Universitat Autònoma de Barcelona

Departament d'Arquitectura de Computadors i Sistemes Operatius

Màster en Computació d'Altes Prestacions

Balanceo Distribuido del Encaminamiento para topologías Fat-tree sobre Redes Infiniband

Memoria del trabajo de investigación del "Máster en Computación de Altas Prestaciones", realizada por Belmar Mex Uc, bajo la dirección de Daniel Franco Puntes. Presentada en la Escuela Técnica Superior de Ingeniería (Departamento de Arquitectura de Computadores y Sistemas Operativos)

2008

Trabajo de investigación

Máster en Computación de Altas Prestaciones

Curso 2007-08

Título: Balanceo Distribuido del Encaminamiento para topologías Fat-tree sobre redes Infiniband

Autor: Belmar Mex Uc
Director: Daniel Franco Puntes
Departamento Arquitectura de Computadores y Sistemas Operativos
Escuela Técnica Superior de Ingeniería (ETSE)
Universidad Autónoma de Barcelona

Firmado

Autor Director

AGRADECIMIENTOS:

A mis padres que son mi modelo a seguir y la motivación principal para continuar prosperando. A mis hermanas que con su apoyo me hacen ser un mejor hermano mayor. A mi abuelita la cual tiene un lugar muy importante en mi corazón por todo el cariño que me brindó. A mis familiares Mex y Uc que siempre se han preocupado por mí y han contribuido mucho en lo que soy. También a mi tío Manuel que esté donde esté se que estará orgulloso de mí. A mis amigos de Campeche que siempre me decían "échale ganas". A la fundación Pablo García que me proporcionó medios para poder hacer mejor mi estancia en Barcelona. Sin olvidar a Dios, y al Cristo Negro de San Román por permitirme lograr el máster y cuidarme durante mi estancia en Barcelona.

Con mucha importancia a mi director de proyecto Daniel Franco, que antes que mi tutor es un amigo, que cuando se está tan lejos de casa es lo mejor que se puede tener. A la coordinadora del máster Lola Rexachs que desde el primer día en la escuela supe que podía contar con ella, me sacó de muchos problemas, y me ha mostrado otras formas de estudiar. A Emilio Luque, con quien tuve el privilegio de trabajar, pues es un ejemplo a seguir; los conocimientos y consejos que él me aportó nunca los olvidaré. A todos los miembros de mi grupo de investigación: Gonzalo, John, Álvaro, Diego y Juan Carlos Moure. Y mis compañeros de máster Álvaro, Andrés, Juan, Leonardo, Ronal, y a todos los otros amigos del laboratorio.
Gracias por todo y a todos.

"Vivir es separarse de lo que fuimos para acercarnos a lo que seremos en el futuro"
Octavio Paz.

Resumen:

Las redes de interconexión juegan un papel importante en el rendimiento de los sistemas de altas prestaciones. Actualmente la gestión del encaminamiento de los mensajes es un factor determinante para mantener las prestaciones de la red. Nuestra propuesta es trabajar sobre un algoritmo de encaminamiento adaptativo, que distribuye el encaminamiento de los mensajes para evitar los problemas de congestión en las redes de interconexión, que aparecen por el gran volumen de comunicaciones de aplicaciones científicas ó comerciales. El objetivo es ajustar el algoritmo a una topología muy utilizada en los sistemas actuales como lo es el fat-tree, e implementarlo en una tecnología Infiniband. En la experimentación realizada comparamos el método de control de congestión de la arquitectura Infiniband, con nuestro algoritmo. Los resultados obtenidos muestran que mejoramos los niveles de latencia por encima de un 50% y de throughput entre un 38% y un 81%.

Abstract:

Interconnection networks play an important role in the throughput of high performance systems. Currently, the message routing management is a key factor to maintain network performance. Our proposal is to work on an adaptive routing algorithm, which distributes message routing to avoid congestion problems on interconnection networks that appear due to the large volume of scientific or commercial application communications. The aim is to adjust the algorithm to a topology that is widely used in existing systems such as fat-tree, and couple it with Infiniband technology. In our experiments we compare the control congestion method on Infiniband architecture, with our algorithm. The results obtained shown that latency levels have been improved above 50% and throughput between 38% and 81%.

Resum:

Les xarxes de interconnexió juguen un paper molt important en el rendiment dels sistemes d'altes prestacions. Actualment la gestió de l' encaminament dels missatges és un factor determinant per mantenir les prestacions de la xarxa. La nostra proposta es dissenyar un algorisme de encaminament adaptatiu que distribueixi el encaminament dels missatges per evitar els problemes de congestió en les xarxes de interconnexió, els quals apareixen pel gran volum de comunicacions de aplicacions científiques o comercials. L'objectiu és ajustar l'algorisme a una topologia molt utilitzada en els sistemes actuals como ho es el fat-tree, i implementar-ho per a una tecnologia Infiniband. En l'experimentació realitzada comparem el mètode de control de congestió de l'arquitectura Infiniband amb el nostre algorisme. Els resultats obtinguts mostren que millorem els nivells de latència per sobre d'un 50% i de throughput entre un 38% i un 81%.

CONTENIDO

ÍNDICE DE FIGURAS

CAPÍTULO 1.
INTRODUCCIÓN

1.1. CONTEXTO

El sector de computación de altas prestaciones es uno de los más significativos dentro del ámbito de la computación, ya que el objetivo es aportar respuesta a las necesidades más exigentes de cómputo. En el mundo del HPC (*High Performance Computing*) cada día se aportan más y más técnicas o soluciones para aumentar las prestaciones.

En primer lugar, el uso de HPC estaba enfocado a aplicaciones científico-técnicas limitadas a la realización de un gran volumen de cálculos matemáticos, pero desde hace un tiempo se ha empezado a ampliar a otras zonas de aplicación a consecuencia de los progresos que se han conseguido en potencia de cómputo y facilidad de uso, al igual que es cada vez más accesible. Áreas como el procesamiento gráfico, la distribución de imágenes en movimiento, los negocios, la telemedicina, el teletrabajo, la teleenseñanza, distribución de películas bajo demanda, la extracción y resumen de grandes bases de datos, etc.; adoptan los sistemas de altas prestaciones por que suelen requerir gran capacidad de almacenamiento, gran capacidad de cómputo y gran capacidad de comunicaciones. Las aplicaciones multimedia, requieren de respuestas que sean rápidas, o de tiempo real.

De igual forma, no sólo el cómputo, sino también la comunicación de datos, a corta o larga distancia, ha sido y es un aspecto que esta cobrando gran importancia y relevancia.

Los grandes sistemas de procesamiento y clusters de computadores personales son la tecnología que debe cumplir con la demanda de cómputo de altas prestaciones y soportar comunicaciones de las intensivas aplicaciones que se ejecutan en estos sistemas. Estos sistemas deben contar con una red de alta velocidad de comunicación de datos que interconecte todos los nodos del sistemas a través de una serie de elementos como enlaces y encaminadores; ya que es un aspecto de gran importancia para mantener las altas prestaciones; un alto ancho de banda , una latencia baja, mantener un alto rendimiento (throughput) de cómputo. Este tipo de redes son las conocidas como las Redes de Interconexión de altas prestaciones.

Actualmente, tecnologías comerciales como *Infiniband. Myrinet, Quadrics*,.., son las que marcan tendencias a la hora de construir las redes de interconexión dentro de los sistemas

de altas prestaciones. Son capaces de formar redes de interconexión con altas velocidades de transmisión de datos, soportar las diferentes formas de implementación de la red entre todos los nodos, al igual que incorporan sus propias técnicas para mantener adecuados los niveles de las comunicaciones en estos sistemas o clusters.

Pero a pesar de existir estas marcas comerciales las redes de interconexión presentan problemas para controlar congestión en el tránsito de mensajes, lo que implica que la red comience a trabajar al máximo de utilización y si esto no se controla adecuadamente es posible que la red se sature, el throughput de la red disminuya considerablemente y que los recursos disponibles tengan problemas para poder continuar adecuadamente con las comunicaciones. Esta situación de aumento del tiempo en el viaje de los mensajes por la red es conocida como *latencia*. Es por ello que ha aumentado las aportaciones y técnicas de la comunidad investigadora para prevención o control de la congestión para utilizar estas redes al máximo de sus posibilidades.

Actualmente se ha demostrado [10] que se obtiene altos índices de throughput de las Redes de Interconexión gracias a los Encaminamientos Adaptivos, ya que estos toman en cuenta el estado de la red en todo momento para tomar sus decisiones de encaminamiento permitiendo así un mejor balanceo en el tráfico de la Red.

1.2. ANTECEDENTES

El problema que surge con las aplicaciones que se ejecutan en sistemas de altas prestaciones es que las estimaciones de los tiempos de cómputo y comunicación no son todo lo exactas que sería necesario para hacer una asignación óptima de procesos a procesadores, en cuanto que la estimación del tiempo de comunicación es complicado por la naturaleza del comportamiento de la latencia de las redes de interconexión. La latencia de comunicaciones debe ser reducida con objeto de minimizar tal tiempo de ejecución y prevenir casos de congestión dentro de la red.

Existen algunas técnicas que se plantean controlar y solucionar estos problemas de congestión, algunos ejemplos son [2]:

- **Técnicas preventivas.** Esta técnica consiste en contar con toda la información de todos los recursos de la red reservándolos antes de comenzar la transmisión de datos, garantizando así la ausencia de la congestión. Consecuencias que trae esta técnica es que aparte de circular por la red los mensajes de las aplicaciones, se añaden los mensajes de la información de los recursos de la red, esta sobre carga de mensajes es llamado overhead. De igual manera el conocimiento acerca de todos los recursos de la red no es fácil de conseguir. Esta propuestas funciona aplicando las siguientes técnicas:

    - **Incrementar recursos.** Aumentar el número de elementos de la red (conmutadores, encaminadores, nodos, enlaces)

- **Planificar y determinar la máxima inyección de mensajes.** Configurar el número mensajes que se enviaran por cada nodo de la red.

- **Técnicas reactivas o Correctivas.** En este caso la regulación del tráfico se ejerce cuando la red ha alcanzado un estado de ocupación próximo a la saturación de un recurso. Ya que las técnicas reactivas consisten en emplear métodos eficaces e inteligentes para evitar la congestión, son aplicadas con gran frecuencia dentro del área de las redes de interconexión en computadores paralelos de alto rendimiento, es por ello que cada vez se trata de aportar mas propuestas para mantener eficiente el rendimiento de las redes de interconexión.

En general puede descomponerse en tres fases:

- **Monitorización la red y detección de congestión.** Medir características dinámicas de la red como son la Latencia, Canales o Buffers ocupados tanto a nivel global como local, Disminuciones de la velocidad de flujo de mensajes *(Backpressure*).
- **Notificar la existencia de congestión.** Dar a conocer acerca de la congestión para que se actué ya sea de manera local, entre vecinos, los fuentes, o a todos los nodos de la red.
- **Ejecutar acciones correctivas.** Mecanismo para eliminar la congestión y degradación de rendimiento de la red. Unas de las acciones tomadas son:
  - **Message throttling.** Consiste en detener o reducir la cantidad de mensajes inyectados.
  - **Gestionar y optimizar el uso de los Buffers.** En el conmutador se reordenan los paquetes para evitar interferencias.
  - **Encaminamiento adaptivo.** Redistribuye la carga por múltiples trayectorias alternativas entre un par fuente-destino.

El Balanceo Distribuido del Encaminamiento [4] es un método para crear caminos alternativos entre fuente y destino en la red de interconexión. Esta distribución está controlada por los niveles de carga de los caminos posibles dentro de la topología entre un nodo fuente y un nodo destino. Estos nuevos caminos harán uso de los enlaces disponibles de los encaminadores, distribuyendo la carga de los caminos mas cargados hacia los menos cargados.

El objetivo de DRB (*Distributed Routing Balancing*) es una distribución uniforme de la carga de tráfico sobre la red de interconexión entera para mantener una latencia baja y uniforme y prevenir la generación de "hot-spots".

Este mecanismo se basa en la expansión controlada por la latencia de los caminos que usan los canales para enviar los mensajes. Para ellos se evalúa de manera dinámica la carga de la red en todo momento y se actúa localmente pero teniendo en cuenta los efectos globales de manera que se produce una "sintonización" global a partir de actuaciones locales que consideran el comportamiento de sus vecinos.

En practica, el método consiste en monitorizar el estado de los enlaces entre las conexiones entre la comunicación de un par fuente destino, al detectar congestión se notifica al/los nodos/s que inyectan mensajes para que determinen nuevas trayectorias y redistribuyan el tráfico según su estado de carga. La detección de la congestión toma en cuenta algunos parámetros de la red como son: latencia acumulada en un enlace en particular, la ocupación de los buffers en los puertos de los conmutadores, etc. En cambio para la distribución del tráfico de los paquetes puede ser de forma aleatoria, en función de la ocupación de los enlaces, o la velocidad de los enlaces en redes no homogéneas.

1.3. MOTIVACIÓN

Las redes de interconexión desempeñan un rol fundamental en las aplicaciones que se ejecutan sobre sistemas de altas prestaciones, porque permiten el intercambio de mensajes para la colaboración de tareas. Pero los problemas que afectan a las redes de interconexión y sus causantes hacen que sea necesario aplicar técnicas para mantener o aumentar las prestaciones de los computadores de altas prestaciones.

Las variaciones bruscas de la latencia deben ser evitadas. Nuestra propuesta es conseguir que la red de interconexión presente un comportamiento con una latencia baja y controlada de manera que se eviten los "hot-spots", lo cual se logra con la técnica de balancear la carga de comunicación uniformemente por la red de interconexión. El mecanismo diseñado para conseguir el balanceo consiste en la búsqueda de caminos alternativos cuando los caminos originales que se están usando comienzan a saturarse.

Existen antecedentes de esta propuesta en los trabajos de investigación [], donde se continua con el objetivo de poder tomar esta propuesta y poder implementarla en

Actualmente, existen tecnologías que son empleadas en Sistemas o Clusters de centros de investigación de gran renombre a nivel mundial, y que tratan de emplear métodos de conexión de punto a punto, baja latencia, elevado ancho de banda, etc. Algunos ejemplos de estas tecnologías son Infiniband, Myrinet, Quadrics, Gigabit ethernet, etc.

La gran popularidad que ha ganado la Arquitectura Infiniband en computadores de altas prestaciones, según como demuestra el ranking *top500* [1], es debido a la versatilidad que ofrece con las conexiones de varios componentes de red, productos definidos tanto para cobre y fibra óptica, define propiedades para switches y routers, al igual que el ancho de banda que ofrece hace posible la utilización de gran cantidad de conexiones. Por otro lado, maneja un estilo de comunicación de mensajes la cual consiste en dividirlo en paquetes y volverlo a armar al final de su recorrido al destino, todo gracias a las opciones de encaminamiento y gestión de errores que implementa en su estándar. Esto hace a Infiniband independiente de cualquier tecnología en particular, al igual que es una excelente base para poder implementar librerías de comunicación (como *MPI*) encima de él.

Los switches pueden ser configurados para cualquier topología, pero en la práctica está siendo usado mas para topologías *fat-tree*. La topología Fat-tree es una de las mas populares dentro los supercomputadores mas importantes y de mayor rendimiento de la actualidad, como lo demuestra el listado del Top 500 de Noviembre del 2007 [1].

La topología Fat-tree como el nombre lo dice se basa completamente en un árbol, pero a diferencia de los árboles tradicionales, éste se caracteriza principalmente por hacerse mas ancho en cuanto se acerca a la raíz. En la Figura 1.1 podemos observar un ejemplo de un fat-tree.

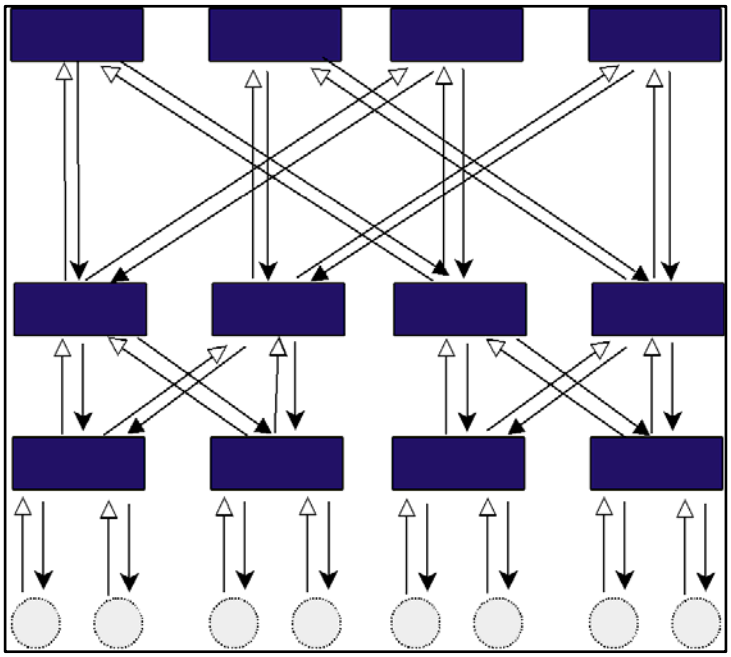


Figura 1.1. Fat-tree

Haciendo referencia a la literatura actual [6], estudios demuestran que al aumentar el grado de los switches jerárquicamente hasta la raíz hace que la implementación física sea en varias ocasiones inaplicable, por ello surge la alternativa de implementarlos con el mismo grado en cada switch, esto deriva en la siguiente nomenclatura *k*-**ary** *n*-**tree.**

En el *k*-**ary** *n*-**tree** el camino mínimo fuente-destino de un paquete enviado a través de la red es el entrar en la fase de subida hasta encontrar el ancestro (switch) en común mas cercano entre el origen y el destino, una vez validado, el switch hace el cambio de dirección (turn-around) el paquete para comenzar la fase de descenso hasta el destino; una vez en esta fase solo un camino es posible hasta el final. Durante la fase de acenso de un paquete por los niveles del árbol, existe la posibilidad de ser encaminado por varios caminos, es por esto que se necesita un encaminamiento adaptivo, ya que sin un adecuado mecanismo de encaminamiento puede ocurrir situaciones de congestión y aumentar los niveles de latencia, lo que conlleva que se degraden las prestaciones.

Por todo esto, en este trabajo de investigación nos hemos planteado la implementación de una técnica de control de congestión inteligente: un algoritmo adaptivo vigente, factible y realista, referenciado en otras publicaciones o artículos de otros autores; sobre

un estándar de arquitectura de red y una topología de red que son actuales y populares. Las razones expuestas en este apartado justifican el interés y necesidad de este trabajo.

1.4. OBJETIVOS

Como se comento antes, nuestra propuesta es un algoritmo adaptivo para el control de congestión, que reduzca la latencia y haga controlar situaciones de "hot-spot" en redes de interconexión de computadores de altas prestaciones. Al igual que contribuir en que este algoritmo puede ser aplicado en tecnologías importantes como Infiniband y que puede desempeñarse en topologías como el Fat-tree.

Los principales objetivos que pretendemos cumplir son:

- Estudiar las características dinámicas y generales de las redes de interconexión.
- Analizar las problemáticas surgidas en el funcionamiento de las redes de interconexión y sus causantes.
- Realizar una propuesta de comportamiento deseable de una red de interconexión a partir de su respuesta en comportamiento de la latencia y el "*throughput*".
- Analizar el estándar Infiniband, sus capacidades y versatilidad al momento de aplicar el concepto del balanceo distribuido del encaminamiento (DRB).
- Solucionar situaciones de congestión y de encaminamiento en topologías multinivel (*MINS*), en este caso topologías Fat-tree.
- Utilizar herramientas de desarrollo y simulación adecuadas para la investigación.
- Experimentar y analizar los resultados obtenidos, comparándolos con otra técnica de control de congestión.
- Demostrar o justificar si nuestra propuesta es factible.

1.5. ORGANIZACIÓN DEL DOCUMENTO

La estructura de este documento comienza con el Capítulo 2 haciendo un estudio de las características estáticas, estructurales, dinámicas y topológicas de las redes de interconexión. De igual manera se muestran los elementos a tomar en cuenta para la funcionalidad, junto con los aspectos en el diseño de una red de interconexión, al igual que mencionamos el estado del arte de nuestra investigación.

En el capítulo 3, se realiza una descripción general de Infiniband, se detallan las características que permiten adaptar e implementar la propuesta de algoritmo de control de congestión, el balanceo distribuido del encaminamiento.

El Capítulo 4 hace un pequeño análisis de la herramienta de simulación que se usara para experimentar y obtener nuestros resultados, con los cuales nuestra propuesta estará sustentada.

En el capítulo 5 planteamos el análisis y diseño del algoritmo de encaminamiento sobre una topología fat-tree y adaptar el algoritmo sobre una tecnología actual como es Infiniband. En el capítulo 6 son los aspectos técnicos que permiten la aplicación del balanceo distribuido del encaminamiento en la arquitectura Infiniband. Los modelos creados en la herramienta de simulación con los cuales se muestra la implementación del algoritmo, son analizados a detalle.

La Experimentación, la obtención y análisis de resultados se encuentra en el capítulo 7, en la cual evaluamos nuestra técnica de control de congestión propuesta, analizamos su comportamiento, en comparación con otra técnica de control de congestión. Para finalizar con las conclusiones obtenidas a lo largo del trabajo y posibles líneas abiertas.

CAPÍTULO 2.

REDES DE INTERCONEXIÓN

2.1. INTRODUCCIÓN

Los sistemas de altas prestaciones dependen de gran medida de las comunicaciones para mantener altos niveles de rendimiento y obtener los resultados con mayor eficiencia. La Comunicación representa el transporte de datos de una localización a otra. Por tanto, una Red de interconexión es el medio o sistema programable que trasporta datos entre terminales o nodos [2].

Las redes de interconexión (RI) están emergiendo como una excelente solución de problemas de comunicación entre niveles de los modernos sistemas de computadores. Originalmente fueron diseñadas para gran número de necesidades de comunicaciones de multicomputadores, pero ahora comienzan a remplazar a los buses dedicados de los sistemas actuales. Así como también se descubre que el encaminamiento de los paquetes es más rápido y económico que el encaminamiento de cable. Las redes de interconexión conectan procesadores a memorias y puertos de entrada/salida a dispositivos compartidos de I/O del sistema. Las RI de igual forma comunican con todos los puertos de entrada y salida de los switches y routers de la red, de esta forma se asegura que cualquier bit podrá ser transportado entre dos componentes fuente/destino.

La importancia de la redes de interconexión es crucial ya que es quién delimita el factor de prestaciones en los sistemas HPC, debido a que la comunicación entre procesadores o memorias determina la latencia y banda ancha del sistema, dos factores fundamentales en las mismas. De igual forma el desempeño de la red de interconexión marca la capacidad del switch, porque la demanda de comunicaciones entre nodos hace que crezca rápidamente la capacidad de los enlaces, y la interconexión se convierte entonces en un cuello de botella en la mayoría de los sistemas.

En estos días las redes de interconexión las podemos encontrar en los siguientes grupos de sistemas de altas prestaciones [2]:

- **Clusters.** Es un sistema procesamiento de tipo paralelo o distribuido, el cual se compone de una colección computadores personales interconectados por una red trabajando juntos como un solo sistema integrado de cómputo.

- **MPP (Massively Parallel Processors).** Un sistema de gran procesamiento paralelo, con una arquitectura compartida. Cosiste en varios cientos de elementos de procesamiento o nodos, los cuales necesitan estar interconectados por una red de alta velocidad. Cada nodo es una variedad de componentes de hardware, pero generalmente todos tienen una memoria principal y uno o más procesadores; cada nodo ejecuta su propia versión de sistema operativo.

Las aplicaciones de cálculo intensivo, principalmente se generalizan por tener una fase previa a la ejecución del programa, de distribución de las tareas entre los nodos de cómputo del computador de altas prestaciones. Esta distribución se realiza tratando de maximizar el uso de los nodos de cómputo, de manera que la mayor parte del tiempo posible esté ejecutando tareas y ningún procesador se quede ocioso sin ninguna tarea disponible mientras otros procesadores tiene tareas que están esperando a ser ejecutadas. Influyen factores en esta asignación de tareas como son las dependencias funcionales de las tareas, los volúmenes de cómputo y de la comunicación de las tareas. La relación volumen de cómputo/volumen de comunicación se llama *granularidad*.

De igual forma se debe llegar a un cierto compromiso y balancear adecuadamente la carga de cómputo. Para ello debe conocerse el tiempo de ejecución de las tareas, ya que si no ocurrirá que durante la ejecución aparecerán procesadores ociosos porque habrá nodos de cómputo con todas sus tareas esperando por mensajes. En este caso, la asignación establecida de tareas será incorrecta y se incrementara el tiempo total de ejecución del programa porque no se hará una utilización adecuada del sistema. Los retardos de comunicaciones pueden hacer que ciertas tareas que esperan por sus mensajes no puedan empezar a ejecutar aunque tengan el procesador disponible, con lo que se alarga el tiempo de ejecución total del programa paralelo; es por esto por lo que se debe evitar que la latencia experimente grandes variaciones.

En aplicaciones multimedia (video bajo demanda, video conferencias, etc.,) se dan requerimientos de tiempo real y de sincronismo. La latencia puede incluso impedir el correcto funcionamiento de dichas aplicaciones, por tanto la latencia es menos tolerada.

