiten construir las redes de interconexión para sistemas de cómputo de altas prestaciones. Estas redes han tenido impacto no sólo en estos sistemas de computo, donde magnitudes adecuadas de velocidad de comunicación y tiempo de viaje de mensajes son de extrema importancia, sino también, en otros con menos requerimientos de performance como redes de sistema (SAN) y Clusters de ordenadores. Los nuevos estándares desarrollados y las implementaciones comerciales deben su existencia a la creciente demanda de aplicaciones con grandes requisitos de cómputo. Dentro del las aplicaciones científico-técnicas, las áreas que tradicionalmente han requerido computo de altas prestaciones son, entre otras, la simulación aero-espacial, la exploración y explotación geológica, el diseño de moléculas, la simulación genética, la dinámica y turbulencia de fluidos, el diseño de medicamentos, la seguridad en reactores nucleares, etcétera [56]. Los modelos utilizados con el fin de describir el comportamiento de sistemas en el campo de la física, astronomía, química, aeronáutica,… están basados en la resolución de sistemas de ecuaciones diferenciales lineales (o no lineales) con coeficientes constantes (o variables), por tanto la simulación de cualquiera de estos sistemas, suele requerir grandes prestaciones de computo y comunicaciones a medida que la exactitud en los resultados y la resolución del sistema aumentan.
Hoy en día las necesidades de computo y comunicaciones se han ampliado desde las aplicaciones puramente científicas, como las mencionadas anteriormente, a otras disciplinas como el procesamiento de imágenes, la gestión de grandes bases de datos en sistemas que atienden en simultaneo múltiples peticiones de usuarios, como es el caso de las aplicaciones multimedia. Este tipo de aplicaciones demandan una serie de requisitos nuevos: un gran volumen de comunicación de datos, una respuesta en tiempo

real y cierto grado de interactividad con el usuario, son algunas de las necesidades más importantes a cubrir por el computador de altas prestaciones.
El inusitado crecimiento de computadores paralelos basados en redes de interconexión de altas prestaciones, ha aumentado el interés y esfuerzo de la comunidad investigadora tanto en el desarrollo de técnicas y conceptos que permitan utilizar estas redes al máximo de sus posibilidades, como en la adaptación de otras técnicas y soluciones ya existentes.

El principal problema en el diseño de las redes de interconexión radica en manejo inadecuado de la congestión de mensajes en transito, que implica que la red esta funcionando cerca del la máxima utilización *(throughput)* que puede alcanzar, puesto que utiliza gran parte de los recursos cerca de su capacidad. Dicha congestión aparece debido al uso compartido de los recursos de la red de interconexión (enlaces y conmutadores) y si esta situación no se controla eficientemente, es posible alcanzar la saturación de dichos recursos. Cuando la red no es capaz de manejar el volumen de comunicaciones que recibe en un momento dado, los mensajes en transito deberán competir por los recursos. Esta situación deriva en un aumento en el tiempo de viaje de los mensajes (latencia) y se propaga rápidamente a toda la red teniendo como efecto principal a un deterioro global en la performance del sistema.

Las consecuencias derivadas de la congestión son aun mas graves en redes que no permiten el descarte de paquetes, como es el caso de las redes que conforman la mayoría de computadores paralelos de altas prestaciones.

La solución tradicional al fenómeno de la congestión, radica en diseñar la red utilizando una cantidad de recursos mayor a la estrictamente necesaria (Sobredimensionar la red) de manera que no exista la necesidad de competir por dichos recursos y evitar la retención de paquetes. Sin embargo esta practica ha quedado obsoleta debido, tanto al aumento en el coste de los componentes de red con respecto al de los procesadores empleados en los computadores paralelos actuales, como al elevado consumo de potencia de dichos componentes, debido fundamentalmente al aumento de la velocidad de los enlaces.

Es por esta razón que el diseño de redes de interconexión sobredimensionadas no es una solución viable en la construcción de sistemas de cómputo reales. Las tecnologías de red actuales permiten realizar interconexión de varios terminales a cada conmutador, disminuyendo el número de componentes de red y el control de la congestión ha vuelto a ser una cuestión clave que requiere nuevas soluciones.

1.2 Antecedentes

La capacidad de gestionar un alto número de mensajes sin que se produzca un gran aumento de la latencia es fundamental en las redes de interconexión de altas prestaciones, mas aun cuando estos soportan aplicaciones en las que la relación de cómputo frente a comunicación es pequeña (granularidad fina), y que por tanto hacen que haya un gran tráfico de mensajes entre los nodos de la red. Por este motivo, el control de congestión ha sido objeto de estudio durante los últimos años.

Las técnicas que intentan manejar y solucionar los problemas asociados a la congestión, pueden dividirse en dos grupos principales [2]:

- **Técnicas preventivas.** Requieren un conocimiento previo de los recursos, reservándolos antes de comenzar la transmisión de datos, para garantizar la ausencia de congestión. Este conocimiento no siempre esta disponible y además, la reserva de recursos puede incurrir en una elevada sobrecarga de paquetes en la red (overhead). En general estas técnicas se basan en :
  - Abrir conexiones
  - Incrementar recursos (sobredimensionar)
  - Reservar ancho de banda
  - Planificar y determinar la máxima inyección de mensajes.

- **Técnicas reactivas o de recuperación.** Estas técnicas se basan en realizar una monitorización del tráfico que circula por la red, o de los recursos que la componen, con el motivo de detectar la congestión, notificar su existencia y llevar a cabo algún mecanismo para controlarla. En general, pueden descomponerse en tres fases.
  - Monitorear la red y **detectar** congestión
  - **Notificar** la existencia de congestión
  - Ejecutar **acciones** correctivas

Las técnicas reactivas, representan un tópico en el área de las redes de interconexión en computadores paralelos de alto rendimiento, que despierta un gran interés en los investigadores y ha permitido el desarrollo de una gran cantidad de propuestas. Dentro de las fases de detección, notificación y ejecución de acciones, pueden encontrarse varias alternativas, que pueden clasificarse de la siguiente manera:

- **Monitorización y detección**. En general miden cierta características dinámicas de la red cuyo estado puede determinar la existencia de congestión, como por ejemplo:
  - Tiempo de bloqueo de mensajes (Latencia).
  - Midiendo recursos ocupados (Canales o *Buffers*).
    - A nivel global (Mayor precisión a costa de overhead y BW).

- A nivel local (Mas limitado pero mejor relación coste/prestac.).

- Disminución de velocidad de flujo de mensajes (Backpressure).

- **Notificación.** Permiten dar a conocer la existencia de congestión al resto (o parte) de la red. Podemos dividir las técnicas según el nivel de notificación de la siguiente manera:
  - **Local** (Detecta la congestión, pero no notifica al resto sino que intentan solucionar el problema localmente).
  - **Vecinos** (Solo los vecinos cercanos son notificados de la existencia de congestión).
  - **Nodos que envían** (Notifica a los nodos cuyo flujo de paquetes contribuye a la congestión).
  - **Todos los nodos**. (Todos los nodos son concientes del estado de la red, en este caso la sobrecarga de mensajes necesarios para transportar la información entre todos los nodos es un factor limitante).

- **Decisiones y acciones correctivas.** Proveen los mecanismos necesarios que permiten eliminar la congestión y evitar la degradación de performance en la red. Algunas posibilidades son:
  - **Message throttling.** Utilizado a nivel de nodo fuente y actúa deteniendo o disminuyendo la cantidad de mensajes inyectados.

    - Durante un tiempo determinado.
    - Hasta que caiga la cantidad de trafico bajo cierto nivel.
  - Gestionar y optimizar el uso de **Buffers.** Utilizado a nivel de conmutador y actúa reordenando los paquetes de diferentes flujos para evitar interferencias.
  - **Encaminamiento adaptativo**. Utilizado a nivel de conexión (punto a punto) y actúa redistribuyendo la carga a través de múltiples trayectorias alternativas entre el mismo par fuente-destino.

Los beneficios y desventajas de estas técnicas serán analizados con más profundidad en capítulos posteriores, aunque podemos destacar en forma general, los principales inconvenientes que se desprenden de su uso.

El factor limitante es función de la técnica de control utilizada y puede incluir algunas de las siguientes *desventajas* [58]:

- *Tiempo de respuesta elevado*. El tiempo en que los mecanismos de control de congestión reaccionan y actúan, debe estar convenientemente acotado para evitar respuestas tardías.

- *Desbalance global de carga.* Algunos mecanismos pueden proveer una respuesta adecuada a nivel local, pero conllevan un desbalance de carga en la red que conduce a una degradación global de las prestaciones.
- *Penalización.* Ciertas técnicas pueden reaccionar correctamente ante situaciones de congestión, pero generan sobrecarga o nuevos fenómenos indeseables (ej. *deadlock, livelock, starvation*) con cargas normales de trafico.
- *Baja eficiencia*. Algunas técnicas no consiguen eliminar el problema completamente o solo lo hacen en algunos casos.
- *No robusto.* El éxito de algunos mecanismos depende en gran parte de las condiciones de tráfico presente en la red, la topología de la red, el tamaño de los mensajes, etc.
- *No escalable*. Algunas técnicas solo son aplicables en la práctica a topologías simples y/o pequeñas.

1.2.1 Balanceo distribuido del encaminamiento.

El punto de partida del trabajo de investigación aquí presentado, consiste en un mecanismo de balanceo del encaminamiento que intenta uniformizar la carga en todos los enlaces de la red de interconexión. Este mecanismo se conocido como: Balanceo Distribuido del Encaminamiento ("*Distributed Routing Balancing*", DRB por sus siglas en inglés) [25] [26] y se basa en la distribución uniforme de la carga en la red mediante la expansión de caminos. Esta expansión es dinámica y está controlada por el nivel de latencia existente en la red. El método establece nuevos caminos alternativos simultáneos entre cada par fuente y destino con objeto de mantener una latencia baja de los mensajes. DRB define cómo crear los caminos alternativos para expandir los caminos simples originales y cuándo y cómo usarlos dependiendo del nivel de carga de tráfico en la red de interconexión. Es importante destacar que el mecanismo produce un efecto de balance colectivo (a nivel global), pues esta expansión se produce para todos los pares fuente-destino de la aplicación que interaccionan entre sí.

Conceptualmente, este método mide el estado de la carga en todas las conexiones entre pares fuente destino, al detectar congestión se notifica al/los nodo/s que inyectan mensajes para que configuren nuevas trayectorias posibles y redistribuyan el trafico según su estado de carga. Es importante destacar el concepto en el que el algoritmo basa su funcionamiento, porque puede ser implementado de diversas maneras. Es decir: la detección puede realizarse mediante la medición de la latencia acumulada a través del enlace, o en función de la ocupación de los buffers en los puertos de los conmutadores, etcétera... Mientras que la redistribución de tráfico puede llevarse a cabo de manera aleatoria, o en función de la ocupación de los enlaces, o de la velocidad de estos en redes no homogéneas.

Esta versatilidad en la implementación del algoritmo, junto a su buen comportamiento hace que su aplicación a tecnologías de interconexión actuales sea un tema de investigación interesante, tanto para la mejora y evaluación de esta técnica, como para desarrollo de nuevas propuestas sobre las tecnologías utilizadas.

1.3 Motivación

En virtud de lo que hemos mencionado dentro del contexto en el cual desarrollamos este trabajo de investigación, sobre la importancia de las aplicaciones de computo de altas prestaciones, que permiten satisfacer las necesidades de todo tipo de ámbitos y campos de la ciencia. Encontramos que los computadores paralelos posibilitan el desarrollo y la plataforma adecuada para estas aplicaciones, pues son aptos para realizar las tareas de cómputo intensivo necesarias en estos casos. Por otra parte, la red de interconexión que permite el intercambio de mensajes para la colaboración de tareas, es un componente principal dentro del computador paralelo. Los problemas que afectan a las redes de interconexión y sus causantes, se han detallado en el punto anterior y se ha mencionado que mediante el uso de mecanismos adecuados es posible aumentar de manera notable las prestaciones alcanzadas por los computadores paralelos. El encaminamiento es una de estas técnicas y permite mejorar la utilización de la red, adaptándose a sus cambios y aprovechando sus recursos [57].

En la última década han aparecido una importante cantidad de especificaciones que acompañadas del avance tecnológico adecuado, pretenden desarrollar y estandarizar redes de comunicaciones con las operaciones y características mencionadas en los párrafos anteriores (Conexiones punto a punto, baja latencia, elevado ancho de banda, etc.…). Puntualmente, la especificación InfiniBand[1] [15] es una nueva y poderosa arquitectura diseñada, no sólo para cubrir con las demandas de performance asociadas con el movimiento de datos en los dispositivos entrada-salida, sino también para conformar los cluster de cómputo de alta performance *(High Performance Computing HPC)*, debido al elevado ancho de banda y la baja latencia de transporte que ofrece.

Los clusters InfiniBand de gran escala están ganado gran popularidad según lo reflejan los rankings de supercomputadores en el *top500* [49], tal como se muestra en las figuras Fig. 1-1, Fig. 1-2 y Fig. 1-3, donde puede verse como ha evolucionado InfiniBand, en el último año hasta alcanzar la tercera posición, entre los sistemas de interconexión usados en los supercomputadores más potentes del mundo. La figura muestra solamente los treinta primeros puestos, y se observa que es uno de los estándares mas utilizados. Al mismo tiempo, las topologías directas (mallas, toros, hipercubos…) y la topología fat-tree [58]se han convertido en las más utilizadas en la interconexión para estos clusters, debido a que permiten múltiples trayectorias disponibles entre un mismo par de nodos. No obstante, incluso en estas topologías, pueden ocurrir situaciones de congestión, que

[1] La versión 1.2 de la especificación es del año 2004

dependen principalmente de la configuración de trayectorias entre nodos y del patrón de comunicación de la aplicación. Para empeorar aun más la situación, la naturaleza determinista del encaminamiento en algunas tecnologías, limita a las aplicaciones del uso eficaz y transparente de trayectorias múltiples que permiten evitar situaciones de congestión.

Interconnect family	Noviembre 2006	Junio 2006	Junio 2005	Noviembre 2005
Gigabit Ethernet	42.60 %	51.20 %	50.00 %	42.80 %
Myrinet	15.80 %	17.40 %	20.20 %	27.80 %
Infiniband	15.60 %	7.20 %	5.40 %	3.20 %
Quadrics	2.80 %	2.80 %	2.80 %	2.60 %

Fig. 1-1 Evolución de IBA en el *top500*

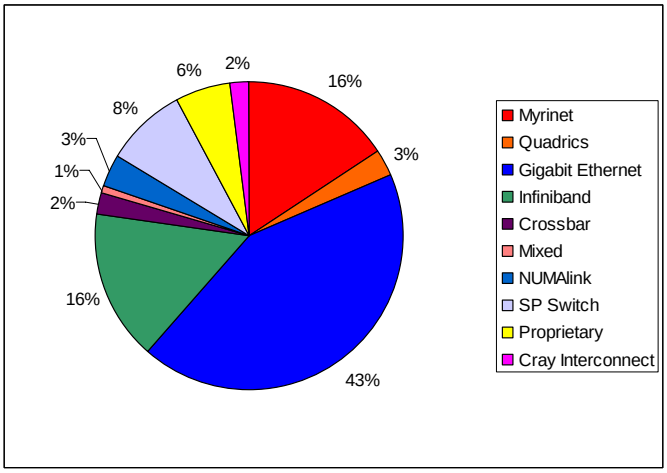


Fig. 1-2 Uso de InfiniBand como red de interconexión

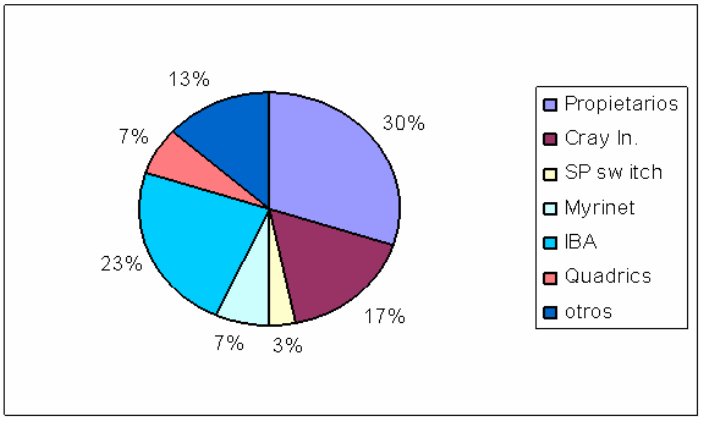


Fig. 1-3 Posición en los treinta primeros puestos

Sin embargo, InfiniBand especifica una manera de establecer varias trayectorias [55] posibles e independientes (mediante la asignación de direcciones y mascaras) que son utilizadas para enviar paquetes entre los diferentes pares de nodos fuente-destino.

Un uso inteligente de estas capacidades otorga a la red de interconexión una utilización eficiente, debido a que permiten conformar un mecanismo de control de congestión que

monitoriza y gestiona los recursos, a fin de evitar la aparición de puntos calientes *(hot-spot)* [7] [20] que deterioran la performance del sistema.

Lo expuesto en los párrafos anteriores, justifica plenamente el estudio propuesto en este trabajo de tesis: La adaptación de un algoritmo de encaminamiento vigente, factible y realista, citado recientemente, en artículos y contribuciones, por otros autores en estudios similares [24] [36], sobre un estándar emergente cuya utilización esta ganando terreno velozmente en el campo de las redes de interconexión para computadores de alta performance, pero que carece de un control de congestión adecuado[2].

1.4 Objetivos

Como se ha señalado previamente, nuestro trabajo se centra en el estudio y la aplicación de un algoritmo de encaminamiento adaptativo sobre la arquitectura especificada por el estándar InfiniBand, y profundiza en la definición de las características deseables de dicho algoritmo con motivo de incorporar sus capacidades en dicha especificación, manteniendo la compatibilidad. Este estudio centra su contexto en el marco de los computadores de altas prestaciones y, concretamente, sobre las redes de interconexión como componente común a todos ellos. Nos centramos en el estudio de las redes de interconexión porque, como hemos mencionado, son un componente fundamental para conseguir los requerimientos que el tipo de aplicaciones actuales demanda sobre los sistemas de altas prestaciones.

En virtud a lo expuesto, podemos desglosar el objetivo principal de este trabajo de tesis en una serie de objetivos parciales, que dan a conocer la metodología empleada para su cumplimiento:

Realizar un estudio general de las redes de interconexión, las características dinámicas, el espacio de diseño y su marco de aplicación.

Realizar un análisis de las problemáticas surgidas en el funcionamiento de las redes de interconexión, los fenómenos que las originan y favorecen su evolución y los efectos provocados por su presencia.

Realizar una evaluación de las soluciones existentes para el control de congestión y la evitación de zonas calientes *(hotspots)*, analizando su espacio de acción, ventajas y desventajas.

Analizar las tecnologías existentes, sus capacidades y versatilidad para la aplicación del concepto de balanceo distribuido del encaminamiento *(DRB).*

[2] InfiniBand define una forma de control de congestión basada en la regulación estática de la inyección en los nodos de paquetes.

Seleccionar las herramientas de desarrollo y simulación adecuadas para el estudio, validación y experimentación de la propuesta establecida.

Experimentación y análisis de los mecanismos introducidos, mediante la comparación a través de simulaciones con otras técnicas existentes. Conformando un conjunto de experimentos que cuantifique y confirme las bondades de nuestras propuestas.

1.5 Organización del documento

Los capítulos que se presentan a continuación, están estructurados de la siguiente manera.

En el capitulo 2, se describen de forma general las características mas relevantes de las redes de interconexión actuales, y se realiza una exposición teórica de los aspectos estáticos estructurales, que definen las redes de interconexión utilizadas en multicomputadores o multiprocesadores. Asimismo en este capitulo, se introducen los diferentes elementos que las conforman y determinan su funcionalidad, junto con los aspectos a tener en cuenta en el diseño de una red de interconexión.
En el capitulo 3, se realiza una descripción general de la especificación propuesta por InfiniBand para el diseño de redes de interconexión, haciendo especial hincapié en las características especiales, que permiten implementar la propuesta de control de congestión utilizada en este trabajo. Mas adelante, en el capitulo 4 se detalla como se utilizan estas características y las adaptaciones que se han realizado sobre la técnica de balanceo distribuido del encaminamiento, con el objetivo de aplicar el algoritmo sobre la arquitectura InfiniBand. Por otra parte, también se analiza el control de congestión propuesto por InfiniBand que servirá como referencia para evaluar el comportamiento de DRB.

Uno de los objetivos planteados en este trabajo, consiste en realizar un estudio general que haga posible la selección de una herramienta de desarrollo y simulación, que permita disponer de una plataforma para la evaluación, experimentación y análisis de la propuesta de encaminamiento de la especificación InfiniBand. A este objetivo se dedica el capitulo 5.

En el capitulo 6, se tratan los aspectos técnicos que permiten la aplicación del balanceo distribuido del encaminamiento en la arquitectura InfiniBand. Asimismo, se describen los modelos que determinan el funcionamiento de la especificación, realizados sobre la plataforma de simulación, que permiten la implementación y la evaluación del mecanismo.

El capitulo 7, abarca un conjunto de experimentos realizados con el fin de evaluar la implementación de la técnica de control de congestión propuesta y analizar su comportamiento, ante diversas características de funcionamiento y diferentes modos de operación de la red de interconexión.

Por ultimo el capitulo 8, recoge las conclusiones globales del estudio junto con las propuestas que las que se pretende continuar trabajando en el futuro.

Capitulo 2 Redes de interconexión

2.1 Introducción

En este capítulo se realiza una exposición teórica de los aspectos estáticos estructurales que definen las redes de interconexión utilizadas en multicomputadores o multiprocesadores. En primer lugar se da una vista general de los diferentes sistemas que emplean este tipo de redes con el motivo de establecer una definición de las mismas y la justificación de su uso. A continuación se introducen los diferentes elementos que las conforman y determinan su funcionalidad, junto con los aspectos de comportamiento a tener en cuenta en el diseño de una red de interconexión y el análisis de las diferentes alternativas existentes en la literatura para cada uno de estos aspectos.

Es posible definir a la red de interconexión como el sistema físico compuesto de una serie de elementos (enlaces y encaminadores) que se comportan e interrelacionan entre ellos de manera específica y que sirven para facilitar la comunicación de los nodos de cómputo del computador paralelo. En este sentido la red es un sistema inteligente [57] que permite la comunicación rápida de datos entre los componentes de dicho computador, y es inteligente en el sentido que permite establecer diversas conexiones entre diferentes puntos, en distintos instantes de tiempo, mediante la gestión de los recursos disponibles.

En virtud de esta definición, el espacio de uso de las redes de interconexión esta enmarcado en los sistemas de computo paralelo, cuyos elementos trabajan en conjunto para cumplir un determinado objetivo, utilizando a la red como el soporte físico para la comunicación y colaboración entre tareas.

Desde este punto de vista nos centraremos en el modelo descriptivo de las redes de interconexión que está caracterizado por el uso de técnicas de conmutación y la utilización de enlaces punto a punto de alta velocidad para conectar sus componentes, junto a un cierto grado de flexibilidad y variabilidad con respecto a las topologías que las conforman.

2.2 Ámbito

Hoy en día, las redes de interconexión de altas prestaciones se utilizan en diferentes sistemas dedicados tanto a la computación como a la comunicación. Es posible clasificar estos sistemas en tres grupos principales [43], en los que es indispensable el uso de tecnologías eficientes de interconexión puesto que la red tiene influencia directa en las prestaciones globales del sistema:

- **Clusters.** Consisten en un conjunto de ordenadores personales conectados a través de una red de interconexión de alta velocidad. Inicialmente fueron concebidos como plataforma para la ejecución de aplicaciones paralelas, aunque posteriormente han sido utilizados para cubrir las necesidades de otro tipo de aplicaciones y servicios, en virtud de su buena respuesta, como es el ejemplo de las redes de almacenamiento *(Storage Area Networks, STANs)* y los servidores de Internet, donde el volumen de acceso de los usuarios y la sofisticación de los servicios ofrecidos demandan, además de tiempos de respuesta acotados, potencia de calculo y capacidad de almacenamiento.

- **Computadores paralelos masivos.** Según lo ya mencionado, el aumento en la versatilidad de las redes de altas prestaciones ha permitido el uso de redes de interconexión en nuevos tipos de sistemas. A pesar de ello, no han dejado de utilizarse en las arquitecturas de computadores masivamente paralelos (multicomputadores y multiprocesadores) para las que fueron concebidas originariamente. En estos sistemas, la red debe ofrecer unas latencias de comunicación reducidas, a fin de evitar el uso ineficiente de procesadores y demás recursos. Además debe ser capaz de manejar eficazmente el volumen de comunicación que ofrece la aplicación, mediante un ancho de banda elevado. Las aplicaciones de calculo intensivo, que requieren una elevada potencia de cómputo y determinada exactitud en los cálculos, justifican el uso de este tipo de computadores paralelos y demandan un desarrollo continuo, tanto en la tecnología, como en la investigación asociada a tal evolución.

- **Routers IP.** El importante crecimiento de usuarios de Internet provoca una demanda elevada de ancho de banda. Las limitaciones tecnológicas en los elementos de conmutación de este tipo de redes no acompaña adecuadamente dicho crecimiento, con lo cual se produce un desfase entre el ancho de banda demandado por los usuarios y el ofrecido por los encaminadores. Este *cuello de botella* ha conducido a un aumento notable en el número de puertos en este tipo de componentes. La opción viable para la construcción de estos elementos de conmutación subyace en las estructuras de red conformadas por varias etapas

(redes indirectas), lo cual hace de las redes de interconexión una alternativa evidente como componente de comunicación de estos dispositivos.

La Fig. 2-1 muestra algunos sistemas actuales que utilizan redes de interconexión para la comunicación entre componentes y que tienen lugar en la clasificación realizada en el párrafo anterior. Cabe destacar que los computadores paralelos *Marenostrum, Earth Simulator y Blue Gene/L*, encabezan la lista de los quinientos supercomputadores más potentes del mundo [49] .

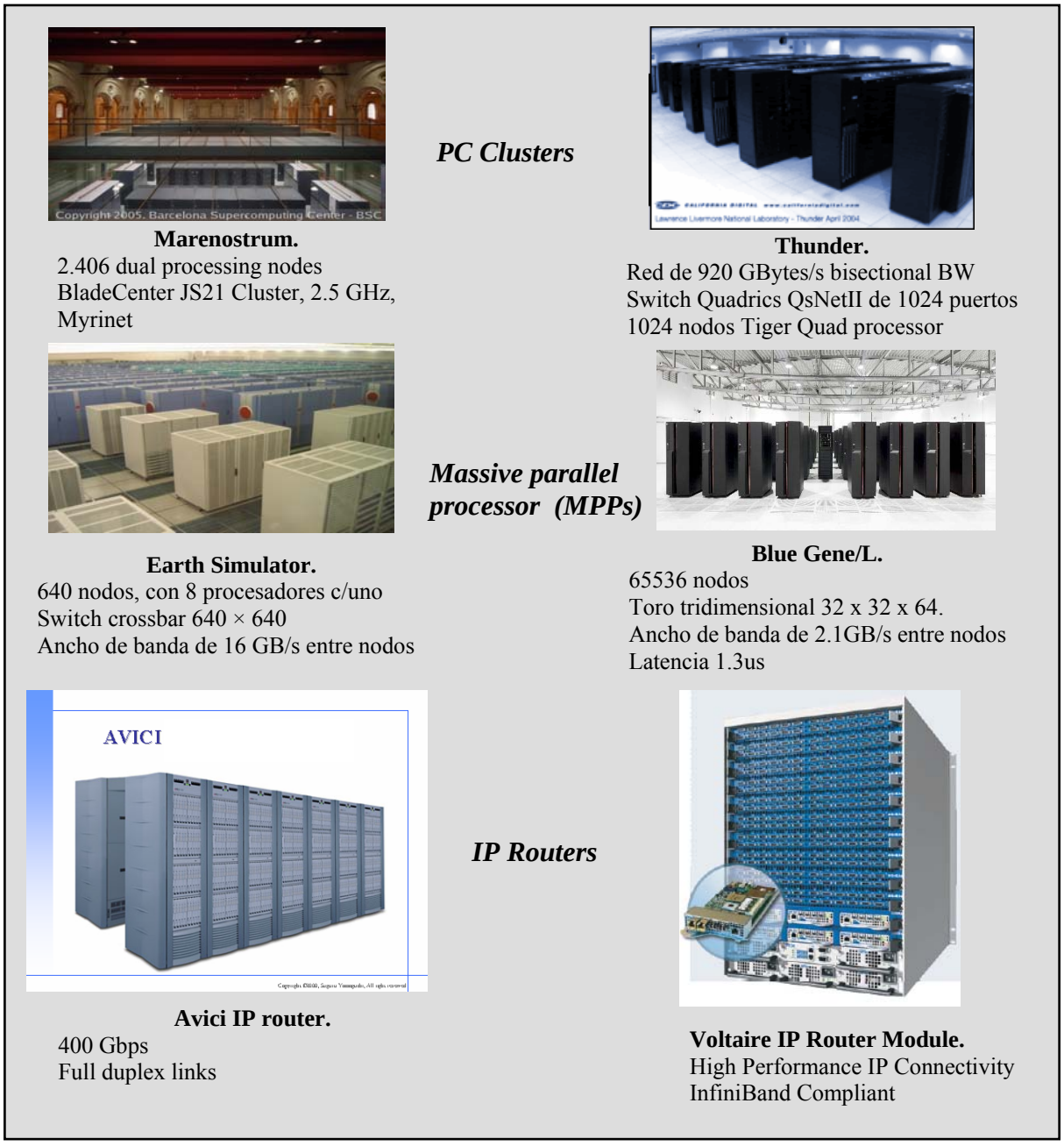


Fig. 2-1 Sistemas que utilizan redes de interconexión de altas prestaciones.

En la se observan algunas de las características técnicas principales en los clusters de ordenadores, computadores paralelos masivos e IP routers mas conocidos. Asimismo,

pueden verse las tecnologías utilizadas (Myrinet, InfiniBand, Quadrics,…), las topologías (crossbar, toro, entre otras), tamaño, ancho de banda y latencia mínimas de la red, etcétera.

2.3 Espacio de diseño en redes de interconexión

El sistema de interconexión de nodos dentro del computador paralelo debe hacer que las comunicaciones entre los procesadores sean independientes del arreglo topológico de los componentes, de esta manera no es necesario hacer cambios en la aplicación, cada vez que se modifica la topología del computador ni cuando se migra directamente a otro computador paralelo diferente. Por lo tanto, el sistema de comunicaciones debe abstraer los detalles físicos del computador en el modelo de comunicaciones.

Durante la ejecución de la aplicación, cada procesador podría desear enviar un mensaje a cualquier otro, por lo tanto la red debe proveer un mecanismo de encaminamiento capaz de hacer avanzar cada mensaje, entre los diferentes nodos de la red de interconexión, hasta alcanzar el destino deseado. Este mecanismo según su principio de funcionamiento, seleccionara una o más trayectorias posibles para establecer la conexión entre el par de nodos fuente-destino.

Lo mencionado anteriormente nos permite establecer una serie de requerimientos, [56] [57] [58] que deben satisfacerse en el diseño de una red de interconexión para cumplir con las características impuestas por los computadores paralelos de altas prestaciones.

2.3.1 Requerimientos de la red de interconexión

- La red debe ofrecer la *mínima latencia* de transito posible para cada mensaje. La latencia tiene un comportamiento dinámico que depende de las condiciones y el volumen de tráfico en la red. Es por esto que los paquetes siempre deben tomar la trayectoria que involucre el menor tiempo de comunicación posible.

- La red debe funcionar cerca de la máxima utilización posible ("*throughput*"), que esta impuesto por el ancho de banda disponible. El "*throughput*" determina la cantidad de trafico por unidad de tiempo que la red es capaz de manejar sin que se produzca saturación en los recursos.

- El mecanismo de encaminamiento utilizado en la red debe *adaptarse* a las condiciones de tráfico y explotar el máximo ancho de banda ofrecido por los enlaces dentro de la topología. Esta característica otorga robustez a la red, ya que provee cierta tolerancia a fallos y la posibilidad de evitar zonas congestionadas.

- Existe un número de fenómenos indeseables que pueden aparecer en la red de interconexión y que deben ser evitados debido a que provocan una degradación significativa en las prestaciones. Estos fenómenos aparecen por el empleo de una técnica de encaminamiento ineficiente y son:

    - "*Deadlock*", esta anomalía provoca una situación indefinida en la que se impide a uno o mas mensaje de la red avanzar hacia su destino.
    - "*Livelock*", este fenómeno provoca la presencia indefinida de un paquete en la red sin llegar nunca a su destino.
    - *"Starvation"*, es la situación en la que un nodo rechaza indefinidamente la inyección de un mensaje o un grupo de mensajes, en la red.

Una vez establecidos los requerimientos impuestos por los computadores paralelos de altas prestaciones a la red de interconexión, se dan las condiciones para definir los diferentes aspectos que dan lugar al espacio de diseño. Es posible dividir a los parámetros principales que intervienen en la definición de una red de interconexión en:

Parámetros estáticos o físicos.

- **Componentes de red.** Son los elementos que conforman el computador paralelo, ejecutan las aplicaciones y almacenan la información. Entre los mas importantes se destacan:
    - Enlaces
    - Conmutadores
    - Encaminadores
    - Nodos de procesamiento o almacenamiento (procesadores, discos, memorias, etc.)
- **Topología.** Define la forma física en la que están interconectados los diferentes nodos del computador paralelo y se pueden clasificar en:
    - Medio Compartido
    - Directas
    - Indirectas

Parámetros dinámicos o de funcionamiento.

- **Control del flujo.** Establece los mecanismos necesarios para controlar la transferencia entre nodos a nivel físico, con el fin de evitar desbordamiento de buffers, regular la cantidad de información ininterrumpida que puede ser enviada dentro de un paquete *(máxima unidad de transferencia, MTU)*, etc.
- **Técnicas de conmutación.** Define la manera en la que se administra el avance de los mensajes de un nodo al siguiente. Existen distintos tipos:
    - Store-and-forward

- Virtual cut-through
- Wormhole

- **Encaminamiento.** Es la entidad encargada de determinar el camino que sigue un mensaje desde el nodo fuente hasta el destino a través de varios nodos intermedios. Se pueden clasificar en:
  - Deterministico
  - *Oblivious* Routing
  - Adaptivo
  - Híbridos

2.3.2 Topologías

Un parámetro importante, que tiene influencia directa en las prestaciones del computador paralelo, es la disposición espacial de los nodos que lo componen. El conjunto de nodos y enlaces se conectan de una determinada manera y definen una estructura topológica. Existen diferentes tipos de topologías posibles, cada una con características diferentes, ventajas y desventajas [21].

2.3.2.1 Redes de medio compartido

Las redes de medio compartido están formadas por un elemento de interconexión único al que se conectan directamente todos los nodos de procesamiento de la red. Estas redes fueron utilizadas en los primeros computadores paralelos pero pronto cayeron en desuso debido principalmente a sus bajas prestaciones. Es posible clasificarlas en redes de área local tradicionales o en redes basadas en "*bus*". La Fig. 2-2 muestra dos ejemplos típicos de este tipo de redes.

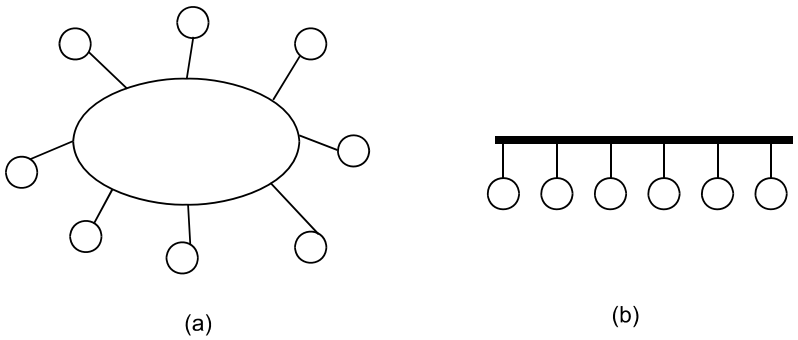


Fig. 2-2 Redes de medio compartido

2.3.2.2 Redes directas

Las redes directas están diseñadas siguiendo un patrón regular y bien definido, a su vez están compuestas por un conjunto de conmutadores con uno o mas nodos de procesamiento, y cada conmutador dispone de varios puertos que permiten establecer varios enlaces con otros conmutadores, conformando topologías sumamente ricas en

recursos, muy versátiles, pero también muy complejas y costosas. Es posible dividir a este tipo de topologías en: Ortogonales y no ortogonales.

Dentro de las redes ortogonales o simétricas existen las mallas abiertas y los toros y los hipercubos, cada una con diversas variantes, pero que siguen un patrón común. En la Fig. 2-3 se representan algunas topologías ortogonales (mallas, anillos, toros e hipercubos).

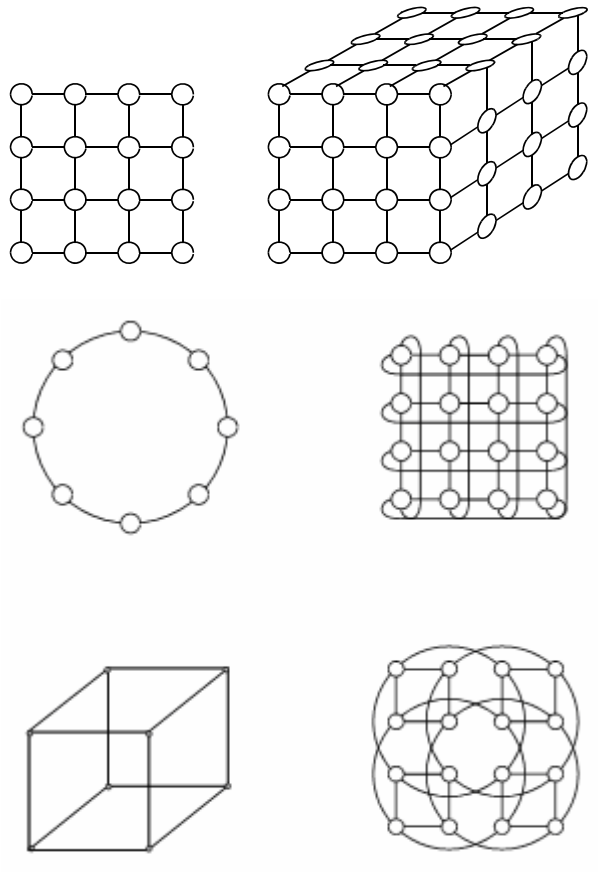


Fig. 2-3 Redes directas ortogonales

Dentro de las redes no ortogonales, existe una gran diversidad de alternativas, sin embargo la mas utilizada es la conocida como "*fat-tree*", esta topología tiene forma de árbol y su característica mas importante radica en el incremento de la cantidad de enlaces cerca de la raíz, como se muestra en la Fig. 2-4.

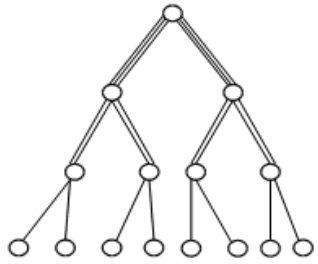


Fig. 2-4 Topología fat tree

Redes indirectas

En el caso de las redes indirectas [57], la comunicación entre nodos de la red se establece a través de varios conmutadores. Cada nodo de la red contiene tiene un adaptador que lo conecta a un conmutador de la red y cada conmutador tiene varios puertos bidireccionales. En el caso de una red indirecta con N nodos, la topología ideal (que permite alcanzar mejores prestaciones) consiste en conectar todos los nodos a través de un único conmutador de N x N puertos. Esta red, denominada "*crossbar*", ofrece una conexión completa entre los nodos de la red. Cuando la cantidad de nodos de la red crece considerablemente, este tipo de topología no posible debido a limitaciones tecnológicas, lo que obliga a utilizar redes multietapa ("*Multi Stage Networks*" ó MINs), como una alternativa de menor coste a los "*crossbar*". En las redes multietapa, no todos los conmutadores tienen conectado un nodo de procesamiento, sino que son utilizados como etapa intermedia. Esto permite desdoblar el número de puertos entre varios conmutadores. En la Fig. 2-5, se observa una red *crossbar* de *nxm* elementos.

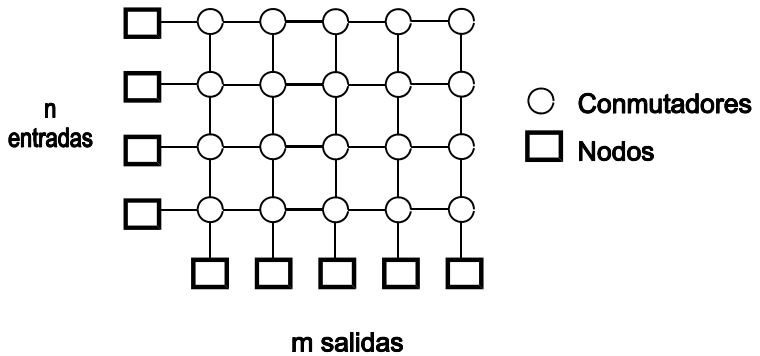


Fig. 2-5 Red tipo *crossbar nxm*

Dentro de las redes multietapa, existe una gran cantidad de propuestas en cuanto a la organización topológica, sin embargo las mas interesantes, desde el punto de vista practico son las conocidas como redes *Delta* [58], cuya principal característica constructiva consiste en el uso de conmutadores idénticos dispuestos en etapas (ej. una red de $N=k^n$ nodos, contiene *n* etapas cada una con *N/k* conmutadores de *k* entradas por *k* salidas, *kxk*). Las redes conocidas como "*Omega*"," *cube*"," *butterfly*" *y*" *baseline*", pertenecen a la familia de redes Delta y son muy utilizadas por sus características topológicas que facilitan el encaminamiento de mensajes. Cada una de estas redes tiene un patrón conexión de los enlaces entre las etapas de conmutadores diferente, aunque topológica y funcionalmente son redes de interconexión equivalentes.

En la Fig. 2-6 se muestran la topología de conexión para cada una de las redes mencionadas, donde pueden verse las diversas formas en que se conectan los enlaces, para cada topología. El comportamiento de la red, ante la carga de comunicaciones esta fuertemente influenciado por el patrón de enlaces entre las diferentes etapas.

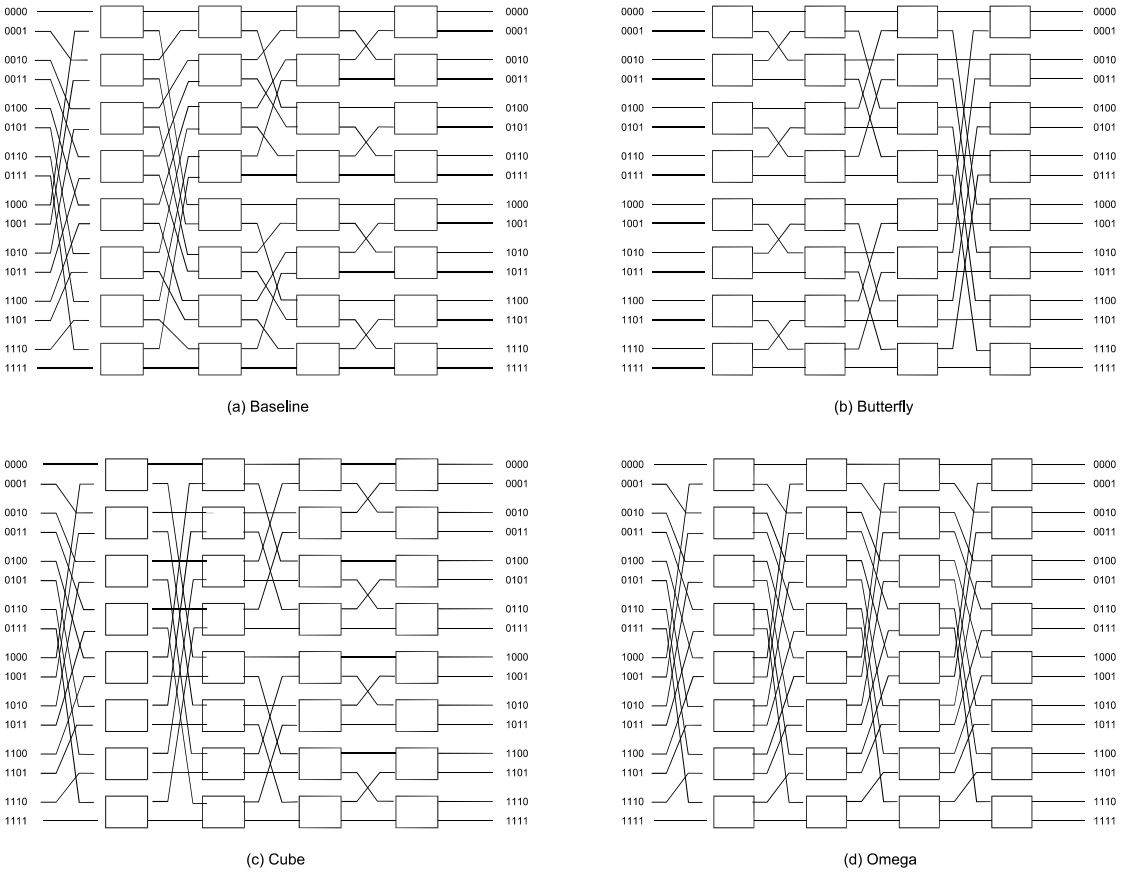


Fig. 2-6 Redes indirectas

2.3.3 Control de flujo

Los puertos son las interfases que conectan los extremos de los enlaces a los conmutadores. Dentro de cada puerto existe una unidad de transmisión y otra de recepción para enviar y recibir, respectivamente, los paquetes que circulan por la red. A su vez, estas unidades contienen *buffers* que permiten contener los datos y evitar la perdida de paquetes ocasionada cuando el puerto no es capaz de enviarlos o recibirlos.

Debido a la presencia de estos *buffers,* es necesario un control de flujo de la información que garantice la capacidad de almacenamiento. Mediante el control de flujo, se notifica al emisor la disponibilidad de espacio dentro del receptor en función de la cantidad máxima de información que puede ser enviada ininterrumpidamente [23]. Los mecanismos mas utilizados para este propósito, pueden dividirse entre los basados en créditos y los basados en marcas.

Los mecanismos de control de flujo basados en **créditos** proporcionan a cada conmutador varios créditos para el envío de paquetes; el número de créditos dependerá obviamente de la cantidad de espacio en el buffer del nodo vecino. En cada transmisión, el emisor consume un crédito hasta que eventualmente los consume todos. Una vez que se libera espacio en el receptor, se envían nuevos créditos al emisor y continúa el envío.

En el caso del control de flujo basado en **marcas**, el receptor detecta que su buffer de entrada esta próximo a desbordarse, debido a que su nivel excede un cierto valor, y envía una señal al nodo emisor para que detenga la inyección de paquetes. Asimismo cuando se recupera el espacio necesario en el receptor y la capacidad es suficiente para recibir paquetes nuevamente, se envía otra señal al emisor para que reestablezca el envío. Este tipo de técnicas se denomina *Stop and Go* o *Xon/Xoff.*

2.3.4 Técnicas de conmutación

Otro de los aspectos que definen el comportamiento de la red es la técnica de conmutación de la información que determina cómo se realiza el avance de los paquetes desde el nodo origen hasta el nodo destino [10]. Concretamente, define cómo se establece el camino entre el nodo fuente y el destino, y cómo se regula el avance de la información en cada nodo de la red. Las técnicas con mayor aceptación dentro de los computadores paralelos de altas prestaciones son:

Store and Forward switching.
Esta técnica determina que el paquete debe ser totalmente recibido y almacenado en el buffer asociado al puerto de entrada antes que pueda ser enviado al puerto de salida. Los enlaces solo se utilizan en el momento en que la información es enviada de un nodo al siguiente, evitando la reserva de dicho enlace más tiempo que el necesario y la consecuente perdida en el uso eficiente del ancho de banda del enlace. Con este mecanismo, la latencia en la transporte del paquete es proporcional a su longitud y al número de conmutadores que atraviesa en el camino. El uso de esta técnica impone ciertas restricciones a los buffers ya que deben contener la totalidad del paquete.