2.2. REQUERIMIENTOS DE LA RED DE INTERCONEXIÓN

Hay dos aspectos en cuestión de requerimientos a tomar en cuenta en una red de interconexión:

- **Requerimientos funcionales.**
  - El protocolo de encaminamiento debe ser libre de "*deadlock*", esto es, que se pueda asegurar que cualquier mensaje pueda llegar hacia su destino o que el camino que necesita para su destino no esta siendo ocupado por otro.
  - No exista el "*livelock*", esto representa que no exista el estado en el que un mensaje se mantiene recorriendo indefinidamente sin llegar a un destino.

- Evitar el "*starvation*", es decir, evitar el tráfico de mensajes injustificadamente por la red de un origen a un destino.

- **Requerimientos de prestaciones.**
  - La ruta mas corta para todo paquete en la red, esto puede depender de la distancia física de la topología o del tiempo de comunicación.
  - Reducir los niveles de latencia y en la medida de lo posible mantenerla acotada bajo un cierto valor máximo y con la menor variación posible (latencia uniforme). La latencia representa el tiempo desde que el origen inyecta la cabecera del mensaje en la red, hasta que la cola de éste llega al destino.
  - Mantener los niveles de "*throughput*" siempre altos. El throughput es el número de mensajes por unidad de tiempo que la red es capaz de gestionar y depende en gran medida del ancho de banda disponible.
  - El mecanismo de encaminamiento debe ser capaz de adaptarse a las condiciones de tráfico, ser capaz de evitar los nodos saturados y presentar cierta tolerancia a fallos.

Las ventajas que se esperan conseguir de la latencia uniforme son:

- La granuralidad de la aplicación será válida en un rango más amplio de la carga de la red. La carga de la red es el número medio de mensajes circulando por unidad de tiempo.
- Se consigue aprovechar al máximo el ancho de banda de toda la red de interconexión durante el mayor intervalo posible de carga de la red. De hecho, el punto de saturación se sitúa en un punto de carga mas elevado.
- La simplificación de la asignación de tareas a los nodos de cómputo porque se pueden predecir los retardos reales de comunicación a partir de los volúmenes de comunicación, ya que se conoce (o se puede predecir) el comportamiento de la latencia.

Existen elementos importantes a considerar en el rendimiento de una red de interconexión, los cuales son la topología, el control de flujo, el encaminamiento.

2.2.1. Topología

La topología de red se refiere a la forma física o estática en la que está conectado los enlaces y nodos de la red de interconexión del computador paralelo. Seleccionar la topología es el primer paso en el diseño de la red por que la estrategia de encaminamiento y el método de control de flujo dependen en gran medida de la topología. No solo especifica el tipo de red, si no que también otros detalles como es el grado ó nivel jerárquico que puede tener un switch dentro de la red, el ancho/bit de cada enlace. [2].

Existen algunas categorías importantes dentro de las topologías de las redes de interconexión. Algunas son:

2.2.1.1. Redes de Medio Compartido.

La estructura menos usada, por que el medio de transmisión es compartido por todos los dispositivos de comunicación, por lo que la red solo permite trabajar a un elemento a la vez. Debido a su limitado ancho de banda, este tipo de red solo puede soportar un cierto número de elementos, lo que la hace no escalable, además de que el medio frecuente mente se vuelve un cuello de botella. Un ejemplo de estas son las LAN tradicionales y las basadas en bus. A continuación la Figura 2.1 un ejemplo de esta redes.

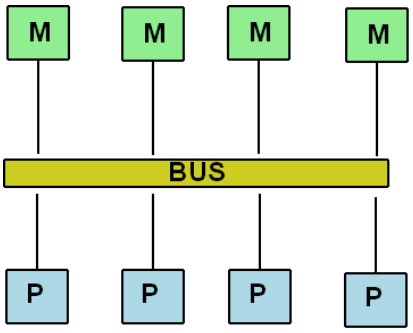


Figura 2.1. Red de Medio compartido

2.2.1.2. Redes Directas (punto a punto).

Un tipo de red escalable con cualquier número de procesadores. Consiste en un conjunto de nodos, cada uno conectado a otra subred de otros nodos de la red. Cada nodo con sus propios procesadores, memoria local, y otros dispositivos de soporte. El componente en común de estos nodos es un router, el cual se encarga de gestionar los mensajes entre estos. Usualmente dos nodos vecinos están conectados por un par de canales unidireccionales en direcciones opuestas o bien por canales bidireccionales. En el caso de encaminadores dedicados en esta red pueden solapar cómputo y comunicación con cada nodo. Es por esto que este tipo de redes son populares para construir computadores paralelos de gran tamaño. Este tipo de redes se puede dividir en ***Ortogonales y no ortogonales***. En las *ortogonales* los nodos son numerados usando su coordenada de *n-dimensiones.* Tomando en cuenta que la distancia entre dos nodos puede ser la suma de desplazamientos entre dimensiones, implementar un algoritmo de encaminamiento es fácil, en cambio en las no ortogonales el encaminamiento puede llegar hacer un poco más complicado [3]. Un ejemplo de estas redes directas en la figura 2.2.

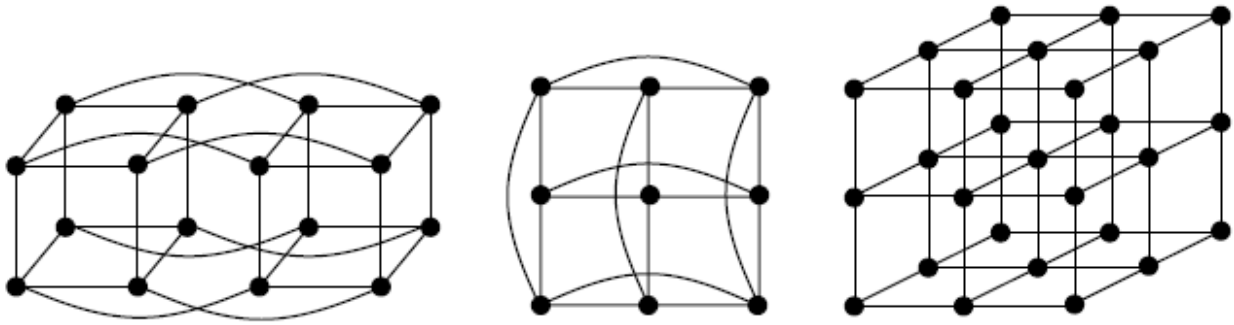


Figura 2.2. Redes Directas

2.2.1.3. Redes Indirectas.

La comunicación entre nodos es gestionada por switches, por tanto cada nodo tiene un adaptador de red que lo conecta al switch, o bien al set de puertos del switch. Cada puerto consiste en un puerto de entrada y otro de salida. Estos switches están interconectados entre si; la forma de conectarlos define varias topologías.

Una es el **Crossbar que** permite a cualquier procesador en el sistema conectarse con cualquier otro procesador o unidad de memoria, es por esto que es posible que los procesadores se comuniquen simultáneamente sin controversia. Una nueva comunicación puede ser establecida siempre y cuando para esta solicitud existan puertos de entrada y salida disponible. Este tipo de red es utilizada para construir sistemas de multiprocesadores de altas prestaciones de pequeña escala [3]. La Figura 2.3 representa este tipo de topología.

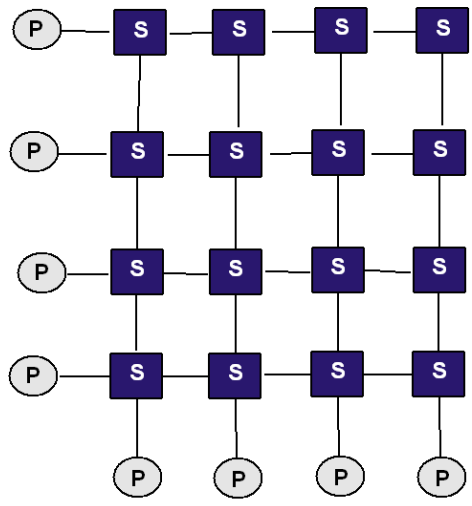


Figura 2.3. Crossbar

Otro tipo de redes indirectas son las *Multistage Interconnection Networks* (**MINS)** las cuales conectan dispositivos de entrada a dispositivos de salida por medio de un número de switches con cierto grado, donde cada switch es un pequeño crossbar. El número de

niveles y el diseño de conexión entre los niveles determina las características de encaminamiento de la red.

En la práctica, los switches son idénticos, es decir tienen el mismo grado, ya que esto ayuda considerablemente en reducir los costos de diseño de la red. La red *Banyan* es un tipo de MINs con la propiedad que solo hay un único camino entre un par fuente-destino. La red Delta es una subclase de la anterior, por lo que el diseño es similar, k x k switches en *n* niveles, donde cada nivel cuenta con *N/k* switches. Muchas de las MINs utilizadas actualmente, como la *Omega, cube, butterfly, y baseline*, son resultado de la Delta. Figura [3].

Pero como se menciono el presente trabajo de investigación esta enfocado en una topología de este tipo, un **Fat-tree,** este tipo de red la cual se caracteriza por que los procesadores se encuentra en las hojas, los vértices internos son switches y las ramas son enlaces bidireccionales. El número de enlaces y el ancho de banda se incrementan conforme se avanza a un nivel superior del árbol, hasta alcanzar la parte más robusta en la raíz del árbol. Para encaminar un mensaje de un procesador a otro, es necesario mandarlo al primer nivel superior del origen (*Forward* dirección), así subirá por todo el árbol hasta encontrar el ancestro en común entre ese par fuente-destino, para comenzar el descenso hasta el procesador destino (*Backward* dirección); al cambio de direcciones en el árbol se le llama *turnaround* [3]. Pero como fue mencionado en la introducción en la realidad usa la nomenclatura *k*-**ary** *n*-**tree,** donde **ary** son los vértices del árbol y **tree** el nivel del árbol, este tipo de árbol esta compuesto de $\boldsymbol{N = k^n}$ nodos de procesamiento y $\boldsymbol{nk^{n-1}}$ switches, con *k* puertos de entrada y *k* puertos de salida. Así que actualmente algunas publicaciones y documentos hacen mención de la Topología Fat-tree y la vinculan directamente al término de *k*-**ary** *n*-**tree**. Un ejemplo la Figura 2.4 de una topología de un árbol 2-ary 3-tree, y en la Figura 2.5 es una representación matemática de links o arys en un switch de una red k-ary n-tree.

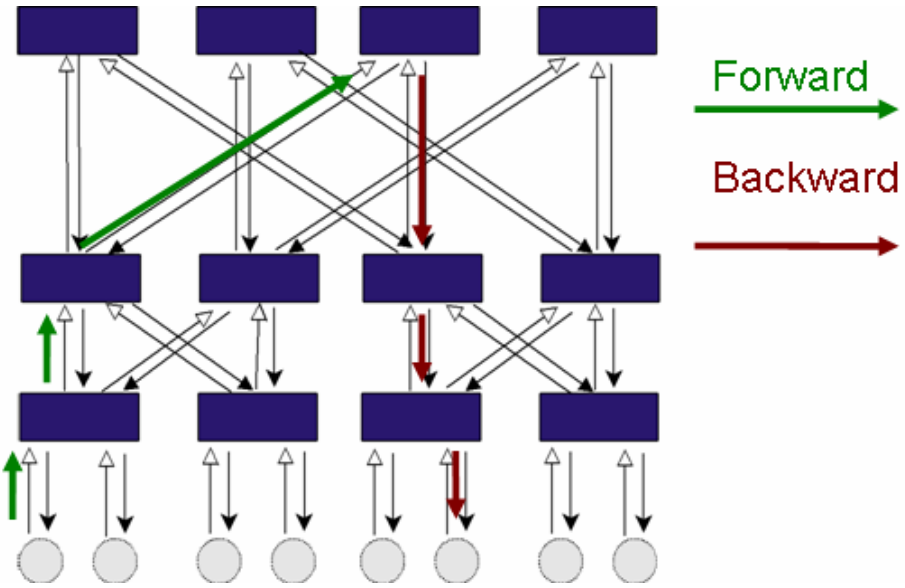


Figura 2.4. Un árbol 2-ary 3-tree

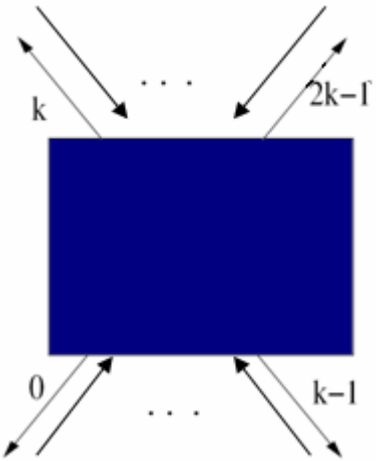


Figura 2.5. Numeración de enlaces en el k-ary n-tree

2.2.2 Control de Flujo

Determina cómo un recurso de la red (un enlace de banda ancha, capacidad del buffer y control de estado) es destinado para que un paquete atraviese la red. Un buen método de control de flujo gestiona estos recursos de una manera eficiente de tal manera que la red pueda conseguir una fracción alta de su ancho de banda ideal y una baja o previsible latencia de los paquetes. Un ejemplo de cómo actúa un control de flujo es de la siguiente manera: dos paquetes arriban a diferentes puertos de entrada de un router al mismo tiempo, pero ambos desean el mismo puerto de salida para continuar hacia su destino, en este caso el mecanismo de control de flujo resuelve esto de tal manera que a uno le reserva el canal de salida y al otro lo deja de tal manera en un estado de paquete bloqueado [3].

Existen dos mecanismo de control de flujo, uno es el basado en *créditos,* este mecanismo asigna a cada uno de los switches de la red una cantidad de créditos para el envío de paquetes, la cantidad de créditos cambia de acuerdo al nivel de ocupación de los buffers del switch vecino; por cada envío el switch emisor consumen créditos, así hasta consumirlos todos. Una vez que el receptor cuenta con espacio libre para recibir paquetes, este envía créditos al emisor para continuar la transmisión.

El otro mecanismo trabaja con *marcas,* cuando un switch descubre que un buffer excede un nivel ocupación y esta próximo a desbordar, este switch envía una marca al emisor para que regule o detenga el envío a este switch. A medida que el switch en conflicto recupera su capacidad de recepción, este envía otra marca al emisor para continuar con la transmisión.

2.2.3 Técnicas de Conmutación

Técnicas que definen la forma en que un paquete progresa a través de la red entre un par fuente-destino. Algunas de las técnicas usadas en la actualidad son [3]:

- **Store and Forward Switchg.** Consiste en que el contenido de todo un paquete debe ser almacenado en el puerto de entrada del switch para después poder tomar el puerto de salida hacia su destino. Esto presenta problemas de perdida de ancho de banda, al igual que la latencia se incrementa en varias ocasiones, ya que este factor va determinado por la longitud de los paquetes y el número de conmutadores que se encuentra en la trayectoria del paquete hacia su destino.

- **Virtual Cut-Through (VCT) Switching.** El router puede empezar a enviar la cabecera del paquete y los bytes de datos del mismo tan pronto como la decisión de encaminamiento ha sido tomada y el buffer de salida está libre. Se consigue que el paquete no necesita ser almacenado en un puerto de salida y pueda separarse en el siguiente puerto de entrada del router enviado para poder así seguir su camino hasta el destino. Solo que en caso de que el puerto de salido se encuentre ocupado, se debe contar con espacio suficiente para almacenar el paquete en lo que espera hacer enviado, lo que pasa con Store and Forward switching.

- **Wormhole Switching.** En esta técnica el paquete del mensaje es dividido en unidades mas pequeñas llamadas *flits*; así los buffers de entrada y salida de un router normal es capaz de almacenar una serie de estos flits; e igual como en VCT, con solo tener la cabecera del paquete puede ser enviado hacia su destino sin necesidad de esperar mas tiempo. En el caso de que un buffer de salida se encuentre ocupado, el mensaje es bloqueado "al momento", es decir que si al momento que los flits de un paquete A llegan a un buffer de salida que esta siendo utilizado por otros flits de otro paquete B, el mensaje A es bloqueado, pero como los flits son pequeños, cada buffer de router es capaz de almacenar por unos segundos los flits del paquete A. Esta técnica no nos libera de un deadlock, pero si reducir promedios de latencia, ya que no es necesario usar la memoria local del procesador para almacenar el mensaje. Esto permite que actualmente los routers sean pequeños, compactos y rápidos.

- **Canales Virtuales.** Las anteriores técnicas asumen que los buffers de entrada y salidas son canales físicos. Pero actualmente algunos canales físicos son capases de soportar la configuración de *canales lógicos o virtuales* subdividiendo el mismo canal físico. Obviamente al momento de actuar como si fueran canales únicos el nivel de operación del canal físico es reducida a la mitad de velocidad. Presenta las ventajas en técnicas como Wormhole. El protocolo del canal físico debe ser modificado para distinguir los canales virtuales.

2.2.4 Encaminamiento

Un encaminamiento se refiere a la trayectoria que de un nodo fuente a un nodo destino en una topología en particular. El encaminamiento determina en gran medida el desempeño de la red, ya que el **algoritmo de encaminamiento** usado para una red debe estar encargado de balancear la carga a lo largo de todos los enlaces con la presencia de diferentes parámetros de tráfico, principalmente cuando son tráficos no uniformes o cambiantes, de igual manera de tener en cuenta la longitud de los caminos para que sean lo mas cortos posibles, para reducir el número de saltos y la latencia de los mensajes de la red.

Algunas propiedades de la red de interconexión cuyo efecto dependen mucho del algoritmo de encaminamiento son:

- **Conectividad.** La propiedad de encaminar paquetes de cualquier nodo origen a cualquier nodo destino.

- **Adaptabilidad.** Capacidad de encaminar los paquetes a través de caminos alternativos ante la presencia de congestión o el fallo de componentes.

- **Libre de Deadlok y Livelock.** Garantizar que el paquete no será bloqueado o vagar por la red para siempre.

- **Tolerancia a Fallos.** Encaminar el paquete ante la presencia de algún fallo de los componentes de la red.

Por todo esto es que el algoritmo de encaminamiento debe cumplir con las siguientes características.

- **Simplicidad.** Capaz de poder tomar la decisiones en pequeñas fracciones de tiempo.

- **Robustez.** Poder reconocer/adaptarse a cambios en la topología de la red debido a los fallos de nodos o enlaces y mantener su funcionalidad a pesar de los cambios en los patrones de tráfico.

- **Maximizar la utilización de los enlaces.** Ser capaz de que los enlaces se usados de forma equivalente al momento de encaminar los paquetes.

- **Optimalidad.** Minimizar el retardo medio de los paquetes y maximizar la utilización total de la red.

Tomando en cuenta todas las características anteriores, los algoritmos de encaminamiento pueden ser clasificados de acuerdo a varios criterios, esto es una taxonomia de los algoritmos de encaminamiento en la actualidad y en la literatura [3].

- **Número de destinos.**
  - Unicast (único). A un solo nodo es destinado un paquete.
  - Multicast (múltiples). Varios nodos de la red son los destinos.

- **Decisiones de encaminamiento.**
  - Centralizado. Un solo nodo de la red es quien toma la decisión de encaminamiento
  - Origen. El nodo fuente del mensaje es quien toma la decisión.
  - Distribuido. La decisión se toma en cada nodo por el que va alojando al paquete a través de su recorrido al destino.
  - Múltiples fases. El nodo origen toma algunas decisiones de los nodos destinos, pero la decisión es tomada de manera distribuida.

- **Implementación.**
  - En tablas de encaminamiento. Almacenarlo en tablas de encaminamiento.
  - Máquina de estado finito. Ejecutarlo en software o hardware de acuerdo a este tipo de máquina.

- **Adaptabilidad.**
  - Deterministico. El algoritmo siempre toma en cuenta el mismo camino par un par fuente-destino, sin considerar el tráfico en la red, usado en topologías simples.
  - Oblivious. Un algoritmo que incluye una filosofía del deterministico, envía a routers aleatoriamente un paquete pero sin tomar en cuenta el estado de a red, sacrificando de esta manera la localidad o el balanceo de de carga.
  - Adaptativo. Se utiliza a nivel de conexión (punto a punto), recolecta información del estado de la red y actúan 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 a estos recursos, para que utilice otro camino hacia el destino.

    - Estado de la red. Si la información que utiliza de la red puede ser solo del nodo en transito (nivel local) o de un conjunto de nodos (nivel global)
    - Progresividad puede ser de dos tipos, *progresivo* siempre encamina la cabecera hacia delante reservando un nuevo canal por cada vez que avanza. El *backtracking* pueden permitirse retornar la cabecera de un paquete por un camino tomado, realizando previamente una reserva del canal.
    - Minimalidad. Clasificados en *provechoso (minimal)* siempre encamina por los caminos mas cercanos al destino. *Misrouting (no minimal)* escoge las trayectorias más lejanas al destino.

- Número de enlaces. Pueden tomar en cuenta todos los enlaces de la red o solo parcialmente algunos.

2.3 CONSECUENCIAS DEL "HOT-SPOT"

Las situaciones que colocan a la red en situaciones límites variando la carga desde valores bajos hasta llegar a un nivel de carga extremo donde se pide a la red la mayor capacidad posible. Esta situación de carga extrema provoca en la red lo que se conoce como "*hot-spot*" o "punto caliente". El "*hot-spot*" produce que las latencias que sufren los mensajes tomen valores muy altos respecto de las latencias cuando no se está en zona de carga de saturación. En la zona de "*hot-spot*" la latencia crece muy rápidamente ante cambios pequeños de la carga. En la figura 2.6 podemos observar un conflicto o puntos de saturación dentro de una red fat-tree, donde los switchs marcados con rojo son los caminos donde se interfecta el tráfico de paquetes encaminados diferentes nodos del sistema y crean el hot-spot. En la Figura 2.7 se presenta una gráfica de comportamiento de latencia que sufren los mensajes. [2] [3].

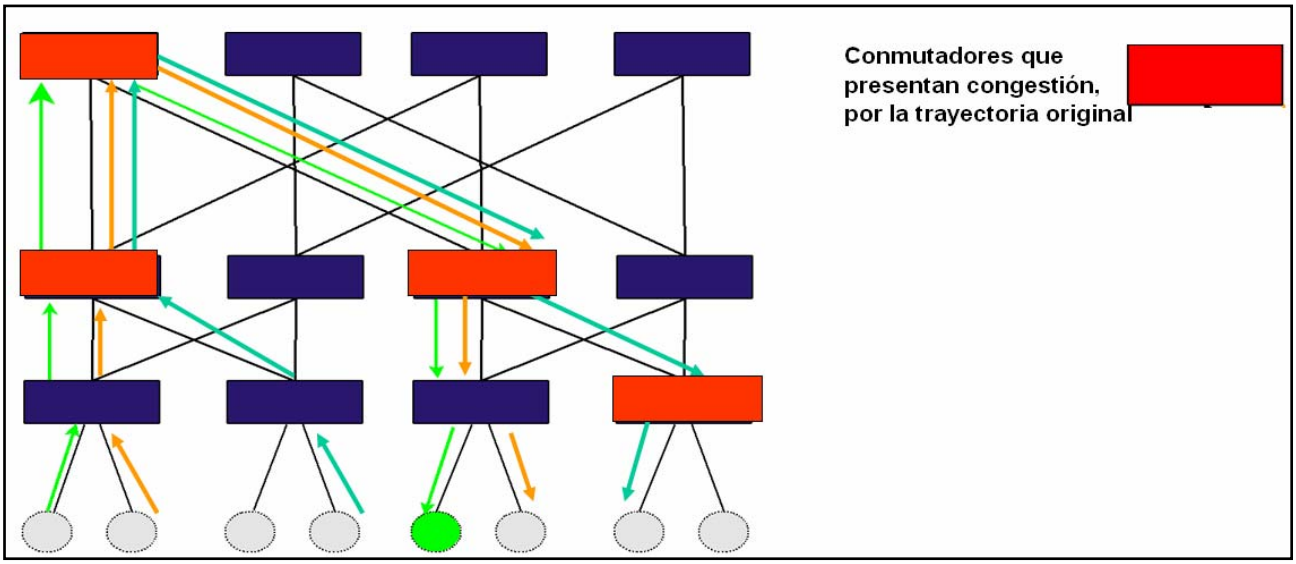


Figura 2.6. Situación de hot-spot en un red fat-tree.

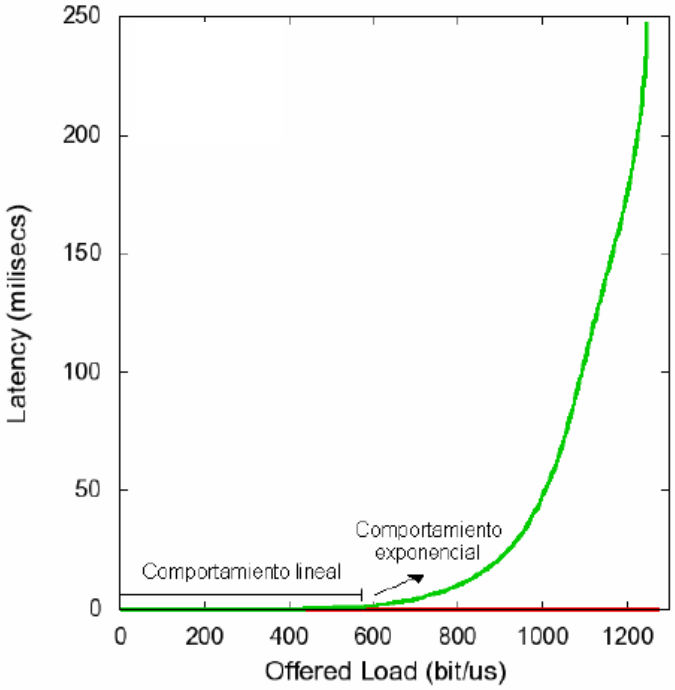


Figura 2.7. Comportamiento de la latencia (Latencia promedio vs. Tráfico ofrecido)

Respecto a la carga de la red, la latencia presenta un comportamiento que se puede describir mediante una curva no lineal en la que se pueden distinguir dos regiones o zonas de comportamiento. La primera zona, cuando la carga de mensajes es baja, es una zona básicamente plana con un comportamiento casi lineal de la latencia, en la cual grandes cambios de la carga no provocan grandes cambios en la latencia. En esta zona, la red de interconexión no esta saturada y no se usa su capacidad al 100% y los incrementos de carga pueden ser absorbidos por recursos disponibles de la red.