Virtual cut-through switching (VCT).
En este caso, el paquete se retransmite hacia el siguiente conmutador tan pronto como se recibe y se interpreta su cabecera, sin la necesidad de esperar el paquete completo. No obstante, el buffer debe disponer de espacio suficiente para almacenar todo el paquete debido a que existe la posibilidad que el canal de salida este ocupado. Por este motivo, en caso de ocupación, el mecanismo se comporta de igual manera que el analizado anteriormente *(store and forward)*. La influencia de la distancia al nodo destino en la latencia de transporte, es considerablemente menor que en el caso anterior debido a que la ventaja de esta técnica radica en que permite el rápido avance de los paquetes una vez que se procesa su cabecera.

Wormhole switching.
Al igual que en Virtual cut-through, esta técnica permite reenviar el paquete antes de recibirlo por completo, sin embargo el tamaño de los buffers no queda determinado por el tamaño del paquete, sino que es menor y solo es capaz de contener una parte del mismo, justo la parte que contiene la información de encaminamiento. Esta técnica de

conmutación es adecuada para sistemas con requerimientos de tamaño como es el caso de las redes on chip *NOCs*, en los que la cantidad de memoria esta fuertemente limitada por el tamaño del encapsulado [14].

La necesidad de almacenamiento temporal en redes *wormhole* es mínima, debido a que los buffers sólo deben ser capaces de almacenar una parte fija y determinada de cada paquete. Esta técnica, al igual que VCT es capaz de reducir significativamente la latencia de los mensajes.

Como hemos visto los *buffers* solo pueden contener una parte de los mensajes, por lo que en caso de congestión un solo paquete puede quedar almacenado en varios nodos diferentes (que haya visitado), esto provoca que varios enlaces se vean perjudicados por esta situación de congestión, con la degradación de prestaciones que implica.

Hasta aquí se han comentado las principales técnicas de conmutación del flujo de la información. La técnica "*store-and-forward*" es la más antigua y clásica y se usaba en la primera generación de redes de interconexión para computadores paralelos. Como se ha comentado, sus desventajas son la latencia elevada y la gran necesidad de espacio de almacenamiento para guardar los paquetes enteros.

Frente a esto, la técnica mas utilizada en la actualidad es la de "*wormhole*", que presenta menores latencias y necesidades de espacio de almacenamiento en los nodos intermedios. A medida que la tecnología permita aumentar el espacio de almacenamiento de los nodos, se permitirá utilizar la técnica de "*virtual cut-through*", que tiene la misma latencia base que "*wormhole*", pero cuando el paquete se bloquea no permanece en la red ocupando un alto número de enlaces, ya que se almacena en el nodo bloqueado.

2.3.5 Encaminamiento

Es el mecanismo encargado de dirigir los mensajes desde el nodo fuente y, a lo largo de todo el camino, hasta alcanzar su destino. A lo largo de la trayectoria puede surgir la necesidad de hacer varios saltos en nodos intermedios. Es por este motivo que la función de encaminamiento deberá conocer la información obtenida de la topología de la red y seleccionar trayectoria (o trayectorias) apropiada a través de ella.

Esta información debe generarse y procesarse siguiendo cierta planificación o estrategia definida en un *algoritmo de encaminamiento* que establezca el mecanismo mediante el cual se guían los paquetes a sus destinos a través de la red. El principal objetivo del algoritmo de encaminamiento es seleccionar el camino que represente el mínimo retardo total para cada paquete, intentando seleccionar las rutas de manera tal que se eviten las sobrecargas en algunas de las líneas de comunicación, utilizando otras inactivas [13].

De forma general un algoritmo de encaminamiento debe cumplir con las siguientes propiedades:

- Simplicidad y rapidez
- Robustez
- Conectividad
- Maximizar la utilización de enlaces.
- Optimalidad.

Las características de simplicidad y robustez implican que el algoritmo debe realizar decisiones de encaminamiento correctas en un tiempo limitado, por lo que no puede ser un algoritmo computacionalmente complejo [9]. Asimismo la robustez del algoritmo significa que debe tener la capacidad de reconocer cambios de topología debido a fallos de nodos o enlaces y adaptarse a ellos. También nos referimos con el término robustez a que sus características de funcionamiento se mantengan independientemente del patrón de tráfico en la red.

La conectividad en un algoritmo de encaminamiento, determina la existencia de al menos una ruta valida para cada par de nodos de la red. Por otra parte, el algoritmo debe ser capaz de garantizar la igualdad en el acceso a los enlaces por parte de los paquetes, lo que permite un uso justo de los enlaces maximizando su utilización. La optimalidad se aplica manejando correctamente la relación de compromiso existente entre dos características deseables, pero a la vez contradictorias en un algoritmo. Estas características son: minimizar el retardo medio de los paquetes y maximizar la utilización total de la red. El aumento en el uso de la red mediante la inyección de más paquetes implica elevar el tiempo de espera que sufren éstos en los buffers de los conmutadores (dado que, cada vez mas paquetes compiten por el acceso a los enlaces), incrementando el valor de latencia promedio.

La descripción de las propiedades que deben incorporar los algoritmos de encaminamiento realizada en los párrafos anteriores, permite establecer una clasificación o taxonomía de los algoritmos de encaminamiento, en diferentes categorías según una serie de criterios.

Las diferentes alternativas posibles [58], para cada criterio considerado se enumeran a continuación:

- ***Número de destinos***
  - Único
  - Múltiple

- ***Ubicación del algoritmo***
  - Centralizado
  - No centralizado

- ***Función de encaminamiento***
  - Consulta en tabla
  - Cálculo algorítmico

- ***Decisiones***
  - Oblivious
    - Estático/Determinista.
    - Aleatorio
  - Adaptativo
    - *Estado de la red*
    - *Progresividad*
    - *Minimalidad*
    - *Número de enlaces*

El primer criterio enumerado es el **número de destinos** posibles a los que se envía un solo paquete. En caso que un solo destino sea el receptor del paquete tenemos comunicación "*unicast*". En el caso de varios destinos, las comunicaciones se denominan "*multicast*" ó"*broadcast*", según se destine a un subconjunto de varios nodos o a todos los nodos de la red.

El segundo criterio esta basado en determinar quién y dónde se toman las **decisiones de encaminamiento**, es decir, la determinación del camino que seguirán los mensajes. Esta decisión puede hacerse de manera *centralizada*, en el caso de que exista un nodo especial que centralice las decisiones, o *no centralizado*, donde las decisiones se deciden localmente entre todos los nodos de la red. En el caso de encaminamiento no centralizado, existen dos posibilidades, debido a que la determinación del camino a seguir puede hacerse en cada *nodo fuente* que envía mensajes o, de manera *distribuida* a través de cada nodo que es visitado por el mensaje.

En cuanto a la implementación de la **función de encaminamiento**, es posible realizarla tanto a través de una *tabla* que almacena los caminos posibles o mediante una *función* lógica o aritmética que determine la trayectoria a recorrer en cada nodo, según la conveniencia y las facilidades que otorga la topología. En general, se utilizan tablas en redes topologicamente irregulares, mientras que la simetría de las redes directas y las

redes multietapa, permite el uso de alguna función simple que calcula en cada nodo el camino a seguir.

La categoría más importante de clasificación de los algoritmos de encaminamiento, es la que tiene en cuenta la información de los recursos de la red, en el momento de tomar las **decisiones** sobre las trayectorias a seleccionar para el encaminamiento de mensajes. Este aspecto hace referencia a si se tiene en cuenta el tráfico presente en la red y/o la ocupación de los enlaces para determinar los caminos o no se tiene en cuenta dicho estado.

En el primer caso tenemos algoritmos *adaptativos* y en el segundo caso, tenemos algoritmos *oblivious [40]*, en los cuales, las decisiones de encaminamiento no se basan en mediciones o estimaciones del trafico existente en un instante dado, sino que seleccionan aleatoriamente la trayectoria o bien determinan el camino de manera estática según una topología determinada.

La característica sobresaliente de los algoritmos adaptativos [9] consiste en su capacidad de cambiar o adaptar las rutas a las condiciones de tráfico de la red, buscando evitar situaciones de congestión que degraden las prestaciones del sistema.

Dentro de los algoritmos *adaptativos*, existen varias alternativas según la *información de la red*, la *progresividad*, la *minimalidad* y el *número de caminos* que utilizan. Si la información del estado de la red se realiza utilizando información únicamente del nodo en tránsito, el encaminamiento es adaptativo a nivel local; en caso de que se use información de un conjunto de nodos o todos ellos, el encaminamiento es adaptativo a nivel global. En los algoritmos *progresivos*, la cabecera siempre avanza reservando nuevos enlaces, sin la posibilidad de volver hacia atrás a los nodos ya visitados, mientras que, en los algoritmos "*backtracking*", sí es posible esta vuelta hacia atrás deshaciendo parte del camino utilizado para tomar nuevos caminos.

La *minimalidad*, es un aspecto importante y beneficioso en los algoritmos adaptativos. Se denominan *provechosos* a los que siempre utilizan caminos de longitud mínima de manera que cada paso acerca los mensajes al destino, por el contrario los algoritmos *no mínimos* permiten alargar los caminos mediante un alejamiento del destino. Por ultimo cabe decir que existen algoritmos que permiten configurar caminos teniendo en cuenta *todos los enlaces* de la red, es decir la información de todas las posibles trayectorias, mientras que otros sólo permiten utilizar un cierto *subconjunto parcial y conveniente de los enlaces* de la red.

2.4 Control de congestión

El comportamiento de las redes de interconexión tiene una serie de implicaciones directas sobre las prestaciones del sistema y por supuesto sobre las aplicaciones que se ejecutan sobre el computador paralelo. Como se ha comentado anteriormente, el programa del usuario se compone de una serie de tareas independientes que se ejecutan en paralelo sobre los nodos de cómputo y que intercambian mensajes por la red de interconexión.

A medida que aumenta el volumen de comunicación demandado por los nodos de cómputo, se incrementará también la demanda sobre los recursos de la red de interconexión, debido a que existen varios mensajes que compiten por acceder a un determinado enlace [2]. En estos casos, la unidad que controla el acceso a los enlaces deberá seleccionar qué paquete, entre los solicitantes, cruzara hacia el puerto de salida requerido, mientras que los demás deberán esperar en los buffers de entrada.

Los paquetes implicados en esta situación están en estado de contención, según lo muestra la Fig. 2-7

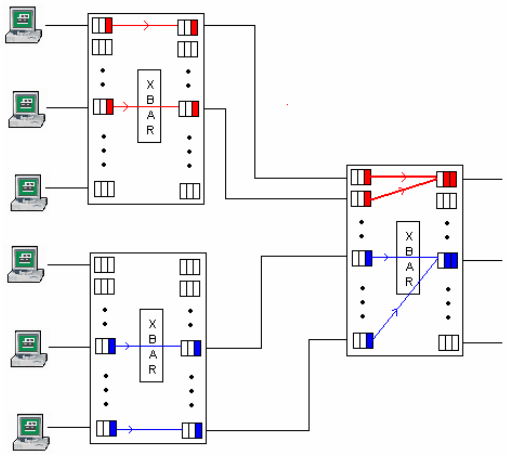


Fig. 2-7 Situación de contención de los paquetes en la red

Ahora bien, si esta situación se mantiene en el tiempo, la acumulación de paquetes en los buffers puede crecer de manera excesiva, llegando incluso a saturarlos. Esta situación es denominada congestión y provoca retardos inadmisibles debidos principalmente al comportamiento exponencial de la latencia de mensajes que deben esperar en los buffers de los conmutadores.

El comportamiento de la latencia que sufren los mensajes, puede ser representado [57] [58] por dos zonas claramente distintas según se observa en la Fig. 2-8.

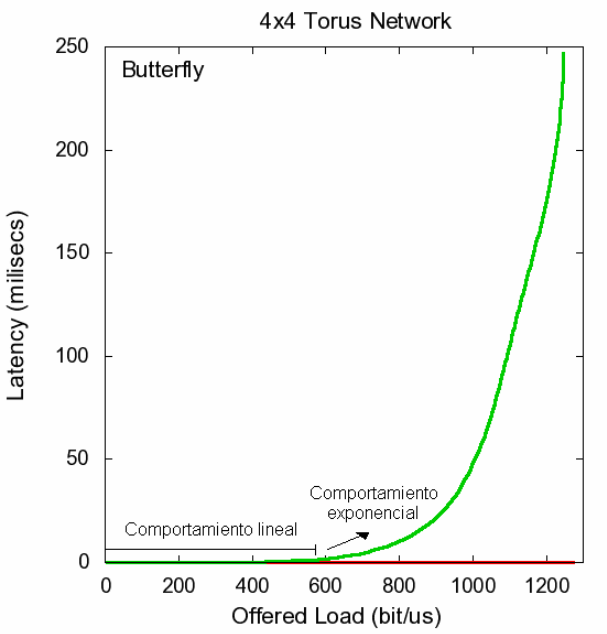


Fig. 2-8 Latencia promedio vs. Trafico ofrecido

Cuando la inyección de mensajes es baja, se observa una zona básicamente plana con un comportamiento prácticamente lineal de la latencia, cuyos valores se mantienen prácticamente acotados aun con grandes variaciones de la carga. En esta zona, la red de interconexión no está saturada, y el volumen de carga puede ser absorbido por los recursos disponibles de la red.

A medida que se incrementa el tráfico, se observa un cambio notable en la curva, que muestra una segunda zona de comportamiento con una gran pendiente, donde la latencia experimenta grandes cambios ante mínimas variaciones de la carga de entrada a la red de interconexión. En este caso, se ha alcanzado la zona de congestión de la red y el volumen de tráfico no puede ser absorbido. Como se ha mencionado, los paquetes deben esperar gran cantidad de tiempo en los buffers antes de entrar en la red, provocando esta zona de crecimiento exponencial de la latencia. Claramente, esta zona de comportamiento no es deseable, debido a la inestabilidad del valor de latencia.

La situación de congestión, puede presentarse aun en casos donde la red está utilizada muy por debajo de su utilización máxima, y esto se debe al desbalance de trafico que provoca una disparidad en el uso de algunos enlaces, pudiendo incluso llegar a saturarlos. Al introducir carga elevada en un solo enlace de la red, ésta se comporta como si toda ella estuviese altamente cargada.

Evidentemente, el retardo sobre los paquetes tendrá una influencia directa sobre el retardo de ejecución total del programa, porque las tareas deberán esperar a que les lleguen sus mensajes para poder operar y esto puede provocar tiempo procesadores ociosos, indeterminación en la aplicación, timeouts, etc.

En este sentido, es primordial el uso de una técnica que controle y solucione esta situación de congestión [37], y que permita evitar el comportamiento exponencial de la latencia, manteniéndola acotada incluso ante valores elevados de carga de tráfico.

2.4.1 Soluciones al problema de congestión

Existen diferentes técnicas que intentan solucionar el problema que ha sido definido y caracterizado, en el párrafo anterior. Estas técnicas consisten en un conjunto de conceptos, con diferentes ventajas y desventajas, que pueden clasificarse globalmente en dos grupos principales: Técnicas preventivas y técnicas reactivas.

Las **técnicas preventivas**, incluyen el tipo de soluciones que intentan resolver el problema de la congestión a través del aumento de los recursos disponibles en el sistema, (sobredimensionamiento) de manera que el volumen de trafico inyectado por los nodos de la red pueda ser entregado a los destinos de forma eficiente, evitando la congestión en todo momento. El empleo de estas técnicas requiere un conocimiento previo del comportamiento de la aplicación con el objetivo de estimar los recursos necesarios para garantizar la ausencia de congestión. Una desventaja de este tipo de soluciones radica en que no siempre es posible disponer del conocimiento mencionado y por otro lado, la reserva de recursos puede incurrir en una elevada sobrecarga de paquetes[3] (*overhead*) y en un uso ineficiente de la red.

Si bien esta solución basada en sobredimensionar la red parece ser la más simple, las características que presentan ciertas tecnologías de interconexión actualmente, dificultan enormemente la posibilidad de sobredimensionamiento. El aumento en el coste de los componentes de red con respecto al de los procesadores que conforman los nodos de cómputo es ciertamente elevado y es un factor que limita fuertemente el sobredimensionamiento de recursos. Asimismo, la velocidad de transmisión en los enlaces, junto con la integración de puertos y la velocidad de conmutación de los propios conmutadores, hacen que el consumo de energía sea también un factor a tener en cuenta en el diseño de la red.

Además del incremento de recursos, es posible encontrar otras opciones dentro de las técnicas preventivas, como es el caso de las soluciones basadas en la apertura de conexiones dedicadas entre diferentes pares de nodos para garantizar la calidad de servicio en las comunicaciones. Asimismo, es posible encontrar otras técnicas que consisten en reservar ancho de banda en función de la utilización requerida para cada enlace. Por ultimo, hemos de mencionar las soluciones que consisten en la planificación y determinación previa de la máxima inyección de mensajes, en forma estática, con el fin de asegurar la disponibilidad de conexión durante la ejecución de la aplicación.

[3] Esta sobrecarga es debida a que la gestión de búsqueda y reserva de recursos se realiza mediante el uso de paquetes.

Técnicas reactivas. Este grupo abarca las soluciones que intentan gestionar los recursos existentes en la red de interconexión de manera eficiente adaptándose a situaciones adversas de tráfico sin utilizar más componentes que los estrictamente necesarios. Por esta razón realizan una monitorización de la cantidad de tráfico que circula por la red, o de la ocupación de los recursos que la componen, con el motivo de detectar situaciones próximas a la congestión y, de esta manera notificar su existencia a los nodos de la red para que puedan llevar a cabo algún mecanismo para controlarla. De esta manera, es posible descomponer este tipo de técnicas en tres fases.

- **Monitorizar** la red y **detectar** congestión
- **Notificar** la existencia de congestión
- Ejecutar **acciones** correctivas

Existen diversas soluciones, cada una con diferentes características de funcionamiento para cada fase. Es por este motivo que dentro de las fases de detección, notificación y ejecución de acciones, pueden encontrarse varias alternativas.

En la fase de **monitorización y detección,** ciertas características dinámicas de la red son evaluadas con el fin de determinar la existencia de congestión. Algunas técnicas realizan la medición del tiempo de transmisión de los mensajes *(Latencia)*, mientras que otras detectan la disminución de la velocidad del flujo de paquetes *(Backpressure)* implícita en el fenómeno de congestión. Los dos mecanismos mencionados actúan a nivel de conexión entre fuente y destino, dando a conocer el estado de los enlaces que la componen. A otro nivel pueden encontrarse técnicas que monitorizan recursos individuales como puertos o buffers y permiten determinar el grado de ocupación de los mismos con el fin de detectar congestión. Esta monitorización puede hacerse tanto a nivel global como a nivel local. En el caso global se analiza un determinado conjunto de recursos y se toman decisiones en función del estado de todos ellos. Esto brinda una información mas precisa del nivel de congestión en la red, aunque implica una sobrecarga *(overhead)* de mensajes en la red destinado a distribuir dicha información, con el consecuente gasto en el ancho de banda destinado a tal efecto.

Con respecto a la fase de **notificación**, también es posible encontrar varias alternativas que permiten dar a conocer la existencia de congestión al resto (o parte) de la red, según el nivel de notificación. El tratamiento de congestión puede ser **local** en algunos casos, en el que un determinado nodo de la red detecta la congestión, pero no notifica al resto sino que intenta solucionar el problema localmente, mediante la gestión adecuada de sus recursos (enlaces, buffers). Por otra parte, existe una gran diversidad de técnicas que luego de la detección, informan a los demás nodos de la red. La cantidad de nodos que reciben la información de la existencia de congestión, depende de la técnica empleada y pueden dividirse en: Notificación a **vecinos,** donde solo un conjunto de nodos cercanos es notificado de la existencia de congestión, y que actuarán para eliminarla. Otro

mecanismo consiste en notificar solo a los **nodos que envían** y cuyos flujos de paquetes contribuyen a aumentar la situación de congestión. Por ultimo, existen mecanismos que notifican a **todos los nodos**, de manera que todos los componentes son concientes del estado de la red. También en este caso, la sobrecarga de mensajes necesarios para transportar la información entre todos los nodos es un factor limitante.

Una vez que la congestión, ha sido detectada e informada, los nodos deberán tomar **decisiones** que permitan ejecutar **acciones correctivas** mediante los mecanismos necesarios, con el fin de eliminar la congestión y evitar la degradación de performance en la red. Algunas alternativas utilizadas, se basan en la *regulación dinámica de la inyección* de mensajes en el nodo fuente *(Message throttling),* y cuya actuación permite detener o disminuir la cantidad de mensajes inyectados durante un tiempo determinado o hasta que caiga la cantidad de trafico bajo cierto nivel. Este tipo de acciones, si bien elimina la congestión, conduce a obtener bajas prestaciones en el comportamiento global del computador paralelo, debido a que los mensajes seguirán contenidos ya no en los conmutadores de la red, pero sí en los nodos fuente, lo que de todas maneras incrementa la latencia. Otra posibilidad consiste en gestionar y optimizar la utilización de los *buffers*. Esta técnica se utiliza dentro del conmutador y actúa reordenando los paquetes de diferentes flujos para evitar interferencias. Los mecanismos de este tipo buscan retransmitir rápidamente los paquetes que no están congestionados pero que comparten algún recurso con un flujo que si lo está. Si bien la gestión de los paquetes en los buffers produce cierto efecto beneficioso, no permite solucionar la causa que provoca la congestión. Es por esto que además de gestionar los mensajes que han entrado en la zona de congestión, el mecanismo utilizado, debe solucionar el problema que conduce a los componentes a entrar en estado de congestión.

El **encaminamiento adaptativo**, se utiliza a nivel de conexión (punto a punto) y actúa redistribuyendo la carga a través de múltiples trayectorias alternativas entre el mismo par fuente-destino. Cuando el mecanismo detecta la presencia de congestión en los recursos utilizados, notifica a cada nodo responsable de la inyección de paquetes en estos recursos para que utilice otro camino hacia el destino. De esta forma, la latencia se mantiene en valores acotados ya que no se detiene la inyección y la utilización de la red se incrementa notablemente debido al uso de trayectorias alternativas.

2.4.2 Características deseables en los mecanismos de control.

En general las técnicas empleadas en el control de congestión deberán cumplir con una serie de requerimientos que garanticen el funcionamiento y las prestaciones de la red. El tiempo en que los mecanismos de control de congestión reaccionan y actúan, debe estar convenientemente acotado para evitar respuestas tardías en la solución del problema de congestión.

Asimismo, algunos mecanismos pueden proveer una respuesta adecuada a nivel local, pero conllevan un desbalance de carga en la red que conduce a una degradación global de las prestaciones, por lo que es necesario que tengan un funcionamiento adecuado tanto en zonas puntuales, como en toda la red.

Ciertas técnicas pueden reaccionar correctamente ante situaciones de congestión, pero generan sobrecarga o nuevos fenómenos indeseables (ej. deadlock, livelock, starvation) con cargas normales de trafico. Esta *penalización* debe ser evitada por la técnica de control.

Por otra parte las soluciones utilizadas, deben ser independientes de las condiciones de tráfico presente en la red, la topología, el tamaño de los mensajes, etc. obteniendo así una técnica escalable y robusta.

A continuación se intenta ilustrar, mediante algunas técnicas existentes, el contenido hasta aquí expuesto, mostrando las técnicas empleadas en las fases de detección, notificación y corrección.

2.4.3 Ejemplos

Encaminamiento basado en colas *(Channel Queue Routing, CQR)* [35]
Esta técnica propone cambiar de política de encaminamiento en función del estado de ocupación en las colas de los puertos. Estima congestión a través de la ocupación promedio en todos los puertos del conmutador y si ésta supera un umbral se utiliza encaminamiento no mínimo. Por tanto, proporciona detección a nivel local, y conmuta entre encaminamiento mínimo y no mínimo según el estado de los buffers del conmutador.

Canales útiles *(U-Channels)* [12]
Estima el volumen de tráfico a través del número de canales virtuales libres en el conmutador. Solo utiliza canales útiles (provechosos) y habilita la inyección si el nº de canales útiles libres supera un valor umbral determinado. La detección se produce a nivel local y controla la congestión regulando la inyección (Inyección/No inyección). Cuando la inyección esta habilitada utiliza encaminamiento mínimo.

En ambos ejemplos vemos que las técnicas utilizan información local, lo que no permite el conocimiento de áreas alejadas pudiendo detener la inyección en el nodo (localmente) aunque éste no sea quien provoca la congestión.

Injection And Network congestión

Este mecanismo detecta congestión midiendo la velocidad de avance de mensajes (flits/seg), mediante un contador asociado. Una vez superado un valor de umbral, la congestión se trata en función de dos políticas diferentes según el estado de dos flag´s:

- El Flag de inyección. Activado cuando hay congestión en algún canal virtual que el nodo usa para inyectar mensajes a la red.
- El Flag de red. Activado cuando hay congestión en algún canal virtual usado para recibir mensajes desde la red.

Como se puede observa el flag de inyección da lugar a una política mas restrictiva, debido a que el nodo que lo contiene es uno de los culpables de la aparición de congestión.

Para finalizar esta colección de ejemplos, podemos mencionar las características principales de los mecanismos:

***P**ing – **B**ubble (PB)* [33]

- Basado en incrementar la prioridad de inyección de paquetes.
- Detecta congestión utilizando paquetes de control, *pings*.
- Libera lugar en los buffers (Bubble) y lo propaga hacia la congestión.

***R**egional **E**xplicit **C**ongestion **N**otification (RECN)* [24]

- Detección **local** de ocupación de colas en puertos de salida.
- Propagación de notificación a nodos vecinos.
- Separación los flujos de paquetes congestionados para que no interfieran con los demás.

Self – Tuned [39]

- Detecta congestión a nivel **global** midiendo el número la ocupación de los buffers en la red.
- Aplica una regulación de inyección en todos los nodos.
- Ajusta automáticamente el umbral según diversas mediciones para adaptarse a la carga de tráfico.

2.4.4 Balanceo distribuido del encaminamiento

Tal y como se comentó en la introducción, el método utilizado como punto de partida en este trabajo es el conocido como **balanceo distribuido del encaminamiento** (DRB por sus siglas en inglés) [25] [26] [27] [28]. DRB es un método para crear caminos alternativos entre los diferentes nodos fuente y destino en la red de interconexión y distribuye la carga de mensajes de cada par fuente-destino en un camino constituido por varios caminos simples. La distribución está controlada por el nivel de carga en todos

los caminos alternativos; estos nuevos caminos harán uso de los enlaces disponibles en los encaminadores.
Los nuevos caminos utilizados pueden ser de distancia mínima o no, dependiendo de la carga de tráfico presente. Por lo tanto, es posible configurar caminos de distancia mínima o permitir un cierto alargamiento de los caminos, siempre de manera controlada.

El objetivo de DRB es crear una distribución uniforme de la carga de tráfico sobre toda la red de interconexión para mantener una latencia baja y uniforme, eliminando situaciones de congestión. En este sentido, la distribución realizada por DRB mantendrá una latencia baja y uniforme sobre toda la red de interconexión siempre que la demanda total de ancho de banda de comunicaciones no supere la capacidad total de los enlaces y encaminadores de la red de interconexión. Dependiendo de la carga de tráfico y de su distribución sobre la red de interconexión, el método DRB distribuye la carga de los caminos mas cargados hacia los menos cargados.

Es importante destacar que el mecanismo produce un efecto de balance colectivo (a nivel global), pues esta expansión se produce para todos los pares fuente-destino de la aplicación que interaccionan entre sí.

Conceptualmente, este método mide el estado de la carga en todas las conexiones entre pares fuente destino; al detectar congestión se notifica al/los nodo/s que inyectan mensajes para que configuren nuevas trayectorias posibles y redistribuyan el trafico según su estado de carga.

El método consiste de tres fases:

- **Monitorización de la Carga de Tráfico**. Los mensajes almacenan la latencia que experimentan en su camino y esta información se reenvía al nodo fuente mediante un mensaje de reconocimiento de máxima prioridad.

- **Configuración Dinámica de Trayectorias Alternativas**. Cuando el nodo fuente recibe la información de latencia se modifica el número de posibles trayectorias si ésta supera un umbral.

- **Selección de Caminos Múltiples**. Teniendo en cuenta la latencia en los caminos y su valor se genera una distribución de probabilidad para seleccionar las trayectorias más convenientes para el envío.

2.4.5 Resumen

A modo de conclusión del capítulo, la Fig. 2-9 muestra un resumen de algunas de las técnicas más importantes, concebidas en los últimos años, diseñadas para el control de congestión y el encaminamiento inteligente de mensajes. En ella se muestra el tipo de encaminamiento que utilizan y el tipo de comportamiento en cada fase del mecanismo (Detección, notificación y acciones correctivas).

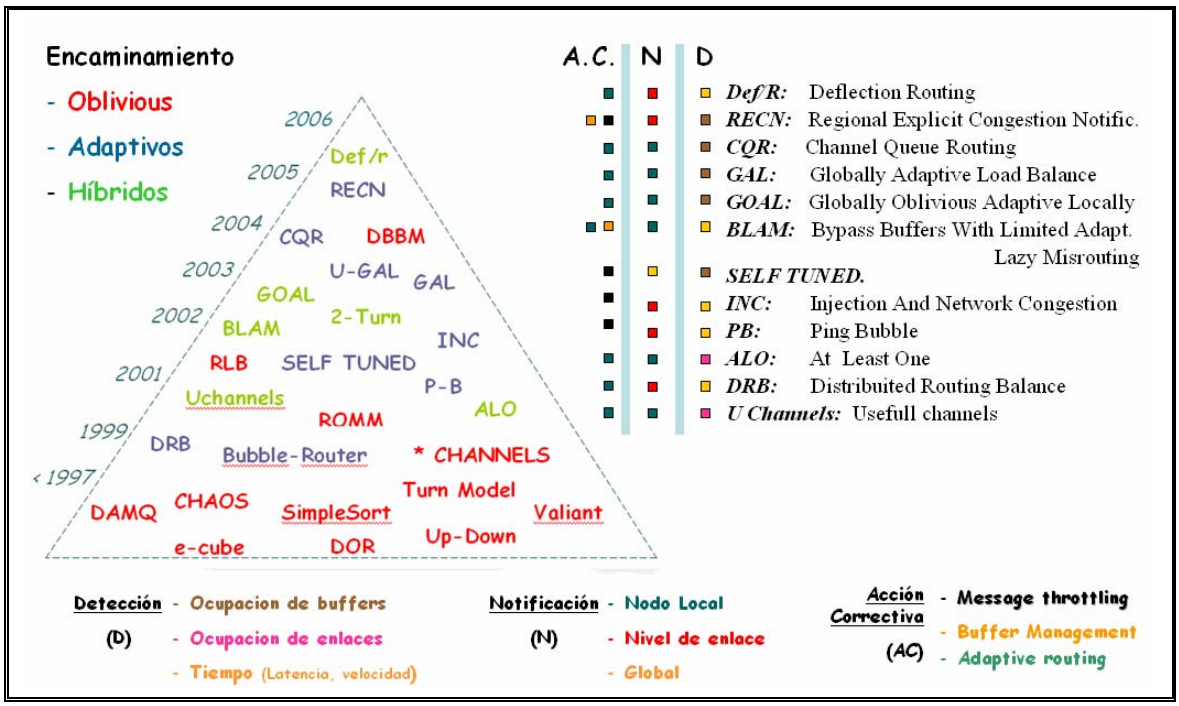


Fig. 2-9 Resumen sobre las técnicas de control de congestión

En la figura puede verse la evolución en el tiempo, hacia los algoritmos de encaminamiento adoptivos. Los algoritmos oblivious, como el método de Valiant [19], e-cube, Turn Model [51], *Channels, etcétera, fueron ampliamente utilizados en el pasado, pero, principalmente por su pobre respuesta en latencia, han dejado lugar a los algoritmos que incluyen cierta inteligencia para realizar el encaminamiento.

Entre los algoritmos adaptivos, puede notarse cierta preferencia por el uso de técnicas que en su fase de detección monitorizan la ocupación en los buffers de los conmutadores o bien el tiempo en que los mensajes, esperan dentro de ellos. Con respecto a la forma en que se lleva a cabo la notificación, la preferencia está en los algoritmos que lo hacen localmente o a nivel de enlace, principalmente porque necesitan menos ajustes y presentan menos requerimientos de comunicación para transmitir a los demás nodos, el estado de los recursos.

Por ultimo, puede observarse las diferentes acciones correctivas que utilizan los algoritmos de la Fig. 2-9. Las acciones basadas distribuir la carga de trafico mediante el

encaminamiento adaptativo presentan, mejores características respecto a la utilización de la red y su respuesta en latencia. Sin embargo, la regulación de mensajes, es una opción utilizada en algunos casos, principalmente por su simplicidad en la implementación.

Capitulo 3 Arquitectura InfiniBand

3.1 Descripción General

Este capítulo proporciona una descripción cualitativa de las características, capacidades, componentes y elementos de la arquitectura, junto con el principio de funcionamiento y operación de la especificación InfiniBand *(IBA)*, ofreciendo, de esta manera una descripción de alto nivel, donde se excluyen intencionalmente ciertos detalles con el fin de clarificar los puntos más importantes del estándar (Para una información más detallada remítase a [15] ó [55]).

Principalmente nos enfocamos en aquellas características, cuyo modo de operación es necesario entender y conocer, con el fin de realizar la implementación del balanceo distribuido del encaminamiento que proponemos en este trabajo, sobre la arquitectura IBA. Por tanto, este capítulo conforma una explicación de las bondades del estándar, que nos permitieron tomar la decisión y seleccionar a InfiniBand como plataforma para la aplicación de DRB.

InfiniBand define una red de área de sistema (SAN) que permite conectar múltiples nodos de procesamiento independientes (Host processors nodes), plataformas y dispositivos entrada-salida (Fig. 3-1). La SAN definida por InfiniBand consiste en una infraestructura diseñada para gestionar y dar soporte tanto a comunicaciones entrada-salida como a comunicaciones interprocesador (IPC) en sistemas de computo. Un sistema InfiniBand puede extenderse desde un pequeño servidor con un solo procesador y algunos dispositivos de entrada-salida, hasta un supercomputador paralelo masivo (Massively Parallel Supercomputer) con centenares de procesadores y dispositivos de entrada-salida. Por otra parte, la naturaleza compatible de la especificación IBA con el protocolo de Internet (IP), permite establecer conexiones sobre Internet o Intranet con sistemas remotos.

La arquitectura IBA propone una red de interconexión conmutada punto a punto permitiendo, de esta manera, la comunicación entre varios dispositivos con un elevado ancho de banda y una latencia muy baja, en un ambiente protegido y gestionado remotamente. Un nodo determinado puede comunicarse con cualquier otro, mediante diferentes puertos utilizando trayectorias múltiples, a través de la red InfiniBand. Las aplicaciones que requieren tolerancia a fallos, gran ancho de banda de transferencia de datos, balanceo de carga o de comunicaciones, explotan al máximo esta multiplicidad de puertos y trayectorias.

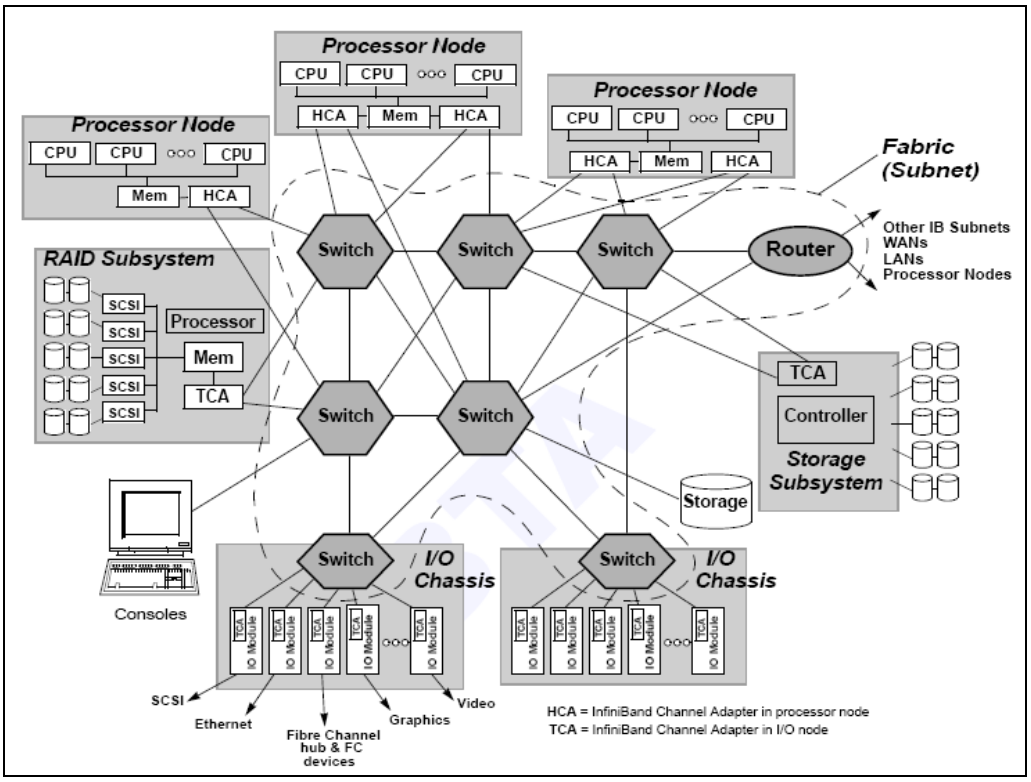


Fig. 3-1 Red de área de sistema InfiniBand

El hardware propuesto ha sido diseñado para liberar al procesador de la carga relacionada con el manejo de datos entrada-salida, lo que permite múltiples comunicaciones concurrentes sin la sobrecarga (*overhead*) tradicional asociada a los protocolos de comunicación. La red contiene dos características clave y diferenciales para el alto rendimiento de InfiniBand, dado que provee a los nodos IPC o I/O transferencias de datos sin involucrar al kernel del sistema operativo, y utiliza hardware dedicado para proveer al sistema, comunicaciones altamente confiables y tolerantes a fallos, liberando al procesador de estas tareas de comunicación.

El sistema de comunicaciones IBA consiste de un grupo de nodos de procesamiento y unidades entrada-salida conectados a través de una red compuesta de conmutadores (switches) y encaminadores (routers) conectados en cascada.

Las unidades de entrada-salida pueden abarcar desde dispositivos ASIC (Adaptadores SCSI o adaptadores LAN) hasta los grandes subsistemas RAID que compiten en complejidad con los nodos de procesamiento

La especificación sólo se limita a definir la red de interconexión, los encaminadores y elementos de conmutación, los nodos (*endnodes,* mas adelante se presenta una definición de este termino), la infraestructura de gestión, el formato y el protocolo de comunicación. No especifica los comandos ni el funcionamiento de los dispositivos. Es decir, InfiniBand no define los comandos de entrada-salida de los discos, ni cómo los

adaptadores SCSI se comunican con éstos, o de que manera el sistema operativo (OS) ve los dispositivos, ni qué nodo dentro del sistema contiene los discos. En cambio, la especificación sí define la forma en que los datos y comandos se transportan entre los nodos de procesamiento y los controladores de entrada-salida.

IBA maneja las comunicaciones de datos IPC y entrada-salida en entornos multi-computador, soportando el elevado ancho de banda y la escalabilidad requerida junto a un nivel de latencia y overhead en los CPU´s, extremadamente bajos. Con InfiniBand, el sistema operativo puede proveer mecanismos de comunicación que no utilizan el kernel y acceden directamente al hardware de red permitiendo operaciones eficientes de paso de mensajes.

3.1.1 La Subred InfiniBand (IBA Subnet).

A alto nivel, InfiniBand se utiliza para interconectar los nodos de la red según lo ilustrado en la Fig. 3-2. Cada nodo puede ser un elemento de procesamiento, una unidad entrada-salida, o un encaminador a otra red.

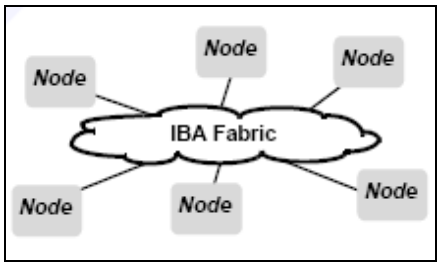


Fig. 3-2 Red InfiniBand

A su vez, la especificación divide a toda la red en subredes (SUBNET's) que se interconectan mediante encaminadores (IBA routers) como se observa en la Fig. 3-3, y donde cada nodo puede estar asociado a una o más subredes.

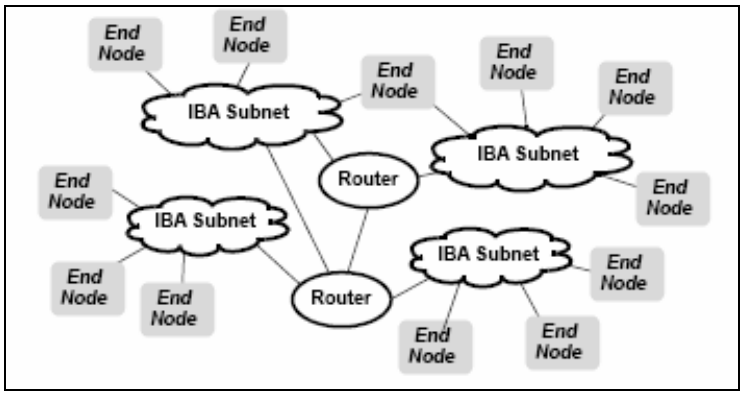


Fig. 3-3 Componentes de red

Cada una de las subredes mencionadas está compuesta de nodos, conmutadores, encaminadores y elementos de gestión de la subred *(Subnet managers)* interconectados

mediante enlaces *(Links)*, ver Fig. 3-4. Los dispositivos pueden unirse a uno o mas conmutadores y/o a otros dispositivos directamente[4], permitiendo múltiples conexiones entre ellos.

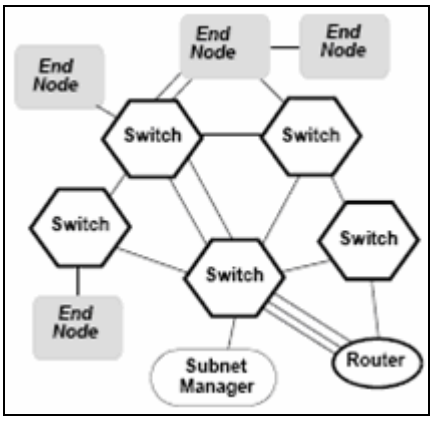


Fig. 3-4 Componentes de la subred

La arquitectura esta optimizada para unidades de cómputo que contienen múltiples procesos independientes y *threads (consumers)* según lo ilustrado en la Fig. 3-5. Dichas unidades, se conectan a través de los adaptadores de canal *(Channels Adapters)* que constituyen un nodo de la red. La arquitectura soporta múltiples adaptadores de canal por unidad y cada adaptador de canal proporciona unos o más puertos para la conexión[5].

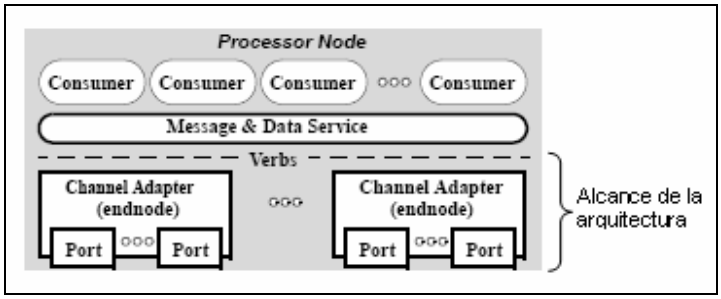


Fig. 3-5 Nodo de procesamiento

Dentro del nodo de procesamiento, el servicio de envío de mensajes y comunicación de datos (message and data service) es un componente del sistema operativo y por lo tanto está fuera del alcance de la especificación. La interfase semántica entre el servicio de envió mensajes y comunicación de datos y el adaptador de canal esta definida y es denominada capa de verbos *(IBA verbs layer)*. Los verbos describen las funciones necesarias para configurar, manejar, y para operar un adaptador de canal. Estos verbos identifican los parámetros apropiados que necesitan incluirse y configurarse en cada

[4] Las conexiones directas entre nodos conforman una subred independiente, sin conectividad con el resto de los dispositivos. En este caso uno de los nodos interconectados funcionara como el gestor de subred para dicha conexión

[5] En este caso, la unidad es vista por la red como múltiples nodos independientes.

función particular. La capa de verbos no es una API, pero provee un *framework* para que sea especificada.

La arquitectura, a su vez define el comportamiento del *host* (mediante verbos) y define la operación de memoria de tal forma que el adaptador de canal puede situarse tan cerca del sistema de memoria como sea posible, proporcionando un acceso directo e independiente entre los consumidores, sin importar si esos consumidores son nodos o controladores entrada-salida o procesos de software. IBA proporciona la semántica de canal (operaciones de envío y recepción) y el acceso directo a memoria con un nivel de protección y aislamiento que impide el acceso a consumidores que no participan en la comunicación (comunicación segura).

3.2 Comunicación

3.2.1 Arquitectura De Colas (Queueing)

El fundamento de operación de IBA esta basado en la capacidad de un consumidor de encolar el conjunto de instrucciones que es ejecutado por el hardware. El estándar denomina a este mecanismo como cola de trabajo *(work queue)*. Las colas de trabajo son creadas siempre en pares, (pares de colas *(QP)*) una destinada a operaciones de envío y otra a operaciones de recepción. En general, la cola de trabajo de envío (*send work queue)* almacena instrucciones utilizadas para la transferencia de datos entre la memoria local del consumidor que envía y la de otro consumidor de la red, mientras que la cola de trabajo de recepción *(receive work queue)* almacena instrucciones que indican en que lugar de la memoria deben colocarse los datos recibidos desde un consumidor remoto (*remote consumer*). Un consumidor remoto puede estar localizado en el mismo nodo pero utiliza la red para enviar datos. IBA especifica el manejo de colas en el adaptador de canal del *host* (HCA). A continuación se describe dicho modelo (Fig. 3-6).

El consumidor realiza una petición de trabajo (WR), a través de una instrucción llamada elemento de cola de trabajo (WQE). El adaptador de canal ejecuta estos elementos respetando el orden en que fueron colocados en la cola de trabajo. Una vez terminada la ejecución del elemento el adaptador de canal coloca un elemento de cola de terminación *(Completion Queue Element, CQE)* en la cola de terminación. Cada CQE especifica toda la información necesaria para terminar la instrucción de trabajo.

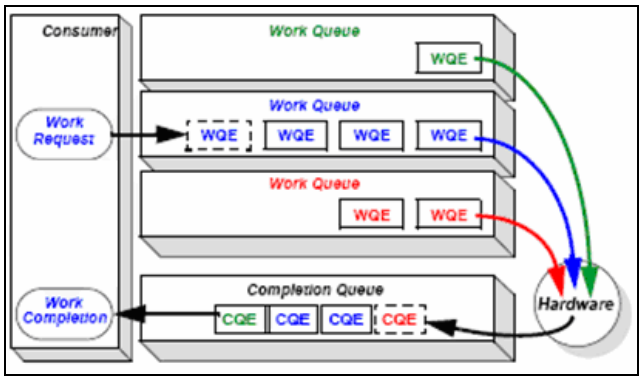


Fig. 3-6 Modelo de colas del consumidor

Cada consumidor, en el nodo, puede tener su propio conjunto independiente de colas de trabajo. Además crea una o más colas de terminación y las asocia con las colas de envío y recepción.

Debido a que algunas colas de trabajo requieren un mensaje de reconocimiento del nodo remoto (comunicaciones fiables) y a que algunos elementos de cola de trabajo requieren el uso de paquetes múltiples para transferir datos, el adaptador de canal puede tener múltiples WQEs trabajando al mismo tiempo, incluso pertenecientes a la misma cola de trabajo.

La especificación define tres clases de operaciones de envío: *operación de envío* (*send operation), acceso remoto a memoria (RDMA) y Asociación de memoria (memory binding).* En el caso de la recepción existe una operación solamente.

- En la operación de envío, el elemento de cola de trabajo WQE, especifica un bloque de datos en la memoria local del consumidor para que el hardware lo envíe al nodo destino, donde existirá otro elemento WQE en la cola de recepción que especifique el lugar donde se almacenaran los datos recibidos.

- En las operaciones de acceso remoto a memoria (RDMA), el elemento de cola de trabajo también especifica la dirección en la memoria del consumidor remoto. De esta manera una operación RDMA no involucra el uso de un WQE dentro de la memoria del consumidor remoto. Existen tres tipos de operaciones RDMA: *RDMA-WRITE, RDMA-READ, y ATÓMIC OPERATION.*

  - La operación RDMA-WRITE, la memoria del consumidor local, hacia la memoria del consumidor remoto.
  - La operación RDMA-READ, especifica la transferencia de datos entre la memoria del consumidor remoto, hacia la memoria del consumidor local.
  - La operación ATÓMICA lleva a cabo la lectura de una posición de memoria remota de 64 bits. El destino responde con el valor leído y eventualmente se

modifica o reemplaza el contenido de la memoria remota, escribiendo un valor actualizado en dicha posición.

- La Asociación de memoria *(memory binding)* asocia una ventana de memoria dentro de un rango especificado, que permite a los consumidores definir porciones dentro de dicha memoria para compartir con otros nodos y especificar permisos de lectura/escritura.

Sólo existe una operación para la cola de recepción que permite especificar el buffer de recepción en la memoria.

- En la operación de recepción, el elemento de cola de trabajo especifica el lugar en memoria donde el hardware coloca los datos recibidos desde otro consumidor que ha ejecutado previamente una operación de envío. Cada vez que un consumidor remoto ejecuta con éxito una operación de envío, el hardware ejecuta la siguiente instrucción de la cola de recepción, coloca los datos recibidos en la posición de memoria especificada por el elemento de trabajo en la cola de recepción, y coloca un elemento de terminación CQE en la cola, indicando al consumidor que la operación de recepción ha terminado. De esta manera, la ejecución de una operación de envío causa una operación en la cola de recepción de un consumidor remoto.

Normalmente una operación de acceso directo a memoria no consume elementos WQE en la cola de recepción del destino, pero existe una excepción que tiene lugar cuando se ejecuta una operación de escritura (RDMA write) con un dato inmediato (*Immediate data*). Los datos inmediatos contienen 32 bits de información que se proporcionan como parte de una instrucción de envío o escritura RDMA. Este dato en vez de ser escrito en la memoria, se trata como otro tipo de información (información de estado) y es devuelto como un campo especial de estado en la cola de terminación del receptor. Esto que implica que, en este caso, una operación de escritura RDMA con datos inmediatos consumirá un elemento de trabajo WQE en la cola de recepción.

3.3 Conexiones

InfiniBand soporta tanto el servicio orientado a conexiones como el servicio de comunicación mediante datagramas. El servicio orientado a conexiones asocia cada par de colas (QP) exactamente a un consumidor remoto. En este caso, el ámbito de existencia de los pares de colas se configura con el identificador del par de colas del consumidor remoto. El consumidor remoto se identifica por un puerto y un número que identifica al par de colas dentro de ese puerto. La especificación permite identificar a cada puerto mediante un identificador local LID *(Local IDentificator)* y opcionalmente

un identificador global GID (para nodos situados en subredes diferentes). En el proceso que permite establecer la conexión esta información es intercambiada entre los nodos.

Contrariamente, el servicio de comunicación mediante datagramas no establece una asociación directa con el par de colas de un consumidor remoto, sino que la información en el elemento de trabajo WQE identifica el destino. En este caso, el proceso de inicio de comunicación debe realizarse cada vez que sea necesario intercambiar nueva información con el mismo destino o uno nuevo.

3.4 Protocolo de Comunicaciones

La arquitectura define un protocolo de comunicaciones (Fig. 3-7) y proporciona un número de transacciones que el consumidor puede utilizar para ejecutar con otro consumidor remoto. Cada consumidor establece un elemento en la cola de trabajo WQE al par de colas que el adaptador de canal interpreta para llevar a cabo la operación.

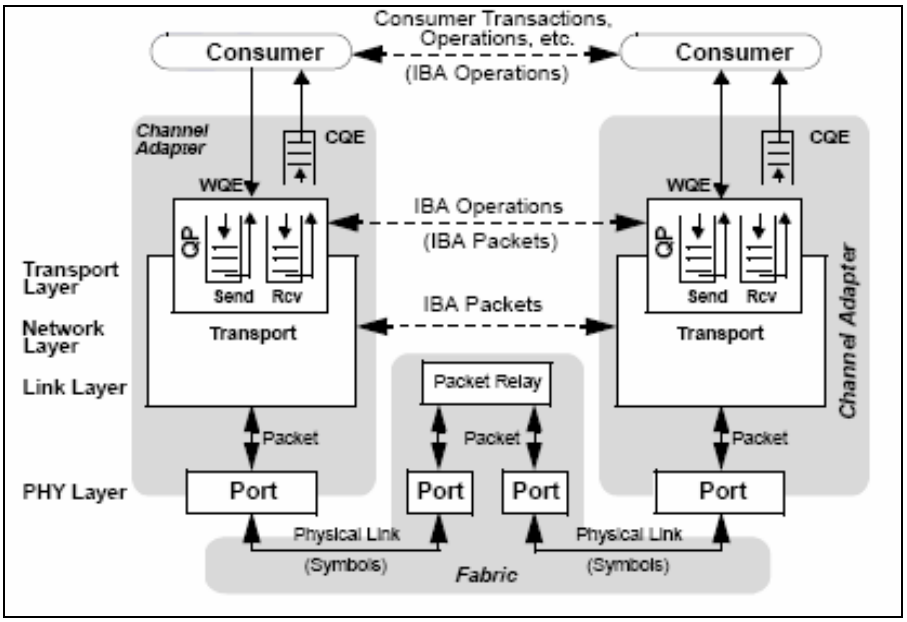


Fig. 3-7 Protocolo de comunicaciones *(stack)*

En las operaciones de envío, el adaptador de canal interpreta el WQE, crea un mensaje de petición *(request message)* que divide en múltiples paquetes si es necesario, agrega las cabeceras necesarias para encaminar el mensaje, y envía el paquete hacia la red a través del puerto apropiado.

La lógica del puerto envía el paquete hacia los enlaces, los conmutadores y encaminadores lo retransmiten a través de la red hasta el destino.

Una vez que el destino recibe un paquete, la lógica del puerto valida la integridad del paquete. El adaptador de canal asocia el paquete recibido al par de colas

correspondiente y procesa el paquete con el fin de ejecutar la operación. Si es necesario, el adaptador de canal crea un mensaje de respuesta *(acknowledgement)* y lo envía hacia el nodo fuente.

La recepción de ciertos mensajes causa que el adaptador de canal consuma un WQE de la cola de recepción. Cuando lo hace, se coloca un elemento en la cola de terminación CQE correspondiente al WQE consumido, lo que implica que se ha realizado con éxito la operación requerida por el consumidor que posee ese par de colas.

3.4.1.1 Arquitectura Modelada por Capas

El funcionamiento y la operación de la arquitectura, puede describirse mediante una serie de capas (Fig. 3-8). El protocolo utilizado en cada capa es independiente de los utilizados en las demás y permite que una capa superior proporcione diferentes servicios a las situadas debajo en el modelo.

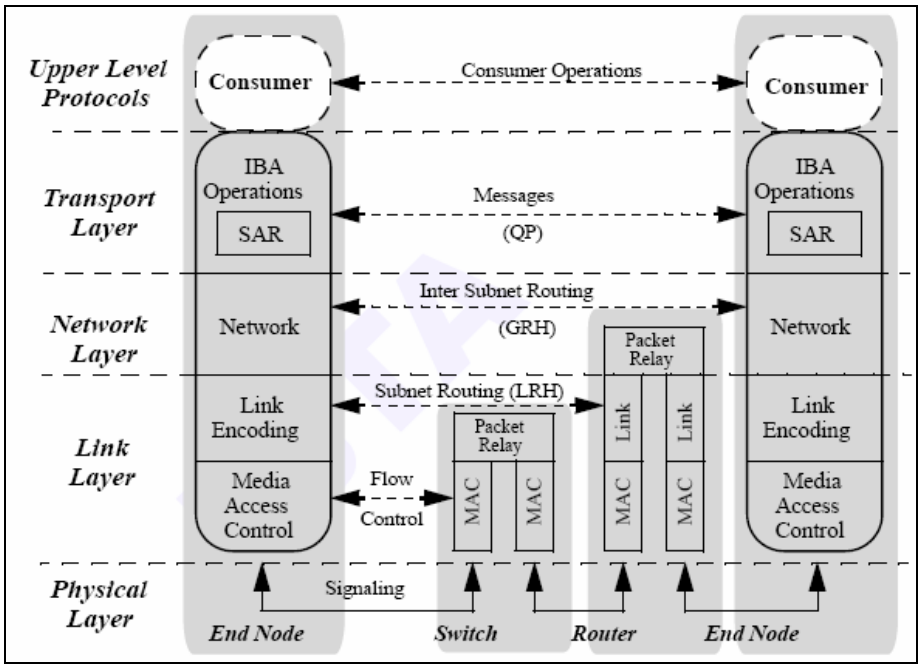


Fig. 3-8 Capas de la arquitectura IBA

La *capa física* especifica cómo se colocan los bits en el medio conductor para conformar los símbolos (ej, comienzo y fin del paquete) que serán enviados (Fig. 3-9), también determina las señales a utilizar y el método de sincronización.

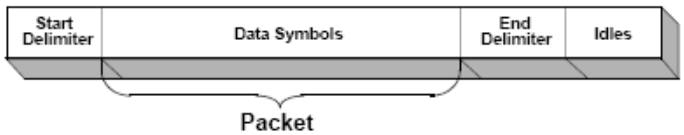


Fig. 3-9 Formato a nivel físico

La *capa de enlace* establece el formato y los protocolos del paquete, su tratamiento y operación (ej. control de flujo y encaminamiento dentro de la subred). A este nivel existen dos tipos de paquetes:

Paquetes de gestión de enlace. Utilizados para negociar parámetros de operación entre puertos y monitorizar la integridad del enlace.

Paquetes de datos. Son los paquetes que transportan operaciones en InfiniBand y están conformados por varias cabeceras que pueden o no estar presentes. (Fig. 3-10)

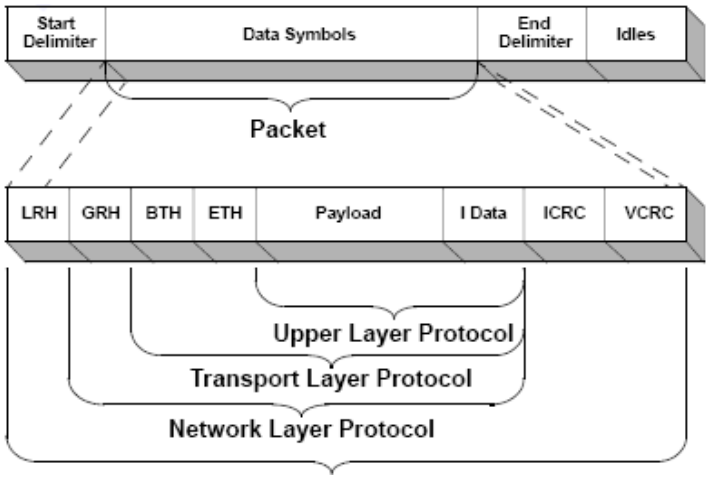


Fig. 3-10 Formato del paquete de datos

La *capa de red*, describe el protocolo para encaminar un paquete entre subredes diferentes, utilizando cabeceras especiales.

La *capa de transporte* es la responsable de la operación de segmentación de los mensajes en varios paquetes cuando la cantidad de datos supera la máxima unidad de transferencia *MTU* de la trayectoria que une los nodos, además configura los puertos para que procesen correctamente los paquetes recibidos y convertirlos nuevamente en mensajes. (Fig. 3-11)

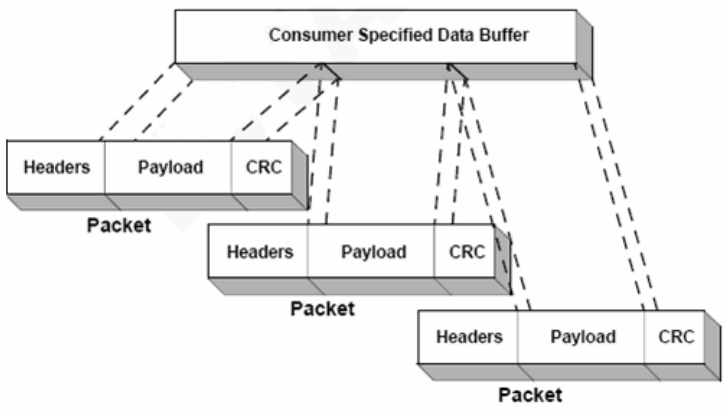


Fig. 3-11 Segmentación del mensaje

Protocolos de Capas Superiores. Según se ve en la Fig. 3-12, *IBA* soporta un gran número de protocolos de capas superiores para varios consumidores de un mismo usuario, junto con mensajes y protocolos para ciertas funciones de gestión. Estos protocolos de gestión se dividen en dos: El gestor de subred y los servicios de subred, ambos con características únicas, que serán descriptas en breve.

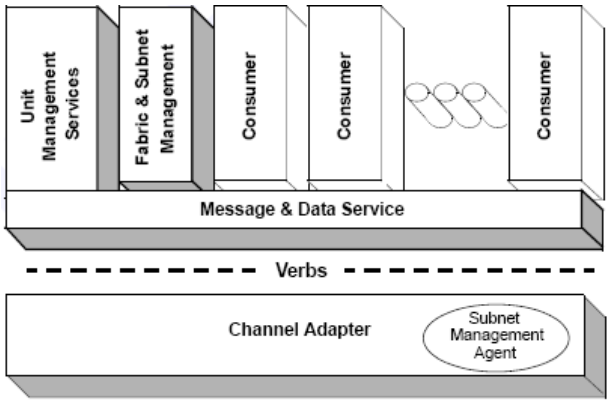


Fig. 3-12 Capas superiores

3.5 Componentes

Los dispositivos en un sistema de IBA se clasifican de la siguiente manera:

- Enlaces y Repetidores
- Adaptadores de canal *(Channels adapters)*
- Conmutadores *(Switches)*
- Encaminadores *(Routers)*

La infraestructura de gestión incluye:

- Agentes generales de servicios
- Gestores de subred

3.5.1 Enlaces y Repetidores

Los enlaces interconectan los adaptadores de canal, conmutadores, repetidores y los dispositivos de encaminamiento para conformar la red de interconexión. Físicamente un enlace puede ser un cable de cobre, un cable óptico, o un circuito impreso o cableado en una placa madre *(backplane)*. Por otra parte, los repetidores son dispositivos transparentes[6] que prolongan las características de propagación de un enlace (Una extensa descripción de la capa física y de los requisitos del repetidor para los varios

[6] Transparente en el sentido que solo implementan el nivel de capa física del protocolo y los nodos no detectan su presencia.

tipos de enlaces, así como los tamaños y dimensiones de los dispositivos de entrada-salida puede encontrarse en [15]).

3.5.2 Adaptadores de Canal

Los adaptadores de canal (Fig. 3-13) son los dispositivos InfiniBand a través de los cuales los nodos de procesamiento y las unidades entrada-salida generan y consumen los paquetes hacia o desde la red. La especificación define dos tipos de adaptadores de canal, a saber: *Host channel adapter* (HCA) y *Target channel adapter* (TCA). El HCA, diseñado para soportar los nodos de procesamiento, proporciona al consumidor una interfase con las funciones especificadas por la capa de verbos, mientras que el TCA diseñado para soportar los nodos entrada-salida, no especifica esta semántica.

La diferencia principal entre el Host channel adapter y el Target channel adapter es la manera en que los consumidores se conectan a la capa de transporte. Específicamente, el HCA soporta la capa de verbos de IBA, mientras que el TCA utiliza un interfaz dependiente de la implementación como interfase a la capa de transporte.

Un adaptador de canal es una entidad programable para controlar las operaciones de acceso directo de memoria, DMA y posee características de protección especiales que permiten que dichas operaciones sean iniciadas tanto local como remotamente.

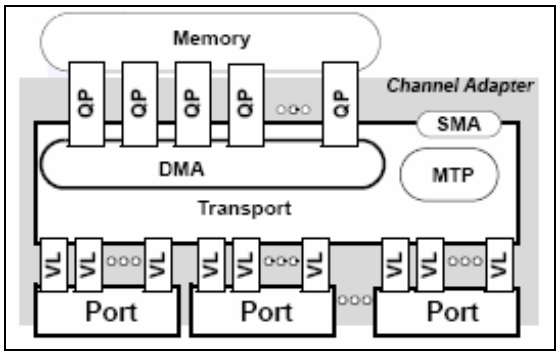


Fig. 3-13 Adaptador de canal

El adaptador de canal esta diseñado para soportar multiplicidad de puertos. A cada uno de estos puertos se le asigna una identificación local (o un rango de identificaciones) denominadas *Local Identifications* o LID´s. Cada puerto tiene su propio conjunto de buffers de transmisión y recepción dotándolos de la capacidad de enviar y recibir paquetes en forma concurrente. A su vez, cada buffer se segmenta en canales o líneas virtuales *(Virtual Lanes, VL)* cada uno con su propio control del flujo.

El adaptador de canal proporciona una mecanismo denominado *Memory Translation & Protection* (MTP) que traduce direcciones virtuales a direcciones físicas, y valida los derechos de acceso a la memoria local de un determinado nodo. Los mecanismos

específicos para la gestión de memoria no están descritos en la arquitectura y los requisitos para tales mecanismos no se especifican para los TCA´s.

El adaptador de canal proporciona una interfase con múltiples instancias de comunicaciones a los consumidores bajo la forma de pares de la cola, (QP) compuestos por una cola de trabajo para el envío y otra para la recepción.

Un potente gestor de subred permite configurar los adaptadores de canal con una o mas direcciones locales para cada puerto físico (LID del puerto). La entidad que se comunica con el gestor de subred, con el fin de configurar el adaptador de canal, se denomina agente de gestión de Subred (SMA) y actúa como interfase para los mensajes entre el nodo y el gestor de subred.

Cada adaptador de canal tiene un identificador global único (GUID) asignado por el fabricante del dispositivo. Debido a que las identificaciones locales asignadas por el gestor de subred no son persistentes (puede ser modificados dentro del mismo ciclo de alimentación), el GUID del adaptador de canal (GUID del nodo) se convierte en el objeto principal para la identificación persistente de un adaptador de canal.

Los adaptadores de canal son los componentes principales de la arquitectura (los conmutadores y encaminadores simplemente controlan el trafico de la red). Cuando un adaptador de canal tiene que mandar información o leerla desde la memoria local de otro adaptador genera una petición a los puertos en forma de paquete de petición *(request packet)* que contiene la dirección del puerto del CA destino.

Una vez que el paquete es inyectado, es guiado dentro de la subred por los conmutadores (*switches*), si el destino esta ubicado en una subred diferente, el paquete deberá atravesar uno o más encaminadores (*routers*) para alcanzarlo.

Ahora que hemos definido el funcionamiento y las características del adaptador de canal, presentamos la definición que la especificación da a los nodos finales *(Endnodes)*[55].
Un *endnode* es cualquier nodo de la red que contiene un adaptador de canal con múltiples pares de colas, que permite establecer conexiones punto a punto y generar mensajes. Los nodos que contiene al *Host Channel Adapter* y al *Target Channel Adapter* son dos tipos específicos de endnodes.

3.5.3 Conmutadores

La función principal de los conmutadores *(switches)* es la dirigir los paquetes basados en la dirección de destino, localizada en la cabecera de los mismos. Existe un caso, en el

que el conmutador genera y consume paquetes, y se presenta cuando el conmutador se configura por el mecanismo de gestión de subred.

Los conmutadores (Fig. 3-14) son el componente fundamental para el encaminamiento dentro de una subred *(intra-subred routing)*, ya que el encaminamiento fuera de la subred es responsabilidad de los encaminadores o routers *(inter-subnet routing*). La interconexión de los conmutadores se compone de enlaces por los cuales se retransmiten los paquetes.

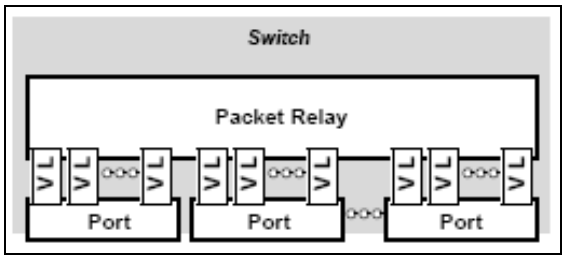


Fig. 3-14 Elementos del conmutador

Cada destino dentro de la subred se configura con uno o más identificadores locales únicos o LID´s. Los paquetes contienen, en su cabecera, la dirección que especifica el LID del puerto del nodo destino, desde el punto de vista del conmutador, el LID del destino representa la trayectoria que seguirá el mensaje a través del mismo. Los elementos del conmutador son configurados en función de tablas de encaminamiento *(forwarding tables)* y los paquetes se remiten individualmente dentro de un conmutador hacia un puerto o puertos de salida basados en el LID destino de la cabecera del paquete y en la tabla de encaminamiento.

Las comunicaciones *unicast* y *multicast* están implementadas en el conmutador, la primera consiste en la entrega de un paquete a un solo nodo destino, mientras que la segunda representa la capacidad de la red de entregar un solo paquete a múltiples nodos.

El gestor de subred es el responsable de entregar y configurar las tablas de encaminamiento a los conmutadores.

Para maximizar características tales como disponibilidad, redundancia, tolerancia a fallos y balance de carga, es posible dotar a los conmutadores de trayectorias múltiples entre los diferentes pares de nodos. El gestor de subred puede utilizar estas trayectorias para reencaminar los paquetes alrededor de enlaces en presencia de fallos, recalculando y entregando las tablas de encaminamiento corregidas a los conmutadores en el área afectada de la red.

Los conmutadores son transparentes a los nodos de la red y no se pueden direccionarse directamente (a excepción de operaciones de gestión de subred). Por otra parte los

conmutadores de InfiniBand soportan un estilo de encaminamiento fuente, denominado encaminamiento directo *(direct routing)*, utilizado para encaminar paquetes del gestor de subred. Ésto permite la configuración de una subred sin entradas válidas en la tabla de encaminamiento (situación típica que se da en la fase de *startup* de la red). La entidad que se comunica con el SM para propósitos de configuración dentro del conmutador se denomina agente del gestor de *Subred (Subnet Manager Agent, SMA).* Es obligatorio que cada conmutador de la red tenga un SMA.

3.5.3.1 El Puerto 0 del Conmutador

Además de los puertos externos *(IBA provee un máximo de 254 puertos de datos posibles)* que puede tener un conmutador InfiniBand, existe un puerto adicional y especial denominado *puerto 0*, también llamado *puerto de gestión*. Es un tipo de puerto único dentro de la especificación, diferente de los otros puertos de IB (que son los puertos externos del conmutador, los puertos del CA, o los puertos del encaminador). El puerto 0, es un puerto virtual, pues no se requiere su implementación física dentro del conmutador y es utilizado para comunicarse con el gestor de subred, a fin de configurar el conmutador, asignarle el identificador local, las tablas de encaminamiento, etcétera.

3.5.4 Encaminadores

Al igual que los conmutadores, los encaminadores (Fig. 3-15) no generan ni consumen paquetes (a excepción de paquetes de gestión), simplemente los paquetes hacia el destino, y lo hacen basándose en la información de la cabecera de encaminamiento global del paquete *(Global Routing Header GRH)*. Los encaminadores son el componente fundamental para el encaminamiento entre subredes *(inter-subnet routing).*

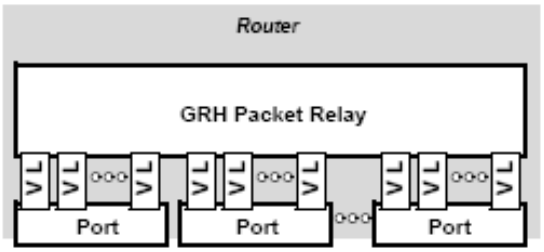


Fig. 3-15 Elementos del encaminador

Los encaminadores no son totalmente transparentes a los endnodes debido a que el nodo fuente debe especificar el LID del encaminador, junto al GID *(Global Identifier, GID)* del destino. Cada subred se identifica únicamente con un identificador de subred conocida como prefijo *(Subnet Prefix)*.

El gestor de subred es el encargado de programar todos los puertos con el prefijo de Subred correspondiente. Por otra parte, dicho gestor configura al encaminador con información importante como canales virtuales utilizables, particiones, etc.…

El prefijo de subred, que forma parte del GID, representa la trayectoria a seguir en el encaminador. Cada paquete se redirecciona dentro del encaminador hacia uno o más puerto de salida basados en el GID destino (localizado en su cabecera) y la tabla de encaminamiento del *router*. De esta manera, los paquetes son reenviados entre routers hasta alcanzar la subred que contiene al nodo destino, donde el paquete termina de encaminarse utilizando el LID destino.

Con el fin de maximizar la disponibilidad de la red, es posible utilizar multiplicidad de trayectorias entre las distintas subredes. De esta forma, los encaminadores pueden utilizar estas trayectorias múltiples con el propósito de balancear la carga de tráfico o reencaminar paquetes alrededor de una subred en fallo.

Los encaminadores operan en la capa de red *(network layer)* del protocolo InfiniBand, que establece la jerarquía para interconectar múltiples subredes (Fig. 3-16). El uso de los encaminadores esta orientado a satisfacer los siguientes requerimientos:

- Escalabilidad
- Reutilización del espacio local de direcciones
- Contención de fallos y cambios de topología
- Confinamiento de la gestión de red a subredes independientes

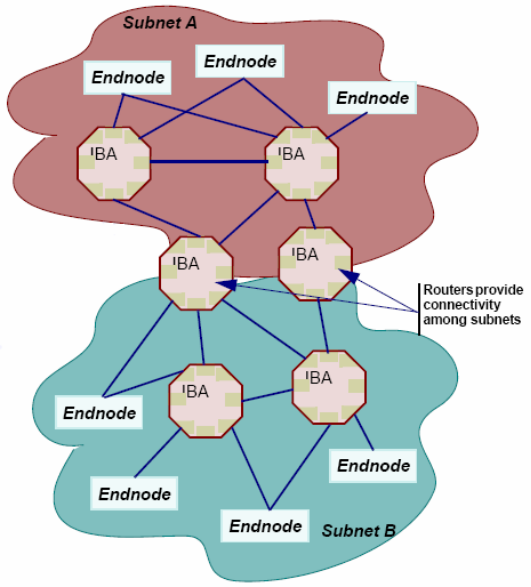


Fig. 3-16 Interconexión entre subredes

Los encaminadores soportan tanto comunicaciones unicast como multicast, y encaminan los paquetes usando encaminamiento basado en destino *(destination based routing)*, en

el cual a todos y cada uno de los puertos de la red se asignan, uno o más identificadores globales *GIDs*. Desde es punto de vista del encaminador, el GID representa tanto un puerto de un nodo como el puerto de otro encaminador dentro de la subred a la que esta conectado, aunque no necesariamente representa una trayectoria a través de la red, debido a que un encaminador puede distribuir el trafico entre varias trayectorias, según la cabecera del paquete.

El encaminador es visible a los dispositivos conectados directamente a la subred y es transparente a los conectados en subredes remotas. La función de resolución de direcciones del gestor de subred, es la que hace visible al encaminador y los nodos utilizan la información de dicha función, para direccionar los paquetes al encaminador local en su camino al destino remoto. Si existen dos o más encaminadores conectados en la misma subred serán visibles entre ellos, con el objetivo de permitir la implementación del protocolo de encaminamiento. Por ultimo, ha de mencionarse que también son visibles al gestor que reside en la subred a la que esta conectado el encaminador.

Hasta aquí se han visto las características generales, que define la especificación y los componentes principales que conforman las topologías de la red de interconexión. A continuación se describe el versátil conjunto de componentes de gestión, que permiten configurar y monitorizar la red de interconexión. Estos componentes posibilitan la aplicación de cada una de las fases (la detección, la notificación y la acción correctiva), del balanceo distribuido del encaminamiento.

3.6 Componentes de Gestión

El modelo de gestión de InfiniBand provee un gestor de subred y una infraestructura que da soporte a un vasto número de servicios de gestión generales (Fig. 3-17). La infraestructura de gestión requiere la existencia de un agente de gestión de *subnet* en cada nodo y define una interfase general de servicios que permite la implementación de agentes de servicio general adicionales.

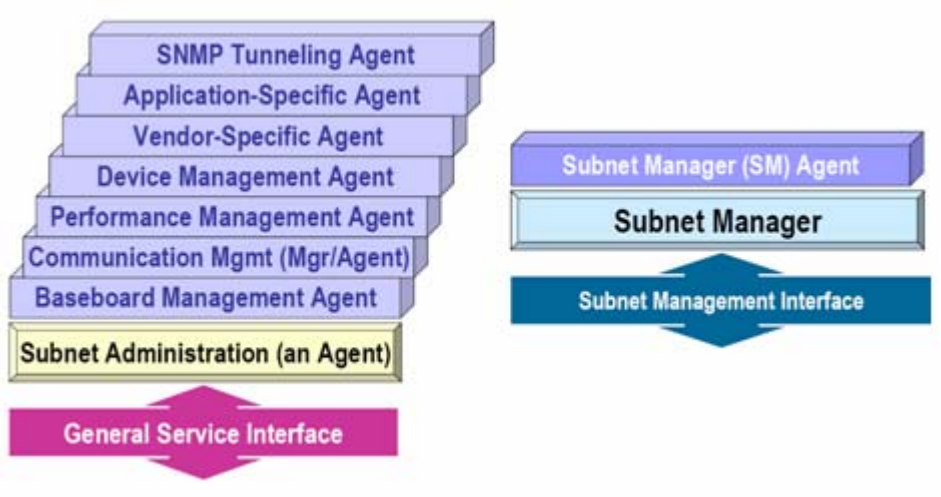


Fig. 3-17 Modelo de gestión

La arquitectura define un datagrama de gestión *(management datagram, MAD)* como la estructura de los mensajes común a todos los servicios, para establecer la comunicación entre los gestores y los agentes de gestión.

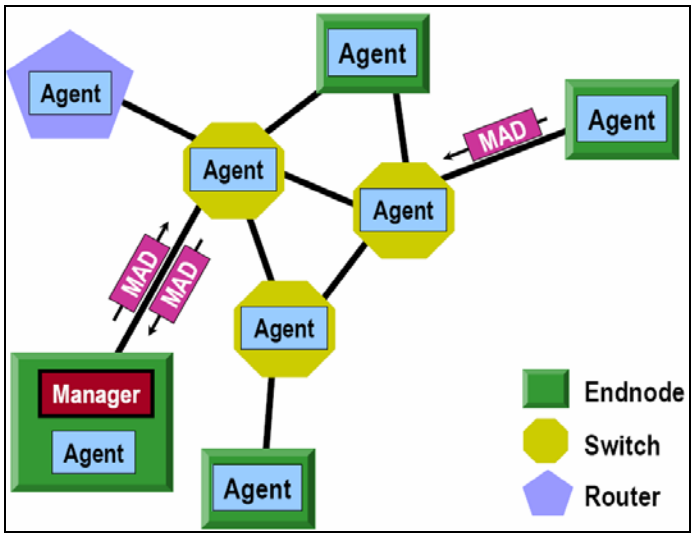


Fig. 3-18 Comunicación entre elementos de gestión

En la Fig. 3-18 se muestran los nodos de la red *(Endnodes, switchs y routers)*, con los agentes de gestión y el gestor de subred. Todos los componentes de gestión se comunican entre ellos mediante los datagramas, de esta manera se recolecta la información y se configura toda la red InfiniBand. Se debe destacar que los componentes de gestión, se definen como entidades en la especificación y pueden estar implementadas tanto en software, como en hardware.

3.6.1 Gestor de Subred (Subnet Manager, SM)

Es la unidad responsable, dentro de la subred, de configurar y gestionar los conmutadores, encaminadores y adaptadores de canal. Puede estar implementado, tanto en software como en hardware, dentro del adaptador de canal o el conmutador.

InfiniBand soporta el concepto de gestores de red múltiples dentro de la subred y especifica cómo se establece la negociación para que solo uno sea establecido como *Master* aunque no prohíbe otros tipos de cooperación para establecer las relaciones entre el master y los secundarios *(standby managers)*. El subnet manager master es el encargado de:

- Descubrir la topología de la subred
- Configurar cada puerto del adaptador de canal con un rango valido de LID's, GID's, prefijos y Claves de seguridad *(P_Keys)*.
- Configurar cada conmutador con un LID, un prefijo de subred, y la base de datos utilizada por las tablas de encaminamiento.
- Mantener y gestionar las bases de datos de los servicios de gestión y de los nodos del servicio en la subred y proporcionar la resolución de nombres LID/GID.

3.6.2 Agentes de Gestión de Subred (Subnet Manager Agent, SMA)

Cada nodo de la red debe contener un agente de gestión de Subred (SMA) que proporciones acceso al SM a través de una interfase denominada interfase de gestión de Subred (*Subnet Manager Interfase,* SMI). El SMI permite a los paquetes encaminados según el destino *(LID Routed Packets)* y a los paquetes encaminados directamente *(Direct routed packets)* acceder al SMA o al SM según el caso. El encaminamiento directo *(Direct routing)* proporciona una forma de comunicación antes que se configuren los elementos de la red, (ej. Puesta en marcha de la red). Una vez que cada nodo y cada conmutador de la red han sido configurados se utilizara el encaminamiento basado en el destino.

3.6.3 Agentes de Servicio General

Cada nodo de la red está dotado de la capacidad de soportar agentes adicionales de gestión denominados Agentes de Servicio General *(General Service Agents, GSA)* a los que puede accederse a través de la interfase denominada interfase de servicio general

(General Service Interfase, GSI), que solo soporta el encaminamiento basado en LID. Las clases de servicio general *(General services)* definidas por InfiniBand son:

- Administración de Subred *(Subnet Administration, SA)* – Este servicio es proporcionado por el SM y permite que los nodos tengan acceso a la información de la subred para descubrir otros nodos activos y/o servicios ofrecidos.

- Gestor de rendimiento *(Performance Management, PM)* – Este servicio es utilizado para monitorizar el estado de los componentes y generar reportes basado en métricas bien definidas utilizando elementos conocidos como contadores de performance.

- Gestor de la Placa Base *(Baseboard Management, BM)* – Provee todo lo relacionado a la gestión del chasis mediante el gestor de enlaces de la especificación *(InfiniBand Management Link, IB-ML)* definido en [15].

- SNMP Tunneling – Proporciona la funcionalidad para definir el método de envío y recepción de mensajes SNMP *(Simple Network Management Protocol)*, que permite la circulación de paquetes de otros estándares o interfaces propietarias.

- Definidos por el fabricantes *(Vendor defined)* – Define extensiones privadas que el fabricante puede utilizar con el objetivo de gestionar y configurar remotamente sus dispositivos.

- Gestor de comunicaciones *(Communication Manager, ComMgt)* – provee al sistema la capacidad de establecer conexiones y otras funciones de gestión entre nodos.

- Gestor de dispositivos *(Device Management, DevMgt)* – Proporciona la gestión de los recursos entrada-salida.

3.6.4 Infraestructura de Gestión en InfiniBand

Los componentes de gestión enumerados en el punto anterior, conforman la infraestructura de gestión de redes InfiniBand, que puede dividirse en dos partes principales:

- Gestión de subred *(Subnet management)*, que proporciona los *métodos* necesarios para que el gestor de subred descubra y configure los dispositivos InfiniBand y gestiones el funcionamiento de la red. En la Fig.

3-19 se muestra el modelo lógico y jerárquico del gestor de subred, y también el modelo físico que deriva en su implementación.

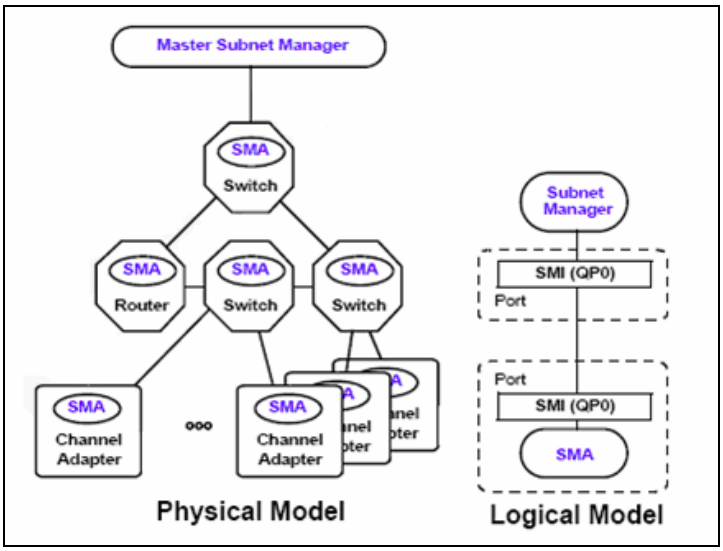


Fig. 3-19 Gestor de subred

- Servicios generales de gestión *(General management services)*, la Fig. 3-20 muestra la distribución lógica de los servicios generales dentro de la subred InfiniBand, conformado por los servicios de:

  - Administración de Subred
  - Gestión de rendimiento
  - Gestión de la Placa base
  - SNMP Tunneling
  - Definidos por el fabricantes
  - Gestión de comunicaciones
  - Gestión de dispositivos

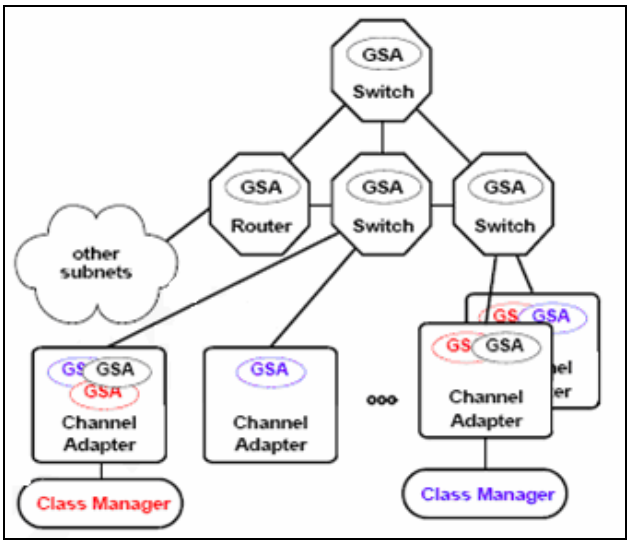


Fig. 3-20 Modelo del gestor de servicios generales

La infraestructura de gestión define las interfaces y los principios de funcionamiento que permiten a los dispositivos descubrir, inicializar y controlar la red, estableciendo un modelo y un marco de gestión común, aplicable a todos los elementos de IBA, identificando esos elementos, y definiendo sus características.

Las características de gestión de InfiniBand involucran una variedad de conceptos, entre los que se incluyen:

- La manera de configurar y recolectar la información de los nodos, conmutadores y encaminadores de la red.
- Un marco de diagnóstico como mecanismo común de gestión de errores.
- Servicios de instalación y configuración que permiten la inicialización de la red y el descubrimiento de la topología.
- Un formato estándar para los paquetes de gestión denominado *datagrama de gestión* a (MAD).
- Un formato de paquetes, que conforma un subconjunto de los MAD's denominados paquetes de gestión de subred *(Subnet Management Packets, SMP)* que permiten establecer la comunicación *(*mediante operaciones *get* y *set)* entre el gestor de subred y los dispositivos de la misma.
- Un formato de paquetes, que conforma el resto de los MAD's denominados paquetes de gestión general *(General Management Packets, GMP)*, que permiten establecer las operaciones de gestión de servicios a los dispositivos de la red.

3.6.5 Interfases de Gestión (SMI, GMI).

La interfase de gestión se realiza a través de dos pares de colas (QP). El par QP0 es reservado para la gestión de subred y el par QP1 se designa a los servicios generales de gestión de la gerencia general.

Cada puerto de la red InfiniBand tiene un par QP dedicado a la gestión de subred. Éste par tiene características especiales que lo hacen único con respecto al resto de pares de colas. El par de colas del gestor de subred QP0 tiene las siguientes características:

- Esta permanentemente configurado para recibir datagramas (de la clase *unreliable*) MADs.
- Cada puerto de un dispositivo esta obligado a tener un par de colas QP0 capaz de enviar y recibir paquetes.
- El par QP0 es miembro de todas las particiones (es decir, puede aceptar cualquier paquete que especifica cualquier partición de la red).
- Solamente los paquetes de la gestión (SMP's) son válidos y serán aceptados en el par QP0.
- El tráfico dirigido a este par utiliza exclusivamente el canal virtual *(VL)* 15, que no está sujeto a control de flujo a nivel de enlace.

Las características del par de colas QP1, utilizadas por los servicios generales son similares a las descriptas anteriormente.

3.7 Características de InfiniBand

En este punto, se comentan las características sobresalientes de la arquitectura, tanto para el establecimiento de las conexiones, como para la configuración de los dispositivos y el nivel de servicio.

3.7.1 Pares de Colas (QP's)

Los pares de colas son la interfase virtual, proporcionada por el hardware a un consumidor de InfiniBand. La arquitectura soporta hasta 224 QP's por adaptador de canal y la operación en cada QP es independiente de la operación en las otras, debido al alto grado de aislamiento y protección respecto a la operación en los pares y nodos restantes. De esta manera un par QP puede ser considerado como un recurso privado,

asignado a un solo consumidor. A su vez cada consumidor puede consumir múltiples pares QP's según se muestra en la Fig. 3-21.

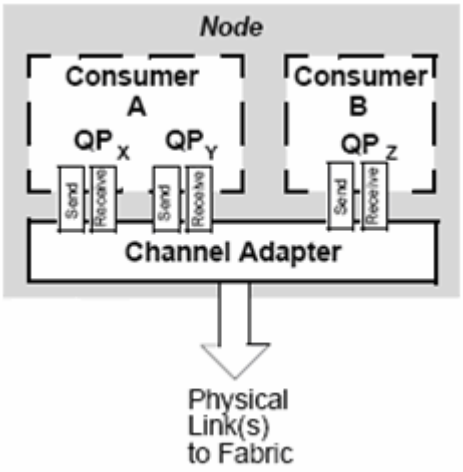


Fig. 3-21 Interfase de comunicación

El consumidor crea un puerto de comunicación virtual asignando un par de colas QP y especificando su nivel de servicio. La comunicación se establece entre un par de colas fuente y un par de colas destino. Para el *servicio orientado a conexión*, cada QP está limitado estricta y exactamente a par QP, generalmente en un nodo diferente. El consumidor inicia el establecimiento de conexión necesario para asignar QP del nodo destino y configura (con información como LID destino, nivel de servicio y limites de operación) el contexto, en el que el QP tendrá validez.

En funcionamiento normal, el consumidor colocara pedidos de trabajo en la cola para establecer la comunicación entre nodos.

3.7.2 Canales Virtuales

Los canales virtuales (VL) proporcionan un mecanismo para la creación de múltiples enlaces virtuales dentro de un solo enlace físico (Fig. 3-22).

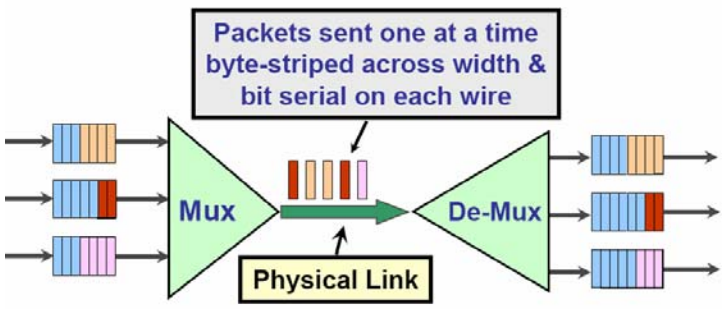


Fig. 3-22 Canales Virtuales

Un canal virtual representa un conjunto de buffers de transmisión y recepción en un puerto determinado, a su vez, todos los puertos soportan el canal virtual denominado *VL15* que esta reservado exclusivamente a la comunicación con el gestor de subred. En total existen 15 canales virtuales de datos *(VL0 a VL14)*, (Fig. 3-23) y todos los puertos deben soportar al menos uno (VL0).

Los canales virtuales que un puerto utiliza están configurados por el SM, basados en el nivel de servicio *(Service Level, SL)* del puerto, característica que está contenida en un campo de la cabecera del paquete. Por defecto se utiliza el canal virtual VL0, hasta que el SM determina el número de VL's soportados a ambos lados del enlace y configura el nivel de servicio del puerto a través de la tabla *SLtoVL mapping Table*, como se verá mas adelante.

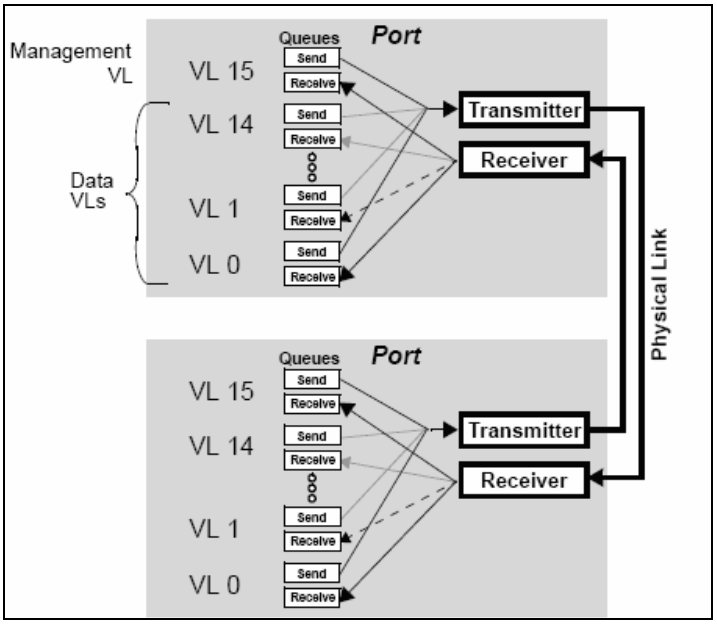


Fig. 3-23 Conexión mediante canales virtuales

Cada puerto mantiene un control de flujo separado sobre cada canal virtual de datos, de tal manera que un exceso de tráfico en un VL no bloquea el tráfico en otro. La asignación de VL's se realiza en los puertos localizados a los extremos de un enlace y es independiente de las asignaciones en los otros enlaces de la red.