La segunda zona de comportamiento es una curva con una gran pendiente donde la latencia experimenta grandes cambios ante pequeños cambios de la carga de entrada a la red de interconexión. En este caso, se ha llegado o se está cerca de la zona de saturación de la red y el nuevo tráfico no puede ser absorbido y los mensajes deben esperar gran cantidad de tiempo en los "*buffers*" de los nodos antes de entrar en la red. En esta zona se tiene un crecimiento de la latencia exponencial llegando a tomar valores imprevisiblemente altos.

Además, en la curva de latencia se puede identificar claramente el punto de cambio de una zona a otra ya que presenta un "codo" muy marcado. El punto de cambio de la latencia de una zona a otra, significa que si la carga sigue aumentando, la latencia de la red entra en la zona de saturación.

Algunos factores que son los que determinan la carga de la red son la frecuencia de inyección de mensajes, el número de canales que inciden sobre un enlace, la longitud del mensaje, el número de puntos de colisión de un canal con otros canales, son muy importantes en la influencia de la latencia.

Finalmente, si en una zona de la red se produce un "*hot-spot*" o zona de saturación de carga, rápidamente esta situación se propagará a zonas vecinas a través de los mensajes que sufren el "*hot-spot*" y de ahí a otras zonas creando un efecto colectivo que colapsará toda la red de interconexión.

2.4 BALANCEO DISTRIBUIDO DEL ENCAMINAMIENTO

Como se comentó anteriormente en el Capítulo 1, nuestra propuesta es aportar una técnica de control de congestión, en este caso un algoritmo adaptivo cuya función sería basada en conseguir un balanceo global de las comunicaciones para conseguir un uso uniforme de la red y se eliminen las zonas de saturación o "*hot-spots*".

El algoritmo "Balanceo Distribuido del Encaminamiento" (DRB siglas en ingles) balancea la carga de comunicaciones entre los enlaces de la red de interconexión. Tras conseguir el balanceo se deben cumplir algunos puntos como son: una latencia uniforme, latencia predecible y prevención de "*hot-spots*". Con una latencia uniforme se quiere que no exista una variación del tiempo de envío de paquetes al distribuir la carga de comunicaciones. Al momento de lograr una latencia uniforme se tiene que es predecible. De esta manera, los enlaces se aprovechan al máximo mientras no esta saturada [4].

DRB es utilizar un método de ampliación de caminos alternativos para cualquier par fuente-destino de nodos en la red de interconexión. La distribución de las comunicaciones esta controlada por el nivel de carga de todos los caminos posibles. Estos nuevos caminos, el algoritmo permite que puedan ser distancia mínima o una posible extensión en la distancia de los caminos, todo dependería de la carga de tráfico presente en la red y cuántos enlaces se encuentren con menos carga de comunicación [5].

El algoritmo emplea una notificación a los nodos que inyectan los mensajes en la red para que comiencen a configurar nuevas trayectorias posibles a los siguientes paquetes y redistribuyan el tráfico según las mediciones hechas de carga de todos los enlaces posibles de la red.

La distribución de mensajes entre caminos disponibles tiene un doble efecto [4]:

- Los nuevos caminos alternativos tendrán menores niveles de carga que el camino original, el camino original reducirá por consecuencia su carga, y la suma de los anchos de bandas entre los múltiples caminos puede igualar el ancho de banda de los enlaces sin congestión

- Al enviar los mensajes por otros caminos alternativos, y estar usando un ancho de banda mayor que el camino original, se produce una reducción de la latencia.

El método de DRB consiste de tres fases principales [5]:

- **Monitorización de la carga de tráfico.** Los mensajes almacenan los niveles de latencia que va registrando en su recorrido y esta información es devuelta al nodo fuente a través de un mensaje de reconocimiento de máxima prioridad.

- **Configuración Dinámica de Trayectorias Alternativas.** Al momento de recibir el mensaje con la información de niveles de latencia del mensaje enviado anteriormente, este nodo modifica el número de nuevas trayectorias posibles para los próximos paquetes si es que se supera el umbral previsto con el nivel de congestión crítico en la red.

- **Selección de Caminos múltiples.** Teniendo en cuenta la latencia en los caminos, se generan posibles canales convenientes para el envío en base a una distribución probabilística.

Ahora supongamos que tenemos un red de interconexión con una carga de mensaje en los canales que están saturados, es decir, la carga es muy alta debido al número de canales que los utilizan, y los otros canales se encuentran trabajando con poco nivel de ocupación y, por lo tanto, no están aprovechando el máximo de capacidad del camino que utilizan. Entonces DRB hace una modificación en la distribución de canales para reducir el tráfico en los caminos más cargados de la red, así los caminos que estaban poco cargados reciben una parte de la carga, pero la latencia no aumentara mucho porque esto se desarrolla en la zona plana de la curva de latencia de la Figura 2.7.

2.5 ALGORITMOS DE CONTROL DE CONGESTIÓN EN TOPOLOGÍAS FAT-TREE

Actualmente existen dentro de la literatura algunas técnicas de control de congestión sobre topologías fat-tree. A continuación hacemos una pequeña reseña de estas técnicas [6]:

- **First Free (FF)**. Una técnica de encaminamiento que selección a el primer enlace físico que cuente con espacio libre de salida para encaminar el paquete. Muy sencilla su función de selección y rápida, pero no toma en cuenta el resto de la trayectorias, es un ejemplo de un algoritmo de encaminamiento Oblivious.

- **Static Switch Priority (SSP)**. Dado un nivel del árbol, la función de selección asigna una prioridad más alta a un diferente enlace de asenso de cada switch en ese nivel. La idea es crear caminos disjuntos de asenso con diferentes prioridades desde el primer nivel del árbol. Por consecuencia los paquetes inyectados encontraran diferentes caminos de ascenso desde el primer nivel hasta el último nivel del árbol, donde los paquetes serán encaminados de forma determinista.

- **Static Destination Priority (SDP)**. La selección de función de caminos asigna prioridad a los enlaces físicos en cada switch de acuerdo solo al destino de los paquetes. El enlace preferido o seleccionado estará dado por el componente menos significativo dentro del destino del paquete, el cual representa el puerto destino asignado en el primer del árbol.

- **Static Origin Priority (SOP).** Su método consiste en asignar prioridades a los enlaces físicos dependiendo el origen del paquete. El enlace preferido o seleccionado estará dado por el componente menos significativo dentro del origen del paquete, el cual representa el puerto destino asignado en el primer del árbol.

- **Static And Destination Priority (SADP).** Este algoritmo de encaminamiento toma en cuenta dos factores de la topología, primero el nivel al que pertenece el switch y el componente destino del paquete correspondiente a ese nivel. Es decir, que cuando un paquete arriba a un switch en cualquier nivel del árbol, el algoritmo se encarga de leer el destino marcado en el paquete y lo compara con el enlace de salida del switch, si el enlace de salida asignado con mayor prioridad presenta niveles de ocupación muy altos en los buffers de salida, la función de selección asigna otro camino, seleccionando el enlace con una prioridad menor que coincida con el destino.

- **Cyclic Priority (CP).** La técnica utiliza un algoritmo de *round robin* dentro del switch para escoger un diferente enlace físico de salida para los paquetes. Este algoritmo nunca tomo los niveles de ocupación de los buffer en los enlaces que selecciona.

- **More Credits (MC).** Se basa en el mecanismo de control de flujo de asignar créditos a los switches para encaminar paquetes. Esta técnica se encarga de seleccionar los enlaces que tienen más créditos para encaminar el paquete.

En el [6] los autores, analizaron el impacto de estas técnicas de algoritmos de control de congestión sobre topologías fat-tree. Los resultados mostrados colocan al algoritmo SADP como el que presenta mejores resultados, ya consigue bajos niveles de latencia y una mejor redistribución de la carga de tráfico en los enlaces de la red. En la Figura 2.8 se presenta una gráfica comparativa de las siete técnicas anteriores.

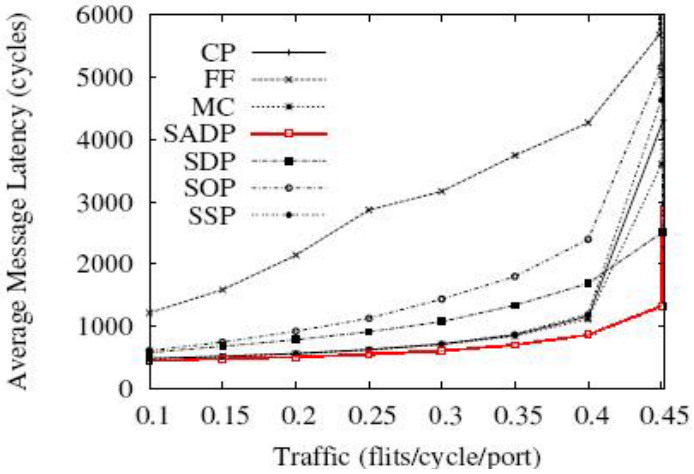


Figura 2.8. Gráfica comparativa de algoritmos en fat-tree

La gráfica muestra como la forma de encaminar los paquetes del algoritmo SADP, presenta niveles bajos de latencia. Los gráfica es el resultado de la simulación una red 4-ary 2-tree con tecnología Myrinet, con patrones de tráfico uniformes y *complement* (nodos envían al nodo opuesto).

CAPÍTULO 3

ARQUITECTURA INFINIBAND

3.1 DESCRIPCIÓN GENERAL

A continuación se describen las características operativas de la tecnología de red Infiniband (IBA), en las cuales se interviene para poder implementar dentro del este estándar el algoritmo de balanceo distribuido del encaminamiento (DRB).

La arquitectura Infiniband es utilizada para crear una red de área de sistema (SAN), lo cual permite la versatilidad de poder diseñar redes con un servidor, un solo nodo de procesamiento y algunos dispositivos de entrada y salida; hasta poder diseñar la red de interconexión un supercomputador, como ocurre actualmente con los sistemas mas potentes de la actualidad [1]. En la Figura 3.1 podemos observar una representación de una red Infiniband, de cómo los nodos de cómputo o de entrada/salida están agrupados en subredes y estas subredes interconectadas a través de routers.

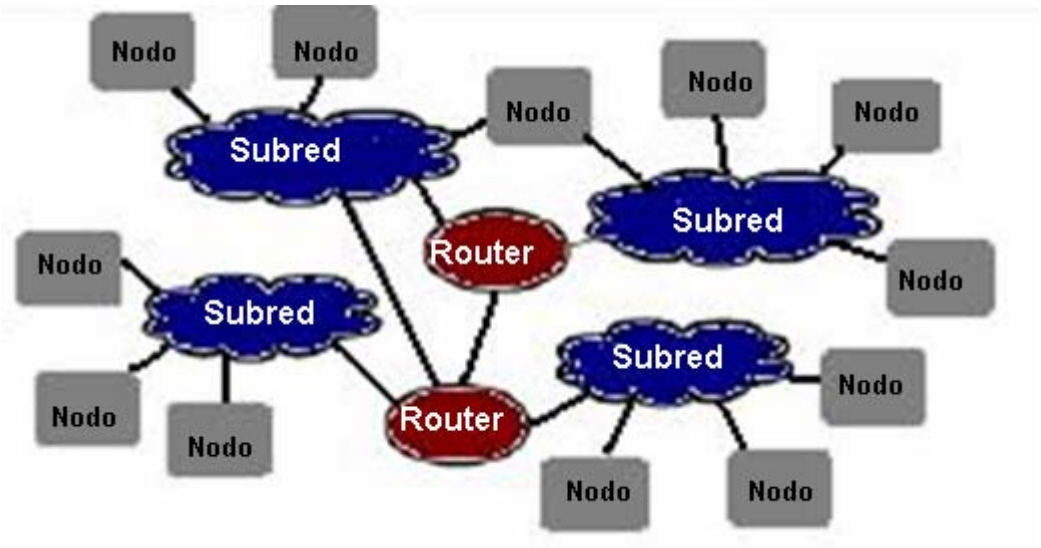


Figura 3.1 Red Infiniband

Se pueden diseñar redes de interconexión conmutadas punto a punto, con conmutadores (switches) y encaminadotes (routers) conectados en forma de cascada; de esta manera se logra la comunicación entre varios dispositivos con un elevado ancho de banda, niveles de latencia y *overhead* muy bajos, se permite la creación de múltiples caminos entre diferentes nodos, en un entorno protegido y gestionado remotamente.

El hardware de IBA permite gestionar comunicaciones altamente confiables y tolerantes a fallos entre nodos o dispositivos de entrada y salida sin involucrar el sistema operativo, lo que significa que permite al procesador del nodo estar dedicado solo al cómputo de la aplicaciones.

Infiniband cuenta con una semántica de canal (operaciones de envío y recepción) y el acceso directo a la memoria cuenta con un nivel de protección y aislamiento que impide el acceso a consumidores que no intervienen en la transacción.

Infiniband permite al usuario definir su red de interconexión con una topología en específico, los encaminadores, los elementos de conmutación, así como también la forma en que los datos y comandos se encaminarán por la red. La especificación de IBA divide la red en subredes (*SUBBETs)* que están interconectadas por medio de IBA routers y donde cada nodo puede comunicarse con cualquier subred existente., En la figura 3.2 se muestra la representación gráfica de una subred.

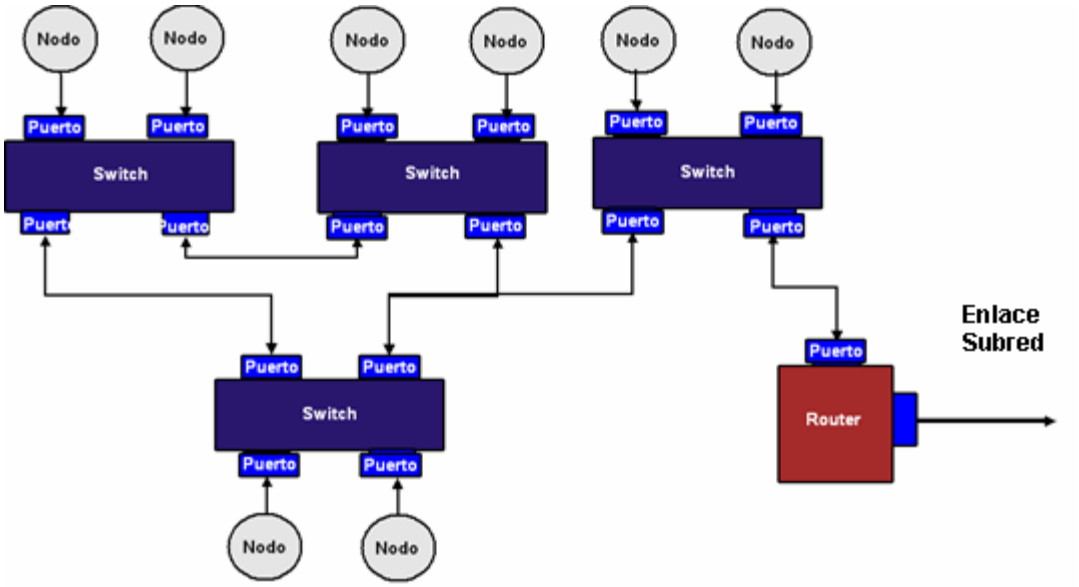


Figura 3.2 Subred Infiniband

3.2 COMPONENTES

3.2.1 Enlaces y Repetidores

Los enlaces son los medios físicos por donde se conectan conmutadores, nodos, encaminadores; estos enlaces pueden ser de cobre, fibra óptica, circuito impreso o cableado a una placa madre (*backplane*). Los repetidores prolongan las características de propagación de un enlace.

3.2.2 Subnet Manager

Cada una de estas subredes esta conformada por elementos de gestión de la subred (*Subset manager, SM*) y enlaces que los unen. El mecanismo de gestión es el encargado de:

- Reconocer la topología física de la subred
- Asignar identificadores locales (*LIDS*) a los puertos de los nodos, conmutadores y encaminadotes
- Establecer las trayectorias posibles entre los diversos nodos de la red
- Detectar cambios en la topología en la subred

Infiniband no define de qué manera se realizan las tareas del SM mencionadas anteriormente, sólo el orden y la estructura de datos que han de manejarse en su realización. El SM puede estar implementado en el nodo (nivel de software o hardware) o bien en un switch o en un router. Es por esto que puede existir más de un SM en una subred. Así que un mecanismo denominado *handover* entra en acción para asignar a uno la condición de *maestro* y ciertas prioridades a los demás SM.

EL gestor de subred es el encargado de la construcción de trayectorias alternativas cuando arranca la red de interconexión por primera vez, las trayectorias serán la primera opción para todos los pares fuente-destino. Infiniband no cuenta con un algoritmo para definir los caminos originales, por lo que se implementa uno, el *Depth-first search (DFS)*, que genera caminos múltiples entre un par fuente-destino, con una función que selecciona los caminos disjuntos, y es almacenado en las tablas de encaminamiento de los switches.

Las tablas de encaminamiento son distribuidas a todos los conmutadores y la red de interconexión comienza a trabajar con normalidad. Y el SM comienza a monitorear la red por si ocurre algún cambio en la topología.

3.2.3 Adaptadores de Canal

La arquitectura IBA permite que las unidades de procesamiento o *threads* de un nodo se conecten a la red Infiniband por medio de los adaptadores de canales (*Channels Adapters*) que proporciona uno o mas puertos IBA para la conexión. La forma de configurar, manejar y operar el adaptador de canal es por medio de una interfaz denominada capa de verbos (*IBA verbs layer*), la cual se encuentra entre el envío de mensajes o comunicaciones y el adaptador de canal.

El Host channel adapter (HCA) es uno de estos adaptadores de canal, encontrado en los nodos de procesamiento y el *Target channel adapter (TCA)* que se encuentra en los nodos de entra/salida (I/O). En la Figura 3.3 se muestra una representación de la arquitectura de un nodo de procesamiento para conectarse a la red infiniband.

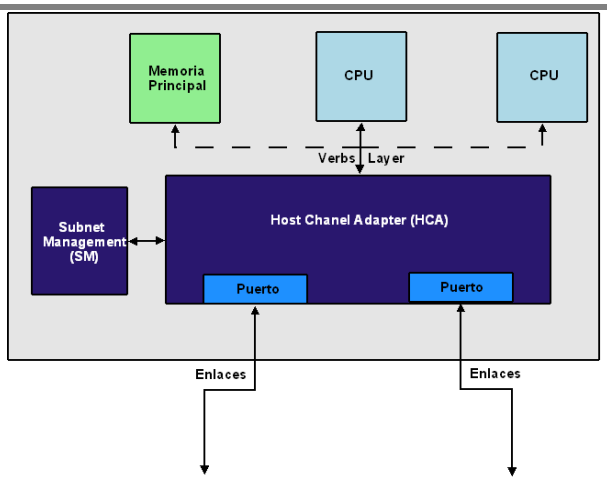


Figura 3.3. Endnode de procesamiento en una red Infiniband

Los procesadores y la memoria principal se encuentran conectados al (HCA) y entre ellos la gestión del HCA la capa de verbos. También podemos observar el SM que se comunica con el HCA para gestionar el envío y la recepción de paquetes del nodo.

La Figura 3.4 se muestra un nodo de entrada y salida con su arquitectura de conexión en una red infiniband. Los controladores de I/O proporcionan la interfaz a varios dispositivos de I/O, un arreglo de almacenamiento masivo es un ejemplo de un IOC. El adaptador de canal de estos nodos es el TCA qua su vez es gestionado por SM.

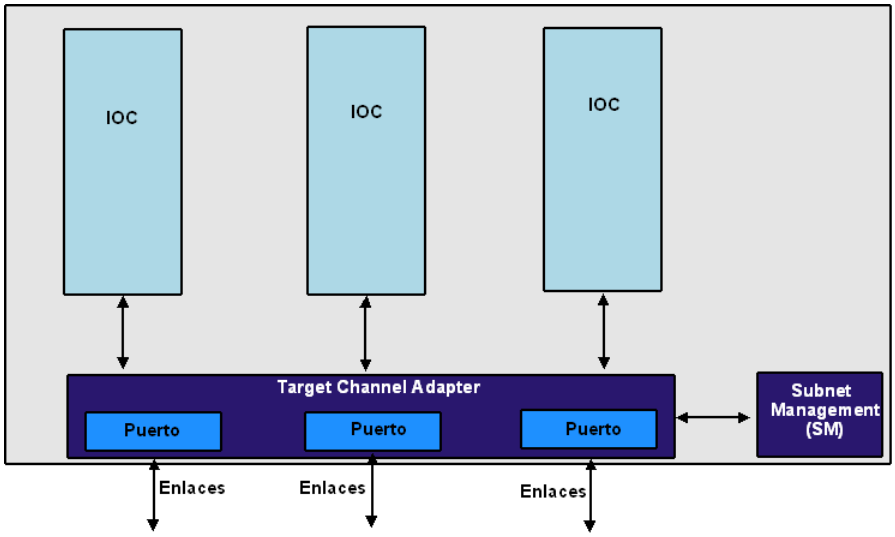


Figura 3.4. Endnode de entrada/salida en una red Infiniband

Estos adaptadores de canal son capaces de soportar múltiples puertos, asignando a cada uno de estos un identificador local (LID). Cada adaptador de canal tiene un identificador global único (GUID) asignado por Infiniband.

Cada puerto cuenta con un conjunto de buffers de transmisión de paquetes y recepción de paquetes, que trabajan de manera concurrente. Cada uno de estos dos buffers puede ser dividido en canales virtuales (*Virtual Lanes, VL)* y cada uno maneja su propio flujo de paquetes de manera independiente.

El recurso por el cual se comunica un gestor de subred para configurar el adaptador de canal, es el Agente de gestión de subred (SMA), que representa el medio en que los mensajes se comunican entre un nodo y otro gestor de subred de un elemento de la red. La arquitectura Infiniband utiliza unos datagramas de gestión (*management datagram, MAD)* para establecer comunicación entre los gestores de subred y los agentes de gestión.

3.2.4 Conmutador o Switch

Los conmutadores cumplen con una tarea principal, que es la de dirigir los paquetes hacia el destino marcado en sus cabeceras. Los elementos del conmutador son configurados en función de tablas de encaminamiento (*forwarding tables*), dentro de estas tablas de encaminamiento se almacenan los LIDs de los puertos en la red. Así cuando un paquete llega al *switch* analiza el LID destino marcado y lo encamina de acuerdo ala información almacenada en la tabla de encaminamiento. Como se ha comentado, el gestor de subred es el responsable de entregar y configurar las tablas de encaminamiento a los conmutadores.

El conmutador es capaz de soportar comunicaciones *unicast* y *multicast.* La comunicación *unicast* es un tipo de comunicación en la que un paquete corresponde a un solo nodo destino. En la comunicación *multicast* por consiguiente el paquete puede ser entregado a varios nodos destinos. En la Figura 3.5 observamos la estructura básica de un conmutador Infiniband

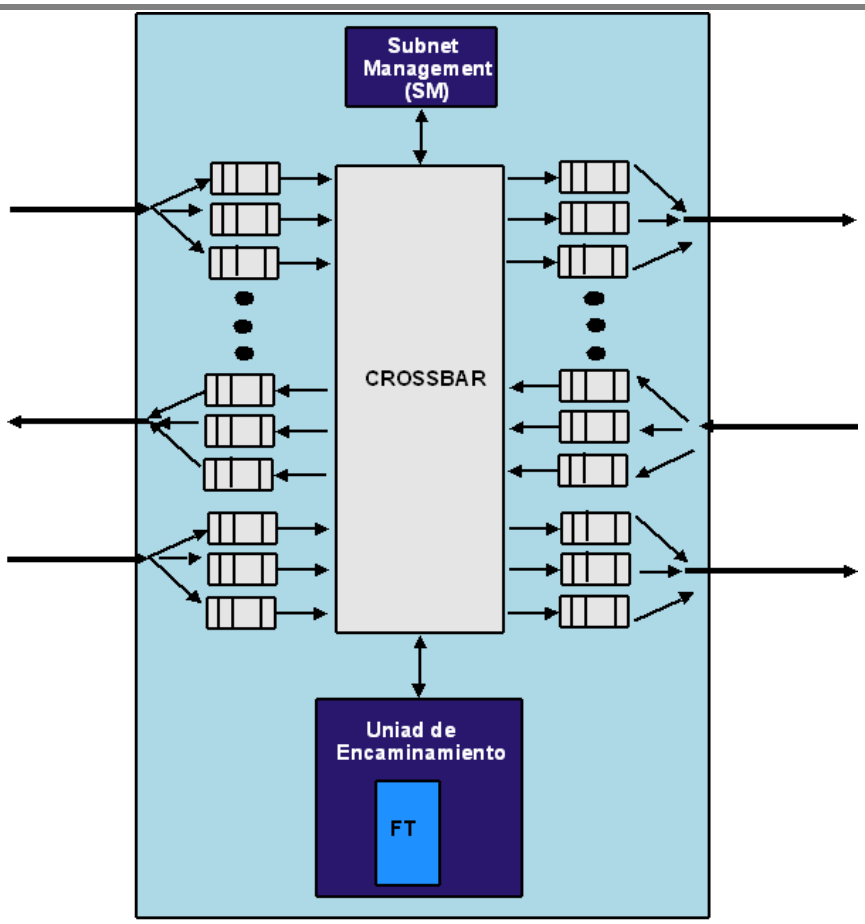


Figura 3.5. Switch Infiniband

En la figura 3.5 el crossbar es el encargado de comunicar todos los enlaces de entrada con todos los de salida. La unidad de encaminamiento es donde entra en acción el algoritmo encaminamiento que se implementa en al red, como podemos ver en la figura el encaminamiento es llevado acabo por tablas. El switch también cuenta con el SM, donde se gestiona la llegada y envío de los paquetes dentro del switch.