Como se ha mencionado, cada paquete tiene especificado un nivel de servicio *(SL)* especificado en su cabecera. A medida que el paquete atraviesa la red, el valor de dicho nivel de servicio determina qué canal virtual ha de ser utilizado en cada enlace. Para ésto, cada puerto contiene una tabla de conversión de nivel de servicio a canal virtual (*SltoVL mapping Table)* que permite el envío del paquete por el VL apropiado.

3.7.3 Calidad de Servicio

IBA proporciona varios mecanismos que permitan a un gestor de subred administrar y garantizar diferentes niveles de calidad de servicio. Estos mecanismos son denominados Nivel de servicio *(Service level)*, Conversión de Nivel de Servicio a Canal Virtual *(Service Level to Virtual Lane Mapping)*, y por ultimo particiones *(Partitions)*.

- *Nivel de servicio (Service level).*
  InfiniBand define el atributo denominado nivel de servicio (SL) que permite a un determinado paquete operar con 16 tipos de niveles. La definición y el propósito de cada nivel de servicio esta fuera del alcance de la arquitectura y queda determinada por las políticas de administración de la red. De esta manera, la asignación de los niveles de servicio, es una función del gestor de comunicaciones del nodo y su negociación con el gestor de subred.

- *Conversión de Nivel de Servicio a Canal Virtual (Service Level to Virtual Lane Mapping).*
  Otro mecanismo que está ligado al nivel de servicio, es el de los canales virtuales. Cada paquete identifica su SL a medida que atraviesa la red, el SL almacenado en el paquete determina qué VL se utilizará en el enlace siguiente. Con este fin, cada puerto de la red (examinadores y nodos) tiene una tabla de conversión de nivel de servicio a canal virtual (Fig. 3-24) que es configurada por el gestor de subred. Naturalmente, para todos los enlaces que terminan en un puerto que soporte solamente un canal virtual de datos, todos los niveles de servicio serán asignados a ese canal virtual (SL's a VL0). De otro modo (es decir, si disponemos de varios canales virtuales), la política de gestión de la subred será quien determine la asignación de cada nivel de servicio al canal virtual disponible. Los paquetes del gestor de subred (SMP) enviados al par de colas QP0 y que usan exclusivamente el canal virtual VL15, ignoran el nivel de servicio, debido a que el canal virtual mencionado no es un canal de datos.

- *Particiones.*
  La capacidad de asignar particiones también esta ligada con el nivel de servicio. El administrador de la red puede asignar cierto nivel de servicio a determinadas particiones. Esto permite que el SM aísle la circulación de paquetes entre esas particiones aunque ambas particiones funcionen con el mismo nivel de QoS, de modo que puede garantizarse a cada partición una justa distribución del ancho de banda sin importar el comportamiento de los otros nodos.

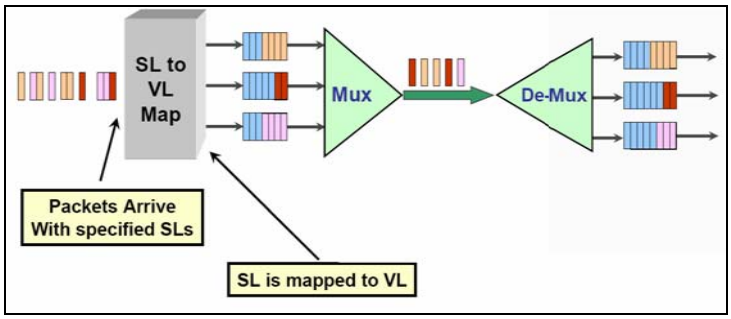


Fig. 3-24 Tabla de conversión

3.7.4 Control Estático de la Velocidad de Inyección

InfiniBand define varias velocidades de transmisión para los enlaces. La velocidad de transmisión mas baja es de 2.5 Gb/sec, a este enlaces se le denomina 1x (veces). Otras velocidades posibles son 10 Gb/sec (4x) y 30 Gb/sec (12x). Para soportar velocidades múltiples en los enlaces de la red, InfiniBand define un mecanismo estático de control de inyección *(Static Rate Control)* que impide que un puerto con un enlace de alta velocidad sobrepase la capacidad de otro puerto con un enlace de velocidad más baja.

Como parte del proceso de descubrimiento de trayectorias, el gestor de subred provee información al nodo sobre la máxima unidad transferencia *(MTU)* y la velocidad de transmisión para dicha trayectoria. La información de las trayectorias se utiliza debido a que el puerto del conmutador o el nodo podrían ser el cuello de botella. El ejemplo ilustrado en la Fig. 3-25, muestra el puerto (A) de un nodo con una velocidad de enlace 12x, que tiene el potencial para inyectar un tráfico tres veces mayor a la capacidad del puerto B y doce veces la capacidad de los puertos C, D, o E y el puerto B tiene además el potencial para inyectar tráfico cuatro veces mayor a la capacidad de los mismos puertos C, D y E. Debido a que el tráfico tiende para ser adverso *(bursty)* cada vez que el puerto A envía a uno de los otros puertos, la red tiene una alta probabilidad de entrar en estado de congestión. El control de flujo del enlace, evita que la red descarte los paquetes que ingresan a la zona en congestión, pero la inyección sostenida *(back pressure)* afectará a otras trayectorias que, de otra manera, no estarían congestionadas.

InfiniBand soluciona este problema definiendo un mecanismo estático para el control de velocidad en los puertos que funcionan a velocidades de enlace mayor que 1x.

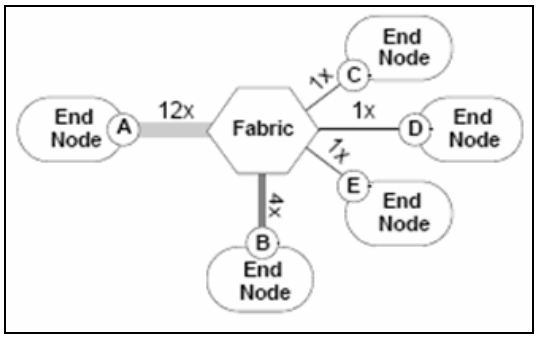


Fig. 3-25 Ejemplo de control de inyección

Cada puerto destino tiene asociado un valor de *time-out*, que se calcula con el cociente entre las velocidades de transmisión de los puertos fuente y destino (cuando las velocidades son iguales, el valor de *time-out* es cero no se necesita control). De otra manera, cuando el puerto transmite un paquete a un puerto destino, asigna en dicho paquete el LID del destino correspondiente y un valor de *time-out*, ubicado en la tabla de control de velocidad estática *(static rate control table)*.

Mientras siga existiendo dicho valor en la tabla ningún paquete se envía a ese destino (pudiendo reencaminarlo hacia otro destino). Después de transcurrido el período de time-out, el puerto quita el valor de la tabla y transmite el paquete colocándolo en la cola del canal virtual de salida correspondiente.

3.7.5 Asignación de Nombres (Addressing)

Cada nodo puede contener uno o más adaptadores de canal con varios puertos, y cada puerto, a su vez, puede contener varios pares de colas. El adaptador de canal asigna a cada par de colas un número (QPN) que lo identifica unívocamente. Los paquetes contienen el QPN del par de colas del nodo destino y cuando llegan al CA se utilizan para procesar paquete. Como hemos visto, cada puerto tiene un identificador local (LID) asignado por el gestor de subred, en la etapa de descubrimiento, y dentro de la subred estos valores son únicos. Los conmutadores utilizan los LID´s para encaminar los paquetes dentro de la subred, mediante el uso de tablas de encaminamiento y la posición del puerto respecto al conmutador. Cada paquete contiene, entre otros, un campo con información del LID fuente (SLID), que identifica el puerto que inyectó el paquete en la subred y un campo con el LID destino (DLID) que identifica el puerto donde este paquete debe ser entregado.

IBA también tiene la capacidad de soportar puertos virtuales múltiples dentro de un puerto físico mediante la definición de una máscara de control de LID *(LID mask control, LMC)*. La mascara LMC específica el número de bits menos significativos del LID que el puerto debe omitir (enmascarar) cuando realiza la validación del campo DLID del paquete que ingresa en el puerto. El conmutador, sin embargo, no ignora estos bits, de esta manera el gestor de subred programa diferentes trayectorias a través de la

red, para el mismo par de nodos fuente-destino. Con esta característica, cada puerto representa 2^{LMC} puertos virtuales que pueden utilizarse para encaminar paquetes a través de la red.

Además de esta asignación local de nombres (dentro de la subred), cada puerto está dotado también de una identificación global y única (GID) que tiene el mismo formato que una dirección IPv6. Los paquetes contienen una cabecera global de encaminamiento *(Global Route Header, GRH)* que es opcional y especifica tanto el puerto que inyectó el paquete en la red (el valor del GID fuente, SGID) como el puerto donde ese paquete debe ser entregado *(GID destino, DGID)*. Esta cabecera se utiliza sólo por los encaminadores para retransmitir los paquetes entre subredes, mientras que los conmutadores no la tienen en cuenta debido a que utilizan sólo la información local residente en el campo LRH (Local Routing Header).

Los nombres (LID, GID) que han sido descriptos en los párrafos anteriores, son asignados dinámicamente por el gestor de subred. Además de éstos, existe un tercer y ultimo nombre especificado por el fabricante, denominado Identificador Global Único *(Globally Unique Identifier, GUID)*, que es asignado, en forma estática, al adaptador de canal y a sus puertos. De tal forma que el GUID combinado con el prefijo de subred conformaran el GID de un determinado puerto y la combinación entre la dirección del puerto (GID + LID) y el QPN conformaran la dirección del par de colas correspondiente QP.

3.8 Servicios de Transporte

Los mecanismos del transporte provistos por InfiniBand proporcionan múltiples clases de servicios de comunicación. Cuando se crea un par de colas, se configura para proporcionar alguna de los siguientes servicios de transporte:

- **Conexión confiable** *(Reliable Connection)*
- **Datagrama confiable** *(Reliable Datagram)*
- **Conexión no confiable** *(Unreliable Connection)*
- **Datagrama no confiable** *(Unreliable Datagram)*
- **Raw Datagram**

El servicio de conexión confiable *(Reliable Connection)* asocia un par de colas local con otro remoto, unívocamente. De esta manera cada consumidor debe crear un par de colas nuevo cada vez que desee comunicarse con un consumidor remoto diferente (Fig. 3-26).

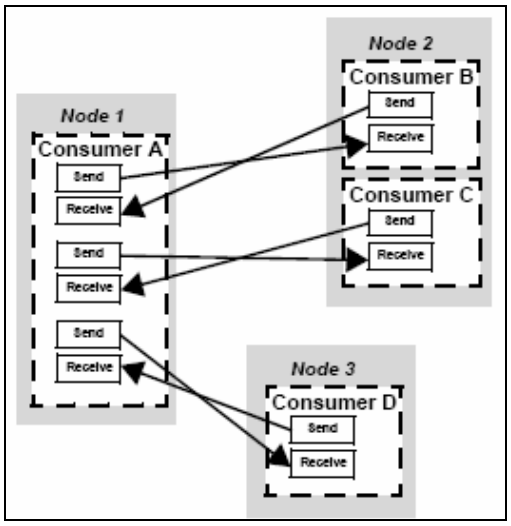


Fig. 3-26 Servicio orientado a conexión

El servicio es confiable porque el adaptador de canal puede establecer secuencias de números y mensajes de reconocimiento para todos los mensajes. De esta manera, los consumidores mantienen comunicaciones confiables aún en presencia de errores, saturación de buffers, congestión y fallos en la red (si existen trayectorias alternativas). Los mensajes de reconocimiento son utilizados para indicar que el dato ha sido recibido en el nodo destino correctamente.

El servicio de conexión no confiable *(Unreliable Connection)* también asocia un par de colas local con sólo uno remoto, pero a diferencia del servicio de conexión confiable, no utiliza mensajes de reconocimiento, perdiendo la habilidad de reenviar los paquetes perdidos o corruptos, aunque mejorando la eficiencia de los enlaces debido a que no se utiliza ancho de banda para el envío de reconocimientos.

El servicio de comunicaciones mediante datagrama no confiable *(Unreliable Datagram)*, permite establecer comunicación sin reservar conexión ni generar mensajes de reconocimiento y permite al par de colas de un consumidor determinado, comunicarse con cualquier otro par en cualquier nodo de la red (Fig. 3-27). Este servicio mejora notablemente la escalabilidad de la arquitectura y es la clase de servicio utilizada por el gestor de subred para descubrir e incorporar nuevos nodos a la red.

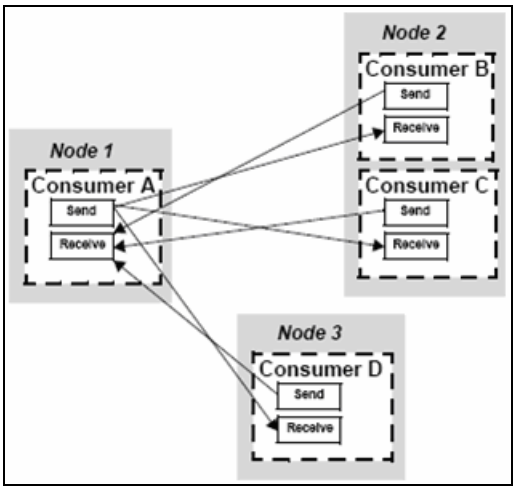


Fig. 3-27 Servicio orientado a datagramas

El servicio de comunicaciones mediante datagrama confiable *(Reliable Datagram)*, ilustrado en la Fig. 3-28 permite establecer comunicaciones multiplexadas entre nodos mediante el concepto de contexto punto a punto *(End-to-end contexts, EEC)* que permite a un determinado par de colas comunicarse con cualquier otro par de la red de manera confiable debido al uso de secuencias de numeración y mensajes de reconocimiento.

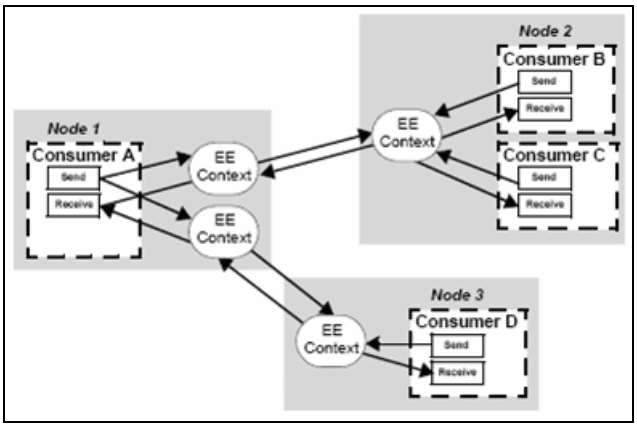


Fig. 3-28 Datagrama confiable

El servicio conocido como *Raw Datagram* no es técnicamente un servicio de transporte, sino que consiste de un servicio de enlace que permite el envío de paquetes cuyo formato no esta definido en el estándar. En la arquitectura existen dos servicios de este tipo conocidos como: EtherType e IPv6 que básicamente permiten el uso de múltiples protocolos estándar, no InfiniBand, como es el caso de TCP, UDP, IPv4 e IPv6.

En este capitulo se ha realizado una descripción general que permite conocer los mecanismos de comunicación y el establecimiento de conexiones, los componentes que conforman la red, los componentes de gestión, y los servicios de transporte de datos.

Los aspectos mencionados aquí, son utilizados con el fin de llevar a cabo la implementación del balanceo distribuido del encaminamiento sobre la arquitectura IBA. La adaptación realizada sobre DRB para su aplicación en el estándar, se expone en el capitulo siguiente.

Capitulo 4 Control de congestión en redes InfiniBand

4.1 Introducción

En este capitulo se enuncian las facilidades descritas, en la especificación InfiniBand [15] para el control de congestión y se analizan las principales desventajas de dicho mecanismo. Estas desventajas son las que motivan la investigación de nuevas soluciones. Por otra parte, presentaremos nuestra propuesta para el control de congestión, la cual soluciona estas desventajas y mejora las prestaciones de las redes diseñadas sobre la arquitectura en estudio.

Asimismo se describen las modificaciones hechas sobre el algoritmo de encaminamiento *Distribuited Routing Balancing*, propuesto para mejorar las prestaciones de la red de interconexión, que permiten su implementación manteniendo la compatibilidad completa con la especificación IBA.

Es importante destacar que la implementación realizada en este trabajo de investigación, no requiere que se realice modificación alguna sobre la arquitectura de la especificación IBA, y respeta todos los aspectos definidos en ésta. Por esta razón, se utilizan una serie de características especificadas en InfiniBand, como la posibilidad de establecimiento de trayectorias múltiples, los mecanismos de monitorización de recursos y la posibilidad de notificación mediante mensajes de reconocimiento, para aplicar un mecanismo de control de congestión eficiente, utilizando un concepto que permite un alto grado de utilización de los enlaces y un bajo valor de latencia de transporte en los mensajes.

4.2 Facilidades para el control de congestión en la especificación

El método propuesto en la especificación es opcional, y deja abiertos varios aspectos del modo de operación como veremos mas adelante en este capítulo. El mecanismo de control de congestión, está implementado dentro del agente de gestión denominado *congestión control agent*, *CCA*, y consiste en un proceso de tres fases, que permite detectar la congestión, notificarla y regular la inyección de mensajes.

La Fig. 4-1, muestra los eventos que ocurren en los componentes de red cuando éstos entran en la zona de congestión [44].

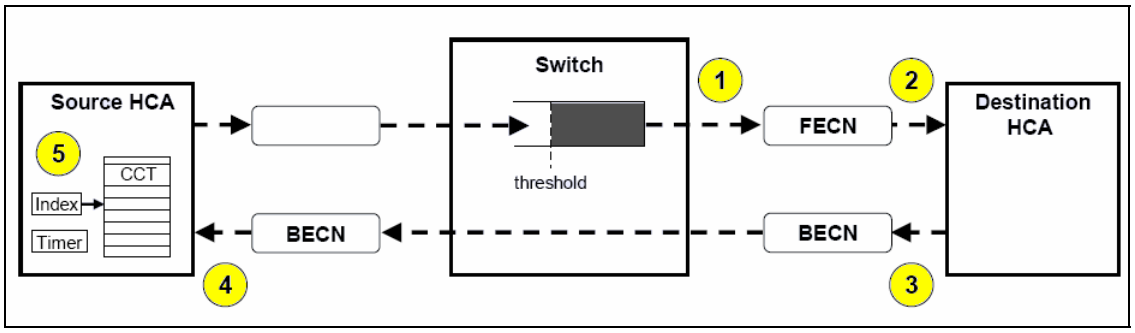


Fig. 4-1 Facilidades ofrecidas por InfiniBand

Mediante el uso de elementos de medición *(performance counters)* cada conmutador conoce, en todo momento, el estado de ocupación en los buffers de cada canal virtual, en todos y cada uno de los puertos que lo componen.

La especificación propone establecer un valor (umbral) de ocupación de los buffers, por encima del cual se activen los mecanismos destinados al control de congestión. El punto 1 de la Fig. 4-1, muestra esta situación.

Cuando se detecta congestión, el conmutador informa de esta situación marcando los paquetes que están situados en el buffer del canal virtual que ha superado el umbral. Dentro de la cabecera de transporte *(transport header)* de todos los paquetes, existe un bit destinado a tal efecto. Este bit es denominado *Forward Explicit Congestión Notification (FECN).*

Cuando el valor almacenado en este campo es un *"uno"*, se interpreta que el paquete ha seguido una trayectoria dentro de la cual, algún enlace sufre congestión y debe ser atendido para evitar esta situación. Por otra parte si el valor de este campo es *"cero"*, el volumen de comunicación que es inyectado en esta trayectoria puede ser manejado correctamente por sus recursos.

Una vez que el paquete es marcado, se reenvía por el puerto correspondiente en función la trayectoria especificada. Eventualmente cada paquete alcanza el adaptador de canal del nodo destino, donde se examina la cabecera que contiene el bit de notificación, como se observa en el punto 2 de la Fig. 4-1.

Si el nodo destino recibe un paquete marcado, el agente de control de congestión de dicho nodo solicita el envío de un mensaje de notificación *(congestión Notification, CN),* con el objetivo de informar al nodo fuente la existencia de congestión en la trayectoria establecida entre ambos nodos. La notificación se hace efectiva mediante el uso de otro bit en la cabecera de transporte del mensaje CN, conocido como *Backward Explicit congestión Notification (BECN).* En el punto 3 de la Fig. 4-1, se muestra el envío del paquete que contiene este bit.

Cuando el valor almacenado en este campo es un *"uno"*, se interpreta que el paquete ha sido enviado desde el nodo destino para informar que existe congestión en el enlace y debe realizarse una acción correctiva.

Según se ha mencionado en el capitulo 3, existen diferentes servicios de transporte para la comunicación de datos. En el caso de comunicaciones confiables *(Reliable connections)* cada paquete de datos recibido, debe estar acompañado de uno de reconocimiento. Por lo tanto, en ausencia de congestión en agente de control debe poner a *"cero"* el campo *BECN,* en la cabecera de trasporte, para evitar que se active el mecanismo de control de congestión, cuando no es necesario.

Una vez marcado, el mensaje de notificación se encamina a través de los conmutados de la red hacia el nodo fuente, parte 4 de la Fig. 4-1. El agente de control de congestión analiza el bit *BECN*, dentro de la cabecera y responde informando al nodo que disminuya la inyección de mensajes. De esta manera, los puertos congestionados pueden recuperarse liberando los paquetes contenidos en sus buffers.

La disminución en la inyección será más restrictiva dependiendo de la cantidad de mensajes de notificación que se reciban en el nodo fuente. InfiniBand propone una manera de regular la inyección, basándose en una tabla parametrizable denominada tabla de control de congestión *(Congestión Control Table, CCT).*

El contenido de cada posición de esta tabla, consiste en un valor de tiempo, que representa en tiempo *inter-envío* destinado a regular la inyección. En la parte 5 de la Fig. 4-1 se muestra esta tabla y el índice utilizado para direccionar sus posiciones. Cuanto mas grande es el valor del índice, mayor es el valor del tiempo inter-envío de la posición asociada. De esta manera, cuantos más mensajes marcados se reciben en el nodo fuente, mas se incrementa el índice de la tabla, seleccionando de esta manera, una posición que contiene un tiempo inter-envío más grande.

Eventualmente, la congestión desaparece y la inyección debe recuperarse. Esta tarea es realizada por el agente de control de congestión residente en el nodo, a través del uso de un temporizador. La especificación permite definir un valor de tiempo, utilizado para incrementar el índice de la tabla de control y recuperar la inyección. Cada vez que transcurre un intervalo de tiempo sin que se hayan recibido mensajes de notificación, el índice se decrementa. De esta manera, si no se reciben más mensajes que indican la presencia de congestión en los enlaces, se recupera la carga de tráfico normal.

El ajuste de los parámetros del mecanismo de control *(Índice, Tabla, umbrales, etc.)* es responsabilidad del gestor de control de congestión *(congestión Control Manager, CCM)*, que establece sus valores y puede configurarlos tanto estática como dinámicamente.

La principal desventaja de este mecanismo, radica en que la congestión se elimina trasladando la congestión desde los conmutadores, hacia los nodos fuentes que inyectan los paquetes. De esta forma el comportamiento global de la latencia promedio se incrementa igualmente, pudiendo alcanzar valores muy elevados en presencia de cargas de tráfico adversas.

En la especificación [15] esta escrito textualmente: *"By limiting the injection rate of ports to something which they can handle, congested ports should not see a significant degradation, after all their packets were just going to wait anyway"*.

Sin embargo, si los paquetes son enviados a través de caminos alternativos disponibles y que no sufren congestión, es posible mantener la inyección. De esta manera las prestaciones de la red se mantienen a nivel global, lo que permite obtener un grado de utilización muy alto de los enlaces.

Teniendo en cuentas estas razones, proponemos la implementación del balanceo distribuido del encaminamiento en redes InfiniBand. A continuación se presentan en detalle las modificaciones realizadas sobre el algoritmo que permiten, en conjunto con las facilidades que ofrece la arquitectura, aplicar un algoritmo de encaminamiento vigente, factible y que ofrece muy buenos resultados [27] [28], sobre un estándar emergente pero que no ofrece un control de congestión adecuado.

4.3 Propuesta de diseño para el control de congestión en InfiniBand

La implementación de nuestro algoritmo de control se realizó dentro del contexto establecido por la arquitectura de InfiniBand, teniendo en cuenta las definiciones, tanto obligatorias como optativas, descriptas en la especificación IBA. Por esta razón las modificaciones fueron hechas en el algoritmo DRB y no en InfiniBand. La adaptación de DRB, fue realizada en función de las facilidades para el control de congestión que ofrece el estándar, pero siempre respetando el concepto de operación y funcionamiento del algoritmo consistente en realizar un balanceo de la carga de tráfico de los enlaces mas cargados a los menos cargados con objeto de distribuir uniformemente la carga y por tanto la latencia que sufren los mensajes en la red.

A continuación describimos cada fase del método, junto a las modificaciones realizadas.

4.3.1 Función de detección de la congestión

Se lleva acabo en los canales virtuales de cada puerto en los conmutadores, monitorizando la ocupación de los mismos con respecto a un umbral relativo al tamaño del buffer. El valor del umbral es establecido por el CCM y los diversos valores posibles, varían entre los números 0 y 15; donde el valor 0 indica que ningún paquete

ha de marcarse en este puerto, y el valor 15 especifica un umbral muy agresivo, ver Fig. 4-2. Debido a que la arquitectura del conmutador es un parámetro que influye en la generación de congestión, y afecta en la determinación de su nivel, el significado exacto de los valores de umbral (0…15) se deja a criterio del fabricante del conmutador.

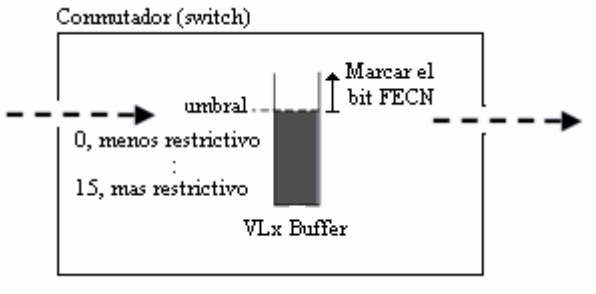


Fig. 4-2 Detección de congestión

Por otra parte, el control de congestión puede también configurarse para marcar solamente paquetes en el conmutador identificado como "raíz de la congestión.". Se denomina raíz de congestión al conmutador cuyo canal virtual de salida ha excedido el umbral y sin embargo dispone de créditos para reenviar paquetes, es decir es el puerto causante de la congestión, como puede observarse en la Fig. 4-3. En cambio, se denomina "victimas de congestión" a aquellos conmutadores cuyos canales virtuales han excedido el umbral pero a causa de la falta de créditos para avanzar. En este caso los buffers comienzan a llenarse debido a que la congestión que se ha producido en un nodo remoto, se ha propagado por la red. El control de congestión puede ser configurando para marcar los paquetes en los buffers de los conmutadores causantes de congestión y así discriminar entre los nodos que son víctimas de la congestión y los que la generan.

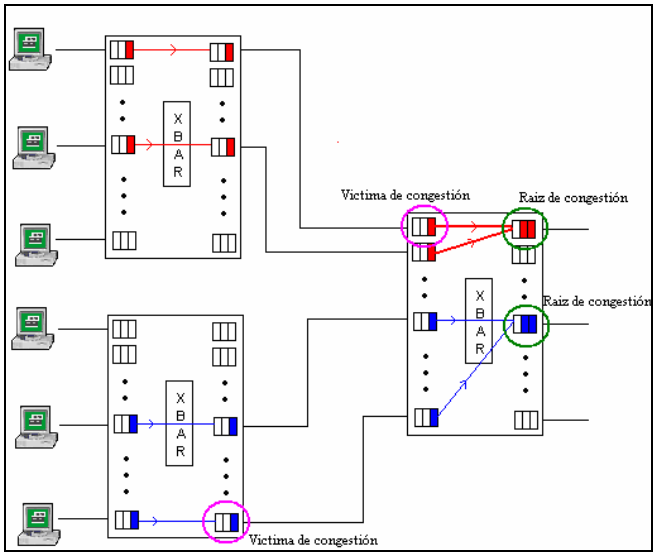


Fig. 4-3 Raíz y victima de congestión

Cuando el mecanismo detecta congestión dentro del conmutador, informa al nodo destino poniendo a *uno* el bit denominado FECN dentro de la cabecera de transporte *(Base Transport Header, BTH)* presente en todos los paquetes del protocolo. A continuación el mensaje sigue siendo transmitido hacia el nodo destino, según la trayectoria especificada inicialmente. Nótese que la información que denota la existencia de congestión se comunica, en primer lugar, al nodo destino y no directamente al nodo fuente.

El motivo de este modo de operación, radica en que la especificación determina que los conmutadores de la arquitectura, no deben generar paquetes (a excepción de algunas funciones de gestión) y es necesario cumplir con esta característica, para mantener compatibilidad con el estándar. Por otra parte, es necesario identificar al par de colas, dentro del adaptador de canal del nodo fuente, que es el causante de la congestión. De esta manera, sólo se aplica la disminución en la velocidad de inyección al causante y no a todos los pares de colas del adaptador. Con este fin, la información del par de colas está contenida dentro de la cabecera en el paquete, y los conmutadores no pueden procesar cabeceras, ni tienen acceso para realizar cambios en ellas.

En virtud a las características de funcionamiento expuestas por InfiniBand, hemos modificado al algoritmo DRB, de manera que valiéndose de las capacidades de la arquitectura, pueda detectar congestión a través de la ocupación en los buffers de los canales virtuales y se ha eliminado el mecanismo de registro y almacenamiento de la latencia de transporte (cuya aplicación no es válida en el estándar).

4.3.2 Notificación de la congestión

Una vez que el paquete con el bit de notificación FECN marcado alcanza el adaptador de canal *(HCA)* del nodo destino, el agente de control de congestión *(CCA)* determina el envío de un mensaje de respuesta al nodo fuente, mediante un paquete de notificación de congestión, como se muestra en la Fig. 4-4. En la cabecera de este paquete, existe un bit denominado BECN que informa explícitamente esta situación.

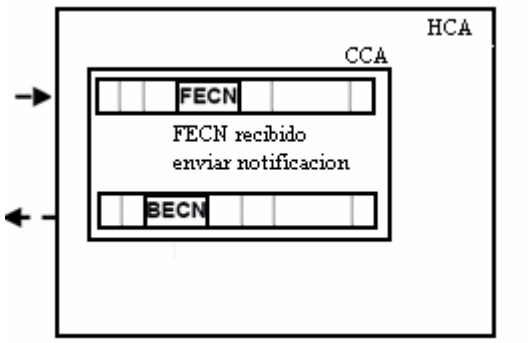


Fig. 4-4 Notificación de congestión

Debido a que este bit esta ubicado en una cabecera común a todos los paquetes, la notificación puede hacerse mediante un mensaje de reconocimiento *(Acknowledge message)* en el caso de comunicaciones confiables (ver *Reliable connections* en el Capítulo 3), o bien a través de un paquete especial de notificación de congestión (CN), en el caso de comunicaciones no confiables.

En este caso, los mensajes de notificación definidos en la especificación cumplen la misma función de los mensajes de reconocimiento de DRB, con la salvedad que la información que contiene es diferente. En el primer caso sólo contienen un bit, mientras que DRB, contiene el valor de la latencia almacenada en la trayectoria del mensaje.

4.3.3 Cálculo y selección de trayectorias alternativas

Los nodos fuente que inyectan paquetes en zonas congestionadas de la red de interconexión comenzaran eventualmente a recibir notificaciones desde los nodos destino. En este momento, el agente de control de congestión (CCA) del nodo fuente debe activar el mecanismo que permita solucionar el problema.

Cuando el nodo fuente detecta la llegada de un paquete con el bit BECN marcado, el CCA evalúa los caminos a seleccionar para planificar el envío siguiente de paquetes. Esta evaluación se realiza mediante el cálculo de la cantidad de paquetes de notificación recibidos y su distribución a través del conjunto de trayectorias alternativas entre el nodo fuente y el nodo destino. De tal manera que la inyección del conjunto siguiente de mensajes se realice de forma inversamente proporcional a dicha distribución, es decir, los caminos menos ocupados son los más utilizados. En la versión original de DRB se funcionaba de la misma manera pero operando con las latencias de cada camino alternativo en lugar de tomar el número de paquetes de notificación de congestión.

Como ejemplo supongamos que existen cuatro caminos disponibles entre un determinado par de nodos, según puede verse en la parte (a) de la Fig. 4-5. Por motivos de simplicidad supongamos que en un dado momento llegan 100 mensajes de notificación al nodo fuente; 55 lo hacen por el *camino1*, 20 por el *camino2*, 15 por el *camino3* y 10 por el *camino4*.

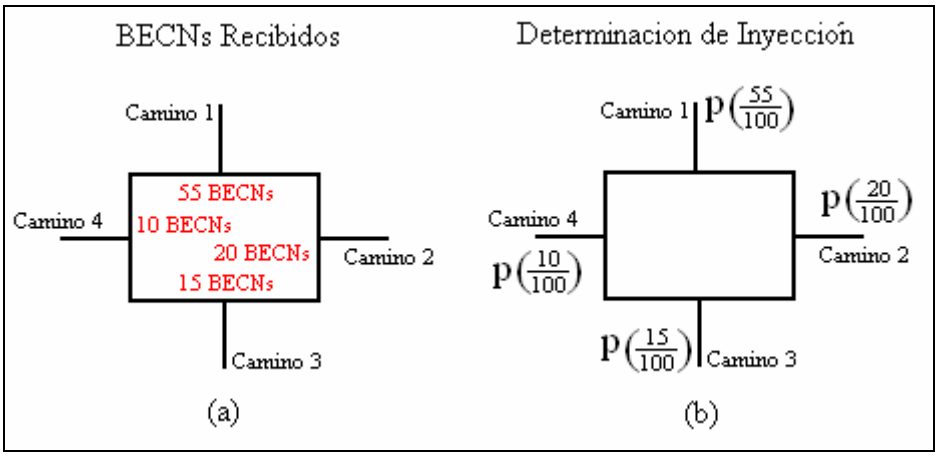


Fig. 4-5 Cálculo de trayectorias alternativas

El agente de control de congestión distribuye la inyección a través de las trayectorias, de tal manera que los paquetes salientes tienen una posibilidad menor de ser encaminados por el camino1, dado que la probabilidad de selección de este enlace tiene un peso menor (en este caso $\frac{100}{55}$). El mismo concepto se aplica a las demás trayectorias, con un peso de $\frac{100}{20}, \frac{100}{15} y \frac{100}{10}$ para los caminos 2, 3 y 4 respectivamente.

La apertura de las trayectorias se hace de forma gradual en función de la cantidad de notificaciones recibidas. Las trayectorias son seleccionadas de manera tal que su largo no involucre un tiempo de transmisión demasiado grande, por lo que deben ser lo mas cortas posible. De manera tal que el problema producido por la contención en los buffers, no se traslade hacia una perdida de prestaciones provocada por un tiempo de viaje elevado. Asimismo, las trayectorias deben ser disjuntas, de manera que la selección de un camino alternativo implique que el tráfico que maneja, circule por una zona que no sufre congestión.

Ha de destacarse que la solución propuesta cumple completamente con los conceptos de funcionamiento del algoritmo DRB, en el cual la selección de trayectorias se realiza en función de la inversa de latencia acumulada en el camino y especifica una información semejante a la otorgada por los mensajes de notificación (utilizando previamente un tratamiento adecuado de dicha información).

Trayectorias múltiples

Para que selección de trayectorias múltiples sea posible, InfiniBand propone un mecanismo denominado: *Mecanismo de gestión de subred* (una descripción detallada de este mecanismo puede encontrarse en el Capitulo 3) [46]. Las operaciones y características principales definidas en este mecanismo nos permiten, en conjunto con el mecanismo de control, aplicar el concepto de encaminamiento adaptativo sobre esta arquitectura.

Como se ha mencionado en capítulos anteriores, el mecanismo de gestión es el encargado de:

- Descubrir la topología física de la subred.
- Asignar los identificadores locales *(LIDs)* a los puertos de los nodos, conmutadores y encaminadores
- Establecer las trayectorias posibles entre los diversos nodos de la red.
- Explorar la subred en busca de cambios en la topología.

InfiniBand no define explícitamente de que manera deben realizarse estas tareas, sino que especifica el orden y la estructura de datos que han de manejarse en su realización.

Cada subred tiene al menos un gestor de subred *(Subnet Manager, SM),* que puede estar implementado tanto en software como en hardware, y puede estar ubicado en un puerto de un CA, un conmutador, o un encaminador indistintamente. Cuando hay más de un SM dentro de la subred, el mecanismo de gestión debe asignar uno como maestro (*master)* y los restantes quedaran en espera.

El SM maestro es el elemento clave en la configuración e inicialización de la subred y el encargado de realizar las tareas descritas anteriormente. Las operaciones de este componente se resumen en el diagrama de estados presente en la Fig. 4-6.

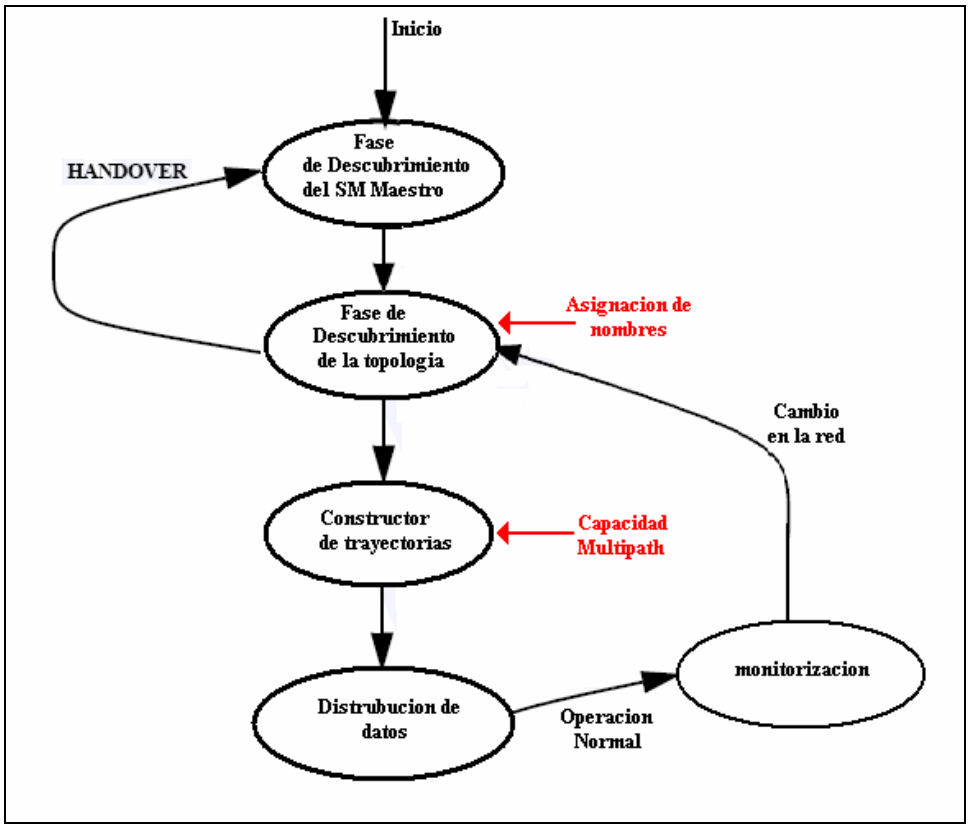


Fig. 4-6 Máquina de estados del SM

En la fase, inicial el mecanismo de gestión de subred verifica la existencia de los diferentes SMs, y asigna a uno de ellos como maestro en función de un parámetro que establece la prioridad de cada uno. Esta fase es conocida como *Handover* y tiene lugar al inicio, o en el momento en que aparezca en la subred un SM con mayor prioridad.

Una vez descubierto, el SM comienza la fase de descubrimiento de la red y sus componentes. Para tal efecto establece comunicaciones mediante paquetes de gestión (conocidos como *Management datagrams, MADs*. Ver capitulo 3 apartado 3.4) con los agentes de gestión de subred presentes en cada uno de los CA, conmutadores y encaminadores de la red.

Durante este proceso se determina el estado de los puertos de cada componente (activo/no activo) junto a sus elementos de configuración: Unidad máxima de transferencia *MTU*, canales virtuales disponibles,…, ancho de banda y uno de suma importancia para nuestra propuesta, conocido como mascara de control local *(Local Control Mask, LMC)*. El valor LMC, esta relacionado directamente con la cantidad de trayectorias alternativas que pueden asignarse a ese puerto.

Se ha mencionado previamente qué el SM asigna un identificador local *(LID)* a cada puerto de la red. El formato de este identificador, situado en las cabeceras de los paquetes, se muestra en la Fig. 4-7.

Fig. 4-7 LID y LMC

Cuando se reciben los paquetes dentro de un conmutador, los 8 bits menos significativos son ignorados, de esta manera es posible modificar el valor de esta máscara, para asignar a los puertos de los CA varios LIDs (es decir, varios nombres). De esta manera el SM puede establecer varias trayectorias para el mismo nodo, en virtud de esta multiplicidad de nombres.

Una vez que se recolecta toda la información de la red, el gestor de subred entra en la fase de construcción de trayectorias [61]. La especificación no define un algoritmo para cumplir con esta finalidad, por este motivo se ha implementado un mecanismo que genera caminos múltiples y selecciona los disjuntos, para cada par de nodos de la red. El mecanismo utilizado es el típico algoritmo de búsqueda en profundidad *(Depth-first search, DFS)* [22], con el agregado de una función que selecciona los caminos disjuntos.

Por ultimo, las tablas de encaminamiento son distribuidas a todos los conmutadores y la red de interconexión entra en el modo de funcionamiento normal. En este estado el gestor de subred queda en fase de monitorización para detectar cualquier cambio en la topología.

Mediante el uso de estas técnicas, se proporciona a la arquitectura InfiniBand, un mecanismo de control de congestión, que mejora notablemente los resultados derivados del empleo del mecanismo especificado en el estándar, como se muestra en el capítulo de experimentación y análisis de prestaciones.

4.3.4 Constricción de trayectorias

En ausencia de congestión, las trayectorias utilizadas para encaminar los paquetes dentro de la red de interconexión deben ser de largo mínimo, con el objeto de ofrecer una baja latencia de transporte. Cuando no existe contención en los buffers de los conmutadores, el valor de latencia esta principalmente determinado por la velocidad de los enlaces. Por otra parte, el uso de caminos alternativos puede incurrir en el desordenamiento de mensajes [57]. Es por esto que posteriormente a que la técnica propuesta en estos párrafos soluciona el problema de congestión, ejecuta un mecanismo de constricción de trayectorias.

La arquitectura especifica que cada adaptador de canal debe contener un contador, que permite establecer una forma de medir el tiempo en el cual el último paquete de notificación ha arribado al nodo. La duración en la que este contador expira, es un parámetro que se establece mediante el gestor de control de congestión.

El mecanismo propuesto aquí utiliza este contador para contraer las trayectorias en ausencia de congestión. Cada vez que el contador expira y no hayan llegado al nodo destino paquetes de notificación, el camino conformado por las múltiples trayectorias disjuntas se contrae.

La constricción se lleva a cabo de forma gradual Fig. 4-8, ya que el camino se contrae totalmente luego de que el contador expira cierto número de veces, ésto brinda cierta histéresis en el mecanismo y evita que se abran trayectorias alternativas cuando no es necesario.

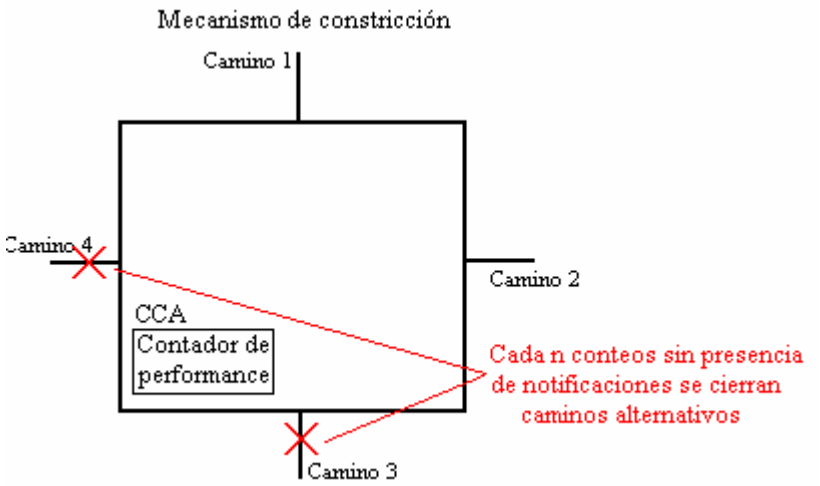


Fig. 4-8 Constricción de trayectorias

Hasta aquí se han descrito, las diversas alternativas adoptadas con el fin de aplicar el balanceo distribuido del encaminamiento a la arquitectura InfiniBand. Utilizando las características beneficiosas y provechosas que facilita la especificación (herramientas de monitorización, asignación de nombres y trayectorias múltiples) ha sido posible adaptar las fases de detección, notificación y corrección propuestas en DRB, manteniendo tanto las ideas conceptuales del algoritmo como la compatibilidad con el estándar.
Con el motivo de evaluar las características y beneficios que emergen de las ideas propuestas en los mecanismos presentados en este capitulo, se han analizado diferentes plataformas y herramientas de desarrollo. El estudio realizado sobre la implementación aquí descrita, ventajas y desventajas de éstas, se desarrolla en el capitulo siguiente.

Capitulo 5 Evaluación, selección y descripción de la plataforma de simulación y desarrollo.

Uno de los objetivos principales de este trabajo de investigación, consiste en el estudio general de las posibles herramientas de desarrollo y simulación, con el fin de seleccionar la plataforma adecuada para la validación y experimentación de la propuesta establecida. Este capítulo muestra las decisiones tomadas para cumplir con el objetivo mencionado. Primero se ofrece una introducción, a fin de definir la terminología en este ámbito y las características buscadas en las herramientas de estudio. Asimismo, se mencionan los requerimientos deseables en los entornos de simulación, junto a las diversas posibilidades existentes.

Por ultimo se describe la herramienta seleccionada para nuestro estudio, detallando sus características y componentes, junto a su funcionamiento y modo de operación.

5.1 Introducción

La simulación es una técnica sumamente útil para el análisis del funcionamiento de los sistemas informáticos. Si el sistema bajo análisis no se encuentra disponible, como es el caso general en la etapa del diseño e investigación de dichos sistemas, un modelo de simulación es capaz de proporcionar de una manera relativamente rápida y simple, una predicción sobre su funcionamiento, junto a una comparación mas o menos precisa de diferentes alternativas [29].

Por otra parte aunque el sistema bajo análisis esté disponible para tomar mediciones reales, el modelo de la simulación puede tener suma importancia, debido a que ofrece la posibilidad de comparar las alternativas con una variedad más amplia de cargas de entrada *(workloads)* e inclusive en diferentes entornos.

Sin embargo, los resultados emergentes de la simulación pueden ser inútiles o analizados equívocamente, debido al desconocimiento en técnicas de programación, o bien a los conocimientos necesarios en estadística y procesos aleatorios. Es por este motivo, que es necesario conocer algunos requerimientos y características deseables de los entornos y técnicas de simulación, con el fin de obtener resultados adecuados y confiables. Entre estos requerimientos podemos nombrar [52]:

- **Nivel adecuado de detalle.** A través del uso de simuladores, es posible realizar un estudio más detallado del sistema, que mediante un modelo analítico. Ésto se debe principalmente a que el modelado analítico requiere varias simplificaciones y

asunciones. En la simulación, el nivel del detalle se limita solamente al tiempo de desarrollo disponible. Ha de mencionarse que una simulación más detallada requiere mayor tiempo de desarrollo y pruebas, y que la probabilidad de errores en su creación aumenta. Por otra parte, el nivel de detalle influye directamente en el tiempo de ejecución, y se convierte en un factor limitante en el caso de simulaciones grandes, donde el tiempo de ejecución puede ser del orden de varias horas o días. Es por este motivo que el nivel de detalle es un factor importante en la simulación y ha de evaluarse con detenimiento en la selección o el desarrollo del simulador a utilizar, con el fin de simular correctamente el funcionamiento y las características del sistema, sin incurrir en largos tiempos de desarrollo o de ejecución y errores de difícil depuración.

- **Lenguaje de programación.** La selección del lenguaje de programación tiene un impacto significativo en el desarrollo correcto del modelo. Los lenguajes de simulación de propósito especial *(Special-purpose simulation languages),* diseñados como herramienta destinada a facilitar el uso de un simulador particular, requieren menos tiempo para el desarrollo de los modelos y facilitan algunas tareas comunes como verificación, obtención de trazas y análisis estadístico. Por otra parte los lenguajes de uso general son portables y proporcionan más control sobre la eficiencia y el tiempo de ejecución de la simulación. Teniendo en cuenta lo expuesto y la experiencia del programador ha de seleccionarse un lenguaje adecuado de programación versátil y de fácil utilización.

- **Utilización de modelos correctos y verificados.** Los modelos utilizados en la simulación son las entidades abstractas que representan los aspectos y el comportamiento del sistema bajo análisis. Por ello requieren un esfuerzo especial en su desarrollo ya que deben proporcionar una descripción precisa de su comportamiento. Los modelos consisten en general de grandes programas de computador y a menos que se tomen las precauciones, es posible tener varios errores de programación que deriven en conclusiones sin sentido. Por otra parte, aunque un programa de simulación no contenga errores, puede darse el caso que no represente el sistema verdadero de forma precisa, debido a asunciones incorrectas o simplificaciones excesivas sobre el comportamiento de dicho sistema. Es por este motivo, que es esencial que cada modelo sea validado para asegurarse que las conclusiones obtenidas sean congruentes con las que se obtendrían del sistema verdadero.

- **Manejo correcto de las condiciones iniciales y el tiempo de simulación.** La parte inicial de la simulación no es generalmente representativa del comportamiento del sistema en un estado estacionario. La parte inicial por lo tanto no ha de tenerse en cuenta. Por otro lado, es deseable que el tiempo en que el simulador entrega los resultados sea lo mas corto posible; con este afán puede tomarse la equívoca

decisión de acortar la ejecución de la simulación. Los resultados obtenidos en tales casos pueden ser muy dependientes de las condiciones iniciales y pueden no representar el comportamiento del sistema verdadero. La longitud temporal, correcta para las simulaciones depende de la exactitud deseada (Intervalos de la confianza) y de la variación de las cantidades observadas.

- **Uso adecuado de técnicas estadísticas.** Los modelos de simulación requieren el uso de variables aleatorias, los procedimientos para generar estas variables son conocidos como generadores de números aleatorios. Es recomendable utilizar un generador bien probado y conocido que haya sido analizado exhaustivamente y no utilizar desarrollos propios que no se han sometido a análisis alguno. Los generadores de números aleatorios son procedimientos que mediante un número dado (o varios) generan otro. Este número debe elegirse cuidadosamente para mantener la independencia entre las variables.

5.1.1 Tipos de simulaciones

Entre la variedad de simulaciones descriptas en la literatura, las que presentan mayor interés para el trabajo de investigación aquí realizado, son la simulación orientada a trazas *(Trace-Driven Simulation)* y la simulación orientada a evento discreto *(Discrete-Event Simulation)*.

La simulación orientada a trazas, utiliza una traza de datos como entrada al simulador. La traza consiste en un conjunto de eventos ordenado temporalmente, que tienen lugar en un sistema real. Este tipo de simulaciones es utilizado para analizar el funcionamiento de algoritmos de gestión de recursos, o en la evaluación de algoritmos de paginación en análisis de cache, o para sintonizar algoritmos de planificación de CPU, o en el análisis de algoritmos para la asignación dinámica de memoria, por solo nombrar algunos ejemplos.

Se utilizan unas trazas de demandas en los recursos como entrada al simulador y mediante la simulación del modelo del sistema, es posible encontrar el conjunto de valores óptimos de sus parámetros.

En la tabla siguiente se hace referencia a las ventajas y desventajas, de este tipo de técnica de simulación.

Tabla 1 Ventajas y desventajas de la simulación orientada a trazas.

Ventajas	Desventajas
Credibilidad	Complejidad
Facil Validacion	Representatividad
Carga precisa de trabajo (Workload)	Finitud (largo de trazas)
Menor Aleatoriedad	Validación no amplia (En otros sistemas)
Comparacion Justa	Nivel de Detalle
Similitud con la implementación	No Permite Cambios

La simulación orientada a evento discreto, utiliza un modelo de estados discreto del sistema para llevar a cabo su análisis. Los modelos basados en eventos discretos son ampliamente utilizados en ciencia computacional, debido a permiten describir sistemas cuyo funcionamiento se basa en un numero determinado de trabajos, realizados por diversos procesos, en diferentes dispositivos. Ha de notarse que *"discreto"* no hace referencia a los valores de tiempo utilizados en la simulación, sino a los diferentes eventos que se generan en cada estado. En este tipo de simulación es posible utilizar valores tanto discretos como continuos, para marcar la evolución temporal de la simulación *(tiempo simulado).* Por otra parte, las herramientas utilizadas en la simulación orientada a eventos, están compuestas por una serie de elementos que se describen a continuación.

- *Planificador de eventos*: Contiene una lista encadenada de los eventos que esperan ser atendidos con el correr del tiempo. Este elemento es uno de los utilizados con más frecuencia en la simulación, debido a que ejecuta ante la aparición de cada evento, y a su vez, puede ser llamado varias veces durante la ejecución de un evento para programar otros nuevos.

- *Reloj de simulación y mecanismo de avance de tiempo*: Cada simulador contiene una variable global que representa el tiempo simulado. El planificador es responsable de hacer que este tiempo avance, y puede hacerlo de dos maneras. La primera es conocida como enfoque por *unidad de tiempo (unit time approach)*, el tiempo avanza a través de intervalos pequeños, comprueba en cada caso la existencia de eventos y los atiende. La segunda alternativa, es conocida como enfoque orientado a eventos y el tiempo se incrementa automáticamente hasta el próximo evento más cercano en la lista.

- *Variables del estado del sistema*: Son las variables globales que describen cada estado del sistema y mantienen la información de su evolución en la simulación. Pueden ser variables simples (un solo valor) o complejas estructuras de datos.
- *Rutinas de eventos*: Cada evento se simula mediante una rutina. Estas rutinas actualizan el valor de las variables del estado del sistema y planifican otros eventos,

si es necesario. Además de éstas, existen otro tipo de rutinas destinadas a tareas tales como: la interacción con el usuario, la inicialización de el conjunto de variables, la generación o adquisición de datos, etcétera.

- *Administración dinámica de la memoria*: El número de eventos y procesos asociados cambia continuamente dentro de la simulación, mientras se generan nuevos eventos debe recuperarse el espacio de memoria utilizado por los ya atendidos. Muchos lenguajes de programación realizan esta tarea automáticamente.

- *Núcleo principal*: Es la entidad destinada a lanzar las rutinas de la entrada, inicializar la simulación, ejecutar iteraciones, y finalmente, recolectar resultados y llamar a las rutinas de la salida.

Se han evaluado diferentes herramientas disponibles actualmente y que forman parte de los tipos de simuladores descriptos en lo párrafos anteriores. Estas herramientas otorgan gran versatilidad y flexibilidad, como parte del proceso de modelado y experimentación en el campo que concierne a la investigación y el desarrollo de las redes de interconexión.

Entre las opciones evaluadas en la búsqueda de la plataforma de simulación se encuentran, el simulador *NS2 (Network Simulator 2)*, la herramienta *Network II.5* y la plataforma *Opnet Modeler 11.5* [50]. Esta última fue la seleccionada para este trabajo y se explica con detalle mas adelante en este capitulo. Además de estos tres, se han analizado otras posibilidades tanto comerciales, como propietarias de uso libre. Las características de esas herramientas ofrecen menos posibilidades de desarrollo que las tres aquí expuestas y por lo tanto no se han incluido en este capitulo

La herramienta de simulación ***Network Simulator 2 (NS2*** [30] fue desarrollada originalmente como una variante del simulador *"REAL Network"*, con el propósito de ser utilizado en investigación y desarrollo. Esta plataforma soporta la simulación de los protocolos TCP y el encaminamiento multicast, y es una herramienta en constante desarrollo. Actualmente no especifica una instalación estandarizada y la versión original consiste en múltiples archivos fuente que es necesario compilar. Los lenguajes utilizados por NS2 son TCL[7] para el control, y C++ para su programación.

Los modelos NS2 se definen mediante *scripts* TCL, las ejecuciones se realizan a través de la línea de comandos y permiten establecer varios parámetros. Esto significa que el desarrollo de un modelo con NS2 implica el conocimiento del lenguaje TCL, y es

[7]"Tool Command Language" o lenguaje de herramientas de comando, es un lenguaje utilizado principalmente para el desarrollo de prototipos, aplicaciones "script", interfases gráficas y de pruebas

necesario utilizarlo para llevar a cabo las acciones deseadas, tanto en la definición de la estructura del modelo como para la recolección de datos de interés.

Debido a su enfoque de programación, NS2 proporciona un ambiente de programación flexible, escalable, modular y extensible para la investigación en redes de interconexión y el diseño de nuevos protocolos. Sin embargo, la herramienta requiere el conocimiento de dos lenguajes de programación nuevos lo que dificulta su selección.

El simulador de redes ***"Network II.5"*** [54], es una herramienta de diseño que describe un sistema de cómputo especificado y proporciona mediciones sobre la utilización del hardware, la ejecución del software, latencia de transporte de los mensajes y contención en la red.

*NETWORK II.5 s*e ha desarrollado como una herramienta basada en el lenguaje de simulación *SIMSCRIPT*[8]. Además de modelar el funcionamiento de la red, también permite la simulación de aspectos relacionados con el hardware y el software de los nodos de cómputo.

Los modelos disponibles se definen según las funcionalidades básicas que ofrecen, asimismo están conformados por una serie de objetos que contienen un amplio conjunto de atributos, que se configuran según el sistema a simular. Esta abstracción permite más flexibilidad en el modelado, pero también causa la necesidad de especificar los valores no relevantes de todos los modelos de la red.

Por otra parte, con esta herramienta no es posible modelar las capas más altas de un protocolo de red. Los proyectos desarrollados se almacenan como archivos de texto y suele tener problemas de ejecución debido a complejidad en la determinación de los atributos. La herramienta que provee los resultados de la simulación, sólo proporciona información del estado del dispositivo y la utilización de los objetos implicados en el modelo, pero no permite recopilar información adicional.

La carencia de librerías con modelos predefinidos y el tipo de parámetros que deben configurarse limitan severamente las posibilidades en el modelado y la simulación de redes de interconexión con esta plataforma.

En la selección de un simulador adecuado, y con el fin de cumplir con la implementación de la propuesta presentada en este trabajo, se ha establecido claramente el objetivo a cumplir con la herramienta seleccionada, especificando el nivel de detalle indicado para alcanzarlo. Mediante el uso de modelos apropiados y validos, se fijaron

[8] Es un lenguaje de programación para simuladores de uso general. Soporta principios de programación como programación estructurada y modularidad que otorgan flexibilidad en la creación de modelos de simulación.

los requerimientos necesarios en la plataforma de desarrollo, que permitan simular y modelar los aspectos necesarios, con el fin de aplicar el balanceo distribuido del encaminamiento a las redes InfiniBand.

La plataforma seleccionada consiste en un entorno de desarrollo para redes de comunicaciones y sistemas distribuidos, incluyendo redes inalámbricas, concebida para solucionar necesidades muy variadas, orientadas al análisis de rendimiento de redes y a la investigación y desarrollo de nuevas arquitecturas y protocolos de comunicaciones. A continuación se describe la plataforma mencionada.

5.2 Entorno de simulación y modelado, OPNET Modeler

La plataforma de software *OPNET Modeler,* provee un entorno destinado al diseño de protocolos y tecnologías de red, así como también las herramientas necesarias para la experimentación y el análisis de diferentes escenarios y configuraciones. Asimismo, permite realizar simulaciones robustas en todo tipo de redes, a través del modelado simple e intuitivo de sus componentes. La herramienta permite analizar eficientemente el funcionamiento de estos modelos a una escala realista, mediante una colección de mecanismos que proporcionan la recolección y presentación de métricas y resultados.

Las características técnicas más importantes, se enumeran a continuación:

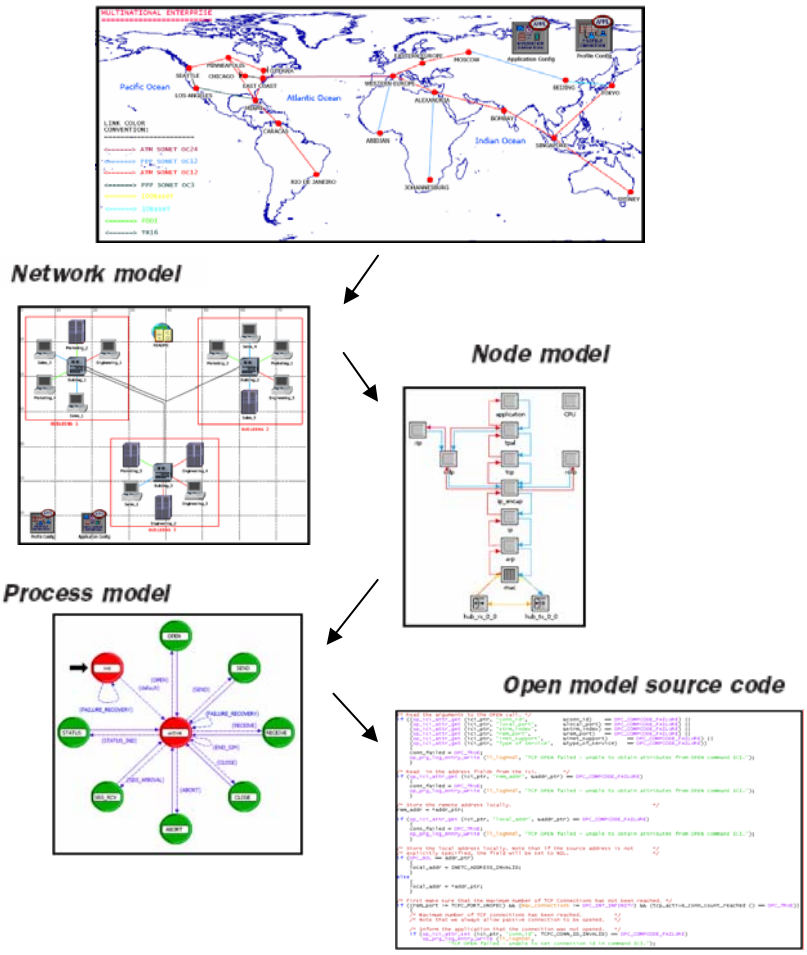
- Motor de simulación orientada a evento discreto, rápido y eficiente.
- Modelado mediante técnicas de programación orientada a objetos.
- Entorno de modelado jerárquico.
- Soporte en 32 y 64 bits para simulaciones paralelas.
- Soporte para simulaciones distribuidas.
- Provee las interfases para enlazar la simulación con sistemas reales *(System-in-the-Loop).*
- Contiene una interfase abierta para la integración de archivos externos, librerías e inclusive otros simuladores.
- Interfase grafica para depuración y análisis.

- Entorno diseñado para modelar protocolos y componentes mediante la técnica de estados *(Finite State Models)*, con el lenguaje de programación C/C++ y un conjunto de librerías.

El motor de simulación permite la ejecución eficiente y escalable de diferentes aspectos de la red a través de diversas técnicas de aceleración. El modelado del comportamiento de cada objeto que forma parte de la red se realiza mediante un paradigma de modelado sumamente simple y claro, las entidades que conforman un elemento de la red son modeladas a *nivel de proceso* y la interconexión entre estas entidades se realiza a *nivel de nodo*, a su vez los elementos de la red se conectan mediante enlaces a *nivel de red.* El comportamiento de los enlaces es programable y permite establecer retardos precisos, diversos niveles de error, características de utilización, etc.

El modelado jerárquico de la red, permite establecer complejas topologías con una gran cantidad de subredes anidadas, la técnica de modelado orientada a objetos modela cada nodo mediante clases (y sus bondades asociadas: herencia, polimorfismo…).

La organización jerárquica es la siguiente:



Editores

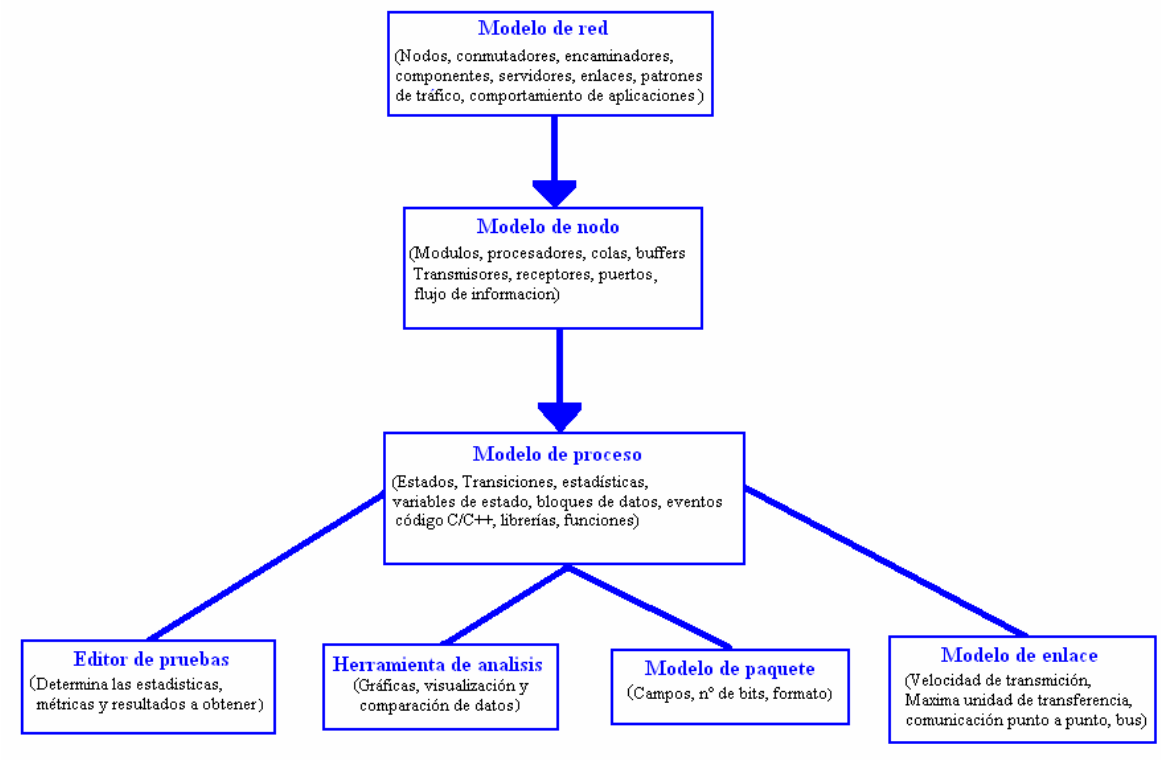


Fig. 5-1 Jerarquía de modelado

Como se puede observar en la Fig. 5-1, el nivel jerárquico superior determina el modelo de red, donde se definen las subredes, componentes, topologías, etcétera. A continuación se observa el modelo de nodos donde se define la estructura interna de cada componente, y nivel más bajo esta el modelo de procesos que determina el comportamiento lógico de los elementos de la red.

A continuación se ofrece una explicación de las herramientas que Opnet provee, con el fin de modelar redes, según la jerarquía establecida.

5.2.1 El editor de proyectos

Conforma el punto de partida para el modelado de una red. Se utiliza para crear la disposición de componentes que conforman la topología, recolectar estadísticas sobre la red, comenzar la simulación y analizar los resultados.

El editor contiene los objetos básicos que determinan el comportamiento del sistema nodos, enlaces, subredes, etcétera. Todos ellos pueden ser adaptados y configurados editando sus parámetros de simulación y cambiando la lógica de sus modelos subyacentes.

La plataforma incorpora un potente depurador interactivo con el que analizar los errores surgidos durante la compilación de los modelos o durante la ejecución de las simulaciones. Tras la compilación se genera un motor de simulación ejecutable basado

en eventos y es posible observar gráficamente su evolución. Asimismo permite visualizar el flujo de paquetes entre los dispositivos dentro de una red o entre los módulos los componen (por ejemplo entre los *buffers* y el *crossbar* en el interior de un conmutador).

Por otra parte, dispone de una amplia variedad de estadísticas de rendimiento predefinidas, que pueden ser recogidas automáticamente durante la simulación, sin intervención del usuario. Por supuesto, el usuario de *Opnet* puede definir estadísticas nuevas e incluirlas en sus propios modelos.

La Fig. 5-2, muestra la potencia del editor de proyectos, en el que se ha modelado una red de área amplia entre dos universidades, utilizando modelos genéricos de redes LAN interconectados a través de enlaces ATM y encaminadores estándar. Con este modelo es posible simular diferentes cargas de tráfico entre dos campus que comparte el acceso a un servidor FTP.

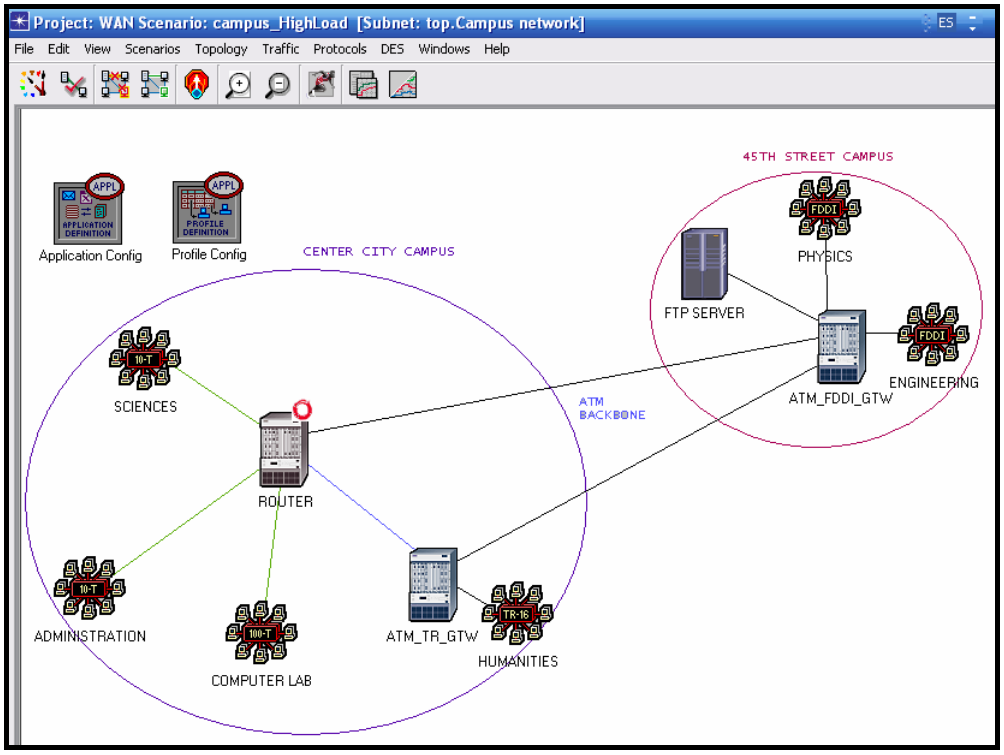


Fig. 5-2 Editor de proyectos

Este editor ofrece una gran variedad de modelos de componentes de red para diferentes fabricantes (Cisco, 3Com, HP, Lucent, por solo nombrar algunos), con la posibilidad de modificarlos y crear otros nuevos.

5.2.2 El editor de nodos

Permite modelar todos y cada uno de los nodos que conforman la red a simular especificando su estructura interna. OPNET denomina *nodo* a cada dispositivo que compone la topología (servidores, terminales, conmutadores,...). Cada nodo se compone de múltiples *módulos* y el editor de nodos establece la forma en la que se interconectan. Esta interconexión permite intercambiar información y paquetes entre los módulos que conforman el modelo.

Cada módulo tiene una función específica dentro del nodo; en general estas funciones consisten en generar paquetes, encolarlos, procesarlos, transmitirlos y recibirlos. Dentro del editor, existe la posibilidad de asignar a cada módulo un conjunto de atributos y parámetros que otorgan gran versatilidad a la plataforma de simulación.

Los módulos que permite crear el editor y que describen a cada componente de la red son:

- *Procesadores:* Son objetos de propósito general, cuyo comportamiento lógico es determinado por una máquina finita de estados (como se muestra mas adelante). Pueden utilizarse para modelar unidades de arbitraje, generadores de paquetes, unidades crossbar, etc.
- *Colas:* Permite realizar el modelado de buffers y otros elementos de almacenamiento, poseen diversos atributos que permiten definir su comportamiento. La plataforma ofrece varios modelos de colas estándar (*FIFO, LIFO,* etc.) listos para utilizarse.
- *Transmisores y receptores:* Controlan la salida y entrada de paquetes hacia y desde los enlaces, y es posible realizar modelos a muy bajo nivel.
- *Conexiones (Packet Stream)*: Definen el flujo de la información entre los módulos del modelo.
- *Cable de estadísticas:* Permite comunicar métricas y valores de interés entre dispositivos.
- *Cable de asociación lógica transmisor-receptor:* Usado para crear un vínculo entre transmisores y receptores de un mismo elemento.

La Fig. 5-3 muestra el editor de nodos, en este caso utilizado para modelar internamente un conmutador *ethernet* de 16 puertos, basado en el modelo *OSI (Open System Interconnection).*

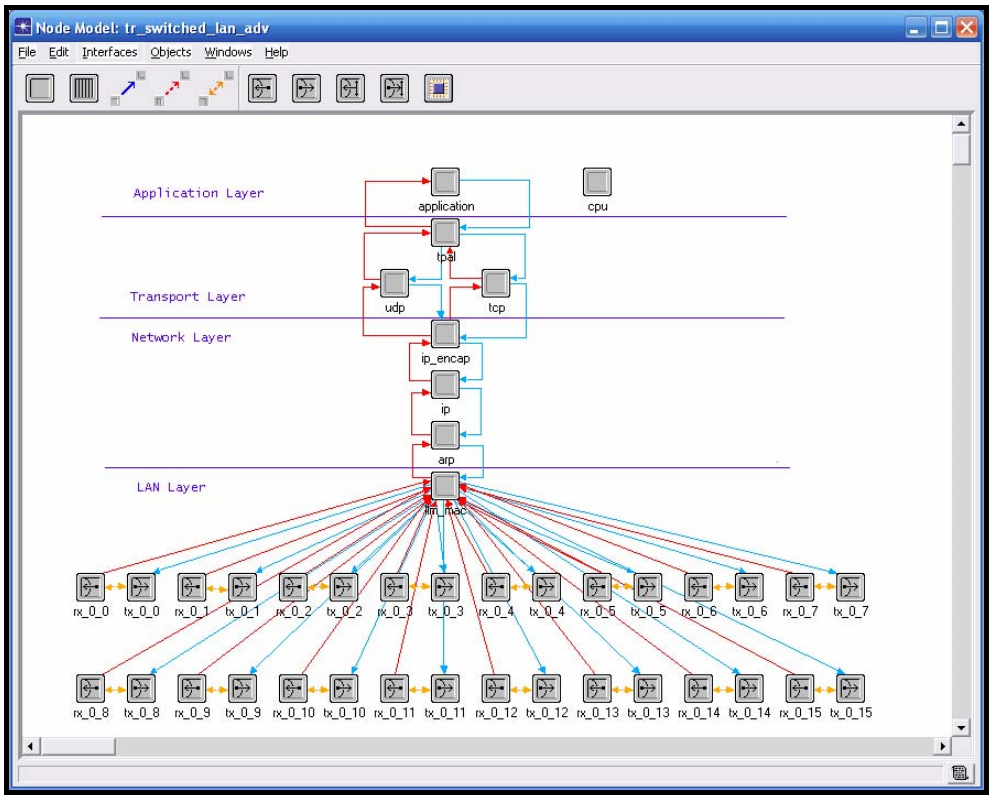


Fig. 5-3 Editor de nodos

En ella pueden observarse los procesadores utilizados para modelar las diferentes capas, junto a los puertos conformados por pares de transmisores y receptores, y las conexiones (en azul) que determinan el flujo de la información.

Hasta aquí se han comentado los editores de mayor nivel jerárquico que ofrece la herramienta, estos editores determinan la disposición física y espacial de los elementos que conforman la topología de red y permiten establecer de que forma se distribuye el flujo de la información entre estos elementos y entre los módulos dentro de ellos.

Continuando con la descripción jerárquica de editores y componentes que otorgan a la plataforma de simulación la versatilidad requerida para el modelado de todo tipos de redes, existe un conjunto de elementos que permiten determinar el comportamiento lógico y temporal de cada modulo, especificando todos los detalles, parámetros y atributos necesarios para la ejecución de la simulación. Estos componentes se describen a continuación.

5.2.3 El editor de procesos

Es un componente de gran importancia dentro del modelo jerárquico propuesto por la plataforma de simulación. Esta importancia radica en el hecho que todos los módulos que conforman cada componente, deben tener una lógica de comportamiento especificada. El editor de procesos permite definir esta lógica de comportamiento,

mediante la creación de una máquina de estados finita *(Finite State Machine, FSM)*. Cada modelo que define un proceso, está diseñado mediante estados y transiciones, como muestra la Fig. 5-4.

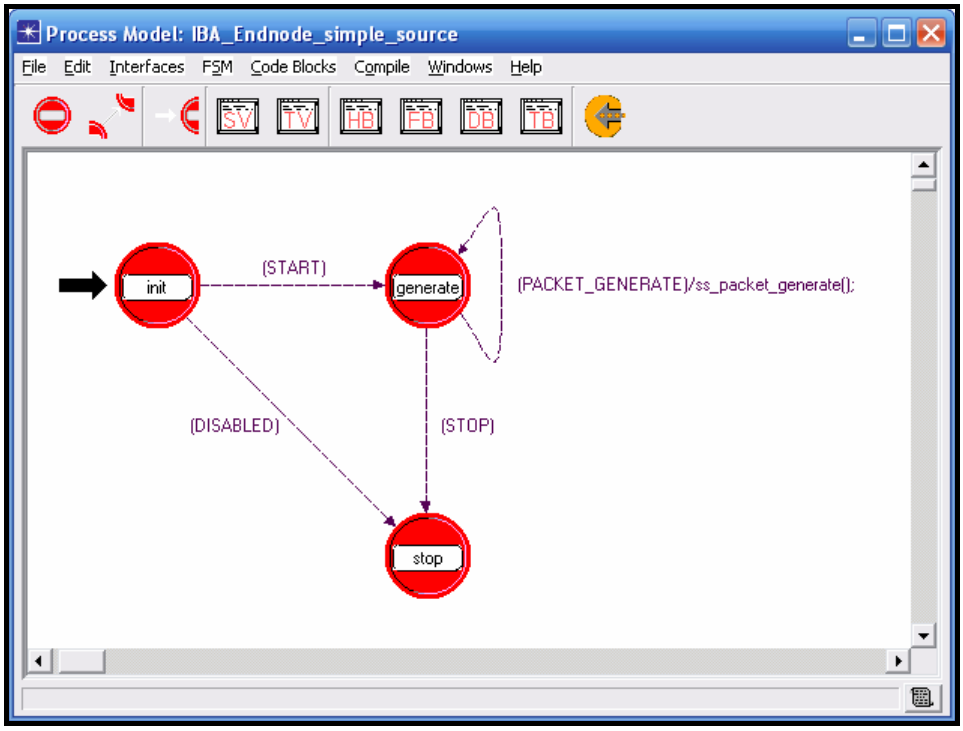


Fig. 5-4 Editor de procesos

En ella puede verse el modelo de estados de un generador simple de paquetes. Luego de recibir una directiva de comienzo en un tiempo de simulación determinado *(START)*, la lógica de funcionamiento pasa desde el estado inicial *(init)* (en el que se inicializan variables, se obtienen los parámetros de entrada al simulador, etc.), al estado de generación de paquetes *(generate)*, en el cual se construye cada paquete con un periodo determinado por la directiva *(PACKET_GENERATE)*[9].

La lógica de operación de cada estado o cada transición están descriptas en lenguaje C++, mejorado por un conjunto de librerías que proporciona la plataforma de simulación, ver Fig. 5-5. Estas librerías ofrecen un potente conjunto de servicios, relacionados con todos los aspectos de la simulación y el modelado, separados por categoría. Alguna de las diversas posibilidades que ofrece son: Conjunto de funciones para animaciones *(Animation Package)*, estadísticas *(Distribution Package)*, eventos *(Event Package)*, control de información *(Interface Control Information Package)*, interrupciones *(Interrupt Package)*, paquetes *(Packet Package)*, programación *(Programming Package)*, procesos *(Process Package)*, simulación *(Simulation Package)*, transmisión de datos *(Transmission Data Package)*, topologías *(Topology Package)*, etcétera.

[9] Este modelo de generación de paquetes, aunque con mayor complejidad agregada, es el utilizado en la implementación de *DRB* sobre InfiniBand, cuyo desarrollo se describe en el siguiente capitulo.

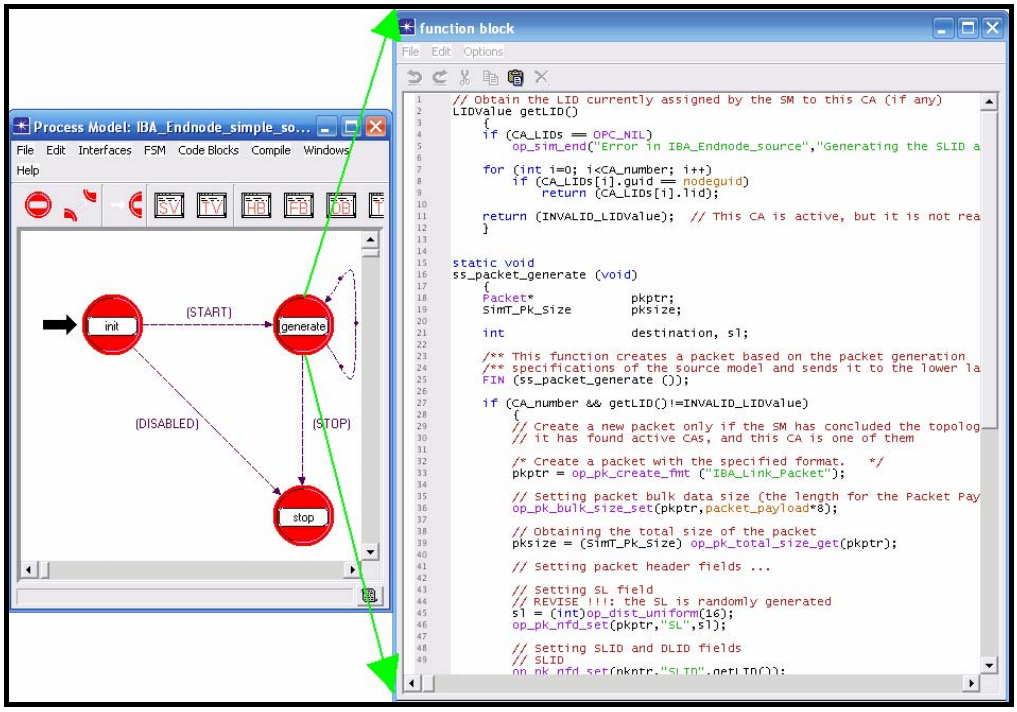


Fig. 5-5 Código asociado al estado

Una vez definido el comportamiento del modulo, a través de la máquina de estados, el editor de procesos permite compilar el código y generar los procesos que definen a cada componente.

5.2.4 El editor de enlaces

Mediante este editor es posible definir las características físicas de todos los enlaces que conforman la red; ancho de banda, latencia, número de canales, porcentaje de errores esperado, velocidad de propagación, atenuación, etc. De esta manera es posible modelar todo tipo de tecnologías (fibra óptica, coaxial, par trenzado, etc.), para cada tipo de conexiones (punto a punto unidireccional y bidireccional, buses, etc.)

5.2.5 El editor de paquetes

Este editor permite crear gráficamente, todo tipo de paquetes mediante la definición de diferentes campos con diverso tamaños (en bits). Los diferentes campos que conforman el paquete pueden ser leídos y modificados durante la simulación mediante el uso de funciones específicas. A su vez están caracterizados por una serie de atributos (nombre, tipo, tamaño, valor), con diversas estructuras y formatos. En la Fig. 5-6, se muestra el editor de paquetes utilizado para modelar y dar formato, a un paquete de datos definido en la arquitectura InfiniBand.

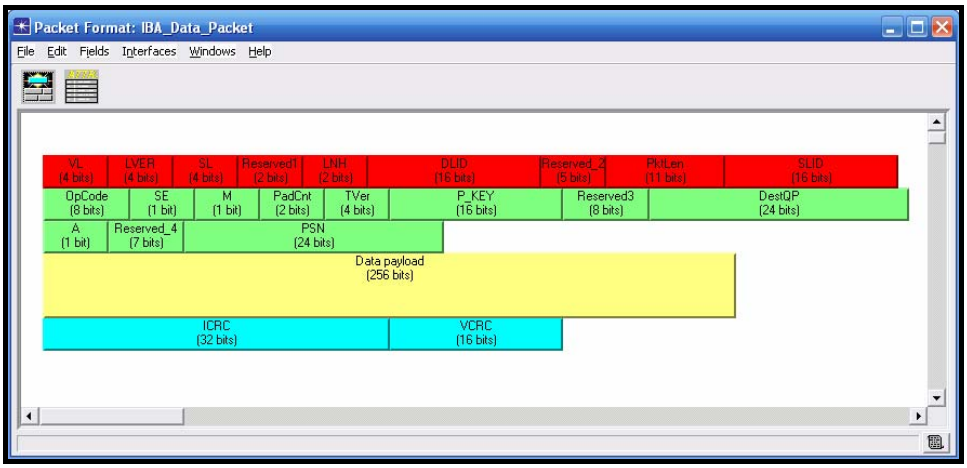


Fig. 5-6 Editor de paquetes

En la figura pueden observarse los diferentes campos que conforman las cabeceras, el área de datos y el conjunto de bits destinado al control de errores.

Las herramientas y características que se han analizado hasta ahora, permiten realizar el modelado completo de una red con el fin de examinar su comportamiento. Sin embargo es necesario en toda plataforma de evaluación, disponer de un conjunto de capacidades que permitan recolectar datos y mostrarlos al usuario, de manera que pueda evaluarse el funcionamiento adecuado del sistema a través del análisis de métricas y comparaciones.

Opnet Modeler no es la excepción y por tanto dispone de estas capacidades, a continuación se muestran las más relevantes.

5.2.6 Editor de pruebas (Probe Editor) y herramienta de análisis (Analysis Tool)

El editor de pruebas es utilizado para especificar cuáles son las estadísticas que van a ser recopiladas durante el proceso de simulación. Las estadísticas pueden ser de diferentes clases, locales o globales. Las estadísticas globales permiten conocer diversos aspectos del comportamiento general de la red (latencia promedio de mensajes recibidos en los nodos, ancho de banda global utilizado, ocupación de buffers, errores de transmisión, entre muchas otras), también es posible establecer estadísticas locales sobre cada nodo o enlace particular, sobre el cual se quiere tener un conocimiento especifico (paquetes inyectados o descartados por nodo, saturación en puertos, etc.). En la Fig. 5-7, se observa el editor de pruebas, junto con las diversas métricas que ofrece, tanto local como globalmente, para cada componente.

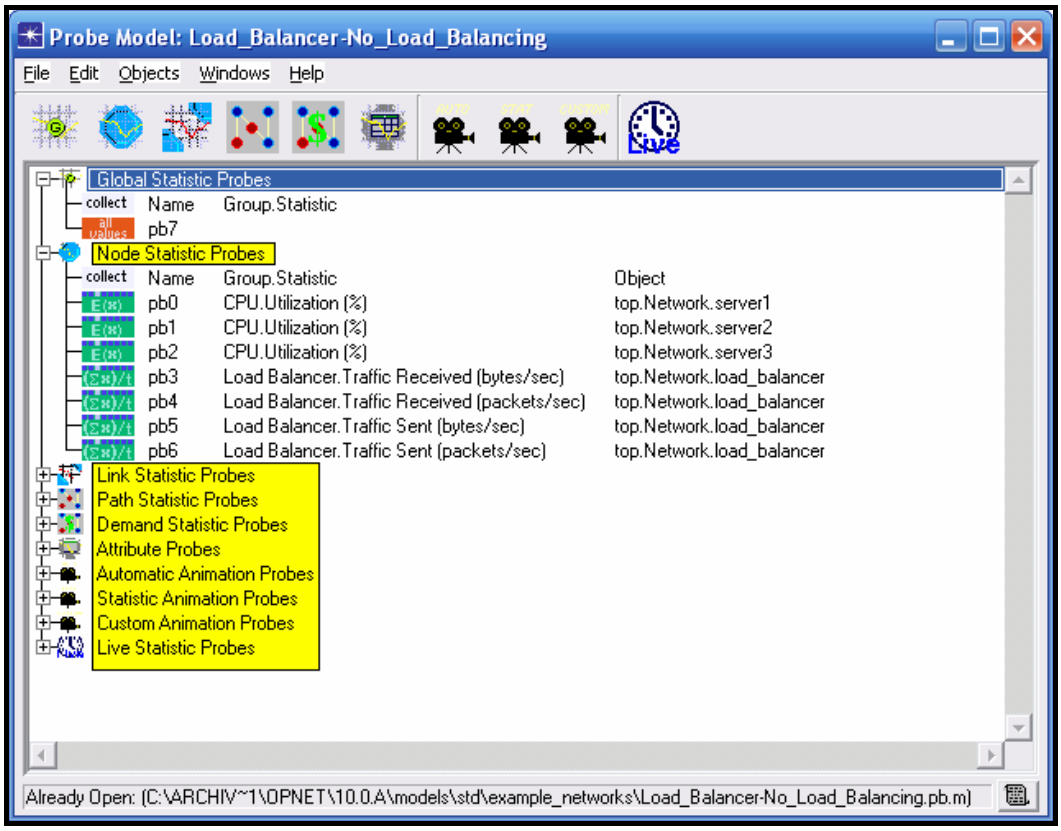


Fig. 5-7 Editor de pruebas

El complemento del editor de pruebas es la denominada herramienta de análisis y se emplea para la creación de gráficos escalares con distintos parámetros, también es utilizada para representar los gráficos emergentes de la simulación con las estadísticas definidas y los resultados. La Fig. 5-8 muestra esta interfase gráfica y algunas curvas de datos que es capaz de proporcionar.

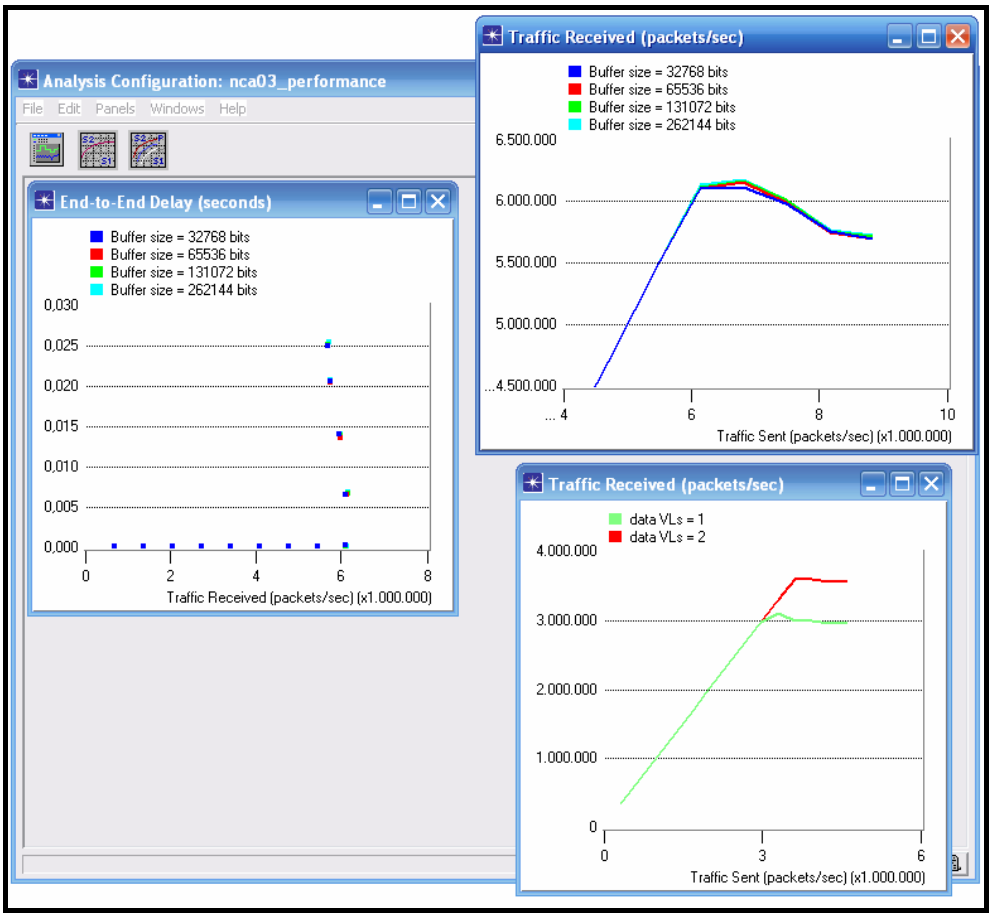


Fig. 5-8 Herramienta de análisis

Hasta aquí se han descrito las características operación principales de funcionamiento y de la plataforma de simulación y desarrollo utilizada para el análisis de la propuesta en este trabajo de investigación. Se ha desarrollado una explicación de los editores y herramientas que permiten el modelado, la caracterización y parametrización de los componentes que conforman la red, junto a las bondades ofrecidas para la recolección y evaluación de datos que otorga la simulación.

Es en virtud del conocimiento adquirido en este capitulo, que se dan las condiciones para mostrar la implementación de nuestra propuesta de encaminamiento adaptativo en la arquitectura InfiniBand. Todo el trabajo realizado con este fin es el tema que se aborda en el capítulo siguiente.