3.2.5 Canales virtuales

Los canales virtuales (VL) es una técnica para crear múltiples enlaces virtuales dentro de un solo enlace físico. Cada puerto de un canal virtual representa un control de flujo independiente, lo que se traduce, en que si uno se congestiona o se cae, no afecta al resto de los caminos virtuales dentro del enlace. En la Figura 3.6 se muestra la representación de los canales virtuales donde se ve la partición de los *buffers* para implementar cada canal virtual.

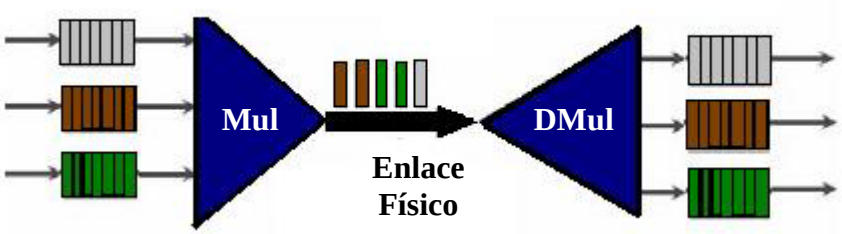


Figura 3.6 Canales virtuales

Los canales virtuales que un puerto utiliza están configurados por el SM, basado en el nivel de servicio (*Service Level, SL)* del puerto. El nivel de servicio es un atributo que tiene un paquete en la cabecera, con 16 tipos de niveles de servicio. A medida que el paquete es encaminado por la red, el valor del nivel de servicio es el que le indica qué canal virtual tomar para continuar su camino. Cada puerto cuenta con una tabla de conversión de nivel de servicio a canal virtual (*SltoVL mapping Table)* que es la que permite el envío por el canal virtual apropiado.

3.2.6 Agente de Control de Congestión

El mecanismo de control de congestión estándar en infiniband está implementado en el *congestion control agent* (CCA). Este agente actúa de manera que detecta la congestión, la notifica y regula la inyección de mensajes. La forma en que conoce el estado de ocupación de los buffers de los canales virtuales de cada uno de los puertos que componen switch es por medio de elementos de medición (*performance counters*). Este mecanismo de congestión entra en acción cuando se supera un cierto umbral de ocupación de los buffers. En la figura 3.7. se muestra gráficamente el funcionamiento del agente de control de congestión.

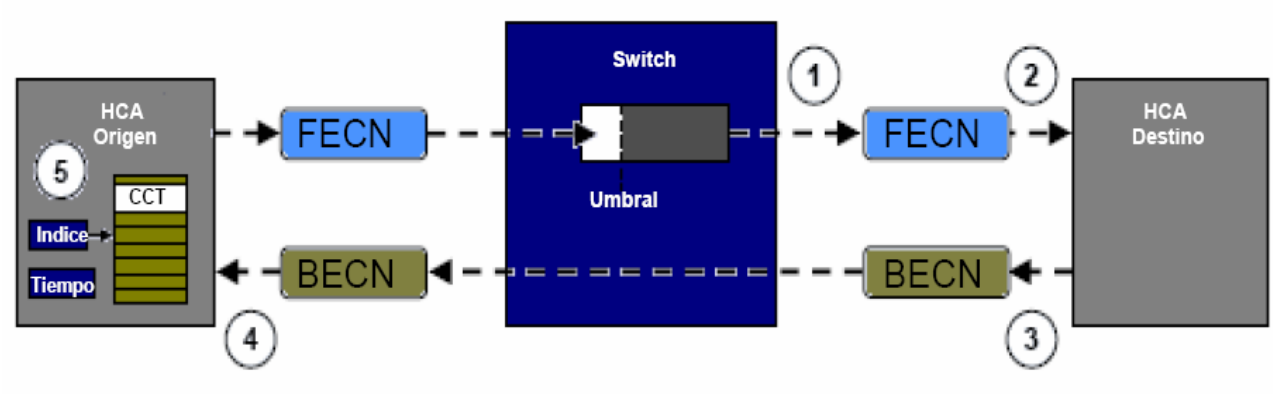


Figura 3.7. Agente de control de congestión de Infiniband

Cuando el switch detecta la congestión (paso 1 de la Figura 3.7.), comienza a marcar con un bit la cabecera (*Base Tranport Header*, BTH, en el estándar Infiniband) de los paquetes que están estacionados en el buffer del canal virtual que ha superado el umbral, este bit es el *Forward Explicit Congestion Notification (FECN).* Registra dos valores, el 0 significa que es perfectamente controlable el volumen de comunicaciones en ese enlace, y se guarda 1 si el paquete ha seguido una trayectoria por donde ha sufrido retrasos debido a problemas de congestión en un enlace.

El paquete continúa su trayectoria original (paso 2 de la Figura 3.7.), hasta llegar a su destino y en el adaptador de canal de ese nodo destino se examina la cabecera del paquete y si el valor almacenado es 1, el agente de control de congestión, detecta que en la trayectoria de este par de nodos existe congestión, la notificación al destino se hace a

través del envío de un mensaje de notificación (*Backward Explicit congestion Notification, BECN*).

Cuando el nodo fuente recibe el paquete marcado con el bit 1 dentro del BECN, el agente de control de congestión indica al nodo que regule la inyección de paquetes a ese nodo destino (paso 4 de la Figura 3.7.), esta regulación tiene como objetivo no colapsar la red, pero los niveles de latencia que resultan de este control de de inyección de paquetes no son aceptables dentro de sistemas de altas prestaciones.

La inyección de paquetes es regulada en Infiniband por medio de una tabla parametrizable denominada *Congestion Control Table* (CCT), cada lugar en la tabla determina un valor de tiempo interenvío para regular la inyección. Si llegan mas paquetes marcados a este nodo, se incrementan las posiciones en la CCT. Conforme la congestión comienza a disminuir sus niveles críticos, la inyección de paquetes del nodo se va normalizando (paso 5 de la Figura 3.7.). Todos estos parámetros en la tabla, umbrales, índices, etc. se configuran de manera estática o dinámica en el *congestion control manager* (CCM).

3.3 CONEXIONES

Dentro de Infiniband se ofrecen servicios de comunicación para el transporte de los mensajes dentro de la red. A continuación alguna de los más usados:

- **Conexiones confiables.** Es un tipo de conexión que asocia un par de colas local con otro par de colas en específico. Por tanto cada consumidor debe crear un par de colas nuevo cada vez que desee comunicarse con un consumidor remoto diferente. El adaptador de canal establece mensajes de reconocimiento de la red. Así la comunicación entre los consumidores se mantiene a pesar de errores, saturación de buffers, congestión, y fallos de la red.

- **Conexiones no confiables.** Conecta un par de colas con otro par remoto, pero a diferencia de las conexiones confiables, el Adaptador de canal no gestiona ningún mensaje de reconocimiento de la red, por consecuencia en caso de algún percance o congestión se pueden perder los paquetes que circulan por la red, pero se gana en ancho de banda al no contar con los mensajes de notificación.

3.4. ARQUITECTURA POR CAPAS

La arquitectura de Infiniband se divide en una serie de capas, cada una es independiente de la otra, pero se permite que una capa superior proporcione servicios a las capas inferiores, las capas son las siguientes:

- **Capa física.** Especifica la colocación de los bits en el medio conductor por el cual serán enviados.
- **Capa de enlace.** Establece el formato, protocolos, tratamiento y operación del paquete.
- **Capa de red.** Se designa el protocolo para encaminar un paquete entre subredes de la red
- **Capa de transporte.** Responsable de operaciones de segmentación de los mensajes en paquetes. De igual manera configura los puertos para que procesen los paquetes recibidos y conformar el mensaje.
- **Capas superiores.** En estas capas se pueden implementar una gran variedad de protocolos para funciones de gestión y servicios de la subred.

A continuación la Figura 3.8 ilustra mejor la arquitectura de capas de Infiniband, donde se detalla de igual forma la función de cada capa.

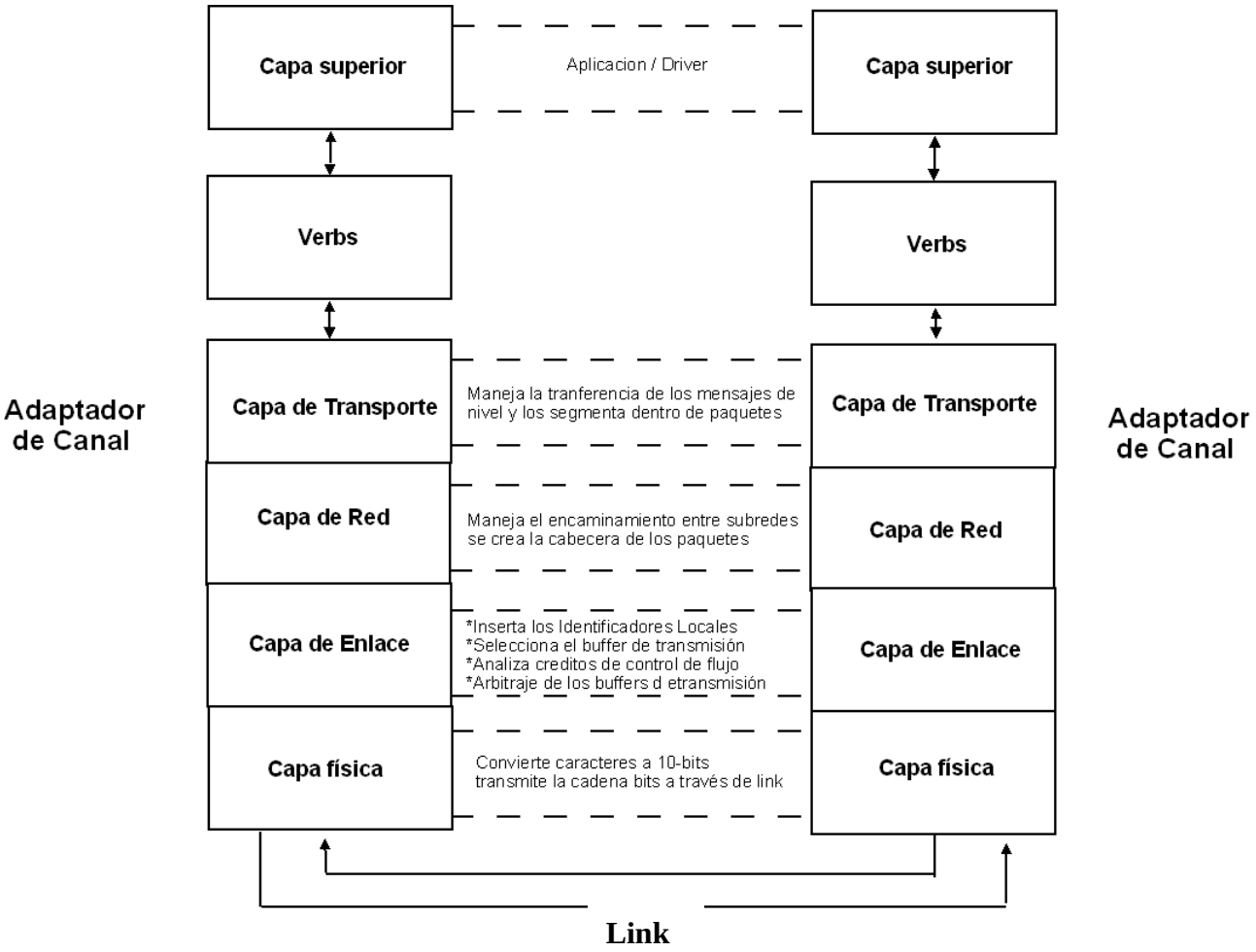


Figura 3.8. Arquitectura de capas en Infiniband

CAPÍTULO 4.
OPNET: PLATAFORMA DE SIMULACIÓN Y DESARROLLO

En el trabajo de búsqueda de un buen simulador en [5], justifican una plataforma que recreara un modelo lo más fiable posible a la realidad, al menos en cuanto a las características en redes de interconexión, para poder aplicar conclusiones sobre los resultados obtenidos mediante la simulación. Se llega a la conclusión que Opnet Modeler, es actualmente un simulador que cumple con los requerimientos para nuestro trabajo de investigación.

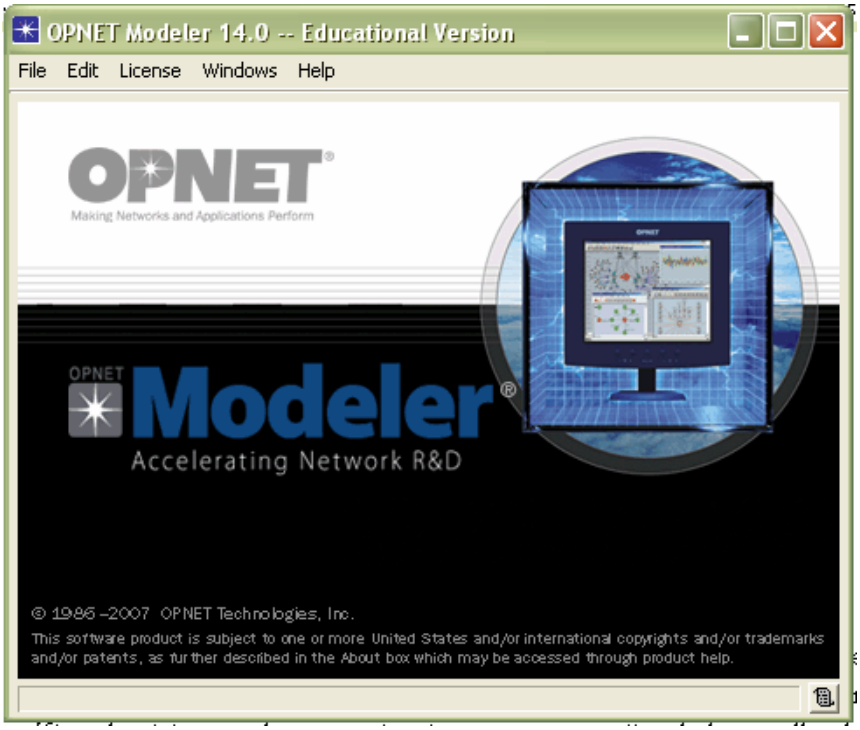


Figura 4.1. Opnet Modeler

Opnet Modeler es un simulador de red que ofrece herramientas potentes con el objetivo de diseñar modelos, simular datos y analizar las redes. Es una potente herramienta de simulación gráfica de sistemas de comunicaciones que permite el desarrollo de modelos complejos de redes, es decir simular una gran variedad de redes, el flujo de mensajes de datos, paquetes perdidos, mensajes de flujo de control, caídas de los enlaces, la conexión de diferentes tipos de nodos, utilizando diferentes tipos de enlaces, etc.

Opnet es un lenguaje de simulación orientado a las comunicaciones. Proporciona acceso directo al código fuente. La herramienta utiliza un entorno de desarrollo de modelación

de redes de interconexión y sistemas distribuidos. El comportamiento y el rendimiento de los sistemas modelados pueden ser analizados mediante la realización de simulaciones específicas sobre cierta clase de evento en la red. El entorno de Opnet Modeler incorpora herramientas para todas las fases de un estudio científico, como son, los modelos de diseño, de simulación, la recopilación de datos y análisis de datos.

OPNET especifica tres fases para llevar acabo una simulación:

1. **Especificación del modelo**, es decir desarrollar la representación del sistema a estudiar, dentro de esta fase se crean los modelos con los editores de opnet.
2. **Selección de estadísticas y simulación.** el siguiente paso es elegir los datos a recolectar
3. **Análisis.** Seguidamente se estudian los resultados obtenidos de la simulación del modelo. En el caso de que estos resultados no fuesen los deseados, se tendría que hacer una re-especificación del modelo donde se modificarían los aspectos erróneos del modelo simulado.

Opnet Modeler se basa en una jerarquía básica para poder plantear las simulaciones:

- **Modelo de Red.** Estarán definidas las redes y subredes de la simulación.
- **Modelo de Nodos.** Irá definida la estructura interna de los nodos.
- **Modelo de Procesos.** Se definen los estados de un nodo en la simulación.

Cada una de estas jerarquías es diseñada por los editores del OPNET Modeler, que proporcionan las herramientas necesarias para la creación de la topología de la red, las tareas distintas de la red y los elementos que influyen en el desempeño de la red. Las herramientas de Opnet Modeler serán descritas a continuación.

4.1. PROJECT EDITOR.

El Project editor es el principal escenario en la creación del entorno de la simulación de la red. Es usado para crear un modelo de red, también podemos crear nodos, construir formatos de paquetes, etc.

Este editor contiene tres tipos básicos de objetos: subredes, nodos y enlaces. Las paletas de objetos (*object palette)* ordenan los objetos disponibles en categorías. Por ejemplo, en la paleta *ethernet*, se encuentran los nodos y enlaces más utilizados para el diseño de este tipo de redes. Al seleccionar la opción de ver resultados (*view results*), aparecen las estadísticas disponibles. En la Figura 4.2 se muestra una imagen del Project Editor.

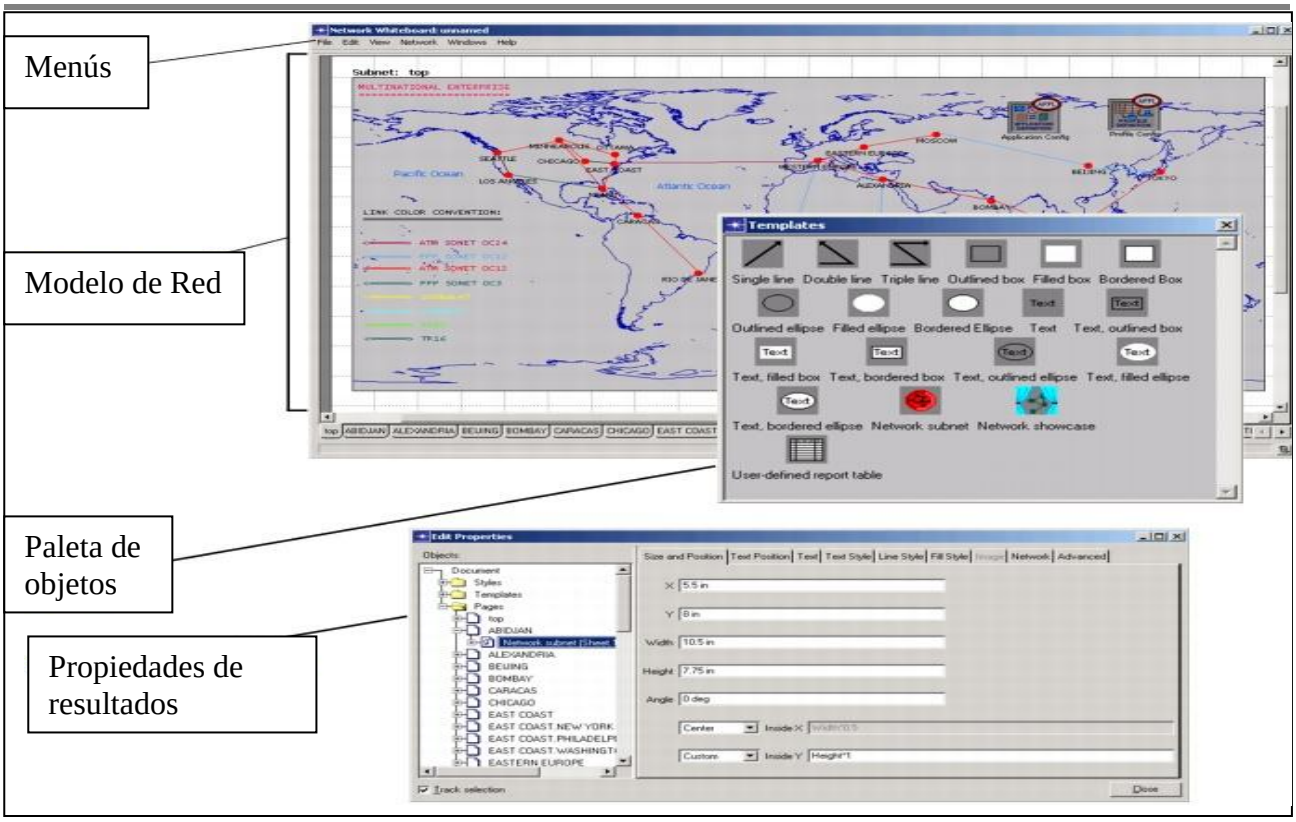


Figura 4.2 Project Editor

4.2. NODE EDITOR.

El Node Editor es un editor que es usado para crear modelos de nodos especificando su estructura interna. Estos modelos se utilizan para simular nodos en el interior de la red en el Project Editor.

Un nodo puede tener en su interior varios módulos. Cada módulo tiene una función específica dentro del nodo, tal como: generar paquetes, encolarlos, procesarlos o transmitirlos y recibirlos. Los módulos que se ofrece son los siguientes:

	Icono estándar para un módulo de procesado. Su función principal es construir bloques de modelo de nodo
	Icono estándar usado para un módulo de cola. Lo que le hace diferente del resto de módulos es que el módulo de colas tiene recursos adicionales internos llamados subcolas
	Icono usado para un módulo de trasmitir. Trasmisor punto a punto
	Icono usado para un módulo de recibir. Receptor punto a punto
	Icono usado para un módulo de trasmitir. Trasmisor tipo bus
	Icono usado para un módulo de recibir. Receptor tipo bus
	Icono usado para un módulo de trasmitir. Transmisor por radiofrecuencia

	Icono usado para un módulo de recibir. Receptor de radiofrecuencia
	Icono estándar usado para las comunicaciones

Los módulos se conectan entre ellos para realizar las funciones específicas de la red que se está modelando. Las conexiones lógicas entre los módulos que ofrece el Node Editor son:

	**Packet stream**: conexiones que llevan a los paquetes desde un módulo fuente a un módulo destino
	**Statistic wires**: transportan datos en un módulo fuente a un módulo destino. Sirven como interfase para que un módulo fuente pueda compartir datos con un módulo destino, y proporcionar información respecto de su estado
	**Logical associations**: su misión es indicar qué relación existe entre dos módulos de la simulación. Por tanto, no transportan datos entre módulos

4.3. PROCESS EDITOR.

Se utiliza en la creación de modelos de procesos, que a su vez controlan los modelos de nodo creados en el Node Editor. La funcionalidad de cada módulo se define a través de modelos de proceso, que se representan mediante máquinas de estados finitos (FSM) y son creados por iconos que representan estos estados y por líneas que representan las transiciones entre ellos.

Las operaciones que realizan cada estado o cada transición se escriben en lenguaje C/C++. A este tipo de simulación se le denomina simulación DES (Discrete event simulation). Más adelante explicaremos en qué consiste ésta. En la figura 4.3 se muestra una representación gráfica del Process editor, donde se visualiza un diagrama de estados y el código relativo a uno de ellos (*init*).