Capitulo 6 Implementación

6.1 Introducción

El presente capítulo trata los aspectos técnicos que han dado lugar a la implementación del balanceo distribuido del encaminamiento en la arquitectura InfiniBand, prestando especial atención en la parte del desarrollo realizado para dotar a los modelos del simulador opnet del control de congestión propuesto en este trabajo de investigación.

Con este propósito, se han de describir los modelos realizados en los diversos niveles de jerarquía, que ofrece la plataforma de desarrollo. La descripción se realiza de manera descendente, es decir, primero se explica cómo se han creado los componentes y sus características, junto a los distintos modelos y topologías de red que pueden obtenerse mediante su utilización. Posteriormente se analiza en detalle cada elemento de la red, los módulos que los conforman, el camino seguido por el flujo de información en la simulación y las máquinas de estado que dan lugar a su funcionamiento.

Con el fin de otorgar versatilidad a los modelos de cada componente, se han definido un conjunto de atributos, algunos de los cuales que pueden modificarse tanto estática como dinámicamente, y permiten realizar simulaciones parametrizables. Los atributos por ejemplo, pueden ser la velocidad de transmisión de un enlace, el tamaño de un buffer de memoria, el tamaño de los paquetes o el valor del umbral de acción del mecanismo de control de congestión. Estos atributos son de gran importancia en las diferentes simulaciones realizadas para evaluar las prestaciones de la red de interconexión modelada; es por esta razón que se describen también en este capítulo.

La plataforma de simulación *Opnet Modeler*, provee un conjunto de modelos InfiniBand entre los que se incluyen las características de la capa física, la capa de enlace y parte del modelo de gestión de subred. Sin embargo, estos modelos no implementan ninguna estrategia de control de congestión, y tampoco permiten establecer diferentes trayectorias entre el mismo par de nodos fuente y destino, debido a que el gestor de subred solo reconoce un camino en su etapa de descubrimiento. Las capacidades de asignación de nombres mediante la mascara de control de LIDs *(LMC)* no están definidas. Por otra parte, no se han modelado los mecanismos para el funcionamiento del gestor de performance, que posibilitan monitorizar la ocupación de los recursos.

Por estas razones y con el fin de aplicar la estrategia de control de congestión, se han desarrollado, en este trabajo los modelos de funcionamiento y operación necesarios para cumplir con los objetivos propuestos.

Los modelos, las máquinas de estado, los enlaces, las topologías, y todo el trabajo relacionado se presenta a continuación:

6.2 Componentes y Modelos de red

Como se ha visto en el capitulo 3, InfiniBand define que una subred esta compuesta por nodos, que contienen los adaptadores de canal *(CAs)*, enlaces bidireccionales y conmutadores.

En parte (a) de la Fig. 6-1, se muestra el elemento correspondiente a un nodo de cómputo y en la (b) el correspondiente a un conmutador, ambos contienen líneas que identifican los enlaces a través de los que se interconectan estos dispositivos.

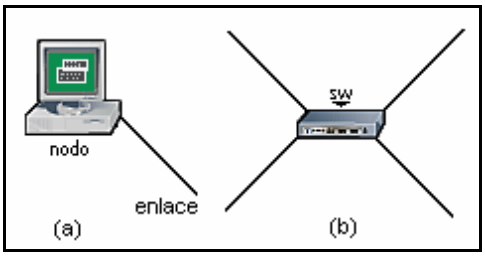


Fig. 6-1 Componentes

Según la disposición espacial de los componentes, es posible conformar una gran variedad de topologías que permiten simular diversas redes de interconexión, limitado solamente por la cantidad de memoria del sistema que ejecuta la simulación.

En la Fig. 6-2, se observa el modelo de red compuesto por 64 nodos que conforman una topología directa en malla (o *grid*) de dos dimensiones.

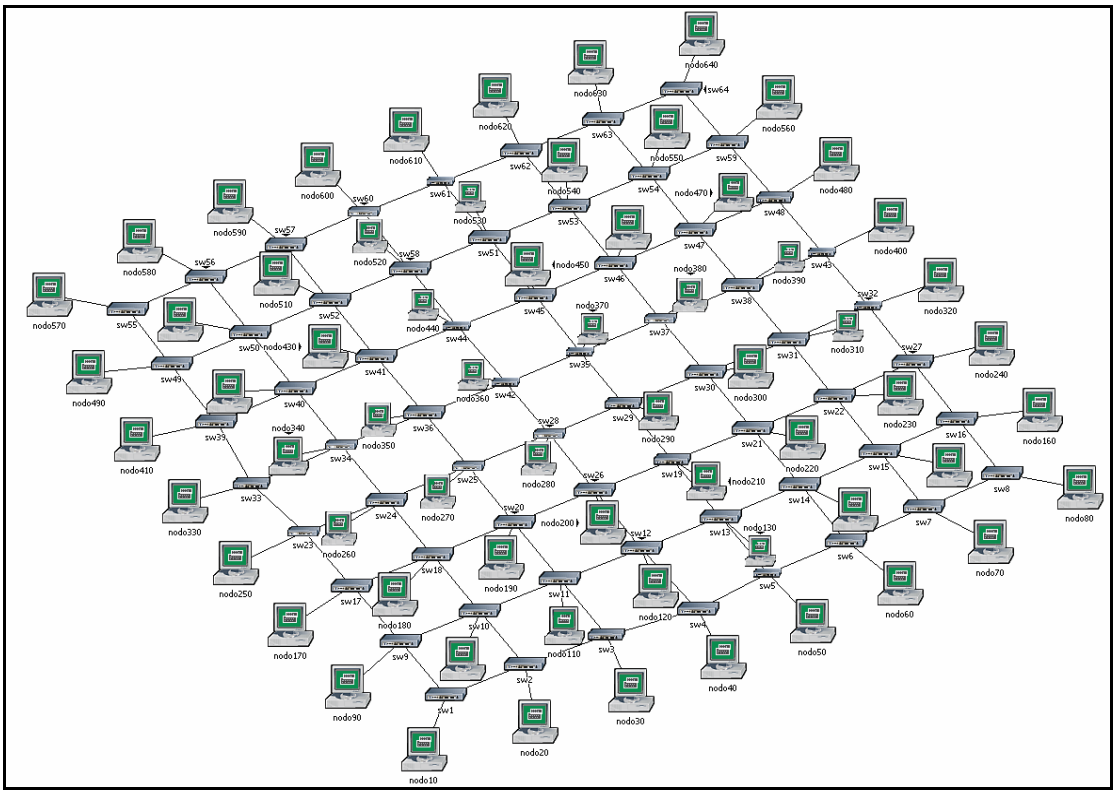


Fig. 6-2 Modelo de red (Malla 8x8)

Asimismo, las características de los modelos permiten desarrollar topologías para conformar redes indirectas, como es el caso de la red tipo *Clos* que muestra la Fig. 6-3, e inclusive topologías irregulares.

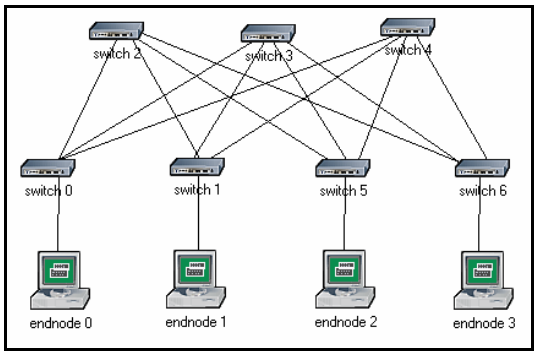


Fig. 6-3 Modelo de red (Clos Network)

6.2.1 Atributos del elemento

Como se ha mencionado, cada uno de estos componentes de red esta diseñado con una serie de atributos que describen su funcionamiento y permiten parametrizar la simulación.

En el caso de los nodos, los atributos más importantes son:

Nombre	Nombre del componente
Modelo	Modelo de nodo asignado
GUID	Identificador global
SM priority	Prioridad para la selección del SM maestro
LMC del puerto	Cantidad de LIDs asignados a este puerto
VLAT entries	Tamaño de la tabla de arbitraje de canales virtuales
Buffer size (bits)	Tamaños del buffer de paquetes de CA
Tiempo de activación	Tiempo de simulación en que se activa el nodo
Canales virtuales disponibles	Determina el número de canales virtuales para este nodo.
Host SM	Especifica que el SM, está activo en este nodo
Tiempo de respuesta	Tiempo entre que el nodo se activa hasta que queda operativo
Puertos físicos	Cantidad de puertos implementados
MTU	Unidad de transferencia máxima (en bits)
Packet payload (bytes)	Determina el tamaño del campo de datos de los paquetes generados en el nodo
Packet generation rate	Especifica la velocidad de inyección de paquetes en el nodo.
Source	Especifica si el nodo inyecta o no paquetes a la red

Cada uno de estos parámetros se define en el editor de procesos y su valor es procesado por el código en C/C++ que contiene cada estado dentro del editor.

En la tabla se muestran varios atributos, en general estos son valores especificados en el estándar InfiniBand y pueden tomar diversos valores, por ejemplo la unidad MTU puede ser de 256, 512, 1024, 2048, o 4096 bytes (que corresponden con los posibles tamaños de paquete especificados en el estándar). Por otra parte, existe un atributo denominado *Modelo* de gran importancia pues especifica el modelo (creado por el *editor de nodos*) que determina el comportamiento del nodo en la red. Mas adelante en este capítulo se muestra en detalle el modelo asignado a este nodo.

En el caso de los conmutadores, el conjunto de atributos viene dado por:

Nombre	Nombre del componente
Modelo	Modelo de nodo asignado
GUID	Identificador global
SM priority	Prioridad para la selección del SM maestro
VLAT entries	Tamaño de la tabla de arbitraje de canales virtuales
Buffer size (bits)	Tamaños del buffer de paquetes en cada puerto del conmutador
Tiempo de activación	Tiempo de simulación en que se activa el nodo
Canales virtuales disponibles	Determina el número de canales virtuales para este nodo.
Host SM	Especifica que el SM, esta activo en este nodo
Tiempo de respuesta	Tiempo entre que el nodo se activa hasta que queda operativo
Puertos físicos	Cantidad de puertos implementados
MTU	Unidad de transferencia máxima (en bits)
Tamaño de la tabla de encaminamiento	Especifica el tamaño de las tablas de encaminamiento.
Periodo de barrido del SM	Especifica cada cuánto tiempo se recorre la red en busca de fallos o nuevos nodos

En la implementación actual, se han dotado a los conmutadores de 5 puertos físicos lo que permite su uso en la evaluación de varias topologías de interés. El número de canales virtuales modelado es de tres, dos para datos y uno para los paquetes de gestión.

Los enlaces, también están caracterizados por diversos atributos. InfiniBand define que las conexiones entre nodos han de realizarse mediante enlaces bidireccionales punto a punto, con una velocidad de transmisión de bits de 2.5 Gbps. Sin embargo la codificación 8b/10b permite 2Gbps efectivos para la transmisión de datos. El retardo físico también se ha implementado (con un valor de 5ns por metro). La Fig. 6-4, muestra el *editor de enlace*, donde se especifican estos valores y el tipo de paquetes que circulan por el enlace.

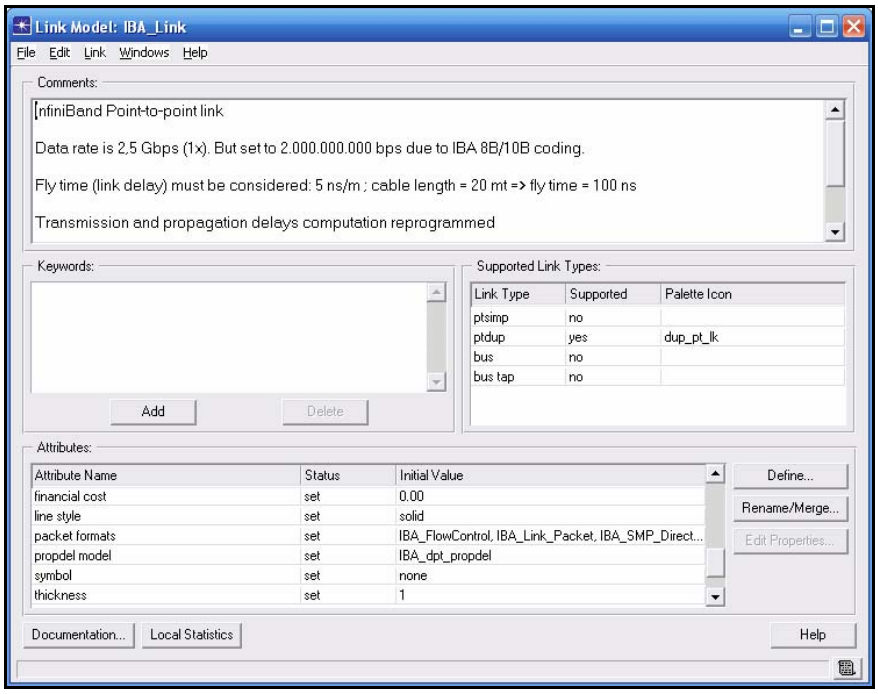


Fig. 6-4 Modelo del enlace

6.2.2 Atributos globales

Los atributos globales tienen una función similar a los atributos de elementos, la diferencia principal radica en que un atributo global define una característica de comportamiento que es común a todos los nodos de un determinado tipo. Por lo tanto, en el caso de los nodos de cómputo existen atributos globales como el patrón de tráfico a utilizar o la cantidad de paquetes a recibir antes de finalizar la simulación, y en el caso de los conmutadores el tipo de encaminamiento, la habilitación o no del componente de control de congestión, etc.

De esta manera, es posible dotar de gran flexibilidad al simulador ya que se pueden simular varios aspectos de la red en estudio, sin la necesidad de cambiar y recompilar el sistema.

En el análisis de las prestaciones del encaminamiento adaptativo sobre redes InfiniBand, utilizamos las bondades de los atributos con el fin de ejecutar varias simulaciones con diferentes parámetros de entrada, como por ejemplo el tamaño de los paquetes, o el patrón de tráfico, o la cantidad de paquetes que deben enviarse, o la activación/desactivación de *DRB*, etc. De esta forma es posible obtener varios resultados correspondientes a las métricas seleccionadas y compararlos.

6.3 Modelado de los nodos

Como se ha visto en el apartado anterior, el modelo de red permite conformar topologías mediante el uso de componentes que pueden ser caracterizados en función de sus atributos. A un nivel mas profundo, dentro de la jerarquía de modelado, Opnet ofrece una herramienta que proporciona la plataforma para desarrollar estos componentes, definir los atributos que contiene, establecer el camino de la información y como procesarla. La herramienta mencionada es el editor de nodos y se ha descrito detalladamente en el Capítulo 3.

A través del editor de nodos, se han creado los modelos que determinan el comportamiento de los componentes de red. A continuación se muestra el trabajo realizado en estos modelos para el nodo *(Channel Adapter CA)* y el conmutador.

6.3.1 Modelo del nodo de cómputo

Los nodos de cómputo han sido modelados con el fin de implementar las especificaciones de la arquitectura InfiniBand. Según la funcionalidad y las características de operación, es posible agrupar los módulos que conforman el modelo en cuatro partes o regiones principales.

De esta manera los módulos que actúan como fuente y sumidero de paquetes, y que posibilitan el modelado del procesador que contiene a la aplicación, conforman la región denominada "nodo procesador" ***(Processor node),*** tal y como se muestra en la Fig. 6-5.

Los módulos destinados a modelar los aspectos de la arquitectura que conforman la capa física y la capa de enlace, están agrupados en la región denominada "adaptador de canal" ***(Host Channel Adapter)***, *ver* Fig. 6-5. Dentro de este grupo se observan los módulos destinados a modelar la tabla de arbitraje para el acceso a los canales virtuales *(VL arbitration unit)*, la unidad de conversión de nivel de servicio a canal virtual *(SLtoVL mapping unit)*, la unidad de control de flujo *(Flow control unit)*, el puerto compuesto por una unidad receptora y una transmisora, y los buffers de recepción y transmisión *(Buffer_rcv y Buffer_xmt)*. Estos buffers son módulos que merecen un comentario adicional. Observe que el flujo de información se divide en tres cadenas o *streams* de paquetes (líneas en azul). De esta manera, *Opnet* permite modelar los canales virtuales, en este caso cada módulo que representa una cola (o buffer) se divide en tres subcolas independientes, y cada una representa un canal virtual.

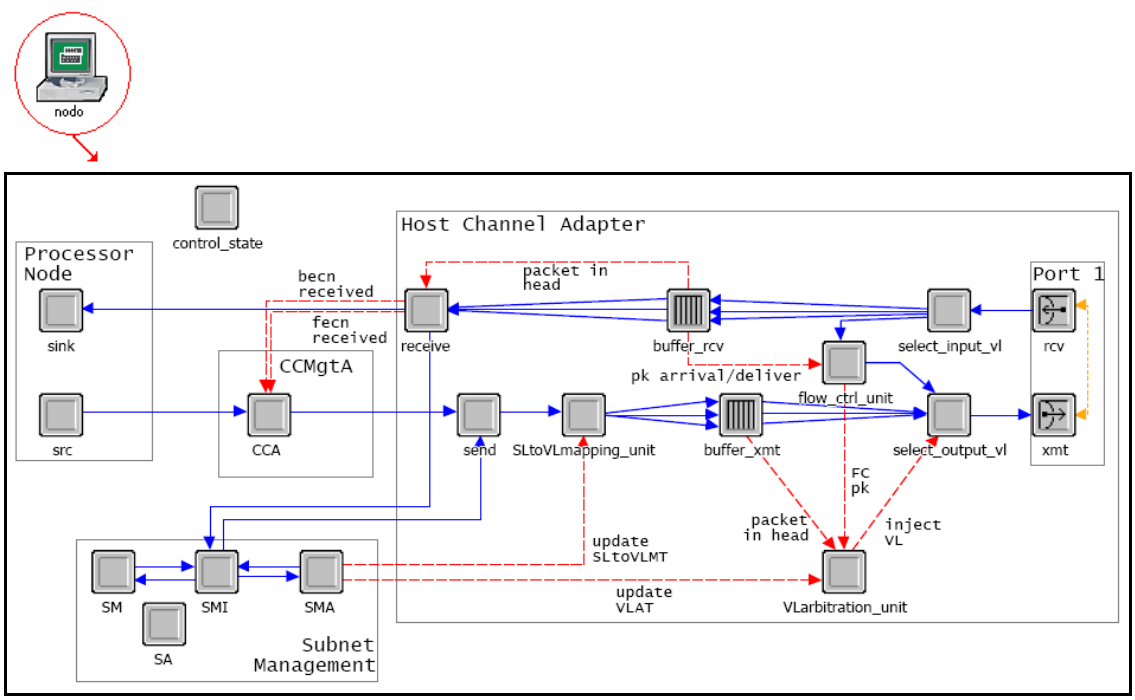


Fig. 6-5 Modelo de nodo

Además de las cadenas o *streams* de paquetes, existen otras entidades que permiten la comunicación entre los módulos. Según el caso, puede ser necesario que un módulo deba comunicar a otro la ocurrencia de cierto evento, que no necesariamente está relacionado con el avance de los paquetes. Por este motivo, la plataforma de desarrollo dispone de los denominados cables de estadísticas o *statistic wires* (líneas de rojo). En el modelado del adaptador de canal utilizamos esta herramienta, para comunicar los módulos entre sí. La figura, muestra como el cable de estadísticas denominado *packet in head,* es utilizado para comunicar al módulo receptor la llegada de un paquete, o para comunicarle a la unidad de arbitraje que existe un paquete en el buffer, con la intención de acceder al puerto de salida. También es posible observar que el cable de estadísticas denominado *pk arrival/deliver* se utiliza para que el buffer de recepción le comunique a la unidad de control de flujo que no hay más sitio para almacenar paquetes.

Según se ha mostrado en el Capítulo 3, InfiniBand define un conjunto de componentes de gestión destinados a administrar el funcionamiento correcto de la red. Las características y operaciones especificadas por el estándar, que permiten llevar a cabo la gestión de subred, se han modelado mediante los módulos que conforman la región denominada "gestión de subred" ***(Subnet Management)***, como se muestra en la Fig. 6-5 .Cada módulo se corresponde con alguna de las entidades específicas que requiere la gestión de subred, así pues, existe el módulo correspondiente al gestor de subred *(SM)*, la interfase de gestión *(SMI)* y el agente de gestión de subred *(SMA)*. En el caso de recepción, el módulo SMI procesa los paquetes de gestión *(SMPs)* y determina si debe enviarlos al SM en el caso que funcione como maestro, o bien al SMA. Los paquetes de

gestión utilizan el canal virtual cero *(VL0)* y son enviados a la tabla de conversión *SLtoVL mapping table*, para acceder a dicho canal virtual en caso de envío.

Por ultimo, en la zona denominada *CCMgtA (*por *Congestión Control Management Agent)*, se encuentra el módulo destinado a implementar las características asociadas con respecto al control de congestión. En este módulo se reciben y procesan los paquetes cuyos bits de notificación *FECN* han sido marcados en algún conmutador debido a la presencia de congestión. También es el encargado de generar los paquetes de notificación *(CN)*, realizar el cálculo y la selección entre las distintas trayectorias alternativas para el envío, acceder a la cabecera de los paquetes en la que se encuentra el destino y modificar este campo en función de la mascara de control *(LMC)*, y finalmente realizar la monitorización del contador que permite contraer trayectorias.

El funcionamiento de cada uno de los módulos que conforman el modelo del nodo de cómputo está descrito a través de una máquina de estados finita *(FSM)* que especifica su lógica y su dinámica de operación, según se ha visto en el capitulo 5.

Las FSMs se desarrollan mediante el uso del editor de procesos que permite definir estados y transiciones, junto a las herramientas necesarias para su programación (variables de estado, estructuras de datos, funciones, librerías, etc.). A continuación se describen las máquinas de estado que determinan el funcionamiento de los módulos que pertenecen al nodo de cómputo.

6.3.1.1 Estados del módulo fuente *(src).*

La Fig. 6-6, muestra los estados y transiciones correspondientes al módulo diseñado que modela el procesador que inyecta trafico a la red. En el estado inicial *(init),* se leen los parámetros de entrada como la velocidad de inyección, el tamaño de paquetes y otros atributos de simulación como el tiempo de activación del nodo y el tiempo de comienzo de la inyección. También se configuran e inicializan las estadísticas a recolectar. Desde este estado inicial, son posibles tres transiciones que se derivan de tres eventos posibles: el comienzo de la inyección, o el fin de ésta (si es que el tiempo de inicio coincide con el del final de la simulación), o la desactivación, si se ha dispuesto que este proceso no genere paquetes. En el estado *idle*, el proceso queda inactivo durante toda la simulación y el nodo no genera paquetes de datos. En el estado *stop*, el nodo queda detenido de forma similar que en el caso anterior, con la diferencia que los nodos pueden tener diversos tiempos de acción durante la simulación, y algunos pueden acabar la inyección antes o después.

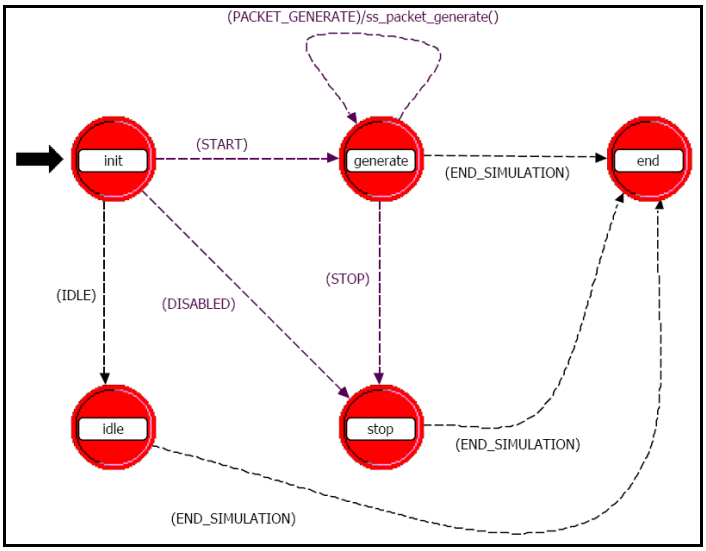


Fig. 6-6 FSM del proceso src

En el estado *generate* se crean los paquetes y se configuran sus campos (LID destino, canal virtual, nivel de servicio, tamaño, etcétera), y se generan los eventos para planificar su envío. Por último se calcula el tiempo inter-envío, en el cual se generará el próximo mensaje.

Cuando se alcanza el final de la simulación, se genera un evento que provoca la transición hacia el estado *end*, donde se recolectan estadísticas y se liberan los recursos utilizados.

6.3.1.2 Estados del módulo SLtoVL mapping unit.

En el estado *init*, se realiza la misma tarea que en el estado homónimo del caso anterior (y este es el caso general, pues como se ha visto es un estado creado para la definición e inicialización de variables y la lectura de parámetros de entrada). Esta máquina de estados se diseña para cumplir con dos funciones. La primera consiste en crear, completar y mantener actualizada la tabla de conversión que otorga el nivel de servicio requerido a la red de interconexión. La segunda y mas frecuente en la simulación, consiste en obtener el paquete de la cadena de datos *(packet stream)* en la entrada, leer el campo del nivel de servicio *(SL)*, simular un tiempo de retardo de búsqueda *(TIMEOUT)* en la tabla y obtener el canal virtual *(VL)*, por el cual el paquete ha de ser enviado. Para cumplir con estas tareas, se han creado transiciones que se asocian a los eventos que las llevan a cabo. La transición *ARRIVAL*, representa la llegada de un paquete y permite comenzar la búsqueda en la tabla, luego de un tiempo fijado como retardo de búsqueda, se planifica el evento que da lugar a la transición *TIMEOUT*, finalizando así la búsqueda y la asignación del canal virtual asociado al nivel de servicio que se fija en el campo SL del paquete. Las transiciones y los estados mencionados se pueden observar en la Fig. 6-7.

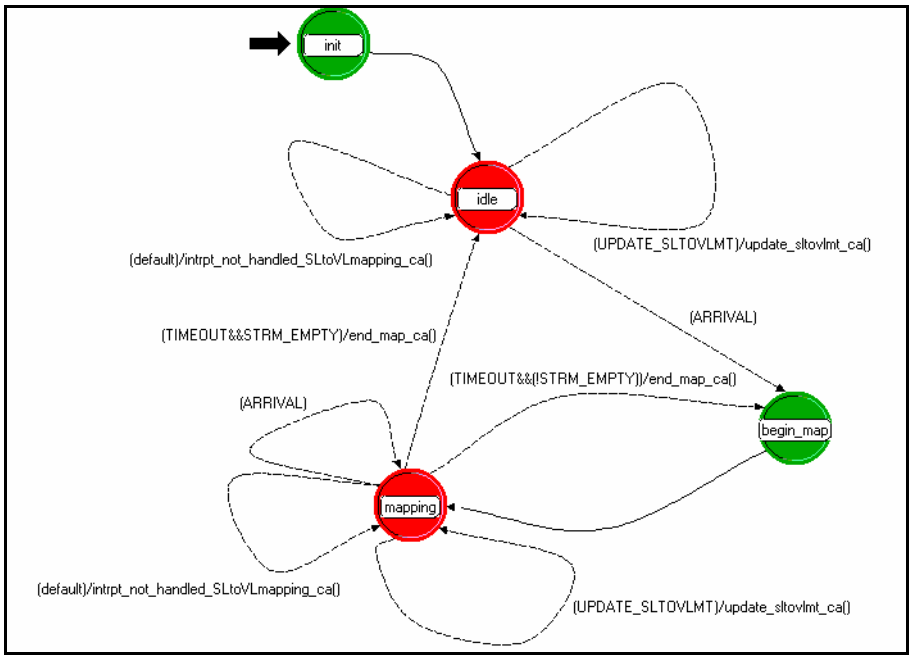


Fig. 6-7 FSM de la unidad de conversión

6.3.1.3 Estados del módulo VL arbitration unit.

La máquina de estados que caracteriza el funcionamiento de este módulo, implementa el arbitraje de acceso a los enlaces. Además de los paquetes de datos y de gestión, existe otro tipo de paquetes que, a diferencia de los anteriores, no transportan datos de la aplicación *(data packets),* o datos de configuración y gestión *(management packets)*.

Estos paquetes son denominados paquetes de control de flujo *(flow control packets)* y tienen importancia a bajo nivel (nivel físico), debido a que permiten establecer las condiciones para el avance de la información. Estos tres tipos diferentes de paquetes, circulan por los enlaces de la red de interconexión y comparten sus recursos. Es por esto que se ha implementado la unidad de arbitraje que permite seleccionar qué paquete será enviado hacia el puerto de salida teniendo en cuenta su prioridad[10].

La Fig. 6-8, muestra los estados y transiciones que conforman la máquina de estados de esta unidad.

[10] InfiniBand define que los paquetes de gestión, tienen la máxima prioridad de acceso al enlace, luego los paquetes de control de flujo y por último los paquetes de datos.

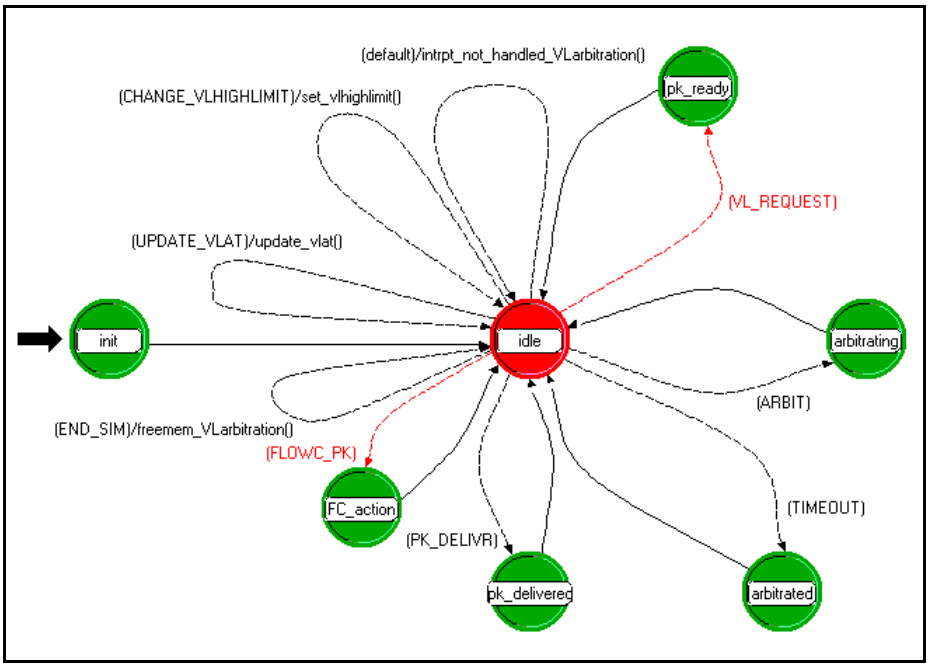


Fig. 6-8 FSM de la unidad de arbitraje

La transición *VL_REQUEST*, tiene lugar cuando un paquete situado en el buffer de transmisión está pidiendo acceso al enlace. Esta transición da lugar al estado *pk_ready* donde se observan los campos de dicho paquete y se obtienen sus características (tipo, canal virtual, tamaño) y se planifica un evento de arbitraje. Lo mismo ocurre cuando el paquete es de control de flujo, solo que el estado donde se analiza el paquete es FC_action, debido a que estos paquetes tiene características diferentes a los demás. El evento de arbitraje da lugar a la transición *ARBIT* que conduce el proceso al estado *arbitrating,* donde se evalúa el tipo de paquete y se planifica un tiempo de retardo, que simula el tiempo que demora la unidad en arbitrar un paquete (TIMEOUT). Los paquetes son enviados al puerto de salida según su prioridad. Una vez que ha transcurrido el tiempo de retardo en el arbitraje, el paquete se envía al transmisor mediante los estados *arbitrated* y *pk_delivered.*

6.3.1.4 Estados del módulo *SM.*

La máquina de estados finita que representa las acciones realizadas por el módulo SM, cumple las tareas que InfiniBand especifica para el mecanismo de gestión de subred. Como se ha descrito en el capitulo 3, el gestor de subred es el encargado de descubrir la topología de la subred, configurar cada puerto del adaptador de canal con un rango valido de LIDs, configurar cada conmutador con un LID, un prefijo de subred, y las tablas de encaminamiento. Por otra parte, debe mantener y gestionar las bases de datos de los servicios de gestión y los de la subred.

Las tareas mencionadas anteriormente, se realizan a través de transacciones utilizando paquetes de gestión *(SMPs)*; por ejemplo para designar el LID de un determinado

puerto, el SM envía un paquete con un campo destinado a tal efecto, hacia el agente de gestión (SMA) residente en ese nodo. Desde este punto de vista, las máquinas de estados con las que se implementa el componente de gestión de subred, consisten en generadores y receptores de paquetes de gestión.

Existen tres procesos destinados a tal fin. Un proceso "padre" y dos procesos "hijos", el proceso padre es invocado primero al inicio de la simulación para cumplir con una serie de tareas, y luego se invocan otros dos procesos (los hijos). Esta característica permite separar a procesos grandes y complejos, en otros más pequeños y facilitar de esta manera su realización y la comprensión de su funcionamiento.

En la Fig. 6-9, se representa el proceso padre denominado *dispatcher*, en este proceso primero se determina si la entidad de gestión (el SM), está o no residente en este nodo en función de un parámetro de entrada a la simulación. Si no lo esta el proceso queda inactivo, en el estado *no_sm*, hasta el final de la simulación, en caso contrario se realizan las funciones que activan al gestor de subred (en el estado *sm_active*), y se comienza con la realización de las tareas pertinentes, mediante la generación y recepción de los paquetes de gestión. En el estado *begin_proc*, se configuran los campos de los paquetes y se lleva a cabo esto último. Asimismo, el estado *begin_proc* invoca la creación de los estados hijos que permiten obtener y procesar la información recolectada por los paquetes de gestión.

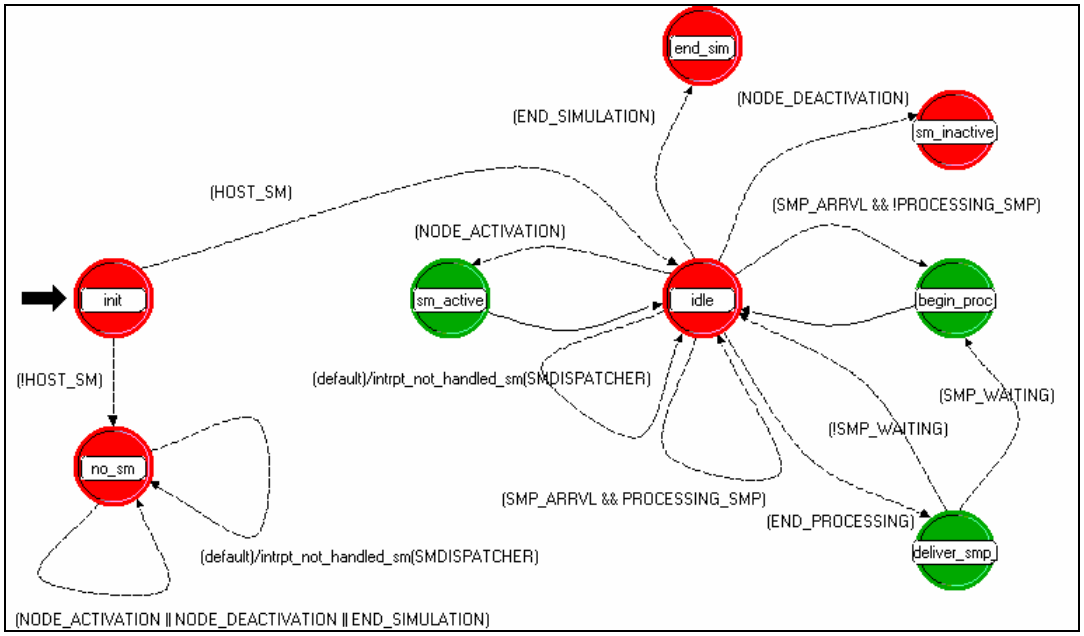


Fig. 6-9 FSM de proceso *dispatcher*

Uno de estos procesos hijo permite recavar información sobre diversos aspectos de los componentes de la red y conformar la topología, este proceso es denominado *discoverer* y se muestra a la izquierda de la Fig. 6-10. El otro proceso hijo, denominado *builder*,

construye las trayectorias a través de la información obtenida de la red, llevando a cabo un análisis que permita encontrar múltiples caminos cortos y disjuntos, entre los diversos pares de nodos fuente y destino de la red. En la parte derecha de la Fig. 6-10, se muestra la máquina de estados que utiliza este proceso.

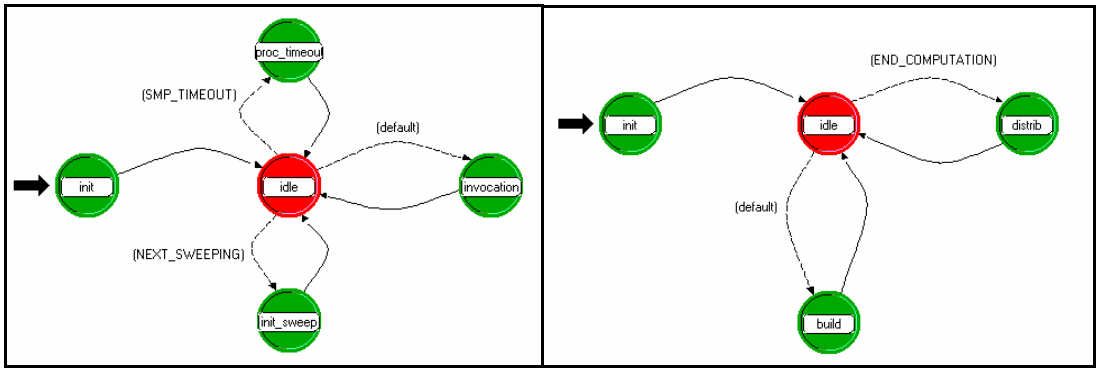


Fig. 6-10 FSMs de los procesos *discovery y builder*

Estos procesos se encargan de la asignación de múltiples nombres a los puertos de los CA's y de la construcción de caminos alternativos, que son características fundamentales para aplicar el balanceo distribuido del encaminamiento a la red.

Las transiciones denominadas *default* representan la invocación desde el proceso padre. Cada vez que se invoca el proceso *discoverer*, se genera un evento que da lugar al descubrimiento de la red. Este evento genera una transición (*NEXT_SWEEPING*) hacia el estado *init_sweep*, donde se configuran las acciones destinadas a tal descubrimiento. El SM recorre la red a intervalos regulares de tiempo (fijado como atributo de la simulación) para monitorizar cambios en la topología o sus componentes. Cada vez que un nodo de la red se activa o desactiva, se invoca a este proceso y se actualiza la información de la red.

Cuando se realiza la invocación del proceso *builder*, se configura el camino múltiple para todos los pares de nodos fuente y destino de la red mediante un algoritmo de búsqueda y selección de trayectorias disjuntas seleccionadas por su longitud implementado en el estado *build*. Una vez terminada la configuración de trayectorias se planifica un evento que da lugar a la transición hacia el estado *distrib*, donde se generan los paquetes necesarios para distribuir esta información, a las tablas de encaminamiento de los conmutadores de la red.

6.3.1.5 Estados del módulo *CCMgtA.*

La máquina de estados que establece el comportamiento del agente de control de congestión, ha sido diseñada para cumplir con las características de diseño especificadas, en cada fase de la propuesta de balanceo distribuido del encaminamiento visto en capitulo 4.

Por esta razón, se han creado cuatro estados, que permiten analizar los paquetes marcados por los conmutadores que detectan congestión, generar los paquetes de notificación correspondientes, configurar y seleccionar los caminos alternativos, y contraer trayectorias cuando no sean necesarias. Estos estados y las transiciones entre ellos pueden verse en la Fig. 6-11

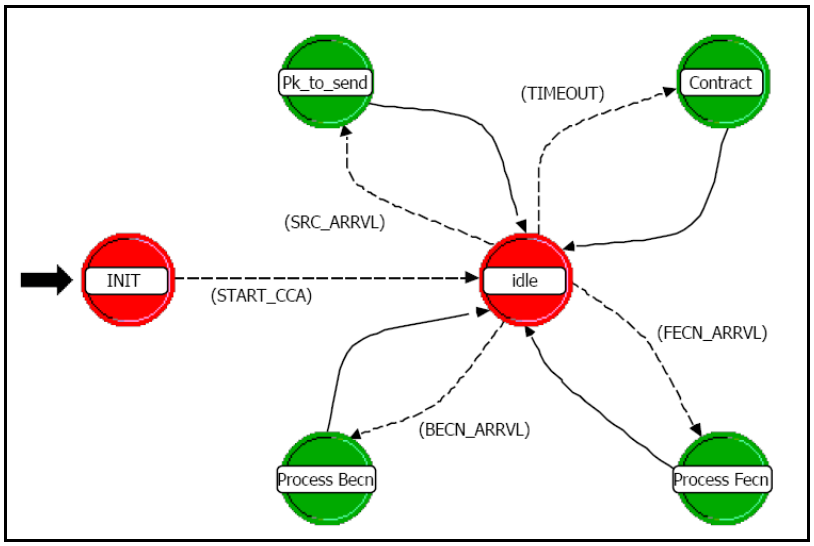


Fig. 6-11 FSM del proceso CCMgtA

El funcionamiento comienza con el evento que determina la habilitación del módulo, este evento esta asociado a la transición START_CCA.

Como se ha mencionado previamente, cada vez que un conmutador detecta congestión, marca el bit FECN correspondiente en los paquetes contenidos en sus buffers. Cuando estos paquetes llegan al adaptador de canal generan un evento que provoca la transición al estado *Process Fecn*, donde se analiza el paquete recibido, se actualizan las estadísticas y se genera un paquete de notificación *(CN)*. Dentro del paquete de notificación se activa el bit BECN y se invierten los campos fuente y destino; de esta manera se notifica al nodo que envió el paquete la existencia de congestión.

Por lo tanto, cada vez que llega un CN al adaptador de canal, se genera un evento que produce la transición hacia el estado *Process Becn*, donde se realiza el procesamiento de dichos paquetes, se determina la ocupación en los caminos alternativos, y se selecciona la inyección a través de éstos, utilizando el criterio explicado en el capitulo 4. Una vez que los caminos alternativos se hayan configurado y seleccionado según su ocupación, se utilizarán para el encaminamiento de paquetes y, por tanto, cada vez que un nodo fuente requiere el envío de un paquete, se genera el evento que permite la transición al estado Pk_to_send, donde se modifican los campos del paquete, involucrados en el encaminamiento. La modificación se realiza según la selección de caminos realizados en el estado, *Process_becn.*

La transición etiquetada como TIMEOUT, se utiliza para llevar al proceso al estado *contract*. El evento asociado con esta transición, esta basado en el tiempo de expiración del contador de performance, este valor de tiempo es pasado como parámetro al simulador y cada vez que se cumple un ciclo de conteo, se ingresa al estado *contract* donde se analiza la cantidad de paquetes de notificación recibidos y se decide la constricción de caminos, en función de esta cantidad.

6.3.2 Modelo del conmutador.

Los conmutadores al igual que los nodos de cómputo, también han sido modelados para cumplir las especificaciones de la arquitectura InfiniBand. Los diferentes módulos que componen el modelo del conmutador, se agrupan en diversas regiones según su funcionalidad y sus características de operación. En la Fig. 6-12, se muestra la implementación del conmutador y las regiones que lo conforman.

La Fig. 6-12 muestra la implementación de un conmutador de 5 puertos bidireccionales, con tres canales virtuales y las unidades necesarias para el encaminamiento de paquetes.

Es posible observar que algunos módulos implementados en este modelo también han sido utilizados en el modelo del adaptador de canal; tal es el caso de los buffers, los componentes de gestión, la unidad de conversión de nivel de servicio, etcétera.

Sin embargo, aparecen regiones nuevas, dos de las cuales merecen especial interés. Estas regiones son la denominadas *crossbar* y *routing*, cuyos módulos han sido creados para simular el comportamiento de la unidad de conmutación (*crossbar*), que permite conectar todas la entradas y las salidas entre sí, y la lógica que determina la búsqueda en la tabla de encaminamiento (*routing*) del puerto de salida por el que se enviará el paquete. Las máquinas de estado que conforman estas regiones, se describen a continuación.

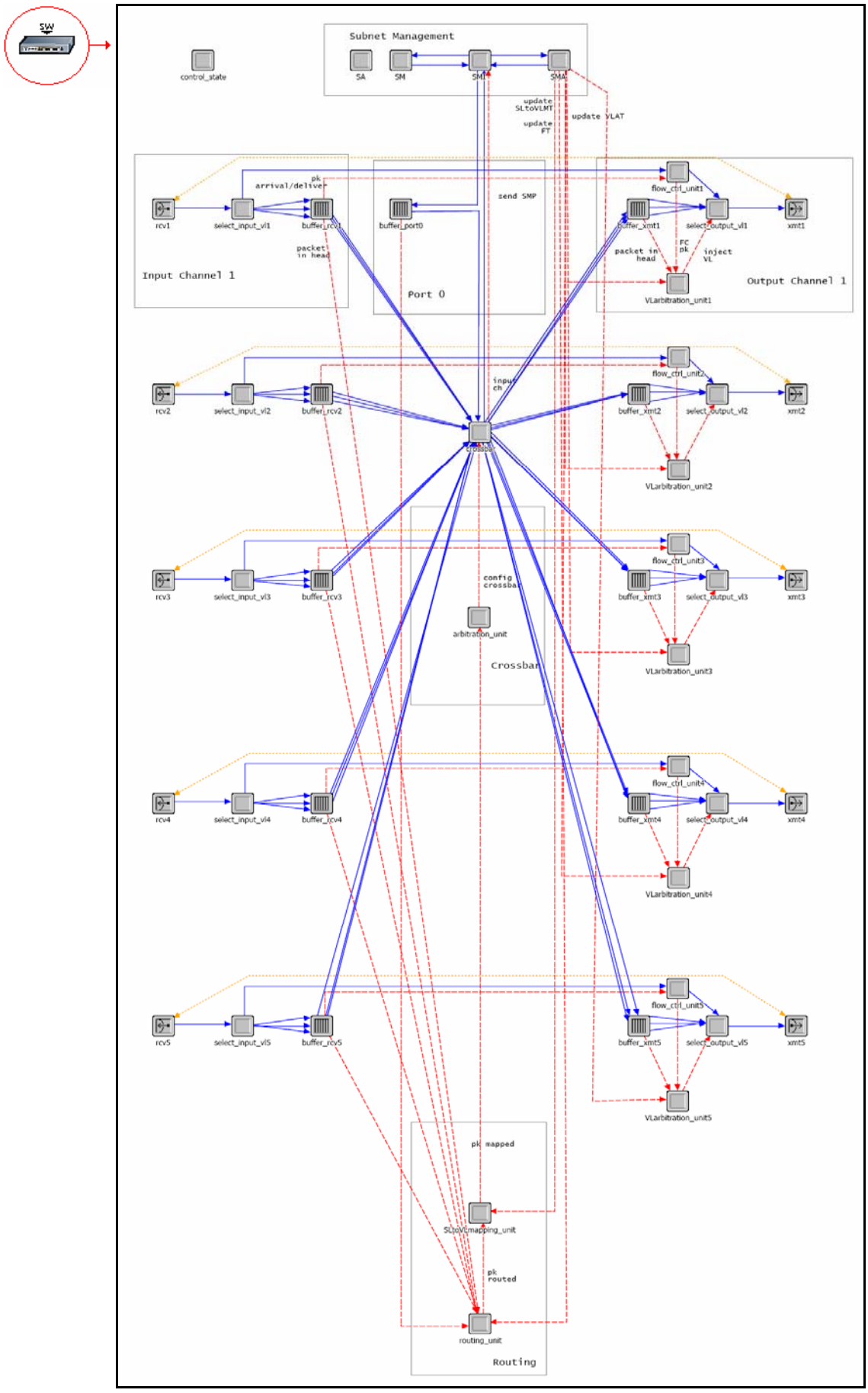


Fig. 6-12 Modelo del conmutador

6.3.2.1 Estados del módulo *Crossbar.*

Cada vez que la unidad de arbitraje detecta la presencia de un paquete en uno de los buffers de entrada del conmutador, genera un evento que permite configurar al *crossbar* y establecer la conexión adecuada entre los puertos de entrada y salida.