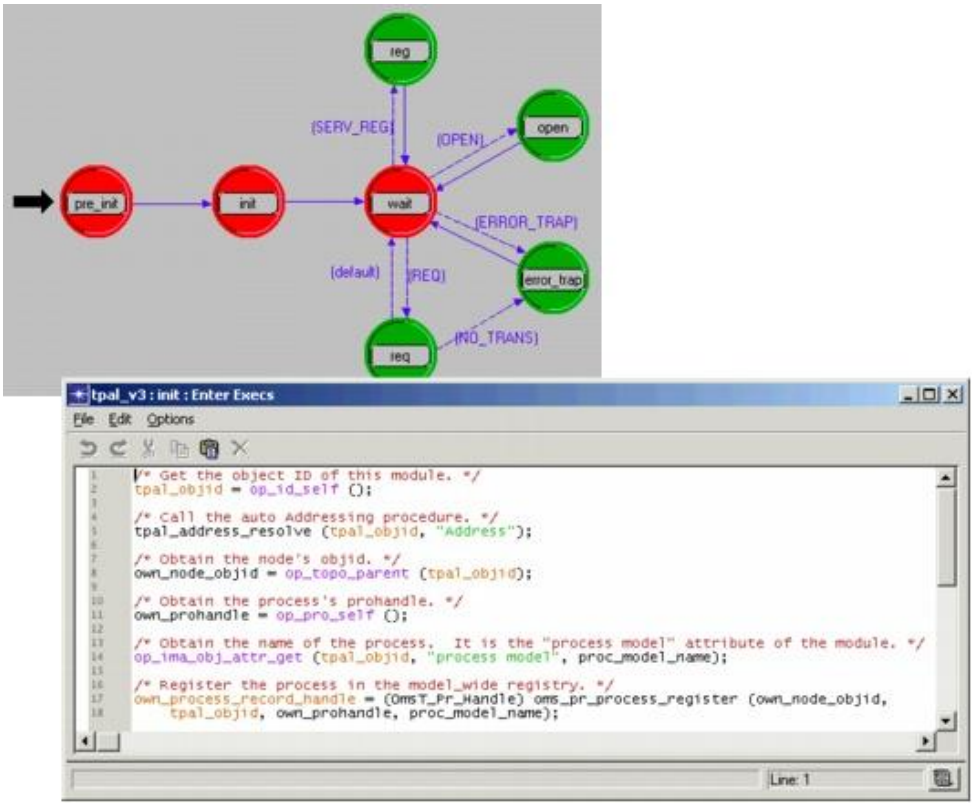


Figura 4.3. Process Editor

En este editor se colocan los diferentes estados. Todos ellos están compuestos de dos partes: Una parte llamada *enter executives,* donde se alberga la programación ejecutada por las transiciones incidentes al estado, y otra parte denominada *exit executive,* en la que se ejecutan sus instrucciones cuando el proceso sale del estado hacia una transición.

Existen dos tipos de estados

- **Estado forzado**. Un estado forzado es aquel que cuando le llega una transacción de un proceso, ejecuta las instrucciones que tiene su *enter executives* e inmediatamente ejecuta las instrucciones que tiene su *exit executives*. A continuación el proceso saldrá por una transición.

- **Estado no forzado.** Un estado no forzado permite hacer una pausa entre el *enter executive* y el *exit executive*. Cuando un proceso llega a este estado, en primer lugar se ejecuta el enter executives y se forma una pausa hasta que sucede una nueva invocación o llamada a este estado. Cuando esta invocación se ha producido, se ejecuta el *exit executive* y el proceso pasa a la transición de salida.

Por lo tanto los objetos más importantes de este editor son:

- **Estados:** Cada uno de ellos representa un proceso. Se definen en él las funciones a realizar durante su ejecución.

- **Transiciones**: Marcan la condición que se necesita para pasar de un estado a otro.

- **Bloques:** Sirven para la programación, la declaración de variables, funciones, etc.

4.4. LINK MODEL EDITOR.

Este editor nos ofrece la posibilidad de crear nuevos tipos de objetos link. Cada nuevo tipo de link puede tener diferentes atributos y representaciones.

Un *link model* especifica la información siguiente:

- **Tipo de enlace soportado:** punto a punto duplex, punto a punto simple, bus.

- **Keyword:** Sirve para simplificar la paleta de objetos del Project editor.

- **Comentarios:** Este apartado nos permite añadir comentarios a nuestros enlaces.

- **Especificación de atributos:** Podemos cambiar los valores de los atributos del enlace.

4.5. PAKET FORMAT EDITOR.

Este editor es utilizado para la definición de la estructura interna de un paquete, con un conjunto de campos. Para cada uno de los campos el *packet format* especifica un único nombre, tipo de datos, valor por defecto, tamaño en bits, comentarios opcionales, etc.

Los *packet formats* son atributos de los módulos de transmisión y recepción del modelo de nodos. El formato de un paquete contiene uno o más campos, representados en el editor como cajas rectangulares. El tamaño de la caja es proporcional al número de bits específicos del campo.

4.6. PROBE EDITOR.

Probe Editor se utiliza para definir la recopilación de datos necesarios antes de ejecutar una simulación. Puesto que la mayoría de modelos de simulación generan grandes volúmenes de datos, si todas las posibles salidas se registran, Modeler proporciona este editor al usuario para que pueda designar explícitamente lo que es de interés para él. Cada resultado deseado se activa mediante la creación de un objeto llamado *probe*. Grupos de *probes* se guardan como una lista de *probe,* a fin de que puedan ser utilizados colectivamente cuando se ejecuta simulaciones.

4.7. SIMULATION SEQUENCE EDITOR.

En este editor proporciona un acceso total a las secuencias de simulaciones, permitiendo con esto:

- Agregar y eliminar conjunto de simulaciones
- Editar los atributos de un número de simulaciones
- Seleccionar un subconjunto de simulaciones para ejecutar

4.8. SIMULACIÓN DES.

En una simulación DES se utilizan eventos para describir sucesos o acciones que tienen lugar en un determinado momento. Cada uno de estos eventos tiene un instante de incidencia en una escala temporal. Esta escala, al igual que el resto de magnitudes, es discreta.

Durante la simulación, se puede distinguir entre el tiempo real y el tiempo de simulación, por ello se utilizan variables contador para representar el momento actual y las cantidades de tiempo. También se utilizan varias variables de estado para representar la fase del sistema simulado. El sistema, evoluciona en la memoria del ordenador, produciéndose diferentes eventos que crean las variables de estado, y éstas a su vez determinan los futuros eventos.

Cada evento es representado sobre la escala temporal discreta de la simulación ocupando una única posición. De este modo, dichos eventos logran ser ordenados cronológicamente, según su instante de incidencia, para ser procesados. En este tipo de simulación, el evento es la unidad de ejecución. Cada uno describe una acción, y el resultado de ésta es la modificación de las variables de estado. Esta característica está especialmente soportada por los lenguajes de programación orientada a objetos.

El simulador DES debe tener un bloque que inicialice todas las variables de estado del sistema simulado, un procesador que ejecute eventos, un modulador que sincronice los bloques, asegurando que los eventos se ejecutan en el orden adecuado y un recolector de datos estadísticos que tome nota de lo ocurrido. Por último, al finalizar la simulación o de manera dinámica durante su ejecución, se podrán procesar los datos recogidos para extraer la información deseada con la posibilidad de representarlos de forma gráfica.

Durante la actualización de contadores de tiempo y la modificación de las variables de estado, pueden generarse nuevos eventos que se insertarán en la lista de eventos a procesar o modificarán los atributos de los que ya existentes.

CAPÍTULO 5.
ANÁLISIS Y DISEÑO

Como fue mencionado con anterioridad el algoritmo DRB basa su funcionamiento principalmente en tres fases: Monitorización de la carga de tráfico, Configuración de dinámica de trayectorias alternativas, y Selección de caminos múltiples. A través de estas etapas describiremos el método en que se ejecutara DRB en una Red de interconexión con topología Fat-tree, tomando en cuenta la arquitectura Infiniband.

A pesar de la técnica de control de congestión estándar de Infiniband, los niveles de latencia de los mensajes no disminuyen, y la congestión que existía en los conmutadores, se traslada a los nodos. Es por esto que la implementación de DRB en el estándar Infiniband tiene como objetivo aportar un técnica mas efectiva al CCM, que es la de aportar trayectorias alternativas para los paquetes del mensaje, como fue demostrado en el trabajo [5]. Continuando con esta investigación, proponemos tres fases principales para implementar DRB sobre la topología Fat-tree, que serán descritas a continuación:

5.1 FASE 1: MONITORIZACIÓN DE LA CARGA DE TRÁFICO

La monitorización será constante y será dentro de los canales virtuales de cada puerto de los switches, se analizaran los niveles de ocupación y se definirá un valor de umbral comprendido entre el 0 (ningún paquete es marcado) y el 15 (nivel de congestión alto).

Cuando se detecta congestión dentro de los buffers del switch, el mecanismo de DRB se encarga de notificar al nodo destino de esta información, es decir, comienza a poner en *uno* el bit de FECN en la cabecera de transporte (Base Transport Header, BTH) de todos los paquetes. Notifica primero al destino por que en Infiniband los switches no deben generar paquetes para no contribuir negativamente en el caso de una congestión.

Una representación de esta fase puede ser vista en la figura 4.1, donde observamos cómo DRB entra en acción al superarse un nivel de umbral predefinido como congestión en uno de los conmutadores. DRB comienza a marcar paquetes con el bit FECN, que provenían del nodo (1) y se dirigen al nodo destino (2). DRB encamina el paquete a su destino original, pero el paquete contiene información sobre los niveles de congestión registrados en la trayectoria original.

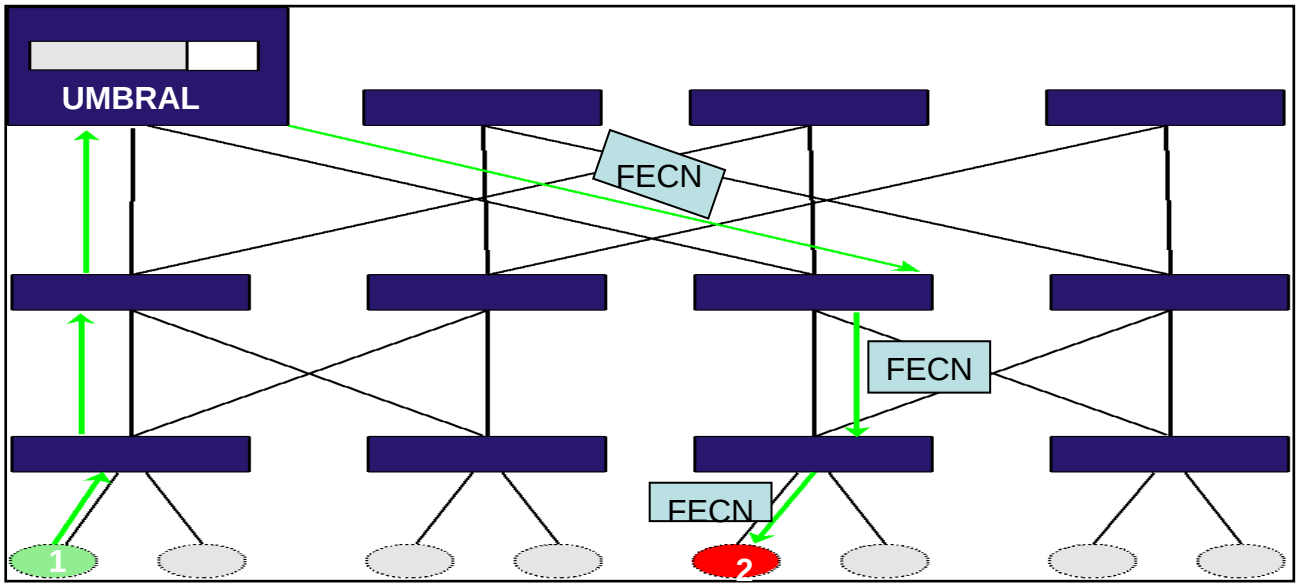


Figura 5.1 Fase de Monitorización, se supera el umbral en una red 2-ary 3-tree

5.2 FASE 2: NOTIFICACIÓN DE LA CONGESTIÓN

Cuando el paquete con el bit de FECN llega al adaptador de canal (HCA) del nodo destino, el agente de control de congestión (CCA) se encarga de enviar un mensaje con la cabecera del paquete marcada con el bit de BECN al nodo fuente. Este paquete con el bit BECN puede ser un mensaje de reconocimiento (*Acknowledge message*) en el caso de que sean comunicaciones confiables o en caso de que sean comunicaciones no confiables es un paquete especial de notificación de congestión.

En la figura 4.2, el nodo (2) recibe los paquetes marcados con la información de congestión en los conmutadores que visitó hasta llegar a este nodo. El CCA de este nodo se encarga de mandar al origen el mensaje de notificación de congestión hasta el nodo origen, marcando este mensaje con el bit BECN.

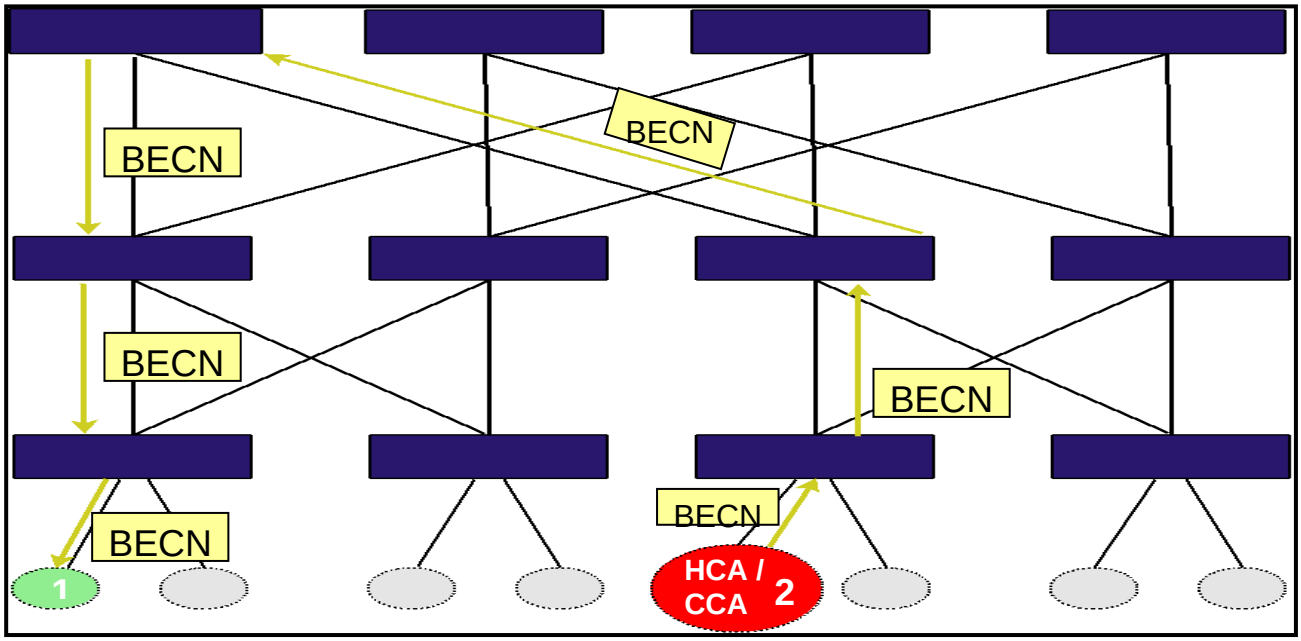


Figura 5.2 Fase de Notificación de congestión del destino al origen

5.3 FASE 3: CÁLCULO Y SELECCIÓN DE TRAYECTORIAS ALTERNATIVAS

Al momento de que el nodo fuente comienza a recibir paquetes con la notificación de congestión o con bits BECN de un destino, el CCA de este nodo se encarga de planificar los siguientes caminos a tomar para los paquetes que se dirijan al destino en cuestión. La evaluación de los siguientes caminos se hace en base al calculo de la cantidad de paquetes de notificación recibidos y a la distribución a través del conjunto de trayectorias alternativas entre el par fuente/destino. DRB se encargará de que los siguientes paquetes se encaminen por los caminos menos ocupados.

Un ejemplo, en un escenario donde entre un par fuente/destino existen cuatro caminos disponibles entre estos, y al nodo fuente ha recibido 55 BECNs por el camino 1, 20 BECNs por el camino 2, 15 BECNs por el camino 3, y 10 BECNs por el camino 4. 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 camino 1, y por lo contrario, el camino 4 es el óptimo para continuar la inyección de paquetes hacia el destino. En DRB la selección del camino alternativo se realiza en función de la inversa de latencia acumulada en el camino. De esta forma se asegura reducir los niveles latencia de los mensajes entre ese par fuente destino, lo descrito anteriormente se refleja en la Figura 4.3.

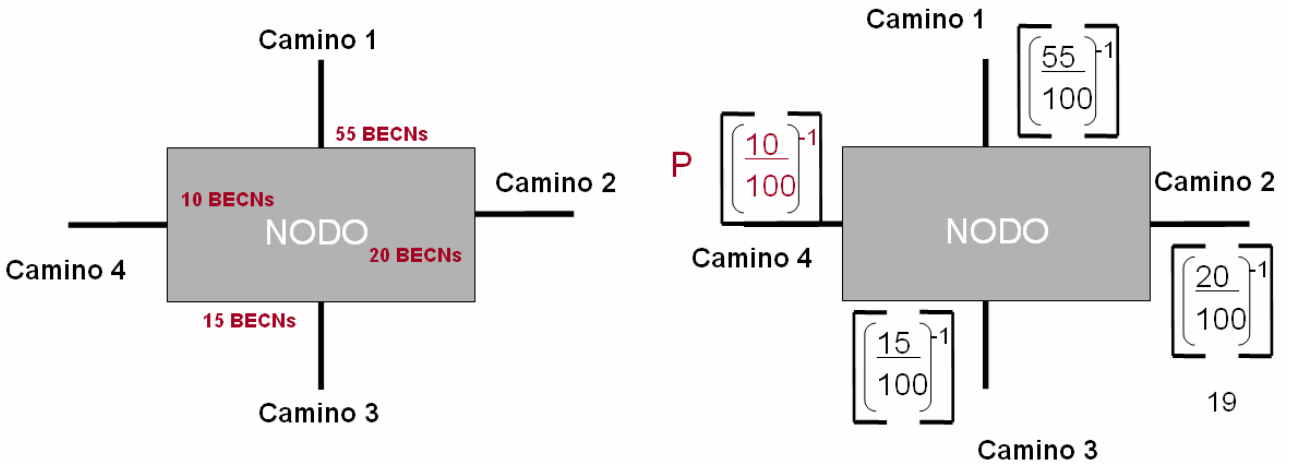


Figura 5.3. Calculo de caminos alternativos entre un par fuente destino

De igual forma se contempla que los caminos seleccionados sean cortos en distancia hasta el destino y disjuntos, ya que de lo contrario podrirá ocurrir que el tiempo de transmisión o el nivel de latencia sea demasiado grande, y se vuelva a pasar por zonas próximas a la congestión.

En la siguiente figura 4.3, podemos observar como el SM del nodo origen, puede encaminar los siguientes paquetes por dos caminos alternativos hasta el destino, en base a los niveles de ocupación de los puertos de salida.

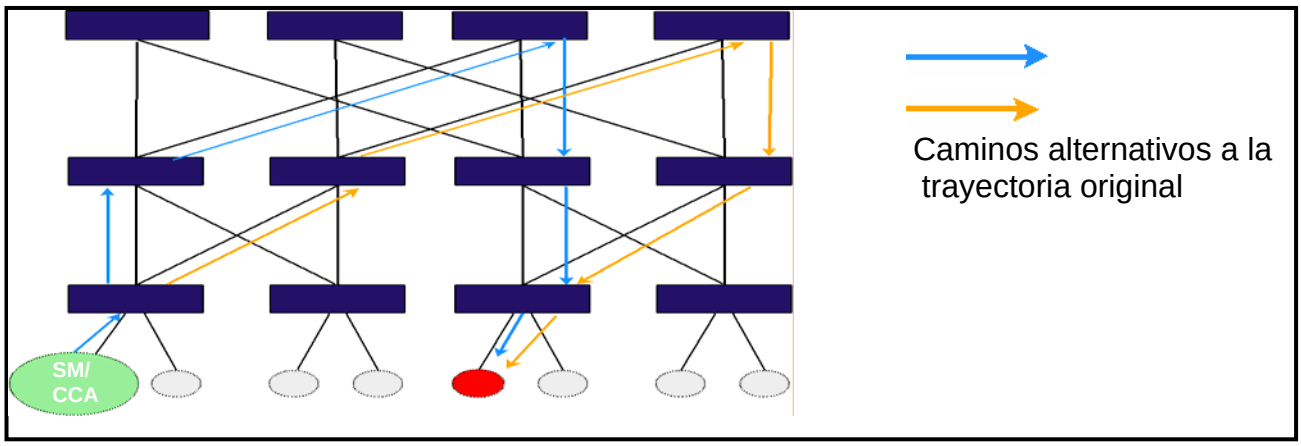


Figura 5.4 Fase de Construcción de caminos alternativos

El SM de las subredes juega una función importante en esta etapa, ya que SM del nodo fuente de congestión se comunica mediante paquetes de gestión (Management datagrams, MADs) con los otros SM de otros componentes 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 la mascara de control (*Local Control Mask, LMC)* con la cantidad de trayectorias alternativas de ese elemento.

El LMC es un campo integrado dentro de un LID de 16 bits, Figura 4.5. Cuando se recibe un paquete dentro de un switch los 8 bits menos significativos son descartados, de esta

forma se modifica el valor de la mascara, para asignar a los puertos de los CA varios LIDs. De esta manera el SM puede establecer varios caminos para el mismo nodo. [5]

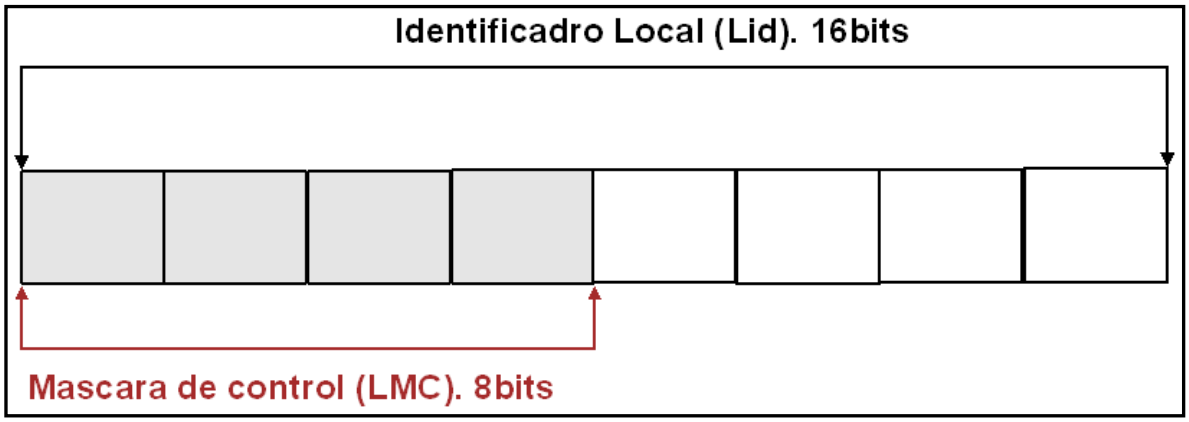


Figura 5.5. Campo LID del paquete y la LMC del mismo LID

Cada adaptador de canal opera con un contador que mide el tiempo transcurrido hasta que el último paquete de notificación de congestión llega al nodo. Este contador concluye en base a un parámetro que se indica en el gestor de control de congestión del nodo.

En este caso cuando el tiempo del contador finaliza y no se ha recibido ningún paquete con la notificación de congestión, el camino conformado por la múltiples trayectorias disjuntas se contrae y se evita así que se tracen caminos alternativos cuando no se requieren.

Hasta aquí se han descrito, las diversas alternativas adoptadas con el fin de aplicar el balanceo distribuido del encaminamiento a la arquitectura Infiniband y la topología Fat-tree. Utilizando las características beneficiosas y provechosas que facilita el estándar Infiniband, y fusionarlas con la filosofía básica de DRB (monitorización de la carga de tráfico, cálculo de nuevas trayectorias, y selección de caminos alternativos) y así respetar la idea conceptual de una tecnología de red y de un algoritmo de encaminamiento para actuar en un tipo de red como fat-tree.

CAPÍTULO 6.

IMPLEMENTACIÓN

En este capítulo abordaran los detalles de implementación del balanceo distribuido del encaminamiento sobre una red infiniband y en una topología fat-tree en el simulador Opnet Modeler. En el [5] fueron desarrollados unos modelos para una red infiniband, son tomados como base y modificados para crear nuevos modelos para poder adaptar DRB a una topología fat-tree.

Como se menciono en el capítulo anterior, Opnet respeta tres jerarquías básicas para desarrollar el modelo de una red. A continuación se describen cada uno de los modelos creados en las tres jerarquías de Opnet.

6.1 MODELO DE RED

El diseño de la red comienza colocando la cantidad de nodos con los que contara la red de interconexión y los conmutadores que gestionaran comunicaciones entre los nodos. En la Figura 6.1 observamos como son representados los nodos y los switches en Opnet.

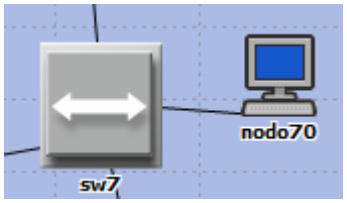


Figura 6.1. Nodo y switch en Opnet

Una vez colocados los nodos y los switches, el siguiente paso es interconectarlos con enlaces bidireccionales y así crear la red de interconexión con la topología prevista, en este caso la topología es un fat-tree, la Figura 6.2 es un ejemplo de una de los fat-tree creados con Opnet Modeler.