La transición denominada *CONFIG_CROSSBAR* (ver Fig. 6-13) se asigna a este evento y permite que el proceso avance hacia el estado *config*, donde se determina en puerto de entrada y el de salida y se obtiene la información del paquete enviada por la unidad de arbitraje. A continuación, se planifica un evento que permita simular el tiempo de cruce del paquete a través de la unidad, el valor de este tiempo se especifica como parámetro del simulador mediante un atributo denominado *crossing_time*.

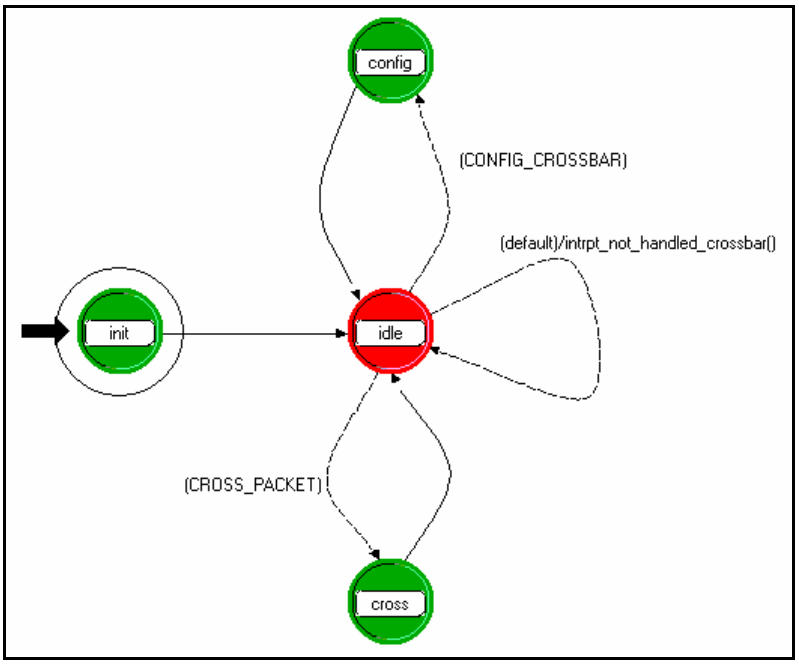


Fig. 6-13 FSM de la unidad crossbar

Una vez transcurrido el tiempo de cruce, el proceso se traslada al estado *cross*, donde el paquete avanza desde el puerto de entrada hacia el puerto de salida a través del *crossbar*.

6.3.2.2 Estados del módulo *Routing unit.*

La función principal del módulo *"routing"* es determinar por qué puerto de salida se ha de encaminar el paquete. Por este motivo cada vez que llega un paquete al buffer de recepción del conmutador se genera un evento de llegada que permite avanzar hacia el estado *route_pk*, según se observa en la Fig. 6-14.

En este estado, se verifica el canal de entrada y se obtienen los campos destino, nivel de servicio y largo del paquete. Con esta información se lleva a cabo la búsqueda del puerto de salida, dentro de la tabla de encaminamiento, por el cual enviar dicho paquete. Por ultimo se lee un atributo de entrada, que permite simular *el tiempo de búsqueda* en la tabla y se planifica un evento en función de su valor.

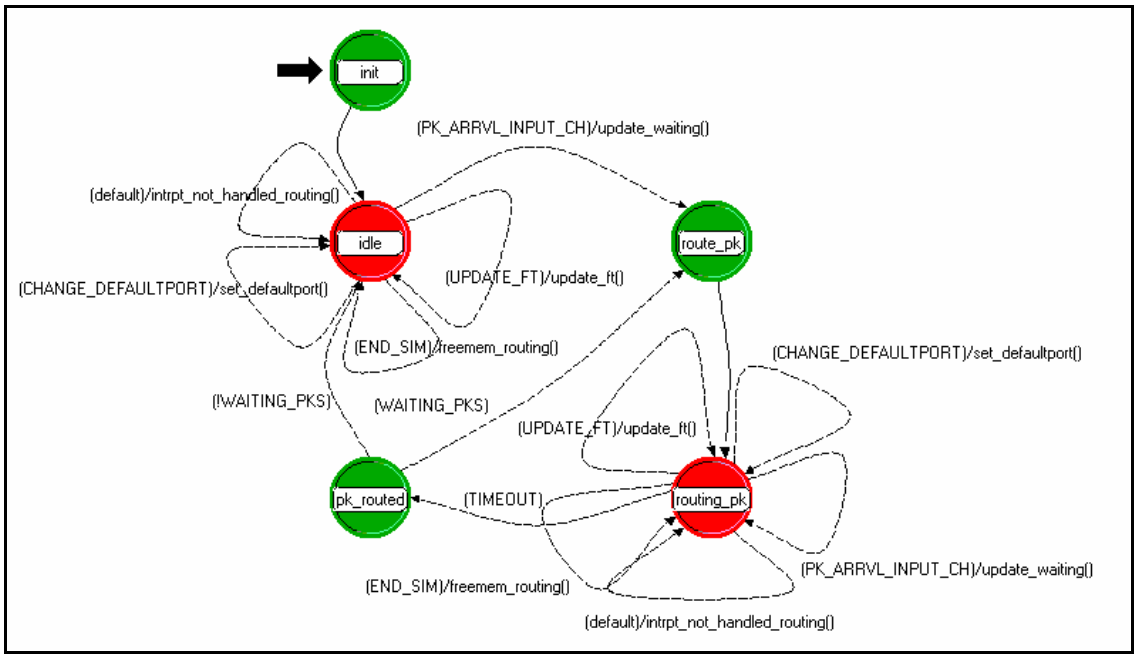


Fig. 6-14 FSM de la unidad de encaminamiento

Cuando el tiempo de búsqueda finaliza, se da lugar a la transición denominada *TIMEOUT*, que termina de encaminar el paquete hacia el puerto de salida especificado en la tabla de encaminamiento, informando a la tabla de conversión de nivel de servicio *(SltoVL mapping table)* por si es necesario realizar la conversión. En funcionamiento normal ningún paquete se descarta, debido a que el funcionamiento de las redes de interconexión de altas prestaciones así lo requiere, sin embargo ante condiciones excepcionales el descarte es posible. Estos casos se dan cuando:

1) No se encuentra el LID destino en la tabla de encaminamiento.

2) Cuando el puerto de salida que figura en la tabla de encaminamiento no es valido o no está soportado.

3) Cuando el puerto de salida que figura en la tabla de encaminamiento es igual al de entrada.

4) Cuando el puerto de salida corresponde al puerto de gestión (puerto 0), pero el paquete ha ingresado por el puerto de entrada mediante un canal virtual de datos.

6.3.2.3 Estados del módulo *Buffer_rcv* y del módulo *Buffer_xmt.*

Esta máquina de estados modela el funcionamiento típico de una cola FIFO *(First In First out)*, con un agregado especial para simular la detección de congestión en los buffers y completar, de esta manera la implementación del balanceo distribuido del encaminamiento sobre la arquitectura InfiniBand.

Los estados y transiciones representados en la Fig. 6-15, modelan el comportamiento de un *buffer* dentro del conmutador (El modelo de buffers utilizado en los CAs es similar, con la salvedad que no contienen el mecanismo de detección de congestión, pues no es necesario). Existe un buffer por canal virtual, por lo tanto, como hay tres canales virtuales implementados, hay tres *buffers* por puerto.

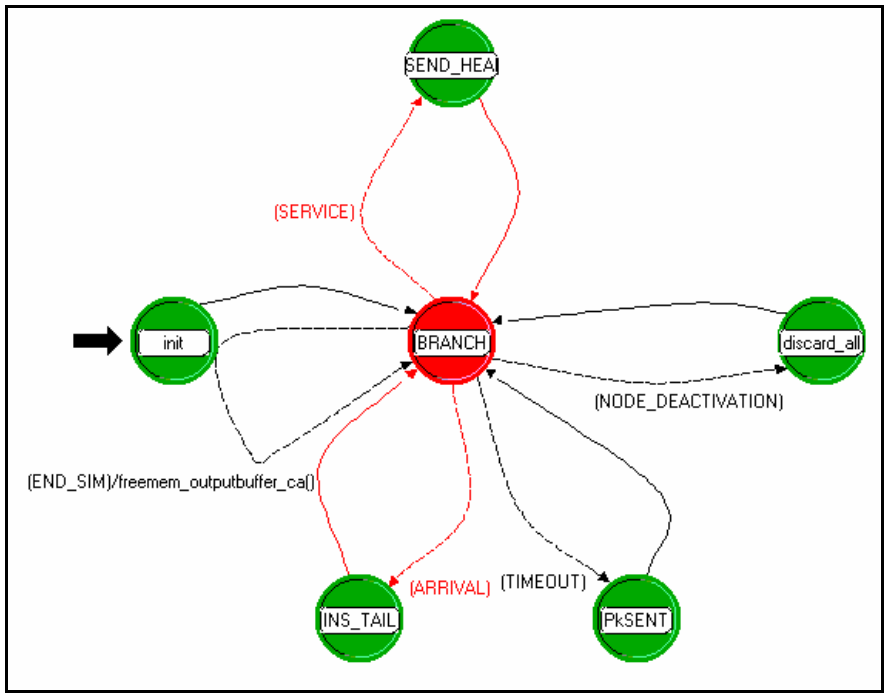


Fig. 6-15 FSM de los buffers

Cada vez que llega un paquete al buffer de un canal virtual, se genera un evento que permite el avance del proceso hacia el estado *ins_tail*. En este estado se verifica el tamaño del paquete y se coloca en la cola del buffer (la ultima posición), eventualmente el paquete alcanza la cabeza del buffer (la primera posición) y debe ser enviado. Cuando ocurre esta situación, se establece un evento relacionado con la transición *SERVICE* que determina que el proceso avance hacia el estado *send_head*,

Dentro de este estado, los paquetes se retiran del buffer del canal virtual correspondiente y se planifica un evento de envío hacia el puerto correspondiente. Este evento no se atiende inmediatamente, sino que se genera luego de un *tiempo de retardo* destinado a modelar el tiempo de servicio de la cola.

Por otra parte, cada vez que un paquete alcanza la cola dentro del estado *send_head*, se monitoriza el tamaño de dicha cola con el objetivo de marcar los bits FECN, ubicados en la cabecera de los paquetes. Esto permite notificar la existencia de congestión en el buffer del conmutador. Los paquetes se marcan mediante la comparación con un valor umbral que se pasa como parámetro al simulador.

A lo largo de este capitulo, se ha presentado el trabajo realizado con la plataforma de evaluación que ha hecho posible, la aplicación del balanceo distribuido del encaminamiento sobre redes InfiniBand. Mediante el desarrollo basado en diferentes niveles de jerarquía, se han mostrado los componentes que permiten crear diversos modelos de red y conformar un rico conjunto de topologías, con el fin de evaluar las bondades de la técnica utilizada para el encaminamiento de mensajes. También se ha visto, como se interconectan los módulos que componen a cada uno de los componentes de la red, y que establecen el flujo de datos y el intercambio de información entre éstos. El comportamiento lógico y dinámico de cada módulo, se determina mediante la elaboración de una máquina de estados; es por esto que se ha dedicado gran parte del capitulo a su explicación. A su vez el conjunto de módulos da "vida" a cada componente y que en conjunto con los diversos atributos de cada nodo, permiten realizar las simulaciones que hacen posible el análisis de la propuesta establecida en este trabajo.

Las pruebas y experimentos que dan lugar a este análisis, junto al conjunto de métricas y patrones de carga utilizados para tal fin, se presentan el capitulo siguiente.

Capitulo 7 Experimentación y análisis de rendimiento

7.1 Introducción

En este capítulo se presentan los experimentos realizados sobre la técnica de balanceo distribuido del encaminamiento en redes InfiniBand y se miden las prestaciones dinámicas de *DRB* en presencia de una carga real de mensajes en la red de interconexión. Para realizar esta evaluación se ha llevado a cabo una experimentación basada en la simulación, utilizando el simulador Opnet Modeler presentado en el capítulo 5 como herramienta de prueba.

Para realizar la experimentación, es necesario disponer de una carga de tráfico de entrada que permita simular el funcionamiento real de las aplicaciones paralelas. El tráfico de mensajes generado por las aplicaciones puede obtenerse de las aplicaciones reales o de los bien conocidos "*benchmarks*", que consisten en un conjunto de casos de prueba específicamente seleccionados.

En este trabajo se ha decidido utilizar los patrones de trafico *("workloads")* generados mediante un conjunto de "*benchmarks*", por las ventajas que presentan. Este tipo de patrones de carga son más representativos y permiten abarcar un amplio rango de situaciones. Por otra parte, presentan una gran versatilidad que posibilita parametrizar su funcionamiento y medir características específicas de la red de interconexión bajo estudio. Por otro lado, de ellos son bien conocidas las características de comunicación que presentan y, por tanto, los resultados obtenidos son factibles de ser analizados y se pueden extraer conclusiones válidas sobre ellos.

Se han utilizado un conjunto de patrones de comunicación que sintetizan el funcionamiento de los programas reales de las aplicaciones científicas y técnicas paralelas más comunes. Estos patrones tienen la propiedad de utilizar la red de interconexión de manera extensiva, con lo cual es posible observar el efecto y el comportamiento global, que tiene el algoritmo de encaminamiento DRB sobre la arquitectura InfiniBand.

La experimentación permite realizar analizar comparativos y por tanto se ha evaluado el comportamiento del control de congestión propuesto, frente al que ofrece la arquitectura

InfiniBand. Ambos mecanismos se han analizado en detalle en el capitulo 4 de este documento.

El enfoque establecido en esta experimentación consiste en dos puntos principales. El primero esta basado en la respuesta de la latencia ante diversos patrones de comunicación tomados de aplicaciones numéricas. Estos patrones son: *"Butterfly", "Bit-Reversal", "Complement", "Perfect Shuffle" y "Matrix Transpose".* En segundo lugar, se ha evaluado la respuesta ante un patrón de comunicaciones que provoca la aparición de un punto caliente *("hot-spot")* en la red. En todos los casos, se evalúa la respuesta de la latencia de transporte y el "*throughput*" de mensajes. Además de los mencionados, se ha evaluado otro aspecto que tiene impacto en las prestaciones de la técnica de control de congestión que se implementado; este factor es la influencia de la longitud del mensaje.

Todos los experimentos mencionados, se han llevado a cabo utilizando la plataforma de simulación de redes de interconexión presentada en el capítulo 5. Por este motivo, se han incorporado a la plataforma mencionada, todos los aspectos de funcionamiento del balanceo distribuido del encaminamiento *DRB* descrito en el capítulo 4, así como, los modelos y la funcionalidad establecidos en la especificación InfiniBand. A continuación se presenta cada uno de los aspectos mencionados en el párrafo anterior.

7.2 Evaluación de prestaciones

La experimentación realizada en este trabajo queda definida según la caracterización de la carga de tráfico que se inyecta a la red, las características que conforman el espacio de diseño de la red de interconexión, las herramientas utilizadas para simular y evaluar el funcionamiento, y la metodología utilizada en la experimentación con el fin de obtener resultados, procesarlos y presentarlos en forma adecuada.

En virtud a lo mencionado, se han elegido parámetros representativos en el marco de las redes de interconexión. Estos parámetros son la topología, el tamaño de la red, el patrón de comunicaciones y el tamaño de los mensajes.

El conjunto de experimentos a realizar representa un espacio de pruebas con las propiedades adecuadas para realizar una evaluación descriptiva. El conjunto de patrones de tráfico seleccionados establece estas propiedades y acerca la evaluación al funcionamiento de aplicaciones reales.

A continuación, se precisan cada uno de los aspectos que definen la experimentación.

7.2.1 Las redes simuladas

En la experimentación, se han utilizado un conjunto de redes de interconexión directas como las presentadas en el capítulo 2, con la finalidad de evaluar las bondades de la técnica propuesta. Por lo tanto se han desarrollado redes con topologías toro y malla (o grid) de diversos tamaños. Como técnica de control del flujo, se ha implementado *"Virtual Cut Throught"* según especifica la arquitectura InfiniBand.

7.2.2 Características de la carga y los patrones de tráfico utilizados

La carga de comunicaciones se especifica por medio de patrones de tráfico sintéticos, que definen los nodos entre los que se envían mensajes, junto a la longitud y la frecuencia de generación de estos mensajes. La frecuencia de generación esta determinada por el tiempo medio de envío de dos mensajes, y se ha implementado mediante una función de distribución de probabilidad exponencial cuya media es el tiempo de envío mencionado. Esta distribución corresponde al envío de mensajes independientes entre sí siguiendo un modelo de proceso de Poisson.

El rango de carga de tráfico, en los experimentos se varía desde valores bajos hasta la saturación. Este rango comienza con valores menores al 10% del ancho de banda máximo[11] especificado por InfiniBand y termina con valores que alcanzan el 120%. Por otra parte, se han elegido diferentes tamaños de paquetes. InfiniBand define un conjunto de valores determinado para los paquetes, estos valores son: 4096, 2048, 1024 512 y 256 bytes, y se han realizado experimentos para los valores de 4096, 1024 y 256 bytes con el objeto de obtener simulaciones representativas.

En la nomenclatura utilizada en las redes de interconexión, se denomina carga de trafico ofrecido *("Offered Load")*, a la velocidad de inyección de los mensajes generados. No obstante, cuando la red llega a la saturación no absorbe toda la carga generada por los nodos, y se produce el rechazo de parte de ella. La parte de la carga inyectada que no es rechazada y circula por la red, se llama denomina *carga aceptada* ("*Accepted Load*"), y como se ha mencionado, en caso de saturación es menor que la carga aplicada. En la experimentación aquí realizada los resultados se presentan en base a la carga aceptada, ya que este último es el valor que realmente esta circulando por la red.

Los patrones de carga de tráfico utilizados, representan el volumen de comunicaciones generado por operaciones de cómputo numérico que se utilizan en un amplio conjunto de aplicaciones. Como ejemplo es posible citar: la transformada rápida de Fourier, la convolución, las operaciones con matrices y el calculo diferencial [22].

Como se ha mencionado estos patrones son: *"Butterfly", "Complement", "Bit-*

[11] El ancho de banda máximo es de 2Gbps, según se ha visto en el capitulo 3.

Reversal", *"Perfect Shuffle"* y *"Matrix Transpose"* [58]. Su operación se basa en la definición de una permutación entre todos los nodos de la red, de manera que cada uno de estos nodos envíe mensajes a algún otro nodo en la red.

La implementación de los patrones de tráfico mencionados se realiza mediante la asignación de un número binario a cada nodo y una transformación del mismo. Las transformaciones que realiza cada uno de los patrones, sobre el número binario asignado, que permiten para determinar el nodo destino son las siguientes 160:

Sea el nodo fuente formado por *n* coordenadas { a_{n-1}, $a_{n-2,}$ ..., a_1, a_0}*(Ej. 1100, para n=4).*

El patrón *"Butterfly"* se forma intercambiando los bits más y menos significativos: el nodo con coordenadas binarias a_{n-1}, $a_{n-2,}$ ..., a_1, a_0 se comunica con el nodo Destino(*Butterfly*)= {a_0, $a_{n-2,}$ ..., a_1, a_{n-1}} *(ej. el nodo 1100 envía al destino 0101).*

En el patrón *"Bit-Reversal"* el nodo con coordenadas binarias a_{n-1}, $a_{n-2,}$ ..., a_1, a_0 se comunica con el nodo Destino(*Bit-Reversal*)= {a_0, $a_{1,}$ ..., a_{n-2}, a_{n-1}}. *(ej. el nodo 1100 envía al destino 0011).*

El patrón *"Perfect Shuffle"* rota un bit a la izquierda: el nodo con coordenadas binarias a_{n-1}, $a_{n-2,}$ ..., a_1, a_0 se comunica con el nodo Destino(*Perfect Shuffle*)= {a_{n-2}, $a_{n-3,}$ ..., a_0, a_{n-1}}. *(ej. el nodo 1100 envía al destino 1001).*

El patrón *"Complement"* consiste en intercambiar los dígitos de manera que el nodo con coordenadas binarias a_{n-1}, $a_{n-2,}$..., a_1, a_0 se comunica con el nodo Destino(*Complement*)= { $\overline{a_{n-1}, a_{n-2,} ..., a_1, a_0}$ }. *(ej. el nodo 1110 envía al destino 0001).*

En el patrón *"Matrix Transpose"* el nodo con coordenadas binarias a_{n-1}, $a_{n-2,}$..., a_1, a_0 se comunica con el nodo Destino(*Matrix Transpose*) = {$a_{n/2-1,}$...,a_0, $a_{n-1,}$...,$a_{n/2}$}. *(ej. el nodo 1001 envía al destino 0110).*

Por otra parte, empleamos también un patrón de generación de "*hot-spot*" específicamente seleccionado. En este patrón, se determinan un conjunto de enlaces que comparten un fragmento del camino que recorren y por tanto produce una gran concentración de paquetes en esos canales (ver Fig. 7-1). Este patrón se evalúa sobre una red toroidal de 64 nodos.

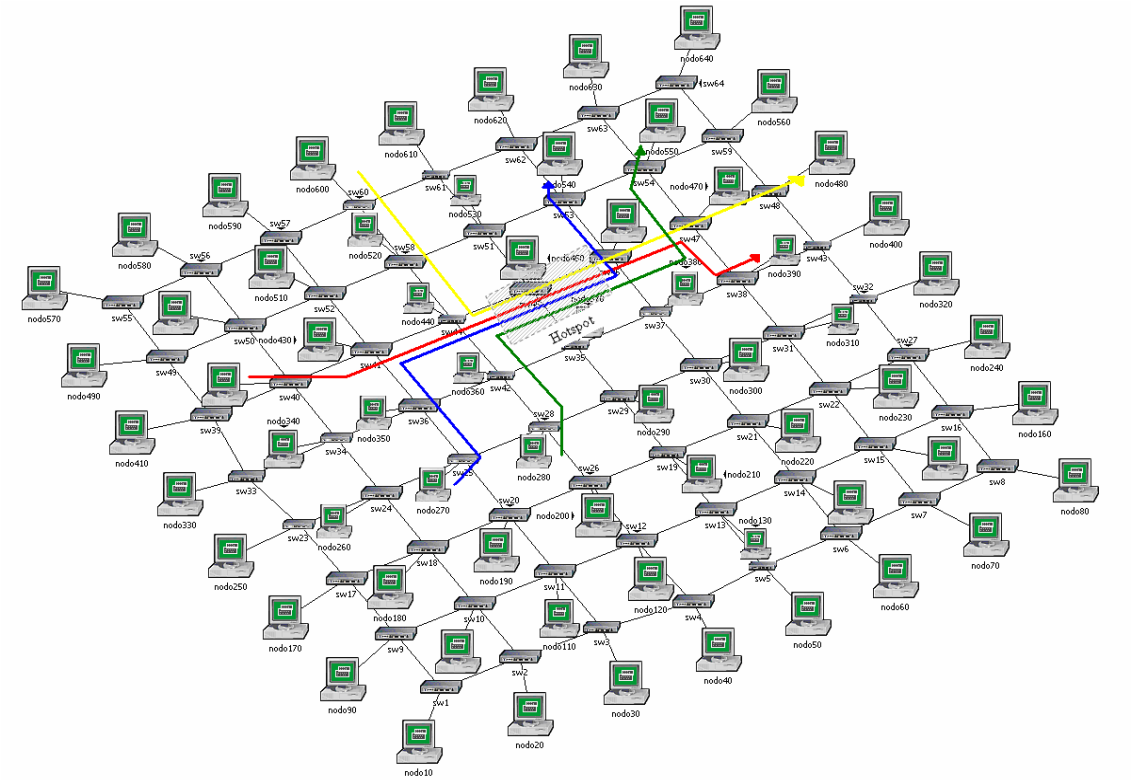


Fig. 7-1 Patrón hot-spot

7.2.3 Técnicas de control de congestión

Las técnicas empleadas en la experimentación son el balanceo distribuido del encaminamiento *DRB* y el mecanismo de control de congestión propuesto por la arquitectura. Ambos han sido explicados en detalle en el capitulo 4. Para realizar la experimentación, se ha configurado en cada caso, el conjunto de características y operaciones requeridas para encontrar los caminos alternativos que representen la mejor utilización de la red. Concretamente, se han utilizado caminos múltiples compuestos por hasta cuatro trayectorias disjuntas y de largo mínimo. El algoritmo de encaminamiento esta basado en el nodo destino según especifica el estándar InfiniBand.

7.2.4 Metodología de trabajo

La experimentación realizada desarrolla varios puntos. En el primero consiste en evaluar DRB para un conjunto de redes de interconexión de diversos tamaños (toros y mallas) y de patrones de comunicación. La experimentación se realiza de forma exhaustiva y se enfoca en dos puntos principales:

- La respuesta en latencia a patrones de comunicación persistentes tomados de aplicaciones numéricas (*"Butterfly"*, *"Bit-Reversal"*, *"Perfect Shuffle"* y *"Matrix Transpose"*)

- La capacidad de entrega de mensajes en la red "*throughput*", que determina el volumen promedio de carga recibido en los nodos destinos con respecto al ofrecido por el patrón de comunicaciones.

El segundo punto, consiste en la evaluación de la respuesta de la red de interconexión respecto a un patrón de comunicaciones que provoca la aparición de un "*hot-spot*". En este análisis se evalúa la respuesta instantánea de la red, tanto para la latencia de transporte, como para la utilización de los enlaces congestionados

También hemos evaluado la influencia y el impacto en las prestaciones del sistema con respecto a la longitud de los mensajes inyectados dentro de la red de interconexión.

7.3 Resultados medidos

Como hemos comentado los resultados de los experimentos, se basan en la medición de tres magnitudes: Primero, la latencia media de las comunicaciones y, segundo, la utilización o "*throughput*" medio en la red de interconexión y por último, la distribución de paquetes en la red.

La *latencia de comunicación* se mide como el tiempo total transcurrido desde que el primer bit del paquete es inyectado a la red por el nodo fuente, hasta el último bit del mismo es recibido en el nodo destino. Para un conjunto de valores de latencia L_i medidos, la "latencia media" $\overline{L}$ se calcula a través del promedio las latencias de todos los mensajes recibidos en cada nodo y se mide en milisegundos:

$$\overline{L} = \frac{1}{n}\sum_{i=1}^{n} L_i$$

El *"throughput"* se mide como la relación porcentual entre la carga recibida (cantidad de información entregada) y la carga de comunicación ofrecida según se ha establecido. Ambas cargas de comunicación se miden bits por unidad de tiempo, en nuestro caso bits por microsegundo (bits/μs).

Finalmente, con objeto de mostrar la *distribución de la carga* de los mensajes en la red, se calcula la utilización *en cada enlace* de la red. Esta es una medida por enlace y no por nodo y se muestra para la experimentación realizada con el patrón de "*hot-spot*".

7.3.1 Experimentación con patrones de comunicación

En este punto, se presentan los resultados obtenidos en la experimentación realizada. Primero se analiza la respuesta ante los patrones sintéticos, luego ante la generación de hotspot y por ultimo la influencia del tamaño del mensaje sobre la red de interconexión.

7.3.1.1 Experimentación con patrones sintéticos.

Los experimentos se han realizado en topologías toros y mallas para tres tamaños: 16, 32 y 64 nodos. En los cinco patrones de comunicación utilizados, el desbalanceo de la carga de mensajes de comunicación es importante y el comportamiento de la técnica DRB y la técnica de control de congestión InfiniBand *(IBA_CC)*, difieren en gran manera.

La tabla siguiente muestra el resumen técnico de la experimentación realizada con los patrones sintéticos de comunicación.

Topología	Tamaño (Nodos)	Patrones	Técnica de control	Figura
Toro	4x4 (16)	Butterfly	IBA_CC	Fig. 7-2
	8x8 (64)	Reversal	DRB	Fig. 7-3
Malla	4x4 (16)	Shuffle		Fig. 7-4
	8x4 (32)	Transpose		Fig. 7-5
		Complement		

Análisis de resultados

Topología Toro.

Los resultados son similares para los cuatro patrones, así que se analizaran de manera conjunta. En general, DRB ofrece mejores resultados que la técnica de control IBA. La diferencia entre las curvas para cada patrón se incrementa a medida que lo hace la carga de tráfico de la red. Se puede observar que a cargas bajas (un ancho de banda menor que 500 bits/μs), las técnicas propuestas se comportan de forma similar. Esto es importante porque implica que DRB no cambia el comportamiento de la red cuando no es necesario, de manera que no introduce ninguna sobrecarga *(overhead)* con demandas bajas de inyección.

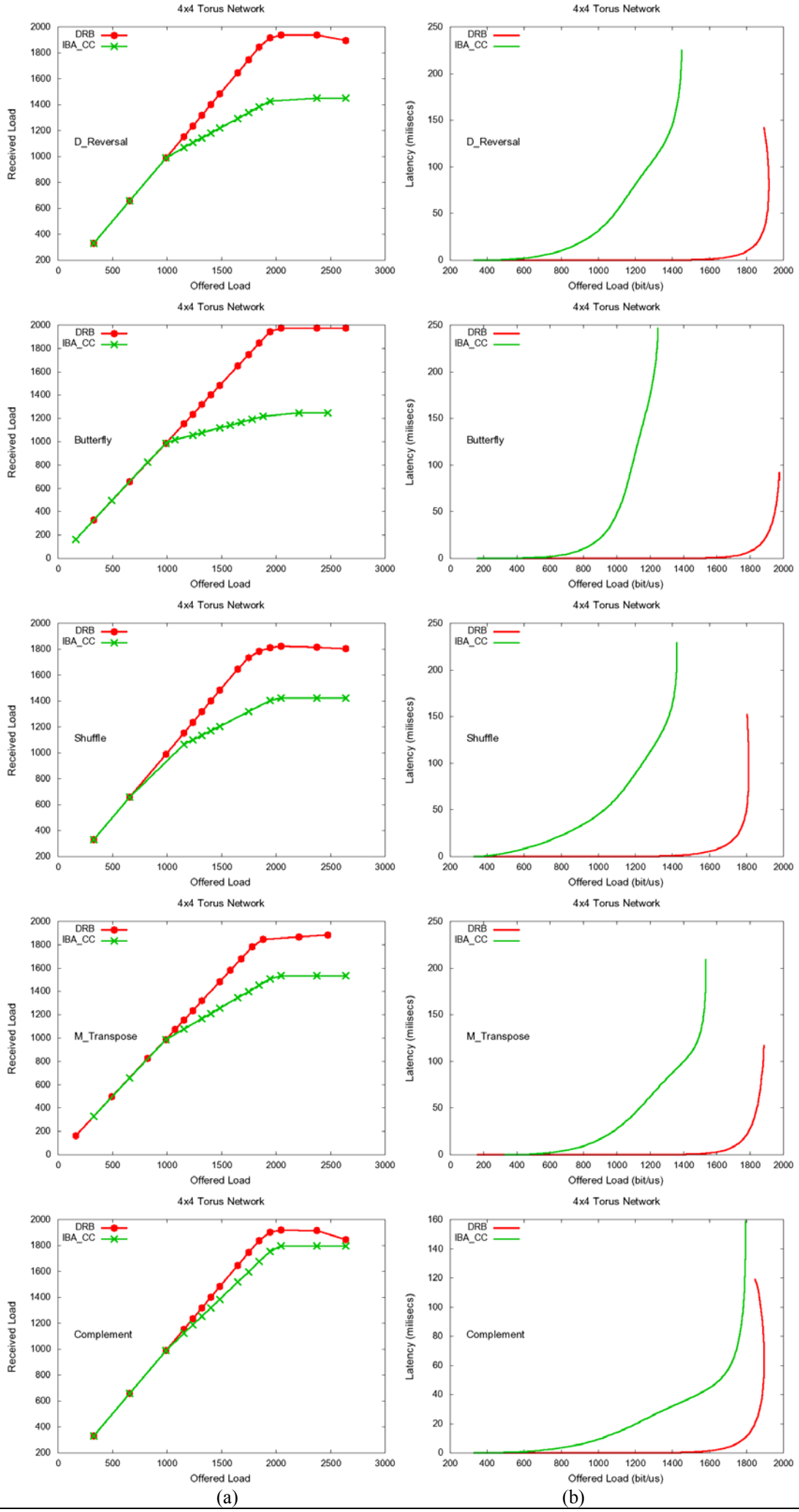


Fig. 7-2 Rendimiento sobre el toro 4x4

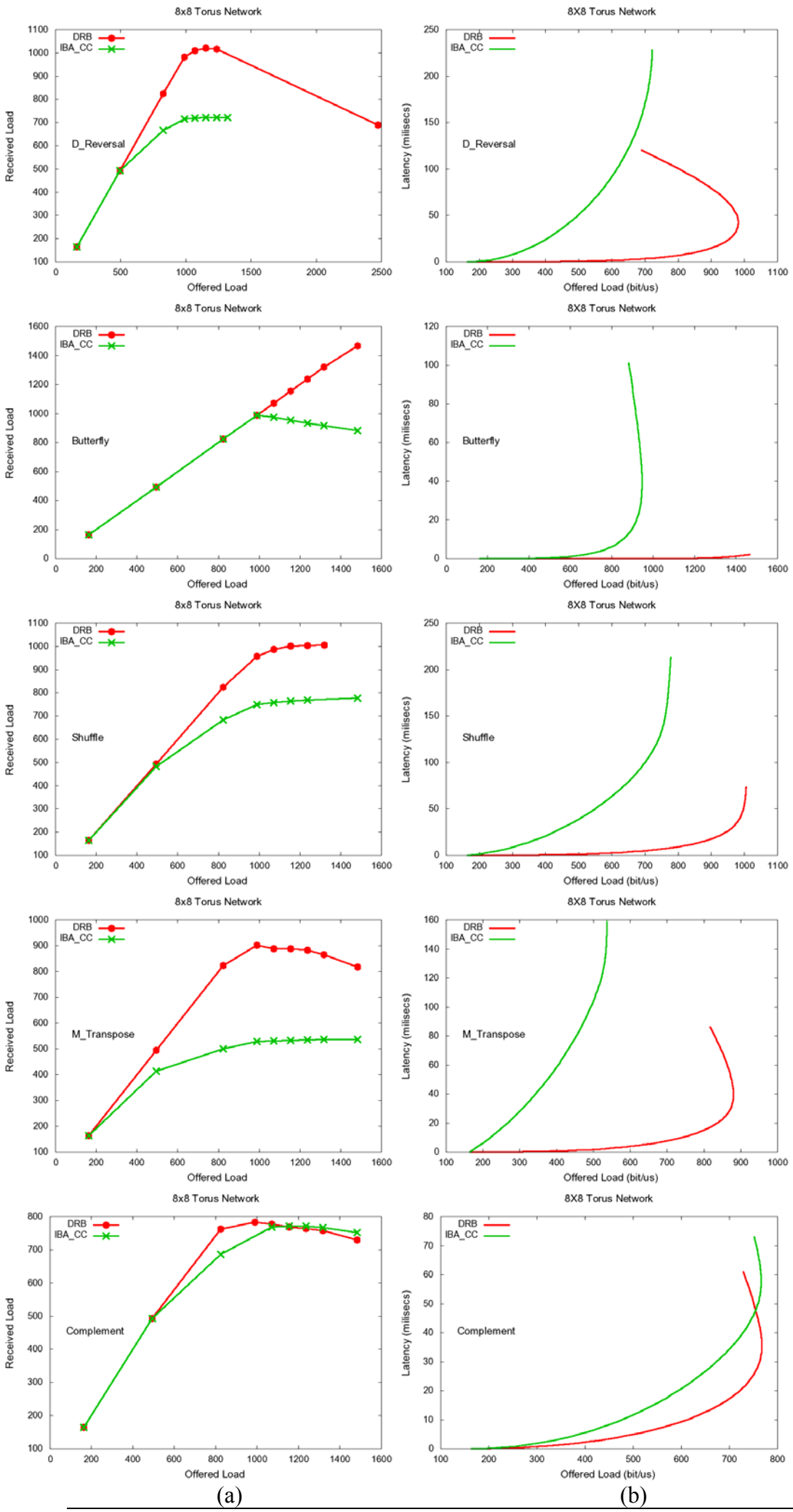


Fig. 7-3 Rendimiento sobre el toro 8x8

Cuando la inyección de trafico en la red se incrementa, con cargas entre 500 y 900 bits/µs, el incremento de la latencia (Fig. 7-2 (b)) obtenido con DRB es notablemente inferior al que se obtiene con la técnica InfiniBand, esto es debido a que, en DRB, los nodos fuente comienzan a utilizar caminos alternativos para el envío de paquetes, mientras que con IBA_CC los paquetes esperan en el nodo fuente. Con cargas elevadas en la red, para valores mayores a 1000 bits/µs en la inyección, DRB utiliza el mayor número de caminos alternativos permitidos en la configuración (en este caso cuatro), resultando en valores de latencia menores respecto a la técnica InfiniBand.

Además, a medida que la carga se incrementa, se observa la mejor capacidad de DRB para manejar el volumen de comunicaciones. En los casos de inyección mayor, se consiguen reducciones de latencia del orden del 100% respecto al obtenido con InfiniBand.

Al mismo tiempo que estas latencias se reducen, el *"throughput"* conseguido se mejora y se observa un incremento notable en la utilización de la red. Este aspecto puede verse en las graficas presentadas en la Fig. 7-2 (a), donde se presenta la carga aceptada como función de la carga aplicada. La curva correspondiente a DRB representa una carga mayor que la correspondiente a IBA_CC, donde la red se satura antes y, por tanto, otorga valores más bajos de carga aceptada.

Con el fin de comprobar la validez del método DRB sobre redes InfiniBand, los experimentos se han realizado para una topología con mayor número de nodos. La Fig. 7-3, muestra los resultados para las redes toroidales de dos dimensiones con 64 nodos. Se observa que con DRB la carga aceptada se mantiene constante, en un rango mayor de inyección, con todos los patrones de tráfico analizados, (lo que se deriva en una mayor utilización de la red) a diferencia del caso IBA_CC donde se observa una disminución de la carga aceptada, lo que significa que la red se satura a partir de un cierto punto en el que el incremento de la carga provoca el aumento de mensajes rechazados. Esta saturación se alcanza para cargas mayores en DRB lo cual permite su utilización en un rango mayor antes de aparecer el fenómeno de la saturación.

Hemos encontrado resultados similares para redes de 16 y 64 nodos que mejoran al aumentar la red de tamaño, lo cual significa una buena escalabilidad de DRB frente a los otros métodos.

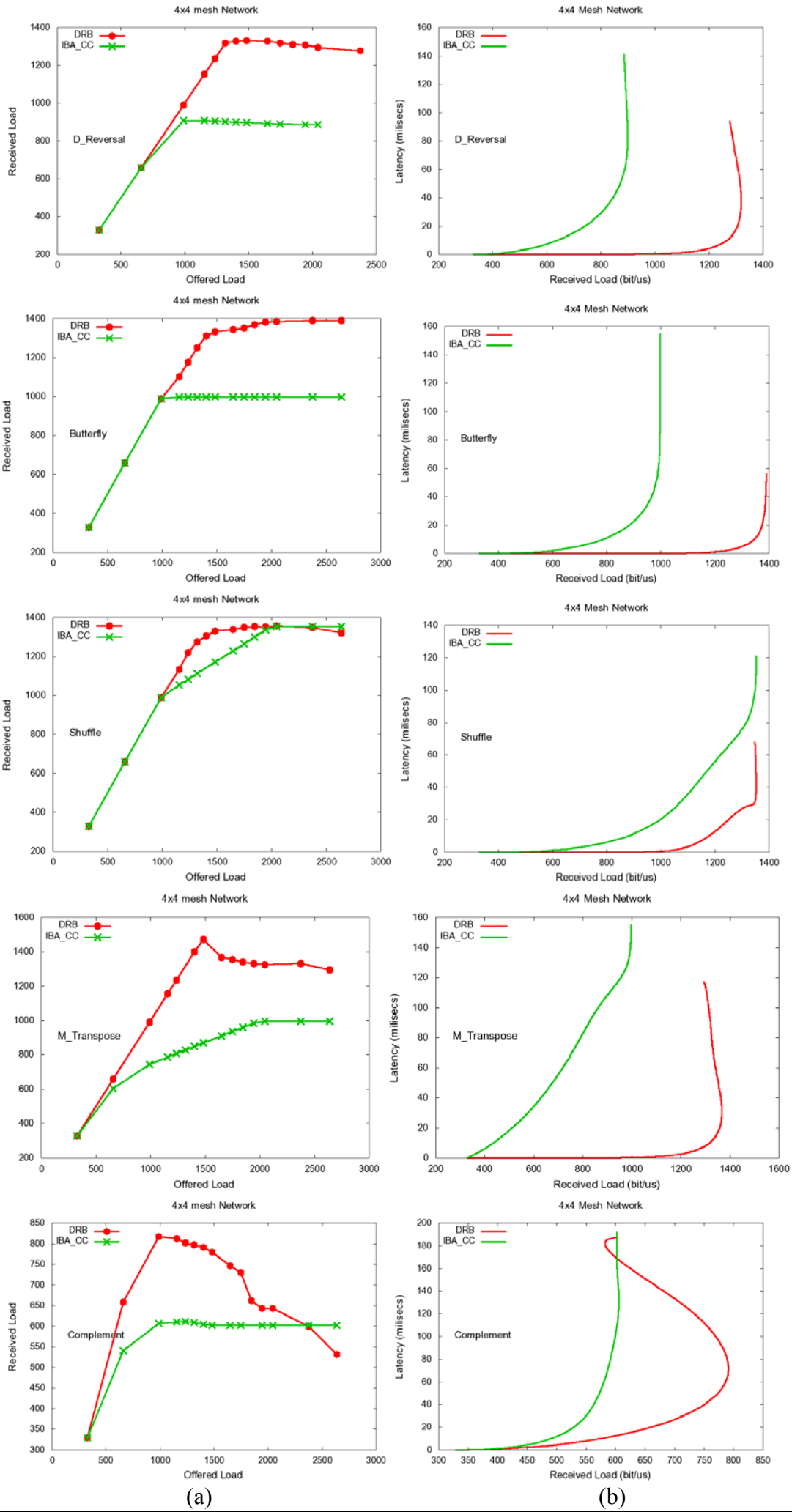
Como conclusión general para el caso del toro, se puede observar que DRB se comporta mejor que el método InfiniBand, sobre todo cuando la carga es muy alta, en cuyo caso la carga rechazada debido a la saturación de la red es mayor. En conjunto, DRB presenta una menor latencia y un mayor rango de utilización (se satura a cargas más altas) lo cual conforma el objetivo perseguido.

Topología malla (o grid).

El mismo tipo de experimentación se ha realizado para la topología malla y se ha encontrado que DRB ofrece mejores resultados que la técnica de control de congestión InfiniBand, para las mallas de diferentes tamaños. La Fig. 7-4 y Fig. 7-5 muestran los resultados para las mallas de dos dimensiones con 16 y 32 nodos, respectivamente.

En el caso de la malla 4x4, el comportamiento de DRB y el de IBA_CC son similares con cargas muy bajas, pero DRB es mejor en la zona de carga máxima, en la que presenta menores latencias y ofrece mejores prestaciones (Fig. 7-4 (b)). En los casos *"Butterfly" "Bit-Reversal"*, *"Perfect Shuffle"*, *"Matrix Transpose" y "Shuffle"* se observa que DRB ofrece menor latencia en el punto de carga máxima. Esto significa que DRB es capaz de soportar cargas mayores y demuestra que, ante condiciones extremas, ofrece mejores prestaciones que el otro método debido a la distribución de caminos utilizada. Con el patrón "*Complement"* la mejora es levemente superior respecto a IBA_CC debido a que la selección de caminos alternativos queda muy limitada a causa del patrón de tráfico que se crea en este caso.

En el caso del *"throughput"* (Fig. 7-4 (a)), DRB presenta mejoras en todos los casos aunque dependiendo del patrón oscilan entre el 25 y 80%.



(a) (b)

Fig. 7-4 Rendimiento sobre la malla 4x4

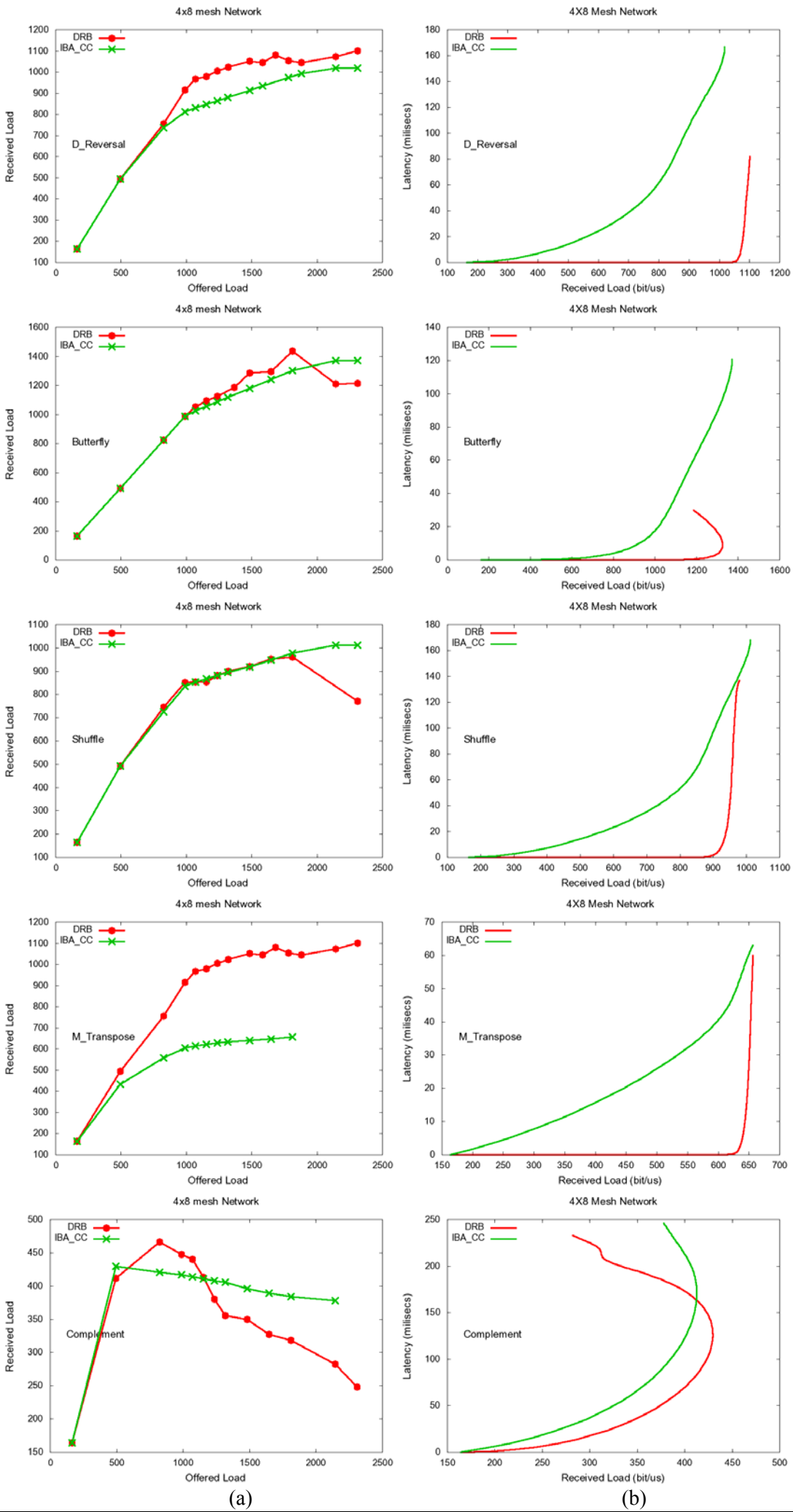


Fig. 7-5 Rendimiento sobre la malla 4x8

Como en el caso del toro, al aumentar el tamaño de la red y considerar la malla de 32 nodos Fig. 7-5, los resultados son cualitativamente similares a la de 16 nodos, pero las diferencias son todavía menos acusadas en el caso del *"throughput",* para los patrones *"Butterfly", "Bit-Reversal"*, *"Perfect Shuffle"*, *"Complement", y "Shuffle",* IBA_CC y DRB presentan comportamientos similares a cargas bajas, y algo mejores para DRB con cargas altas (y Fig. 7-5 (a)) . Esto es debido a la combinación del tipo de patrón y la topología, ya que las redes malla ofrecen menor cantidad de caminos alternativos que los toros. Sin embargo en el caso de "*Matrix Transpose"*, DRB ofrece mejores características respecto al *"throughput"* que la solución de InfiniBand.