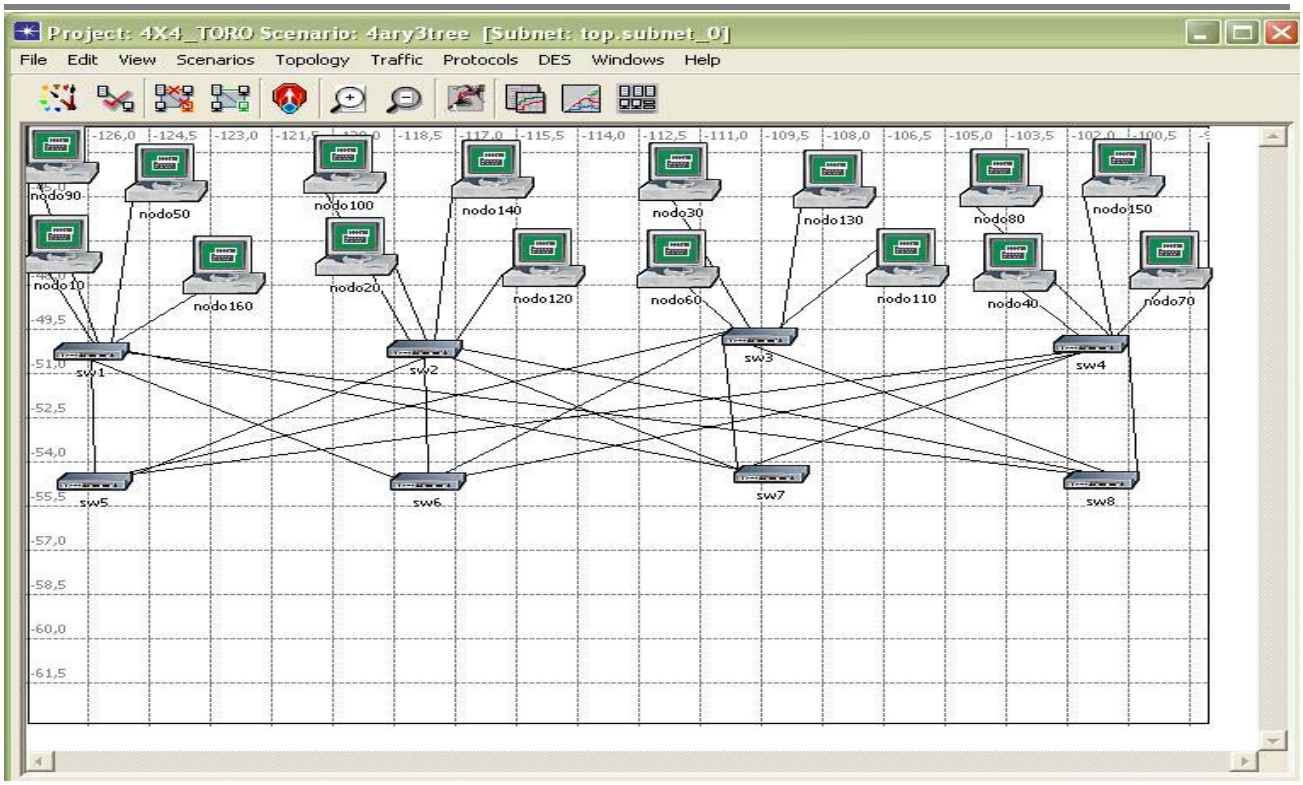


Figura 6.2. 4-ary 2-tree en Opnet modeler

En el editor de red de Opnet se definen unos atributos que regulara el funcionamiento de los nodos, los switch y los enlaces de la red. Existen dos clases de atributos que se definen en el Network editor, los "atributos de elemento" para el nodo y el switch, y los "atributos globales".

- **Atributos del elemento del nodo**.

Nombre	**Nombre del nodo**
Modelo	**Modelo de nodo definido (comportamiento)**
GUID	**Identificador global**
SM priority	**Prioridad para la selección del SM maestro**
LMC	**Cantidad de LIDS asignados al nodo**
VLAT entries	**Tamaño de la tabla de arbitraje de canales virtuales**
Buffer size (bits)	**Tamaño del buffer del 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, esta activo en el 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 paquete)**
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**

- **Atributos de elemento del switch**

Nombre	**Nombre del switch**
Modelo	**Modelo de switch definido (comportamiento)**
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)	**Tamaño de los buffers en el switch**
Tiempo de activación	**Tiempo de simulación en que se activa el nodo**
Canales virtuales disponibles	**Determina el número de canales virtuales para el switch.**
Host SM	**Especifica que el SM, esta activo en el switch**
Tiempo de Respuesta	**Tiempo entre que el switch se activa hasta que queda operativo**
Puertos físicos	**Cantidad de puertos implementados**
MTU	**Unidad de transferencia máxima en bits (tamaño de paquete)**
Tamaño de la tabla de encaminamiento	**Especifica el tamaño de las tablas de encaminamiento**

- **Atributos globales.**
  - Velocidad de transmisión del enlace
  - Patrón de tráfico
  - Cantidad la cantidad de paquetes a recibir
  - Tipo de encaminamiento
  - Habilitar/deshabilitar el CCA

6.2. MODELADO DEL NODO DE CÓMPUTO

En el nodo procesador se incluyen los módulos creados en el Node editor de opnet modeler, que actuaran como fuente y sumidero de paquetes de la aplicación ejecutada en el nodo. En la Figura 6.3 se muestra el diseño de los módulos y las conexiones lógicas para modelar el nodo de la red.

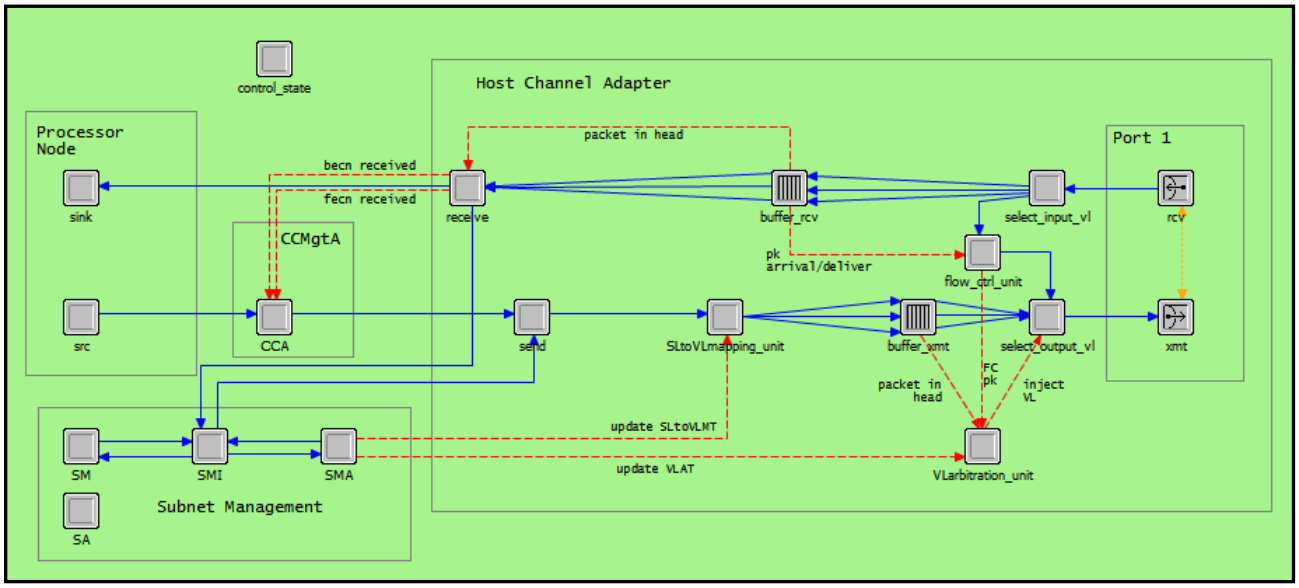


Figura 6.3. Modelo del Nodo

Los módulos más importantes en el modelo del nodo son los siguientes:

- **VL arbitration unit.** En este módulo se almacena la tabla de arbitraje para el acceso a los canales virtuales,

- **SLtoVL mapping unit.** La unidad de conversión de nivel de servicio a canal virtual.

- **Flow control unit**. Unidad de control de flujo

- **Buffer_rcv/Buffer_xmt.** Representan el puerto o la unidad receptora/transmisora, dentro se encuentran los buffers de recepción y transmisión. El flujo de información se divide en tres cadenas (líneas azules) de paquetes, estas cadenas modelan los canales virtuales del enlace, que se pueden dividir en subcolas independientes y cada una representara un canal virtual.

- **SM.** Módulo correspondiente al gestor de subred, Subnet Management, definido para administrar e funcionamiento de la subred.

- **SMI.** Se encarga de procesar los paquetes de gestión (SMPs) y toma la decisión de si debe enviarlos al SM en el caso de que sea el maestro o bien el de enviarlo al agente del Subset Management (SMA).

- **SMA.** El Agente de gestión de subred.

- **CCMgtA.** El módulo cumple con las funciones del agente de control de congestión, es un módulo donde llegan y se procesan los paquetes cuyos bits de notificación FECN han sido marcados en algún switch debido a la presencia de congestión. Al igual que cumple con la función de generar los paquetes de notificación (CN), realizar el cálculo y la selección entre distintas trayectorias posibles para el envío del paquete. Este agente accede a la cabecera de los paquetes en la que se encuentra el destino y modifica este campo de acuerdo a la mascara de control (LMC). De igual forma realiza la monitorización del contador que permite contraer trayectorias.

Además de las cadenas o *streams*, se puede usar los cables de estadísticas o *statistic wires* (líneas rojas) para comunicar los módulos entre si. En la figura 6.3 se muestra el cable ***packet in head***, su función es la de comunicar al módulo receptor la llegada de un paquete, e igualmente de informarle a la unidad de arbitraje que existe un paquete en el buffer para acceder al puerto de salida. El ***pk arrival/deliver*** es usado para que el buffer de recepción notifique a la unidad de control de flujo que no hay mas sitio para almacenar paquetes.

El funcionamiento de cada uno de los módulos del modelo del nodo se lleva acabo en base a máquinas de estado finita (FSM). Ahora analizaremos estas FSM que existen dentro de los módulos de este modelo, creadas en el Process Editor de Opnet Modeler:

6.2.1. Estados del módulo fuente (src).

En la Figura 6.4. se ilustran los estados y transacciones diseñados dentro del módulo *src* para modelar la inyección de paquetes de un nodo.

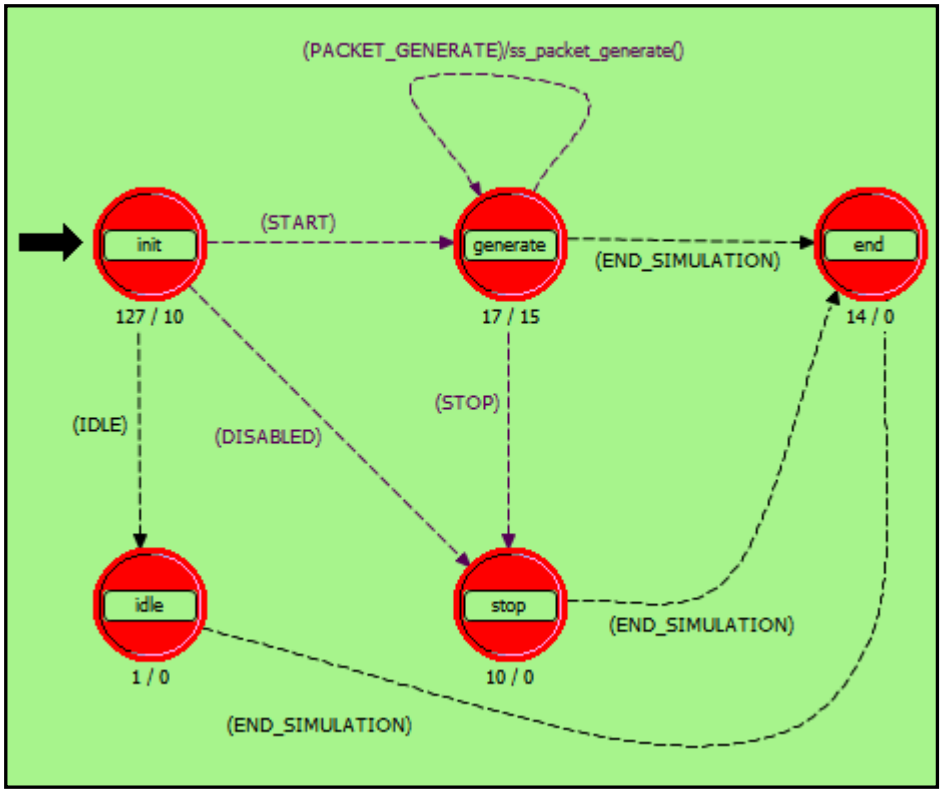


Figura 6.4. Estados y transacciones del Módulo fuente

- **Init.** En este estado se toman los parámetros de entrada como la velocidad de inyección, el tamaño de paquetes, el tiempo de activación del nodo, el tiempo de comienzo de la inyección, se configuran e inicializan las estadísticas a recolectar.

- **Idle.** Estado de proceso inactivo y por tanto no genera paquetes durante o al finalizar la simulación.

- **Stop.** Un estado que se alcanza al finalizar la simulación, de igual manera se para la inyección.

- **Generate.** Se crean los paquetes y se les configura a éstos los campos de LID destino, canal virtual, nivel de servicio, tamaño; todo esto para planificar el envío y su tiempo inter-envío para el próximo mensaje.

- **End.** Se recolectan estadísticas de la simulación y se liberan los recursos utilizados.

6.2.2. Estado del módulo SLtoVL mapping unit.

En la Figura 6.5. Observamos como esta conformada la máquina de estado finita de este módulo.

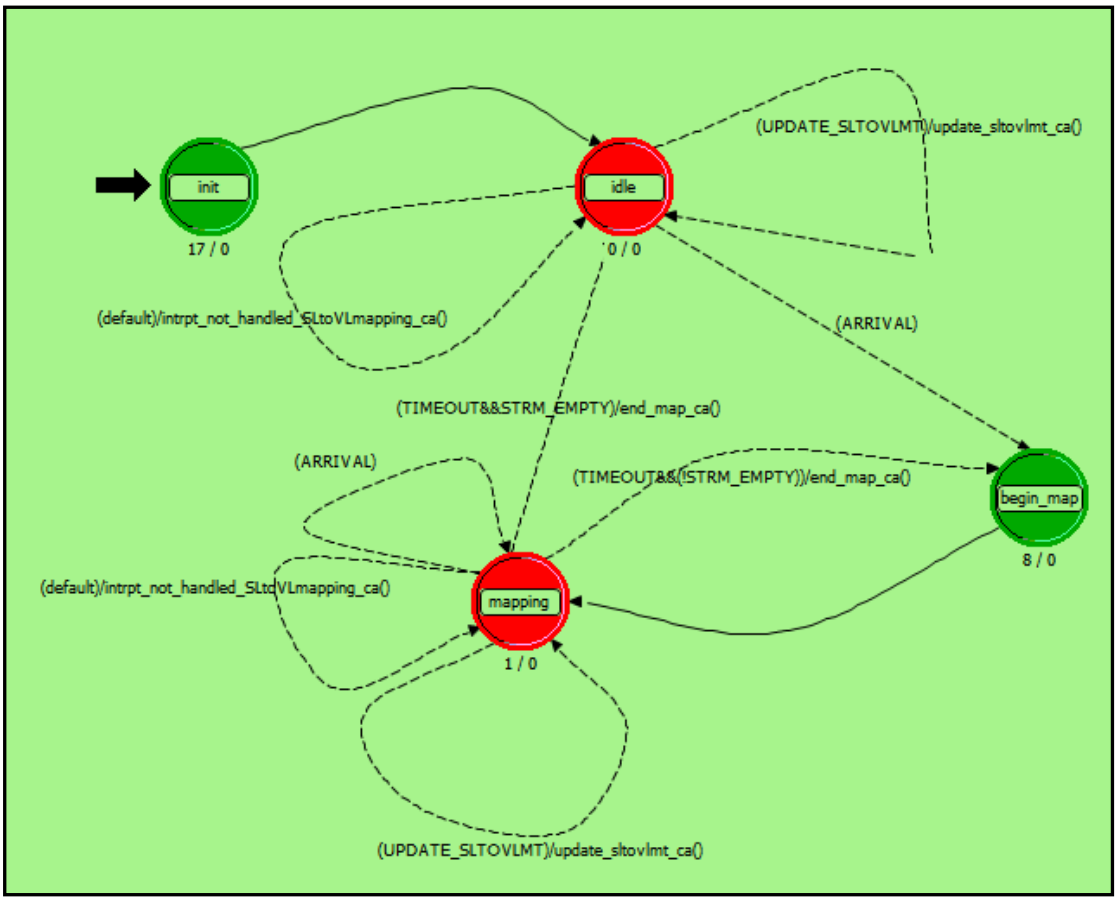


Figura 6.5. Estados y transacciones de la unidad de conversión de nivel de servicio

Esta máquina cumple con dos funciones principales:

- Crear, completar y mantener actualizada la tabla de conversión que otorga el nivel de servicio de la red de interconexión.

- Obtener el paquete del *packet stream* y el nivel de servicio en el campo SL del paquete. Al momento que entran en este módulo, simula un retardo en la búsqueda del canal virtual (VL) que tomara el paquete. Para todo esto existen dos transacciones importantes, *ARRIVAL* que es cuando llega el paquete y se busca en la tabla y se pasa a la transacción *TIMEOUT* donde se finaliza la búsqueda y se toma el canal virtual de acuerdo al nivel de servicio.

6.2.3. Estados del módulo VL arbitration unit.

La Figura 6.6 encontramos el interior de este módulo de unidad de arbitraje. En este módulo se implementa el arbitraje de acceso a los enlaces. Es decir, la unidad de arbitraje permite seleccionar que paquete será enviado hacia un puerto de salida teniendo en cuenta su prioridad. La transacción *VL_REQUEST* entra en acción cuando un paquete se encuentra en el buffer de transmisión y solicita acceso al enlace. Esta transacción inicia el estado *pk_ready* donde se analizan los campos del paquete y se obtienen las características

(tipo, canal virtual, tamaño) de este paquete para planificar su arbitraje. Si el paquete es de notificación de congestión, el estado *FC_action* es quien se encarga de obtener las características de estos paquetes, ya que son diferentes a los paquetes de datos y de gestión. El evento de arbitraje inicia la transacción *ARBIT* para pasar al estado *arbitrating* donde se lee el tipo de paquete y se simula un tiempo de retardo de arbitraje de un paquete (*TIMEOUT)*. Una vez finalizado este tiempo de arbitraje del paquete los estados *arbitrated* y *pk_delivered.*

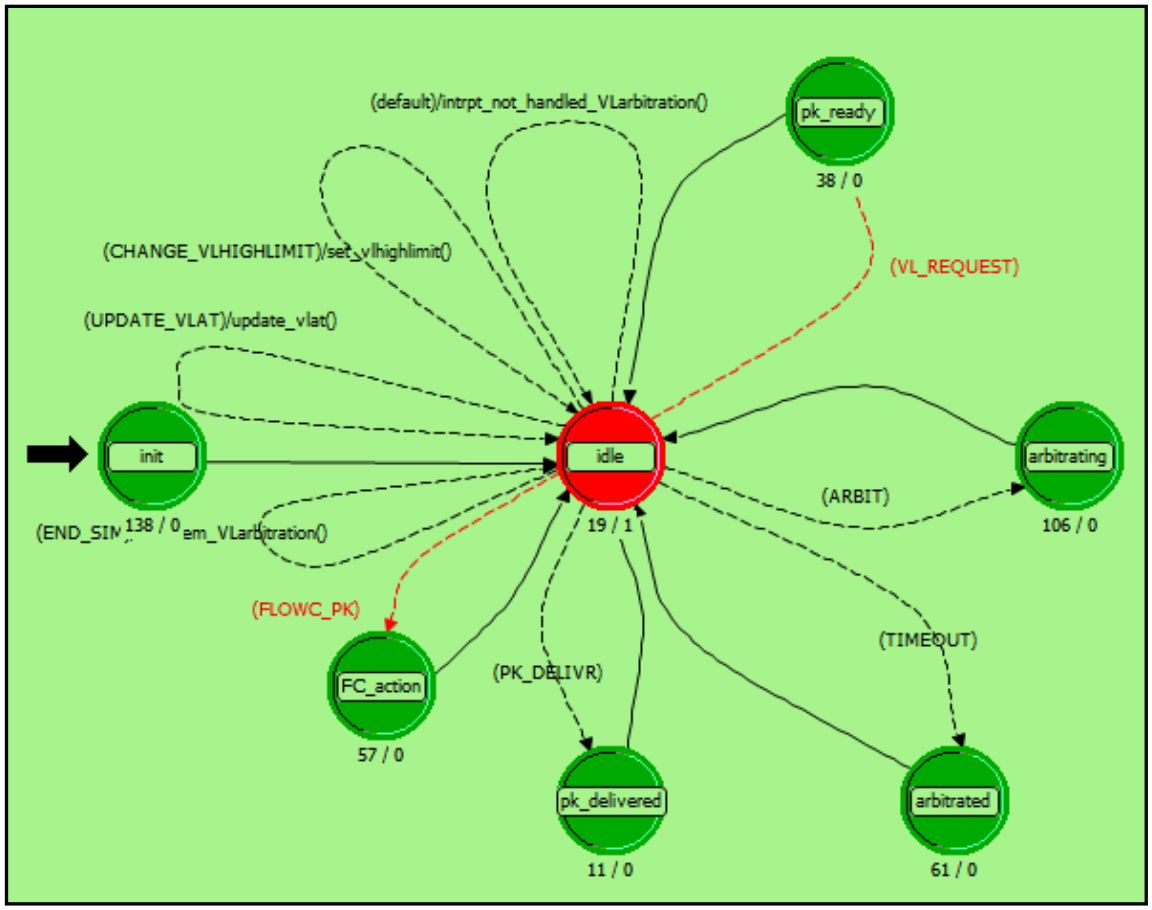


Figura 6.6. Estados y transacciones de la unidad de arbitraje

6.2.4. Estados del módulo SM.

En este módulo donde se encuentra todo el mecanismo de gestión de subred. Todas las transacciones en este módulo se realizan por medio de paquetes de gestión (SMPs), en caso que de designar el LID de un determinado puerto, el SM envía un paquete hacia el agente de gestión (SMA).

Existen tres procesos dentro de este módulo, un proceso padre y dos procesos hijos. En el proceso padre o *dispatcher*, se determina si el SM está o no residente en el nodo. En caso negativo el proceso queda inactivo en el estado *no_sm* hasta el final de la simulación. En caso de que si esté el SM en el nodo se activa el estado *sm_active* que comienza la recepción de paquetes de gestión, pasando al estado *begin_proc* que invoca dos procesos

hijos, las transacciones *default* son la invocación de estos dos procesos hijos. La Figura 6.7 observamos la máquina de estado finita del proceso *dispatcher.*

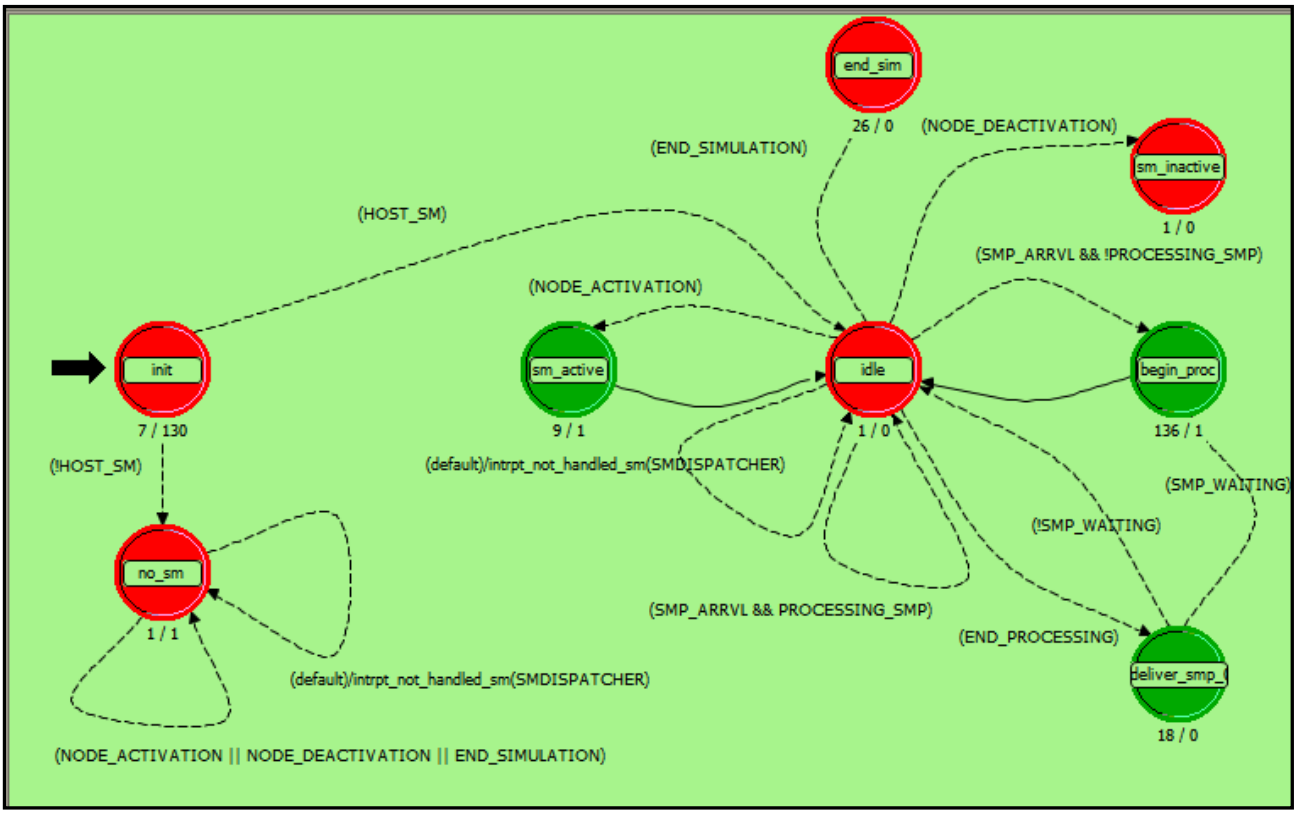


Figura 6.7. Estados y transacciones del proceso dispatcher

El proceso hijo *discover,* Figura 6.8, permite recavar información de los componentes de la red y conformar la topología. Este proceso inicia una transacción *NEXO_SWEEPING* hacia el estado *init_sweep* para iniciar la recolección de información antes mencionada. Este proceso ayuda al SM a monitorizar cambios en la topología o componentes de la red.

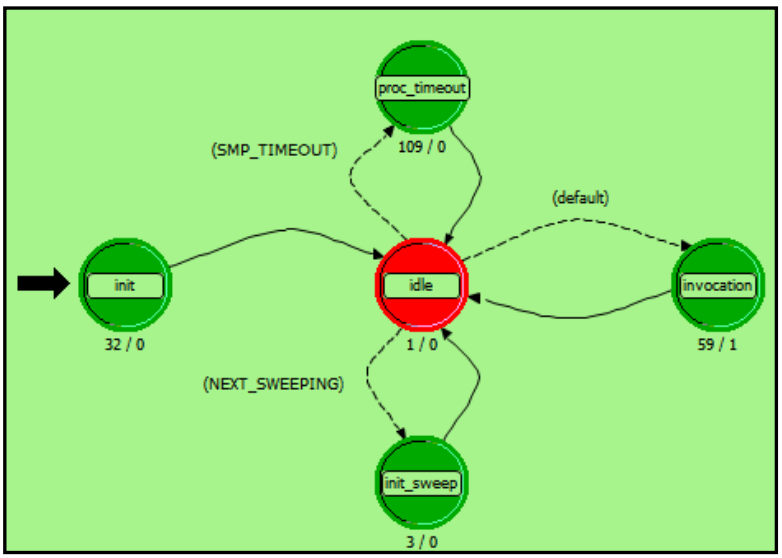


Figura 6.8. Estados y transacciones del proceso discover

El otro proceso hijo *builder,* Figura 6.9, construye las trayectorias a través de la información obtenida de la red, que permite configurar el camino múltiple para todos los pares de nodos fuente-destino, todo esto por un algoritmo dentro del estado *build. P*osteriormente el estado *distrib* se encarga de enviar los paquetes necesarios para distribuir esta información a las tablas de encaminamiento de los switch de la red.

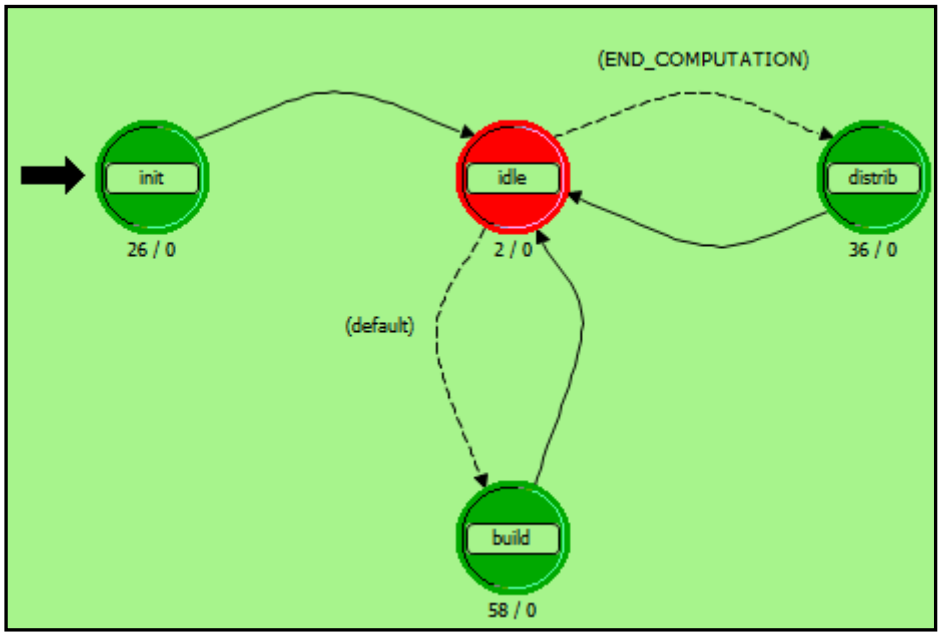


Figura 6.9. Estados y transacciones del builder

6.2.5. Estados del módulo CCMgtA.

En esta máquina se detalla el comportamiento del agente de control de congestión, en este caso se implementa la filosofía de DRB, creando cuatro estados los cuales se encargaran de:

- Analizar los paquetes marcados con un nivel de congestión en un switch recorrido
- Generar los paquetes de notificación de congestión hacia los nodos fuentes
- Configurar y seleccionar las trayectorias alternativas
- Contraer las trayectorias cuando sea necesario.

La transacción *STARt_CCA* empieza el funcionamiento del módulo. Cuando los paquetes con el bit FECN llegan al adaptador de canal una transacción pasa al estado *Process FECN* el cual analiza el paquete recibido y genera un paquete de notificación (CN) donde se guarda el bit de BECN y se invierten los campos fuente y destino, para que de esta forma se notifique al nodo origen de congestión en la trayectoria de ese mensaje.

Cuando un CN llega al adaptador de canal de un nodo en particular, se realiza la transacción *BECN_ARRVL* para pasar al estado *Process Becn,* donde se analizan estos paquetes, y se determina los niveles de ocupación en los caminos alternativos, estos niveles determinaran el encaminamiento de los paquetes. Cada vez que un nodo fuente

requiere el envío de un paquete, éste genera la transición *SRC_ARRVL* e iniciar el estado *Pk_to_send,* donde se modifican los campos del paquete según la selección de caminos realizados en el estado, *Process_becn.* La transacción *TIMEOUT* conduce al estado *contract*, el cual almacena el tiempo de expiración del contador de performance, y con la cantidad de paquetes de notificación recibidos, se decide la construcción de caminos, en función de estas dos cantidades (tiempo de expiración y cantidad de CN). La Figura 6.10 se muestra el proceso del agente de control de cogestión.

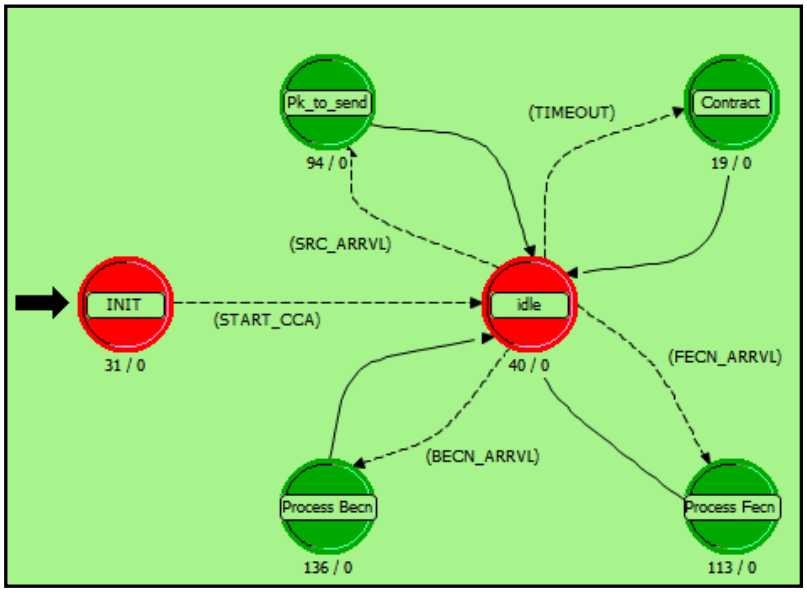


Figura 6.10. Estados y transacciones del agente de control de congestión

6.3. MODELO DEL CONMUTADOR

De igual forma los conmutadores o switches son modelados con las especificaciones de la arquitectura Infiniband. El modelado incluye algunos módulos utilizados en el modelo del nodo. En la figura 6.11 se presenta el modelo del switch.

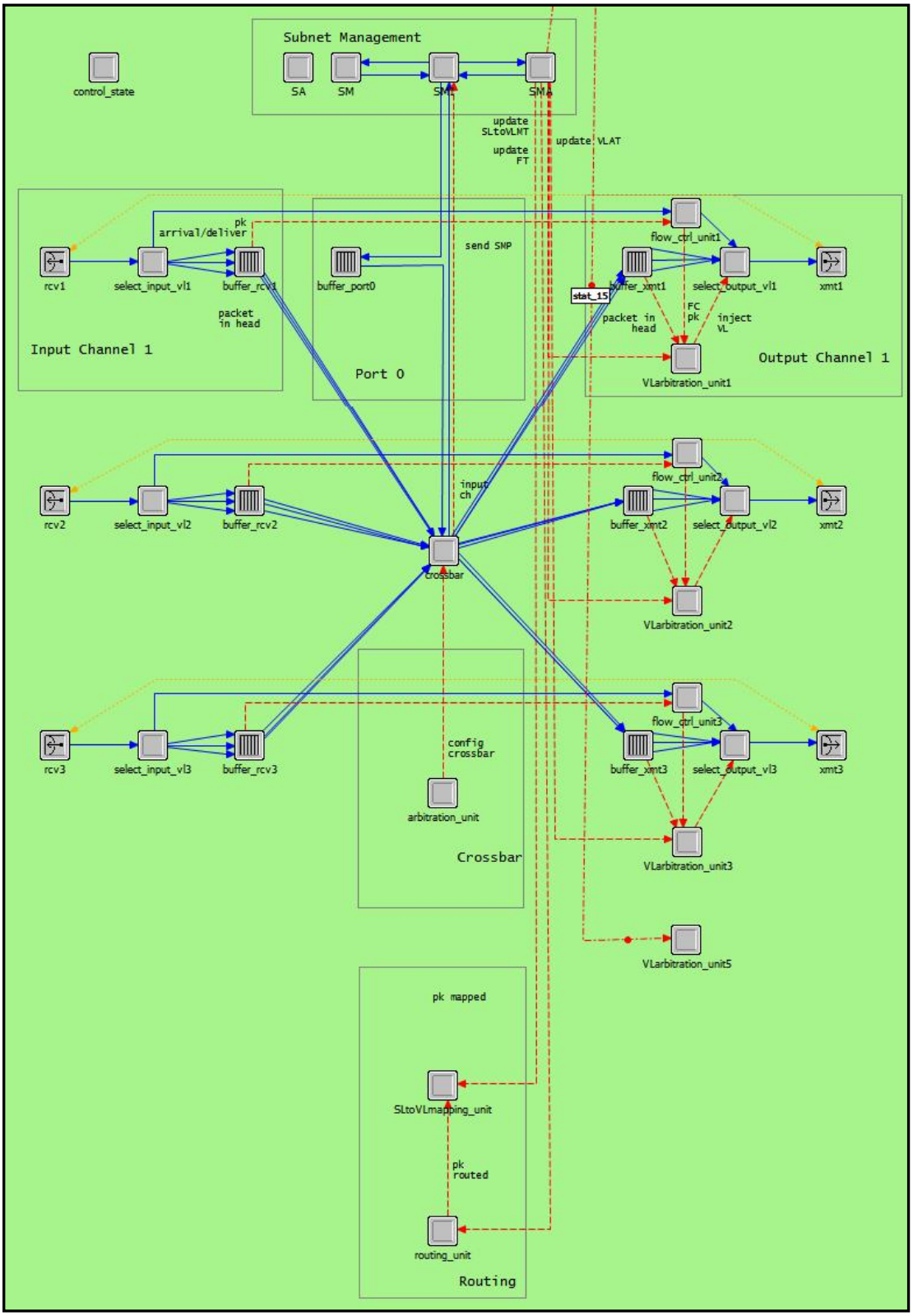


Figura 6.11. Modelo del Switch

El modelo del switch describe un conmutador de ocho a dieciséis puertos bidireccionales, con tres canales virtuales y con sus respectivas unidades para el encaminamiento. Se incluyen dos módulos o unidades que son características de los conmutadores Infiniband, la unidad de conmutación o crossbar y la unidad lógica de

encaminamiento o routing. A continuación serán explicadas las máquinas de estado finito de estos dos módulos nuevos.

6.3.1. Estados del módulo Crossbar

Este proceso entra en acción, cuando la unidad de arbitraje detecta un paquete en uno de los buffers de entrada del switch. La transacción *CONFIG_CROSSBAR* hace activar el estado *config*, donde se le asigna un puerto salida al paquete y con el atributo *crossing_time* se simula el tiempo en que cruza el paquete por esta unidad de Crossbar.

Cuando finaliza el tiempo del *crossing_time*, la transacción *CROSS_PACKET* inicia el estado *cross*, el cual se encarga de que el paquete avance desde el puerto de entrada hacia el puerto de salida. En la Figura 6.12 el proceso de crossbar.

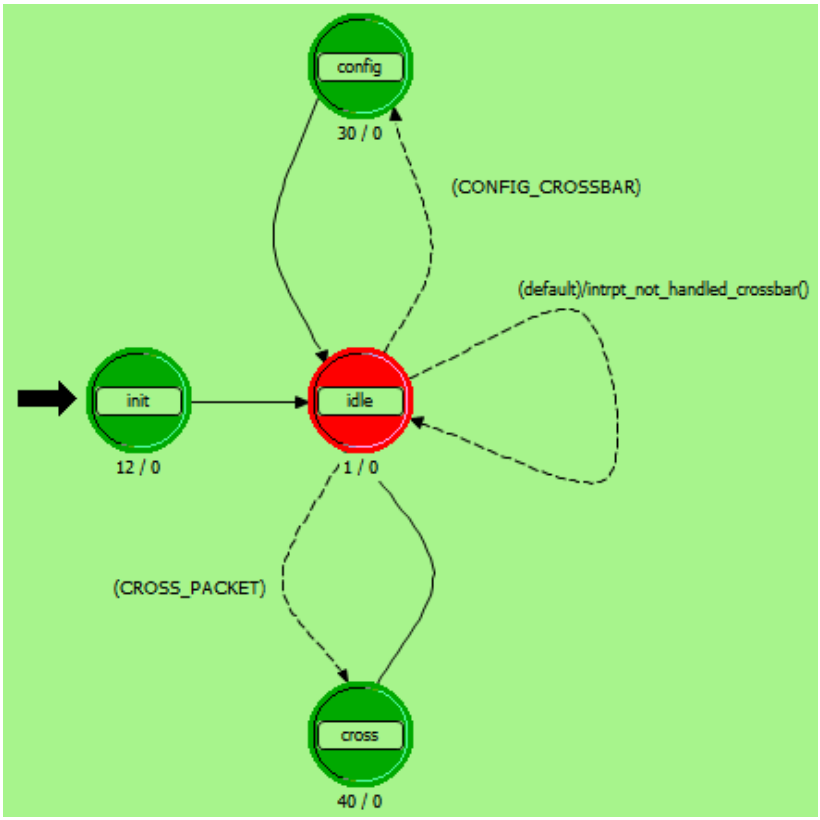


Figura 6.12. Estados y transacciones de la unidad Crossbar

6.3.2. Estados del módulo Routing unit.

Este módulo de unidad de encaminamiento, Figura 6.13, entra en acción cuando un paquete llega a un buffer de entrada, el estado *rote_pk*, identifica el canal por el que entró el paquete, lee los campos de destino del paquete, el nivel de servicio y el tamaño de este.

Obtenida esta información se hace una búsqueda de puerto de salida dentro de la tabla de encaminamiento del switch. Un atributo de entrada permite simular el tiempo de búsqueda en la tabla, cuando este tiempo finaliza la transacción *TIMEOUT* encamina el paquete hacia el puerto de salida asignado e informa de esto a la tabla de conversión de nivel de servicio (*SltoVL mapping table*).

Para tener una red de interconexión de altas prestaciones ningún paquete debe ser descartado. Pero se consideran algunos casos especiales donde sí es necesario tomar esta decisión de descartar paquetes, las excepciones son las siguientes:

- No se encuentra el LID destino en la tabla de encaminamiento
- Cuando la tabla de encaminamiento no puede soportar o no es válido un puerto de salida en especial.
- De igual forma si el puerto de salida es igual al de entrada.
- 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.

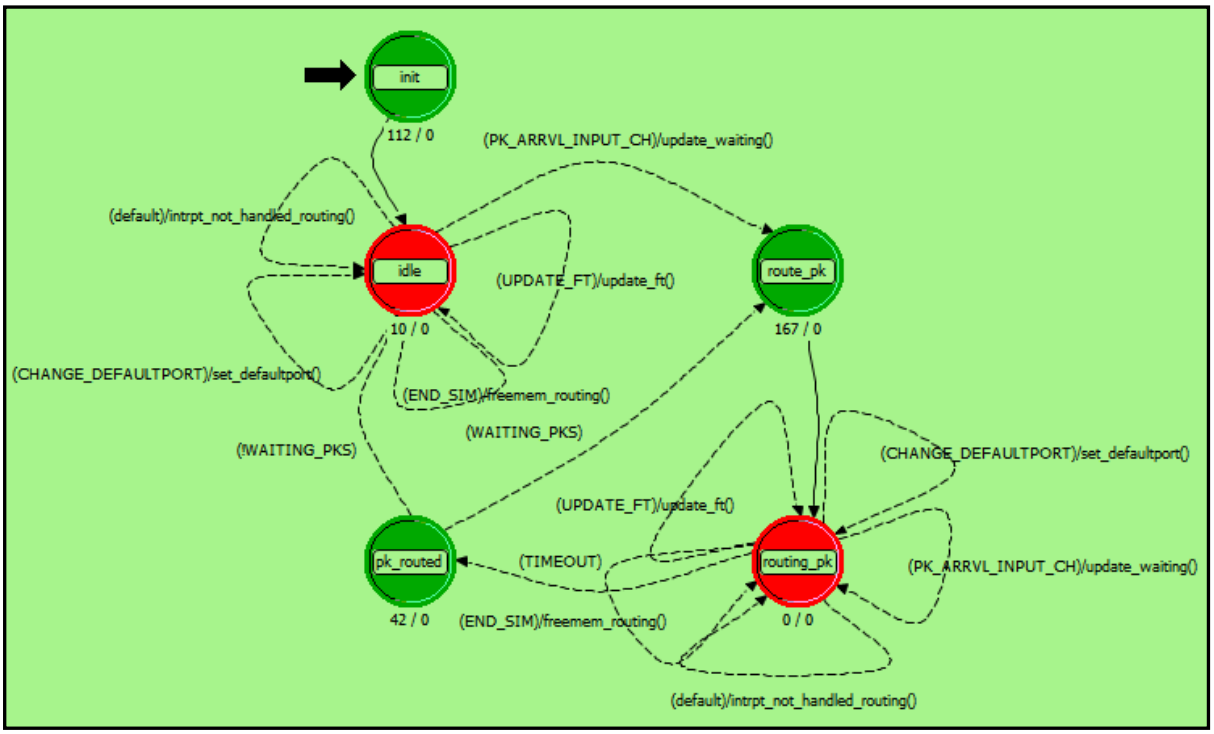


Figura 6.13. Estados y transacciones de la unidad de Encaminamiento

6.3.3. Estados del módulo Buffer_rcv y del módulo Buffer_xmt

Estos buffers de entrada / salida del switch tienen una diferencia con los del nodo, la diferencia es que en los buffers del switch se implementa un mecanismo de detección de

congestión, o bien a lo que se ha nombrado anteriormente como umbral. En los buffers del nodo no es necesarios implementar este mecanismo. El número de buffers creados en el nodo y en el switch es de acuerdo al numeró de canales virtuales diseñados. La Figura 6.14 los estados y las transacciones de los procesos de los buffers.

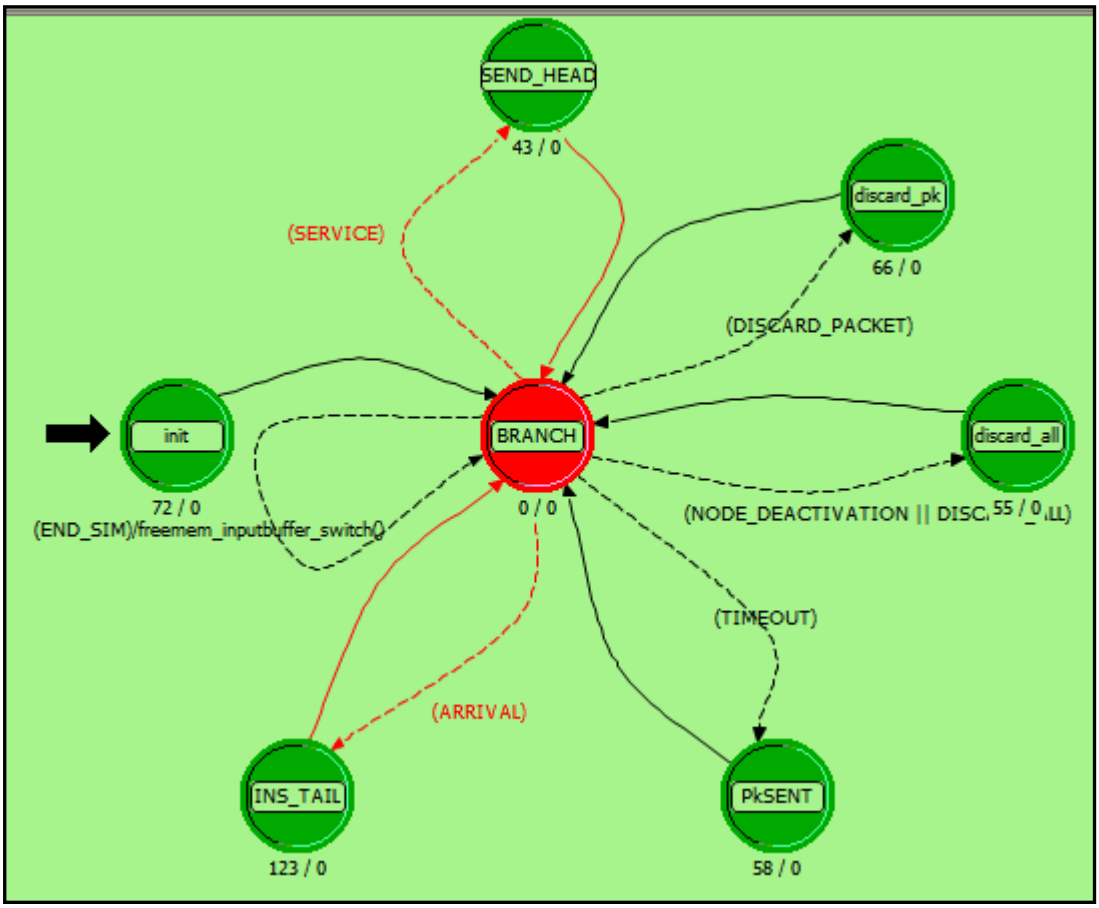


Figura 6.14. Estados y transacciones de los Buffers de recepción/transmisión

En el estado *ins_tail*, si llega un paquete al buffer de un canal virtual, se analiza el tamaño del paquete y se coloca en la cola o ultima posición del buffer, cuando el paquete llega hasta la primera posición de la cola se acciona la transacción *SERVICE* para que comience el estado *send_head*, donde los paquetes son trasladados al puerto de salida asignado. Para simular un tiempo de servicio de la cola del buffer se simula un tiempo de retardo. En este mismo estado *send_head* se monitorea las colas de los buffers para marcar la cabecera de los paquetes con el bit de FECN en caso de que sea necesario notificar una congestión.

Hasta ahora se ha explicado de manera general cada uno de los modelos diseñados en Opnet modeler en cada una de sus jerarquías y editores; se han modelado los nodos con su respectivo adaptador de canal Infiniband, los switches Infiniband y la interconexión de estos dispositivos para modelar un topología fat-tree. Estos modelos fueron diseñados con el objetivo de poder desarrollar la simulación de una red de interconexión bajo ciertos parámetros de entrada, para posteriormente experimentar y analizar el

rendimiento de DRB en una red de interconexión Infiniband con una topología Fat-tree, de DRB.

En el siguiente capítulo, describiremos las pruebas y experimentos desarrollados en Opnet modeler, se detalla cada uno de los parámetros de las simulaciones, como patrones de tráfico, tamaño de red, niveles de carga, tamañote paquetes, etc. En el mismo capítulo presentamos el análisis de los resultados de la simulación.

CAPÍTULO 7.

EXPERIMENTACIÓN Y ANÁLISIS DE RENDIMIENTO

En este capítulo se detalla la serie de pruebas desarrolladas en el simulador Opnet modeler, usando los modelos descritos en el capítulo anterior. Todas estas simulaciones son parte del objetivo de este trabajo, ya que con esta experimentación podremos analizar el desempeño de DRB en una topología fat-tree sobre una red Infiniband. Se presentaran mas adelante en este capítulo los resultados más significativos de toda esta investigación y finalmente se hace un análisis de dichos resultados.