Las latencias (Fig. 7-5 (b)) se comportan de manera similar, excepto con altas cargas que en el caso de IBA_CC se disparan a valores altos, mientras que en DRB se mantienen uniformes. Como puede observarse, sobre la topología malla, DRB presenta resultados aceptables. Sobre esta topología, por sus características físicas y los caminos alternativos que ofrece, las latencias que se producen son ligeramente peores que en el caso del toro.

Concluyendo, ante condiciones adversas se comprueba nuevamente que, DRB ofrece mejores prestaciones que el otro método, debido a la utilización de caminos alternativos.

7.3.1.2 Experimentación con el patrón "*hot-spot*"

Según se ha mencionado anteriormente, hemos diseñado un experimento donde el patrón de comunicación definido provoca la aparición de "*hot-spot*": en este caso varios mensajes compiten por los recursos sobre un camino común. Este patrón permite analizar y comparar las dos técnicas bajo condiciones extremas de carga aplicada. El patrón de "*hot-spot*" presentado en la Fig. 7-1 supone condiciones de comunicación muy rigurosas.

La tabla siguiente muestra el resumen técnico de la experimentación realizada con este patrón.

Topología	**Tamaño (Nodos)**	**Patrones**	**Técnica de control**	**Figura**
Toro	8x8 (64)	Hot-spot	IBA_CC	Fig. 7-6
			DRB	Fig. 7-7

Las Fig. 7-6 y Fig. 7-7 muestran los resultados obtenidos con este patrón, utilizando el mecanismo IBA_CC y DRB respectivamente.

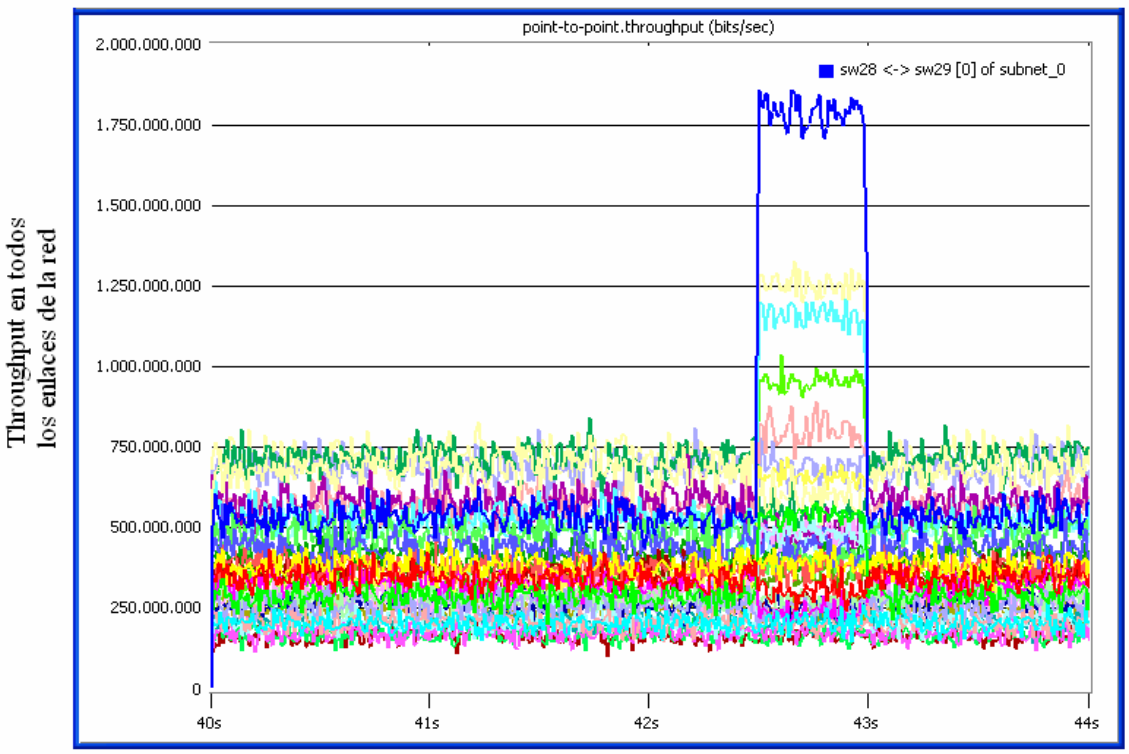


Fig. 7-6 Respuesta de la tecnica IBA_CC ante el patron hot-spot

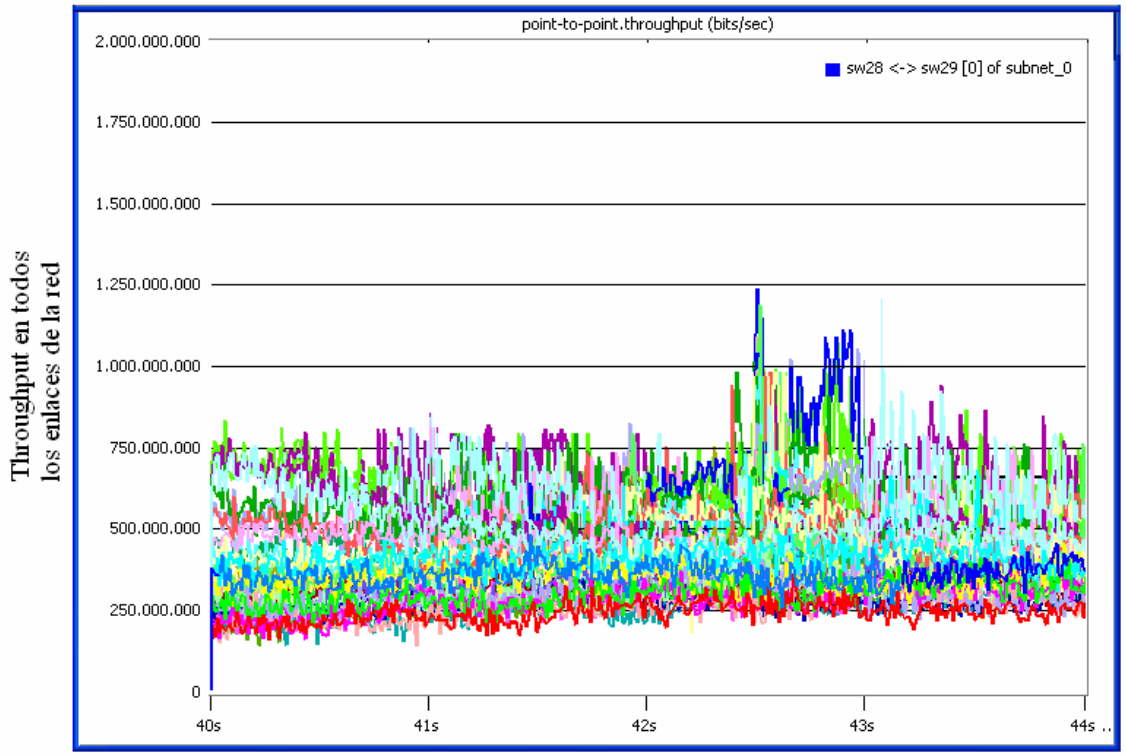


Fig. 7-7 Respuesta de la tecnica DRB ante el patron hot-spot

En ambas figuras puede observarse el tráfico presente en todos los enlaces de la red de interconexión analizada, donde se genera una repentinamente una gran carga localizada en una zona de la red. En este análisis puede verse cómo la aplicación del algoritmo DRB mejora los resultados frente a la técnica de control de congestión utilizada por IBA en aproximadamente un *75%*.

Cuando se utiliza DRB se eliminan efectivamente los picos de comunicación que se traducen en valores elevados de latencia, debido a que los paquetes deben esperar en los buffers a los que se conectan estos enlaces.

Como conclusión es posible decir que DRB ofrece mejores resultados con patrones *"hot- spot"*, que presentan una gran concentración de carga local, ya que es capaz de distribuir el exceso de carga entre los diversos enlaces de la red y balancear de manera eficiente el volumen de comunicación de toda la red de interconexión.

7.3.2 Influencia de la longitud del paquete

La longitud de los paquetes es un parámetro importante a conocer en el funcionamiento de la red de interconexión, debido a que su influencia sobre la latencia del sistema de comunicaciones puede afectar a sus prestaciones. Por este motivo, se ha realizado una experimentación consistente en variar la longitud del paquete y se han configurado longitudes de 4096, 1024, 256 bytes, que son tres de los cinco tamaños posibles especificados en la arquitectura InfiniBand[12].

La tabla siguiente muestra la experimentación realizada para estudiar la influencia de la longitud del paquete.

Topología	**Tamaño (Nodos)**	**Patrones**	**Longitudes**	**Figura**
Toro	8x8 (64)	Butterfly	4096 bytes 1024 bytes 256 bytes	Fig. 7-8 (a)
		Shuffle		Fig. 7-8 (b)

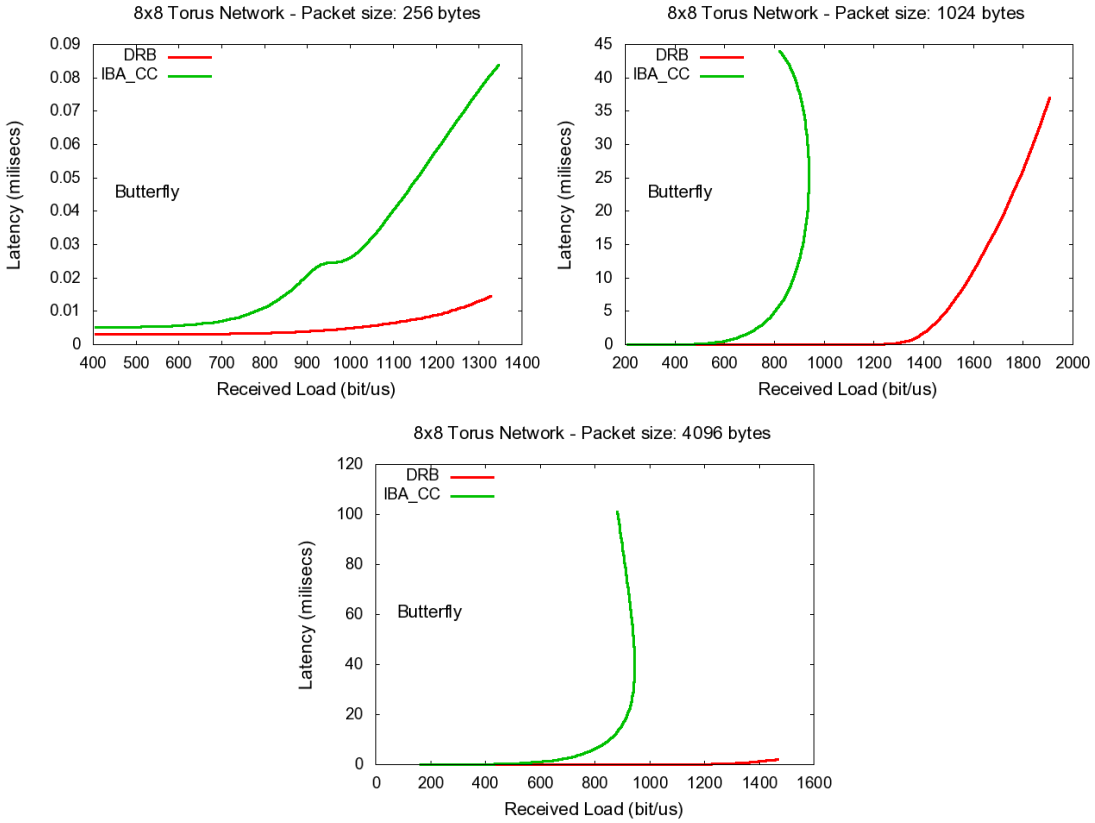
Según puede observarse en la Fig. 7-8, las mejoras en el rendimiento presentado por DRB se incrementan a medida que lo hace el tamaño del paquete en el caso presentado para el patrón de comunicaciones *"Butterfly"*. En este caso se observa una buena respuesta del mecanismo DRB ante las diversas longitudes de paquete, y es debido a que DRB es capaz de sacar partido de las variaciones de carga provocadas por el incremento de la longitud del paquete, para este patrón de comunicaciones.

En la Fig. 7-8 también se muestran los resultados de la evaluación para diferentes longitudes de paquete, pero utilizando el patrón *"Perfect Shuffle"*. Al igual que en el caso anterior, DRB ofrece mejores resultados que la técnica que ofrece InfiniBand. El balanceo distribuido del encaminamiento mejora notablemente los casos de carga muy alta y, es capaz de aprovechar el incremento de longitud de los mensajes.

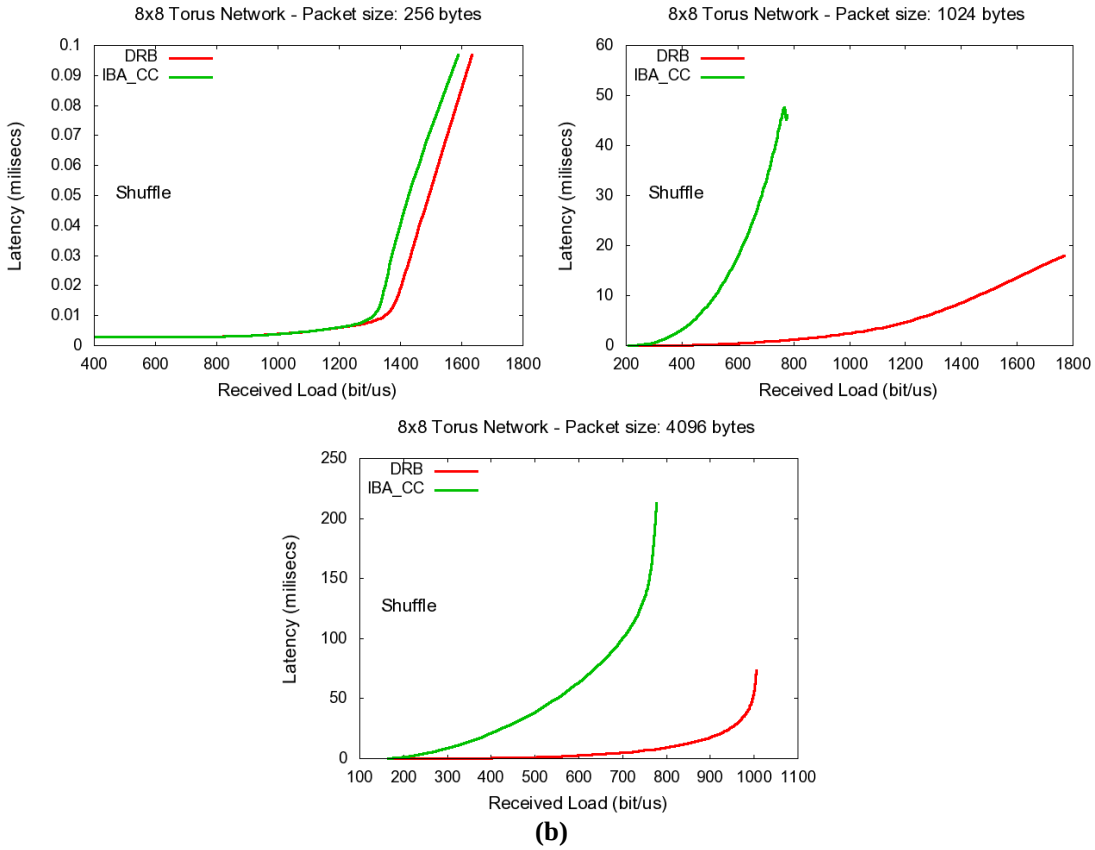
Como conclusión general, se puede observar que el hecho de aumentar la longitud del mensaje no es una buena opción, ya que hace aumentar la latencia para los tres métodos

[12] Los otros tamaños especificados son: 2048 y 512 bytes.

de encaminamiento de manera general. Con DRB, se mejora la disminución de las prestaciones ocasionadas por la variación de la longitud del paquete.



(a)



(b)

Fig. 7-8 Evaluación ante diferentas longitudes de paquete

7.4 Comentarios finales

En este capitulo, se ha realizado la evaluación de la aplicación del balanceo distribuido del encaminamiento en la especificación InfiniBand. Con este fin, se han presentado un conjunto de pruebas que permiten analizar las prestaciones de funcionamiento de la red de interconexión.

La experimentación ha evaluado un conjunto de topologías (toros y mallas) de diversos tamaños (16, 32 y 64 nodos), para un conjunto de patrones estándar de comunicación conocidos como "*Butterfly*", "*Bit-Reversal*", "*Perfect Shuffle*", "*Complement*", y "*Matrix Transpose*". Los resultados obtenidos reflejan la respuesta de la latencia y el "*throughput*" para un rango de carga baja hasta la saturación de dos estrategias de control de congestión: *DRB* e *IBA_CC*. Se ha encontrado, en la evaluación de estos resultados, que DRB ofrece mejores prestaciones que el método ofrecido por InfiniBand. En general estas mejoras alcanzan a duplicar a IBA en resultados de latencia para la topología toro. En cuanto a las mallas, se ha observado que DRB supera al método ofrecido por InfiniBand en promedio un 140% para el caso de la latencia de transporte y en valores entre el 20% y 120% según el caso para el "*throughput*" de mensajes. Como tendencia general, se observa que la diferencia entre DRB y la técnica de InfiniBand, se incrementa al aumentar el tamaño de la red.

En la evaluación ante el patrón hot-spot especifico diseñado para evaluar la respuesta del algoritmo ante picos de localizados carga, se ha observado una mejora del 75% en la utilización de los recursos.

A continuación, se ha evaluado el efecto de la longitud del paquete sobre las prestaciones de la red. Para el patrón "*Butterfly*", la influencia de esta longitud en las prestaciones empeora con su incremento, sin embargo *DRB* es capaz de mantener latencias bajas, debido a la distribución de carga y hace posible el uso de la red en un rango de inyección de tráfico mas elevado. En el caso analizado mediante el patrón "*Perfect Shuffle*", la observación es similar por lo que *DRB* ofrece mejores resultados que la técnica usada en IBA con el incremento de la longitud del paquete.

Según los resultados que arroja el conjunto de experimentos realizados en este capítulo podemos concluir que el balanceo distribuido del encaminamiento es una alternativa eficiente que presenta mejores prestaciones de funcionamiento en redes desarrolladas bajo la especificación InfiniBand y permite un manejo mas eficiente de la congestión.

Capitulo 8 Conclusiones y líneas abiertas

8.1 Conclusiones

Con este capítulo, terminamos la exposición del trabajo de investigación realizado en el ámbito de las redes de interconexión que forman parte de los computadores paralelos de altas prestaciones.

A lo largo del documento, se han desarrollado un conjunto de capítulos que han cubierto desde el inicio, las razones que han motivado la selección de la temática de trabajo elegida dentro de las redes de interconexión, y la tecnología empleada como soporte para la aplicación de las alternativas propuestas con el fin de solucionar el fenómeno de congestión en estas redes. Asimismo se ha desarrollado ampliamente una descripción de este fenómeno, junto a una clasificación de las técnicas existentes que intentan solucionarlo, analizando sus ventajas y desventajas.

También, se han descrito las principales características y el principio de funcionamiento de la arquitectura InfiniBand que permiten la implementación del balanceo distribuido del encaminamiento *DRB* en este versátil y novedoso estándar de comunicación de datos.

Por ultimo se ha justificado la necesidad de una plataforma de evaluación que hiciera factible la definición y el modelado de la arquitectura junto a la implementación, el análisis y la evaluación de la propuesta establecida.

Partimos de la observación del funcionamiento de las redes de interconexión, evaluando el comportamiento de la latencia de transporte como función de la carga de tráfico inyectada y, a partir de ahí, se han puesto de manifiesto una serie de problemas que surgen en el uso de estas redes. Se han extraído sus causas, centradas principalmente en una distribución injusta del volumen de comunicaciones sobre los recursos de la red, de manera que no hay concordancia entre el patrón de comunicaciones de la aplicación y la topología de la red, de manera que los paquetes de datos “quieren” pasar por donde no hay “cable”.

El comportamiento observado puede resumirse en dos aspectos principales. En primer lugar la latencia presenta un valor relativamente plano y acotado en presencia de cargas bajas tráfico, lo que representa que los recursos de la red están en condiciones de manejar el volumen de comunicaciones ofrecido. En segundo lugar la respuesta que presenta la latencia tiene un comportamiento más bien exponencial, y presenta grandes variaciones ante pequeños cambios en la carga de tráfico inyectado. Este

comportamiento se debe principalmente a la congestión que sufren los mensajes como consecuencia del uso compartido de recursos, y conduce a una degradación global de las prestaciones de la red de interconexión.

Por este motivo, la congestión de mensajes representa un problema importante dentro de las redes de interconexión de altas prestaciones, y por esto establece la principal motivación para la realización del trabajo de investigación que aquí se presenta.

Con el objetivo de comprender los aspectos principales del funcionamiento de las redes de interconexión, se han presentado las características y parámetros que conforman el espacio de diseño de dichas redes. Por tanto, se han evaluado los parámetros físicos, como los componentes y las topologías, junto a los parámetros dinámicos o de operación, como las técnicas control de flujo y los algoritmos de encaminamiento.

Asimismo, se han clasificado y evaluado las diversas alternativas que intentan solucionar el problema de la congestión. En esta clasificación, hemos divido las diferentes soluciones en técnicas preventivas basadas en la sobredimención de recursos, y técnicas reactivas que utilizan los recursos disponibles de forma eficiente

A partir de la definición de los objetivos y del análisis del comportamiento de las redes de interconexión realizado, hemos introducido la técnica de encaminamiento con la cuál pretendemos solucionar el problema de la congestión y hemos definido el concepto de balanceo del tráfico para conseguir un uso uniforme del ancho de banda de la red.

La solución consiste en el balanceo de la carga de comunicaciones en la red de interconexión y se denomina Balanceo Distribuido del Encaminamiento o DRB. La técnica del balanceo se basa en la distribución del tráfico usando nuevos caminos alternativos. Es un método dinámico que usa información de la ocupación de los enlaces para determinar el estado de la red. Este mecanismo se basa en la expansión de los caminos controlada por la carga de comunicaciones y esta diseñado para conseguir un valor acotado y uniforme de latencia de transporte, proporcionando a su vez, un método de eliminar la congestión y evitar la contención de mensajes, ya que permite una utilización mayor de los recursos de la red de interconexión.

También hemos remarcado la importancia del efecto colectivo sobre toda la red, debido a la colaboración de los nodos en la distribución de la carga de tráfico entre todos los enlaces de la red de interconexión.

Por otra parte se estudiado en detalle la especificación InfiniBand, cuyas características de funcionamiento proporcionan una nueva y poderosa arquitectura, que permite cubrir con las demandas sobre las prestaciones de las redes de interconexión en los

computadores paralelos de alta performance *(High Performance Compute HPC)*, debido principalmente al elevado ancho de banda y la baja latencia de transporte que ofrece.

Sin embargo, hemos visto que el estándar mencionado carece de una técnica de control de congestión adecuada, lo que ha establecido el principal objetivo en este trabajo: La implementación del balanceo distribuido del encaminamiento sobre la arquitectura InfiniBand.

Con este fin, se han estudiado las diferentes opciones y características que ofrece la arquitectura IBA, y que posibilitan la implementación mencionada. Entre estas, se han descrito la posibilidad de establecer trayectorias alternativas mediante la asignación de diferentes identificadores (LIDs) a cada nodo de la red, y la opción de "marcado" de los paquetes contenidos, que posibilita detectar congestión en los enlaces.

También se ha realizado un estudio que ha permitido la selección de la herramienta utilizada en el desarrollo de los modelos y ha hecho posible la simulación para el análisis y la experimentación de la propuesta establecida. La plataforma seleccionada *(Opnet Modeler)*, permite obtener resultados sobre el comportamiento de la latencia, bajo cualquier patrón de comunicaciones y sobre cualquier topología. Por otra parte, posibilita el modelado de la especificación IBA y el balanceo distribuido del encaminamiento, permitiéndonos cumplir el objetivo de este trabajo de investigación.

Finalmente, hemos realizado un conjunto de experimentos, cuyo análisis permite la evaluación de múltiples aspectos en la técnica de control de congestión aquí presentada. Esta experimentación fue llevada a cabo mediante el uso de la plataforma de desarrollo y simulación mencionada anteriormente.

Se ha experimentado con patrones sintéticos, porque representan los patrones de comunicación que aparecen en las aplicaciones paralelas más comunes, y hacen posible la obtención de la respuesta de la latencia, en la red de interconexión.

A continuación, se ha realizado una experimentación más genérica en la que se ha evaluado para un conjunto de redes de interconexión (toros y mallas) de diversos tamaños (16, 32 y 64 nodos) y para un conjunto de patrones estándar de comunicación ("*Butterfly"*, "*Bit-Reversal"*, "*Perfect Shuffle", "Complement"* y "*Matrix Transpose"*), la respuesta en latencia, y "*throughput*". Se ha encontrado que, para la mayoría de casos, *DRB* ofrece mejores prestaciones que la técnica de control de congestión propuesta por InfiniBand. En general, DRB mejora la técnica IBA entre un 50% y un 100%, tanto en resultados de latencia como de *"throughput*".

Esto significa que la técnica propuesta, permite utilizar la red con un volumen de comunicación mayor, sin que ésta alcance la saturación. La mejora en la utilización se

produce sin la necesidad de sobredimensionar ningún recurso de la red de interconexión. Esta eficiencia en la utilización implica que la red dispone de mayor ancho de banda para enviar mensajes.

DRB mejora en un 75% los resultados con patrones *"hot- spot"*, que presentan una gran concentración de carga local, ya que es capaz de distribuir el exceso de carga entre los diversos enlaces de la red y balancear de manera eficiente el volumen de comunicación de toda la red de interconexión.

Todos estos resultados demuestran la validez del balanceo distribuido del encaminamiento como estrategia de control de congestión en redes InfiniBand, ya que DRB es capaz de reducir la latencia y aumentar el rango de carga de tráfico dentro del cual la red funciona normalmente sin sufrir saturación en sus recursos y manteniendo la compatibilidad con el estándar.

A modo de resumen, se mencionan las principales aportaciones de este trabajo:

Se ha realizado un estudio general de las redes de interconexión, su ámbito de aplicación, sus características físicas y dinámicas y los parámetros de diseño con el fin de analizar las problemáticas surgidas de su funcionamiento y los fenómenos que las originan y favorecen su evolución, y los efectos provocados por su presencia.

Se ha realizado un estudio de las soluciones existentes para el control de congestión y la evitación de zonas calientes *(hot-spots)*, analizando su espacio de acción, ventajas y desventajas. Mediante este análisis destacamos las bondades del balanceo distribuido del encaminamiento y llevamos a cabo su implementación en una nueva y versátil tecnología de red: InfiniBand.

Se han analizado las posibles alternativas de plataformas para la experimentación, simulación y análisis de redes de interconexión donde se pudieran implementar y evaluar nuevas propuestas de diseño de las redes, mostrando las ventajas y desventajas de cada una, y se ha elegido una de ellas OPNET, por ser la mas adecuada para nuestros propósitos.

Se han realizado los modelos necesarios para llevar a cabo la implementación propuesta, mediante una de las herramientas de desarrollo y simulación más potentes que existen para el modelado y análisis de redes de interconexión (y redes en general).

Se ha realizado la experimentación y análisis de la técnica introducida, DRB, mediante la comparación a través de simulaciones con los mecanismos ofrecidos

por InfiniBand, conformando un conjunto de experimentos que ha confirmado la idoneidad de la nuestra propuesta.

En virtud a lo expuesto en estos párrafos, se ha conseguido cumplir con los objetivos impuestos en este trabajo al inicio y expuestos en el primer capítulo de esta memoria.

8.2 Líneas abiertas

Éste, como todo trabajo de cierta envergadura, que cubre una buena cantidad de aspectos dentro de un cierto campo de la ciencia bastante amplio, intenta ser completo en su integridad de manera que sea "cerrado" o "circular". De todas formas, de todo trabajo surgen una serie de líneas abiertas abordables como trabajo futuro para continuar la tarea, las cuales las podemos agrupar en dos tipos: las que surgen de completar ciertos aspectos del trabajo realizado que, por ser específicos o laterales, no se han abordado hasta el momento; y las que surgen de extender el trabajo, debido a que su ámbito de aplicación es de gran interés en la parcela de la ciencia que se ocupa y se pueden vislumbrar ampliaciones o aplicaciones del trabajo mas allá de lo realizado en una primera etapa.

Así pues, dentro del primer grupo de líneas abiertas podemos mencionar:

Extender la experimentación realizada a nuevas topologías y tamaños de red con objeto de disponer de una evaluación exhaustiva del algoritmo DRB sobre InfiniBand lo cual permitirá conocer en profundidad el comportamiento de DRB en un amplio rango de casos y vislumbrar por donde se deben definir nuevas alternativas.

Realizar más experimentación con trazas de aplicaciones reales. Relacionado con el punto anterior, también es importante simular con trazas de aplicaciones reales para conocer la respuesta del algoritmo en casos reales. Esta es una opción posible en OPNET y, aunque tiene la dificultad de que hay que caracterizar y conocer el patrón de tráfico de la aplicación para seleccionar las trazas adecuadamente, su utilidad también se basa en poder sintonizar y diseñar el algoritmo en entornos reales de aplicación.

Determinar los parámetros umbrales del algoritmo DRB. Ciertos parámetros de funcionamiento del algoritmo DRB deben ser determinados para su correcto funcionamiento. Aquí caben tanto una estimación estática previa a la ejecución, cómo un ajuste dinámico en función de la respuesta de la red. El otro aspecto a investigar es cómo determinarlos, es decir, cuál es el algoritmo, o la inteligencia, necesaria para determinar los parámetros y qué información se necesita para ello.

Profundizar en la estimación del *overhead* del algoritmo en la red. Como todo algoritmo de gestión de unos recursos, su aplicación presenta un cierto intrusismo y supone un cierto *overhead*. Aunque ya hemos mostrado en este trabajo que el *overhead* es pequeño y la ganancia obtenida de la aplicación del algoritmo es mayor que el *overhead* introducido, cabe profundizar en este estudio para caracterizar con detalle el mismo con objeto de poder diseñar alternativas que presenten menor *overhead*.

Estudiar la sensibilidad y robustez del algoritmo ante las características del tráfico y la precisión de la información de monitorización de la red. El primer aspecto, sensibilidad y robustez, es importante para asegurar que el algoritmo tiene un buen comportamiento ante cualquier topología o patrón de tráfico, o para conocer en qué casos su rendimiento no es tan bueno. En el segundo caso, respecto la precisión de la información, es importante conocer la robustez del algoritmo frente a este aspecto, ya que éste es un algoritmo donde con información parcial del estado actual de la red se intenta predecir el comportamiento en los estados siguientes para configurar los caminos alternativos y su uso.

Dentro del segundo grupo de líneas abiertas podemos apuntar las siguientes, clasificadas en las que se centran en modificaciones o ampliaciones del algoritmo DRB y las que se dirigen al diseño y definición de redes de interconexión de computadores paralelos.

Dentro de las que se refieren al algoritmo DRB destacamos:

Definir nuevas formas de monitorizar e informar del estado de la red al sistema de control de la congestión que disminuyan el *overhead* y mejoren la precisión e inmediatez de la información sobre el estado de la red. Aquí también cabe estudiar nuevos parámetros que describan el estado de la red: latencia, ocupación de *buffers*, velocidad de avance de los paquetes en la red,…

Definir nuevas formas de construir los posibles caminos alternativos, disjuntos o no, de distancia mínima o no, que se ajusten mejor a ciertas topologías o patrones de tráfico.

Definir nuevas formas de configurar los caminos alternativos cuando hay que realizar la expansión del camino actual debido a un incremento de la latencia, es decir, qué caminos elegir de entre los posibles determinados en el punto anterior, y también cuántos caminos alternativos elegir.

Definir nuevas formas de seleccionar los caminos alternativos cuando se envía un nuevo mensaje, con objeto de minimizar la latencia de los mensajes y maximizar el *throughput* de la red.

Estas líneas aquí expuestas pueden estudiarse en el ámbito de InfiniBand conservando el estándar actual completamente, con objeto de ser implementables y aceptadas por la industria; o pueden realizar propuestas de extensión del estándar InfiniBand, para permitir ampliaciones que superen las limitaciones del estándar actual; o aplicarse a otras tecnologías estándar o propietarias actuales o futuras, pudiéndose definir desde el propio trabajo una propuesta de encaminador o topología de red.

Respecto a las que se refieren al diseño de redes de interconexión:

Desarrollar una metodología para, dada una aplicación, analizar sus necesidades de comunicación, poder determinar cuántos caminos alternativos harían falta para mantener la latencia bajo un cierto umbral y poder así diseñar una red basada en InfiniBand adecuada a la aplicación.

Cabe decir, que con las líneas abiertas aquí descritas, se pretende conformar un trabajo de tesis doctoral que aporte nuevas soluciones a las redes de interconexión de los computadores paralelos y a las necesidades de comunicación de las aplicaciones con gran demanda de ancho de banda y cómputo.

Bibliografía

[1] Arjun Singh, B. T.; Gupta, A. K.; V.Puente, F. & R.Beivide (2004), 'Globally Adaptive Load-Balanced Routing on Tori', IEEE T., 478--866..

[2] Baydal, E. (2005), 'A Family of Mechanisms for Congestion Control in Wormhole Networks', IEEE Trans. Parallel Distrib. Syst. 16(9), 772--784.

[3] Bolding, K.; Fulgham, M. & Snyder, L. (1997), 'The Case for Chaotic Adaptive Routing', IEEE Trans. Comput. 46(12), 1281--1292.

[4] Boppana, R. V. & Chalasani, S. (1993), 'A comparison of adaptive wormhole routing algorithms', SIGARCH Comput. Archit. News 21(2), 351--360.

[5] Dally, W. J. (1990), 'Performance Analysis of k-ary n-cube Interconnection Networks', IEEE Trans. Comput. 39(6), 775--785.

[6] Dally, W. J. & Aoki, H. (1993), 'Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels', IEEE Trans. Parallel Distrib. Syst. 4(4), 466--475.

[7] Dandamudi, S. P. (1999), 'Reducing Hot-Spot Contention in Shared-Memory Multiprocessor Systems', IEEE Concurrency 7(1), 48--59.

[8] Duato, J. (1993), 'A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks', IEEE Trans. Parallel Distrib. Syst. 4(12), 1320--1331.

[9] Gaughan, P. T. & Yalamanchili, S. (1993), 'Adaptive routing protocols for hypercube interconnection networks', Computer 26(5), 12-23.

[10] Gerla, L. (1980), 'Flow Control: A Comparative Survey', Communications, IEEE Transactions on [legacy, pre - 1988]ISSN: 0096-2244 28, 553- 574.

[11] Konstantinidou, S. & Snyder, L. (1994), 'The Chaos Router', IEEE Trans. Comput. 43(12), 1386--1397.

[12] E. Baydal, P. L. & Duato, J. (2001), 'A Congestion Control Mechanism for Wormhole Networks"CICYT'.

[13] M.E. Gomez, J. F. P. L.; A. Robles N.A. Nordbotten, O. L. & Skeie, T. (2004), 'An Efficient Fault-Tolerant Routing Methodology for Meshes and Tori', MCYT -, 4.

[14] Ni, L. M. & McKinley, P. K. (.), 'A Survey of Wormhole Routing Techniques in Direct Networks', Computer 26(2), 62-76.

[15] *'InfiniBand Architecture Specification'* (v. 1.2), June 2001, InfiniBand Trade Association. Disponible en: http://www.InfiniBandta.com/

[16] Sancho, J. C. (2004), 'An Effective Methodology to Improve the Performance of the Up*/Down* Routing Algorithm', IEEE Trans. Parallel Distrib. Syst. 15(8), 740--754.

[17] Seo, D.; Ali, A.; Lim, W.; Rafique, N. & Thottethodi, M. (2005), 'Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks', isca 00, 432-443.

[18] Tamir, Y. & Frazier, G. L. (1992), 'Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches', IEEE Trans. Comput. 41(6), 725--737.

[19] Valiant, L. G. (1990), 'A bridging model for parallel computation', Commun. ACM 33(8), 103--111.

[20] Wang, M.; Siegel, H. J.; Nichols, M. A. & Abraham, S. (1995), 'Using a Multipath Network for Reducing the Effects of Hot Spots', IEEE Transactions on Parallel and Distributed Systems 06(3), 252-268.

[21] Alcover, R.; Lopez, P.; Duato, J. & Zunica, L. (1996), Interconnection Network Design: A Statistical Analysis of Interactions between Factors, in 'PDP '96: Proceedings of the 4th Euromicro Workshop on Parallel and Distributed Processing (PDP '96)', IEEE Computer Society, Washington, DC, USA, pp. 211.

[22] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 22.3: Depth-first search, pp.540–549. P. E..

[23] Dally, W. J. (1990), Virtual-channel flow control, in 'ISCA '90: Proceedings of the 17th annual international symposium on Computer Architecture', ACM Press, New York, NY, USA, pp. 60--68.

[24] Duato, J.; Johnson, I.; Flich, J.; Naven, F.; Garcia, P. & Nachiondo, T. (2005), 'A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks"HPCA '05: Proceedings of the 11th International Symposium on High-Performance Computer Architecture', IEEE Computer Society, Washington, DC, USA, 108--119.

[25] Franco, I. G. D. & Luque, E. (1998), Distributed Routing Balancing for Interconnection Network Communication, in 'HIPC '98: Proceedings of the Fifth International Conference on High Performance Computing', IEEE Computer Society, Washington, DC, USA, pp. 253.

[26] Franco, D.; Garcès, I. & Luque, E. (1999), 'A new method to make communication latency uniform: distributed routing balancing"ICS '99: Proceedings of the 13th international conference on Supercomputing', ACM Press, New York, NY, USA, 210--219.

[27] Franco, D.; Garcés, I. & Luque, E. (1999), Avoiding Communication Hot-Spots in Interconnection Networks, in 'HICSS '99: Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8', IEEE Computer Society, Washington, DC, USA, pp. 8040.

[28] Garcés, I. & Franco, D. (2002), Analysis of Distributed Routing Balancing behavior, in 'SAC '02: Proceedings of the 2002 ACM symposium on Applied computing', ACM Press, New York, NY, USA, pp. 817--824.

[29] R. Jain, 'The Art of Computer Systems Performance Analysis', Wiley Professional Computing, 1991.

[30] The "Network Simulator - NS2." Information Sciences Institute. The University of Southern California. 13 July 2006 , http://www.isi.edu/nsnam/ns

[31] Nachiondo, T.; Flich, J. & Duato, J. (2005), Efficient Reduction of HOL Blocking in Multistage Networks, in 'IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 9', IEEE Computer Society, Washington, DC, USA, pp. 211.2.

[32] Nesson, T. & Johnsson, S. L. (1995), ROMM routing on mesh and torus networks, in 'SPAA '95: Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures', ACM Press, New York, NY, USA, pp. 275--287.

[33] Puente, V.; Beivide, R.; Gregorio, J. A.; Prellezo, J. M.; Duato, J. & Izu, C. (1999), Adaptive Bubble Router: A Design to Improve Performance in Torus Networks, in 'ICPP '99: Proceedings of the 1999 International Conference on Parallel Processing', IEEE Computer Society, Washington, DC, USA, pp. 58.

[34] Ramany, S. & Eager, D. (1994), The interaction between virtual channel flow control and adaptive routing in wormhole networks, in 'ICS '94: Proceedings of the 8th international conference on Supercomputing', ACM Press, New York, NY, USA, pp. 136--145.

[35] Singh, A.; Dally, W. J.; Gupta, A. K. & Towles, B. (2004), Adaptive channel queue routing on k-ary n-cubes, in 'SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures', ACM Press, New York, NY, USA, pp. 11--19.

[36] Singh, A.; Dally, W. J.; Gupta, A. K. & Towles, B. (2003), GOAL: a load-balanced adaptive routing algorithm for torus networks, in 'ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture', ACM Press, New York, NY, USA, pp. 194--205.

[37] Stamatopoulos, J. & Solworth, J. A. (1995), Universal congestion control for meshes, in 'SPAA '95: Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures', ACM Press, New York, NY, USA, pp. 165--174.

[38] Thottethodi, M.; Lebeck, A. R. & Mukherjee, S. S. (2003), BLAM: A High-Performance Routing Algorithm for Virtual Cut-Through Networks, in 'IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing', IEEE Computer Society, Washington, DC, USA, pp. 45.2.

[39] Thottethodi, M.; Lebeck, A. R. & Mukherjee, S. S. (2001), Self-Tuned Congestion Control for Multiprocessor Networks, in 'HPCA '01: Proceedings of the 7th International Symposium on High-Performance Computer Architecture', IEEE Computer Society, Washington, DC, USA, pp. 107.

[40] Towles, B. & Dally, W. J. (2002), Worst-case traffic for oblivious routing functions, in 'SPAA '02: Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures', ACM Press, New York, NY, USA, pp. 1--8.

[41] Towles, B.; Dally, W. J. & Boyd, S. (2003), Throughput-centric routing algorithm design, in 'SPAA '03: Proceedings of the fifteenth annual ACM

symposium on Parallel algorithms and architectures', ACM Press, New York, NY, USA, pp. 200—209

[42] Aurelio Bermudez, F. J. Q. & Duato, J. (2006), 'Fast Routing Computation on InfiniBand Networks', IEEE Trans. Parallel Distrib. Syst. 17(3), 215--226.

[43] Gomez, M. E.; Flich, J.; Robles, A.; Lopez, P. & Duato, J. (2003), VOQSW: A Methodology to Reduce HOL Blocking in InfiniBand Networks, in 'IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing', IEEE Computer Society, Washington, DC, USA, pp. 46.1.

[44] Gusat, M.; Craddock, D.; Denzel, W.; Engbersen, T.; Ni, N.; Pfister, G.; Rooney, W. & Duato, J. (2005), Congestion Control in InfiniBand Networks, in 'HOTI '05: Proceedings of the 13th Symposium on High Performance Interconnects', IEEE Computer Society, Washington, DC, USA, pp. 158--159.

[45] Martinez, J. C.; Flich, J.; Robles, A.; Lopez, P. & Duato, J. (2003), ‘Supporting Fully Adaptive Routing in InfiniBand Networks,’ in 'IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing', IEEE Computer Society, Washington, DC, USA, pp. 44.1.

[46] Bermudez, A.; Casado, R.; Quiles, F. J.; Pinkston, T. M. & Duato, D. (2003), 'Evaluation of a Subnet Management Mechanism for InfiniBand Networks', icpp 00, 117.

[47] Lu, Z.; Zhong, M. & Jantsch, A. (2006), ‘Evaluation of on-chip networks using deflection routing, in’ 'GLSVLSI '06: Proceedings of the 16th ACM Great Lakes symposium on VLSI', ACM Press, New York, NY, USA, pp. 296--301.

[48] T. M. Pinkston, A. F. Benner, M. Krause, I. M. Robinson and T. Sterling, ‘InfiniBand: The De Facto Future Standard for System and Local Area Networks or Just a Scalable Replacement for PCI Buses?’ Cluster Computing, Vol. 6, No. 2, April 2003.

[49] ‘Top500 Supercomputers Sites’, http://www.top500.org.

[50] OPNET Technologies, Inc. ‘Opnet Modeler Accelerating Network R & D’, http://opnet.com/

[51] Glass, C. J. & Ni, L. M. (1994), 'The turn model for adaptive routing', J. ACM 41(5), 874--902.

[52] RM Fujimoto ‘Parallel Discrete Event Simulations’ Comm. of the ACM. Vol33, No 10, 1990.

[53] Yonghao Zhou, Jizhong Han, Jin He, Hongwei Zhang, Zifeng Xiao ‘SADLBoIB: Self-Adaptive Dynamic Load Balance over InfiniBand,’. Workshop on rontier of Computer Science and Technology (FCST'06), 2006.

[54] SIMSCRIPT II.5 solution simulation and modeling, “NETWORK II.5”. http://www.simprocess.com/solutions.

[55] T. Shanley ‘InfiniBand Network Architecture’, Mindshare, Inc. Addison-Wesley, 2003.

[56] Julio Ortega, Mancia Anguita, Alberto Prieto. Arquitectura de computadores. Thomson editores Spain Parainfo S.A. 2005.

[57] William Dally, Brian towles. Principles and practices of interconnection networks. Morgan Kaufmann publishers. 2004.

[58] J. Duato, S. Yalamanchili, and L. M. Ni, Interconnection Networks An Engineering Approach (Revised printing), Morgan Kaufmann Publishers, 2003.

[59] Yong Ho Song and Timothy Mark Pinkston. ‘A New Mechanism for Congestion and Deadlock Resolution,’. Proceedings of the International Conference on Parallel Processing (ICPP’02),2002 IEEE.

[60] J. Pelissier, ‘Providing Quality of Service over InfiniBand Architecture Fabrics,’ Proc. Eighth Symp. Hot Interconnects, Aug. 2000.

[61] Bermudez, R. Casado, F.J. Quiles, T.M. Pinkston, and J. Duato, ‘Modeling InfiniBand with OPNET,’ Proc. Second Ann. Workshop Novel Uses of System Area Networks, Feb. 2003.

[62] E.J. Kim, K.H. Yum, C.R. Das, M. Yousif, and J. Duato ‘Performance Enhancement Techniques for InfiniBand Architecture,’ Proc. Ninth Int’l Symp. High-Performance Computer Architecture (HPCA-9), 2003.