Estas simulaciones fueron llevadas acabo definiendo parámetros de entrada muy similares a los que son sometidas las redes de interconexión de los sistemas de altas prestaciones. Para medir las prestaciones dinámicas de DRB se debe simular la carga real de las aplicaciones paralelas, es decir el tráfico de mensajes generados por las aplicaciones. Este tráfico se obtiene de aplicaciones reales o de "benchmarks", los benchmarks son un conjunto de pruebas específicamente seleccionadas de aplicaciones reales. En esta experimentación se ha tomado en cuenta un conjunto de benchmarks porque presentan ventajas como, que son mas representativos, ya que abarcan un rango mas amplio de situaciones que un conjunto arbitrario de aplicaciones, de igual forma son parametrizables y de este modo se puede tratar de medir cuestiones especificas. Los benchmarks utilizados son patrones de comunicaciones que aparecen en los programas reales de las aplicaciones científicas y técnicas paralelas más comunes de la actualidad. Esto patrones de comunicación tienen la propiedad de utilizar la red de interconexión de manera extensiva.

La experimentación se enfoca en dos puntos principales. El primero es la respuesta en latencia con patrones de comunicación persistentes tomados de los benchmarks de aplicaciones, y el segundo, es la respuesta en throughput de mensajes inyectados en la red de interconexión.

7.1. REDES DE INTERCONEXIÓN

En esta experimentación se han utilizado un conjunto de redes de interconexión Fat-tree o bien k-ary n-tree, concretamente en diverso tamaños de árbol. Como técnica de control de flujo, se ha utilizado la técnica "Virtual Cut Throught" como especifica la arquitectura Infiniband.

7.2. CARGA DE COMUNICACIONES

El rango de carga de tráfico utilizado en las pruebas se modifico desde valores bajos hasta la saturación. El rango mas bajo son valores menores al 10% (2 Gbps) del ancho de banda máximo especificado por Infiniband y los valores máximos hasta alcanzar el 120%. Los tamaños de los paquetes, de igual forma, se variaron entre 4096, 1024, 512 y 256 bytes. La experimentación se basa en la medición y relación de dos factores de carga de tráfico, el primero la carga de tráfico ofrecido (*Offered Load),* representa la velocidad de inyección de los mensajes generados; y el segundo factor la carga aceptada (*Accepted Load),* este factor es la parte de la carga inyectada que no es absorbida por la red, pero que no es rechazada y circula por la red.

7.3. PATRONES DE TRÁFICO

Como se ha mencionado, se han seleccionado una serie de benchmarks sintéticos [12] para evaluar DRB. Estos benchmarks se componen de una serie de patrones de comunicación. Estos patrones son: *Butterfly, Perfect Shuffle.* A continuación, se incluyen las expresiones matemáticas de las transformaciones que realiza cada uno de los patrones para determinar el nodo destino a partir del nodo fuente [13].

Sea el nodo fuente formado por *n* coordenadas $\{a_{n-1}, a_{n-2},..., a_1, a_0\}$ (ej. 1100, 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},..., se comunica con el nodo destino (*Butterfly*)= $\{a_0, a_{n-2}, a0,..., a_1, a_{n-1}\}$ (ej. Origen 1100 envía al destino 0101)

- 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. Origen 1100 envía al destino 1001)

7.4. TÉCNICA DE CONTROL DE CONGESTIÓN

Las técnicas usadas en la experimentación son el balanceo distribuido del encaminamiento DRB y el mecanismo de control de congestión de Infiniband.

7.5. METODOLOGÍA DEL TRABAJO

La experimentación presenta las siguientes partes. Evaluar la respuesta transitoria de DRB, el tiempo de respuesta de DRB y la robustez de DRB frente ala información de ocupación que DRB necesita para seleccionar caminos alternativos, disjuntos y de distancia corta hacia el destino. La experimentación es exhaustiva de DRB frente a patrones y topologías (fat-tree / k-ary n-tree) diferentes y se enfoca en dos puntos principales.

- El primero es la respuesta en latencia con patrones de comunicación persistentes tomados de aplicaciones numéricas (*"Butterfly"*, *"Perfect Shuffle"*)
- El segundo, es el throughput, en cuanto se mejora el rendimiento de la red de interconexión en relación con el agente de control de congestión de Infiniband.

Para todo ello se evalúa la respuesta en latencia, el "*throughput*" de mensajes. Otros aspectos evaluados son la influencia de la longitud del mensaje y la escalabilidad de DRB respecto el aumento del tamaño de la red de interconexión. Los experimentos fueron ejecutados varias ocasiones con diferentes parámetros y promediados para ser consistentes.

7.6. RESULTADOS MEDIDOS

Como resultado de los experimentos, se han medido tres aspectos: Primero, la *latencia media* de las comunicaciones de la red de interconexión, segundo, el *throughput* medio y, tercero, la "distribución de la carga" de tráfico en la red.

La "latencia de comunicación" se mide como el tiempo total transcurrido desde que el mensaje viaja desde el nodo fuente hasta el nodo destino, incluyendo el tiempo que el mensaje espera por ser inyectado en la red. La "latencia media" se calcula promediando todas las latencias de todos los mensajes enviados y se mide en milisegundos:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

El "*throughput*" se mide como la relación porcentual entre la carga aceptada (cantidad de información entregada) y la carga de comunicación aplicada (tasa de inyección). Ambas cargas de comunicación se miden como el número de mensajes por unidad de tiempo.

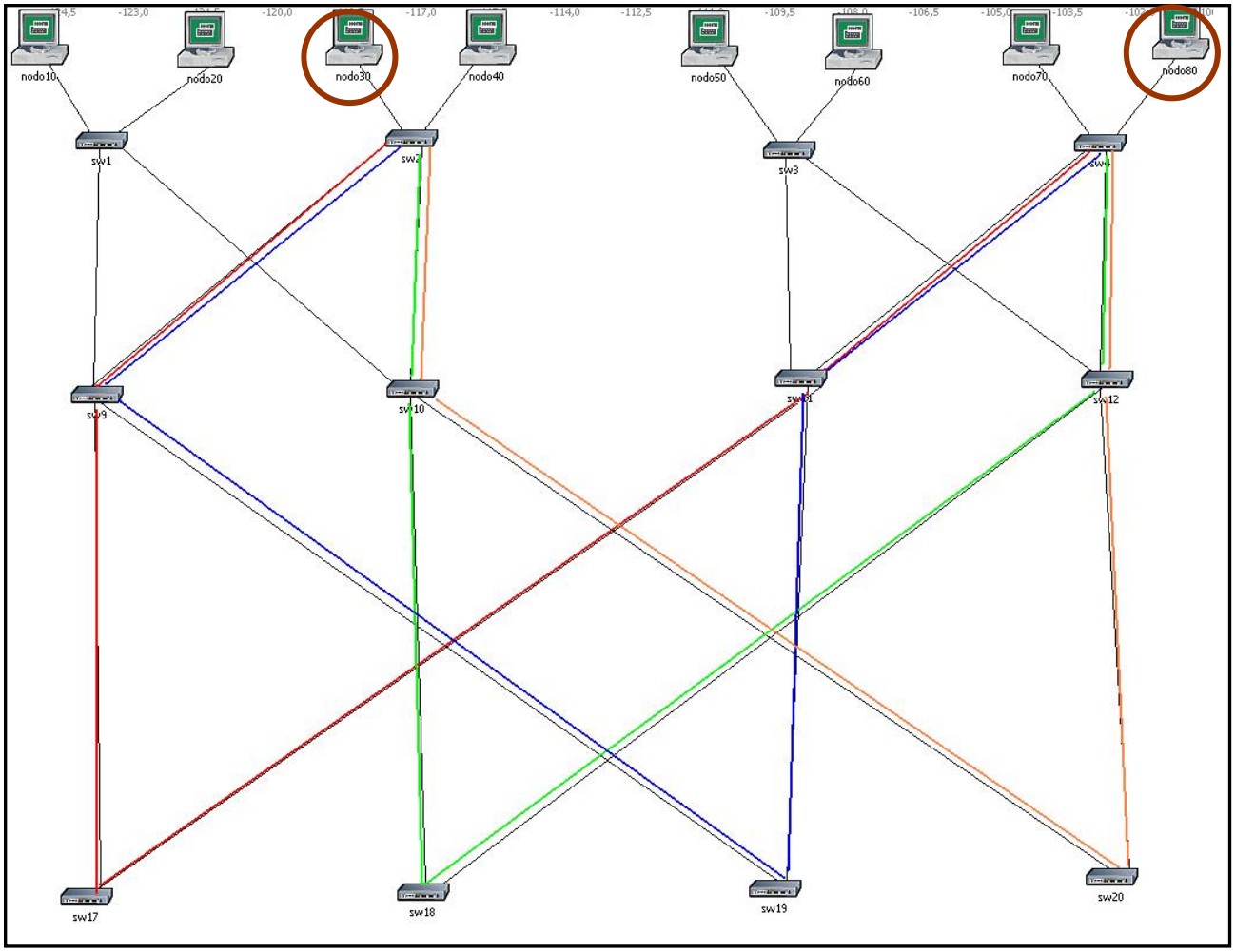
7.7. RESULTADOS

Topología	Tamaño	Patrones	Técnica de control	Figura
2-ary 3-tree	8 nodos--12sw	Butterfly	IBA_CC	Figura 7.2.
2-ary 4-tree	16 nodos--32 sw			Figura 7.4.
8-ary 2-tree	32 nodos--16 sw	Shuffle	DRB	Figura 7.6.

7.7.1. Análisis de resultados del 2-ary 3-tree

Los primeros resultados obtenidos son de una red conformada de 8 nodos, 12 switchs, con un tamaño de paquetes de 4096, con una ratio de generación de paquetes de 10000 hasta 80000 paquetes por segundo por nodo**.** Los dos patrones de comunicación que se

evaluaron son el Butterfly y el Perfect Shuffle. Las gráficas presentan semejanza en los resultados. Esta similitud se presenta en factor de que los patrones de tráfico permiten que DRB pueda disponer de más caminos alternativos y así conseguir mejor desempeño



que el mecanismo de control de congestión de Infiniband.

Figura 7.1. 2-ary 3-tree, cuatro caminos alternativos entre el nodo 30 y el nodo 80

EL 2-ary 3-tree, Figura 7.1., es una red con poca profundidad en niveles del árbol, pero a pesar de que ofrece pocos caminos alternativos, en este caso cuatro caminos alternativos entre un par fuente-destino (siempre y cuando no compartan el mismo switch de primer nivel), en la Figura 7.1 se observa los cuatro diferentes caminos alternativos entre el "nodo 10" y el nodo "110". Las gráficas demuestran que frente al incremento de la carga de tráfico de la red, los mecanismos presentan diferencias al encaminar los paquetes, controlar los niveles de ocupación de los buffers de los switchs, distribuir la utilización de los enlaces.

En el caso de DRB logra en altos niveles de carga balancear la carga de comunicaciones en los enlaces, por lo que el ancho de banda se mantenga uniforme. Estas afirmaciones se pueden observar en ambas gráficas de la Figura 7.2, donde DRB y el mecanismo de control de congestión de IB en los niveles de cargas pequeñas (0-600 bit/μs) el

comportamiento es igual, lo que implica que DRB no esta introduciendo overhead, y cuando no es necesario no cambia el procedimiento normal de la red.
Cuando la carga de mensajes se incrementa (700-1500 bit/µs), el mecanismo de control de congestión empieza a regular la inyección de mensajes entre pares fuente-destino, en consecuencia la curva de latencia de IB hace un curva hacia arriba. Por otro lado DRB se mantiene paralelo al eje X, lo que se traduce en que DRB hace uso de los caminos alternativos hacia los destinos, logrando así controlar la carga inyectada en la red de interconexión.

DRB disminuye su capacidad de controlar los niveles de latencia ante altos niveles de carga (2000 bit/µs), donde los valores de latencia empiezan a subir con mucha rapidez, pero a pesar de esto reduce la latencia de la red en más de un 100% que el mecanismo de control de congestión de IB en cargas excesivas.

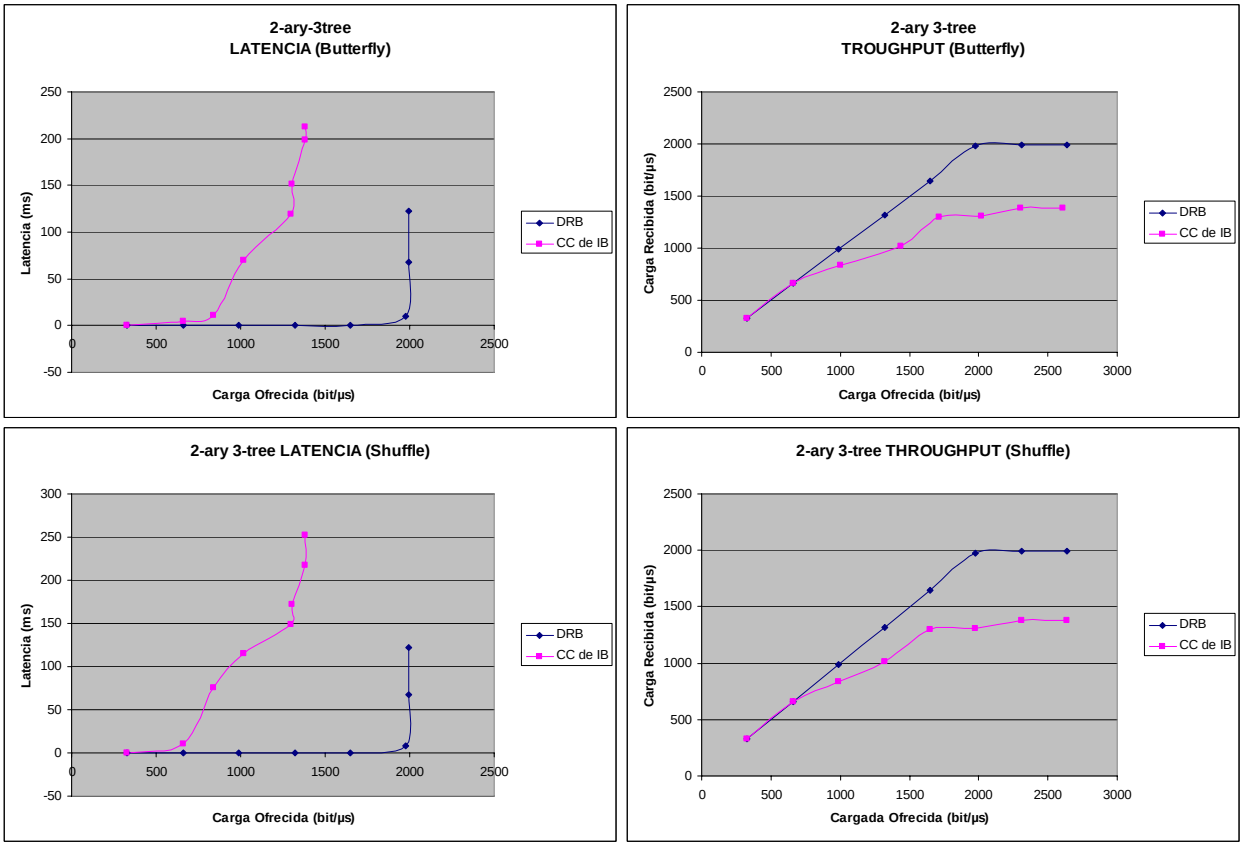


Figura 7.2. Resultados del 2-ary 3-tree

Con respecto al "throughput", se muestra que DRB hace valer los objetivos de tener una red de altas prestaciones ante grandes volúmenes de comunicaciones o grandes cargas de mensajes en la red. El mecanismo de control de congestión de Infiniband hace colapsar la red en niveles de carga donde aun DRB mantiene sin problemas de saturación y una mejor utilización de los recursos de la red. DRB tiene una ganancia de 42% en throughput frente al agente de control de congestión de IB.

7.7.2. Análisis de resultados del 2-ary 4-tree

En estos resultados, la red es un 2-ary 4-tree de 16 nodos, 32 switchs, con una profundidad mayor que la red evaluada anteriormente, en esta red encontramos ochos caminos alternativos entre un par fuente-destino (siempre y cuando no compartan el mismo switch de primer nivel). Se simularon los mismos patrones de tráfico, Butterfly y Shuffle. En la Figura 7.3 la representación de la red 2-ary 4-tree y los 8 caminos alternativos entre el nodo10 y el nodo 120.

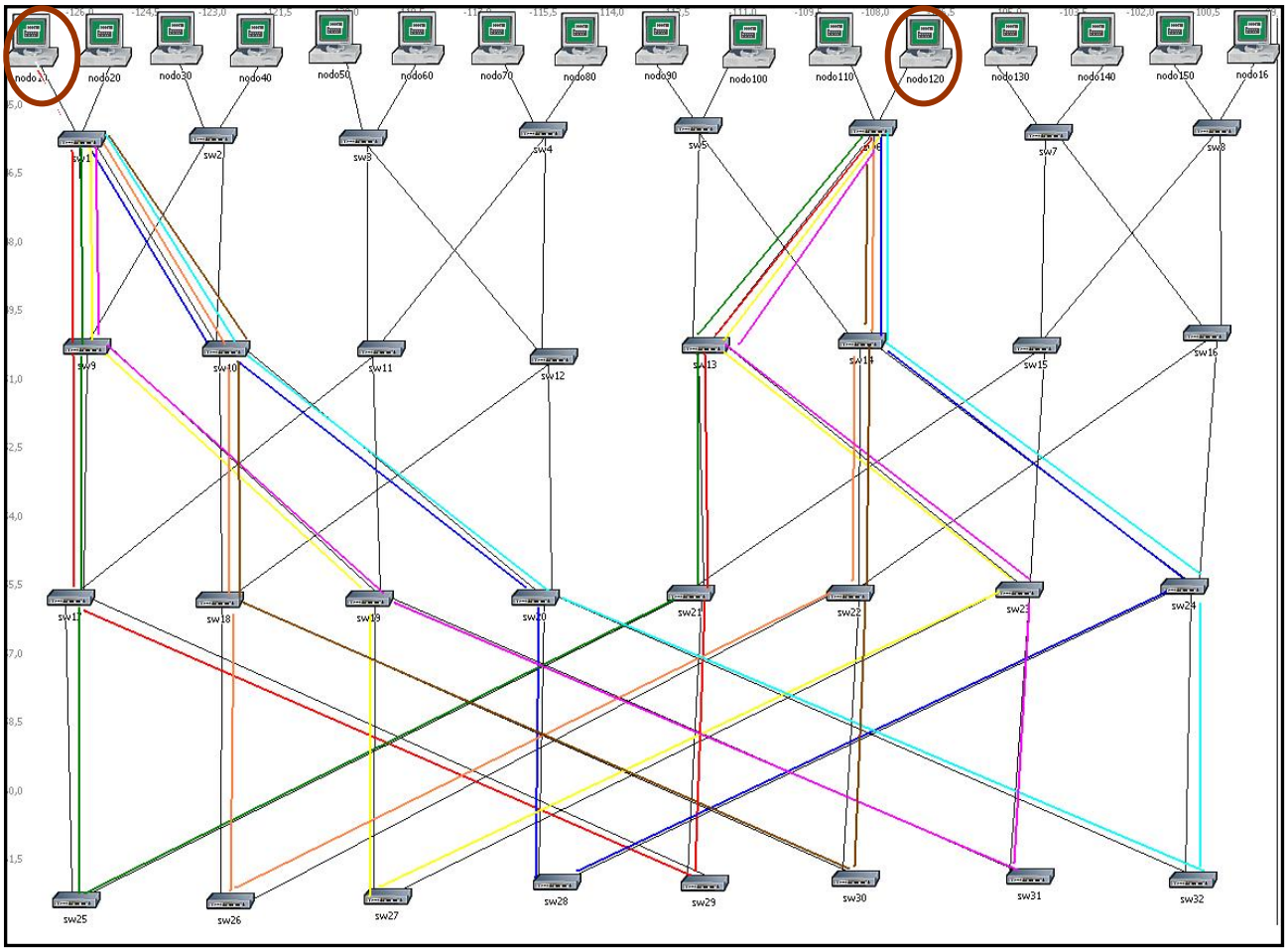


Figura 7.3. 2-ary 4-tree, ocho caminos alternativos entre el nodo 10 y el nodo 120

Los tiempos de latencia como se observa en la Figura 7.4 son muy diferentes entre DRB y el mecanismo de control de congestión de Infiniband. DRB presenta mejores resultados en este tipo de redes con mayor profundidad, ya que existe mayor cantidad de caminos alternativos (8 caminos) entre cualquier par-fuente destino.

Como se menciono anteriormente DRB logra balancear la carga de tráfico en la red al abrir nuevas trayectorias a las originales entre un par fuente-destino. Es por esto que DRB es capaz de mantenerse paralelo al eje X ante niveles de carga muy altos y en cambio el mecanismo de control de congestión de Infiniband no es capaz de gestionar las comunicaciones frente a los grandes niveles de carga. A pesar de que la red cuenta con mayor número de nodos, observamos que DRB se mantiene con un buen desempeño en la red de interconexión, con esto se cumple con unos de los objetivos, la escalabilidad.

De acuerdo a la utilización de la red de interconexión, se muestra que en la gráfica de throughput, que DRB es capaz de mantener un mejor rendimiento de la red en la relación de carga ofrecida y la carga recibida, mejora hasta un 81% la capacidad de utilización de la red en comparación con el agente de control de congestión de infiniband.

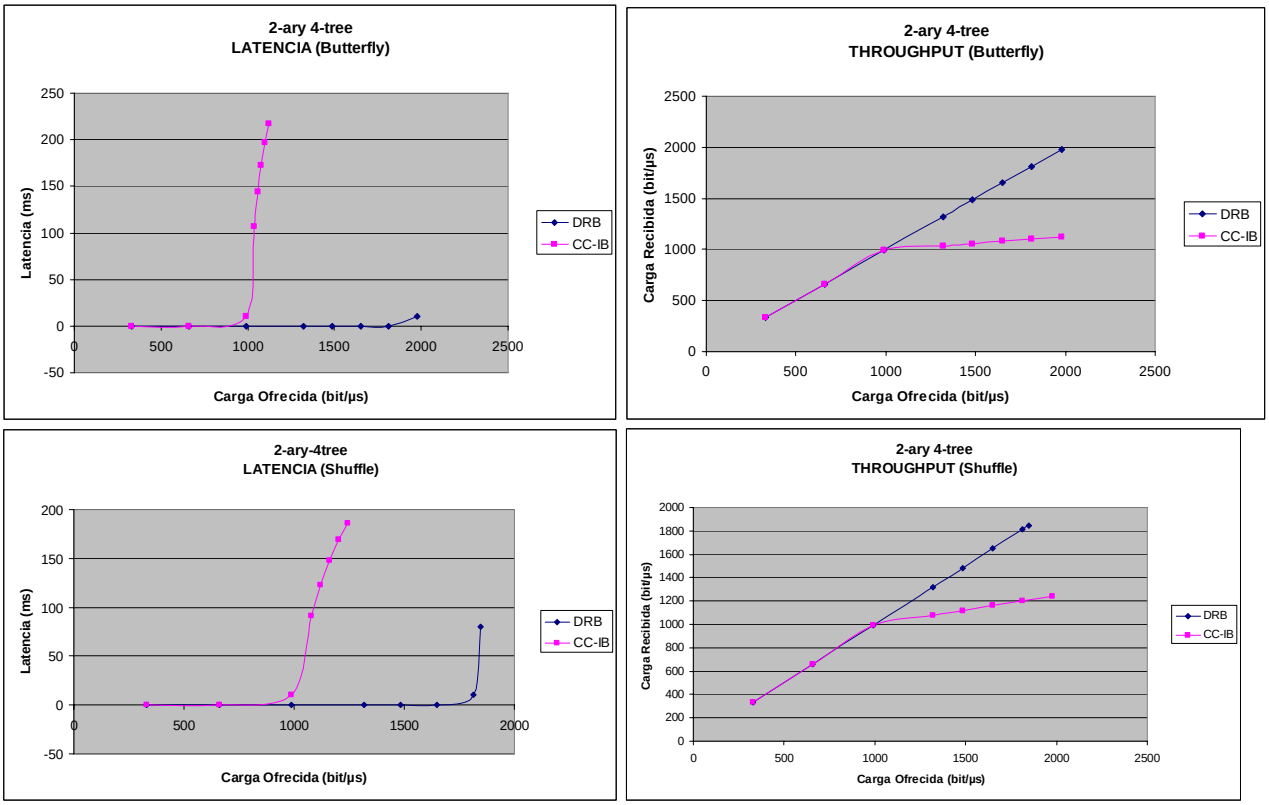


Figura 7.4. Resultados 2-ary 4-tree

7.7.3. Análisis de resultados del 8-ary 2-tree

Continuando con los resultados obtenidos en la simulación, en la red 8-ary 2-tree con 64 nodos, 16 switches, obtenemos medidas similares a los anteriores. Las curvas de latencia en la Figura 7.6 demuestran que en un red con poca profundidad, pero con ocho caminos alternativos entre cualquier par fuente-destino (Figura 7.5 la representación de los ocho caminos alternativos), permiten la escalabilidad de DRB en una red de mayor tamaño. A pesar de que DRB no se mantiene paralelo al eje X como en las gráficas anteriores, DRB tiene una ganancia constante mayor de 100% que el agente de control de congestión de IB. Frente al incremento de la carga de tráfico DRB actúa creando nuevas trayectorias, mientras que el agente de control de congestión de IB regula la inyección de paquetes para controlar la congestión; esto se refleja en las curvas de latencia de las Figuras 7.2-7.4-7.6.

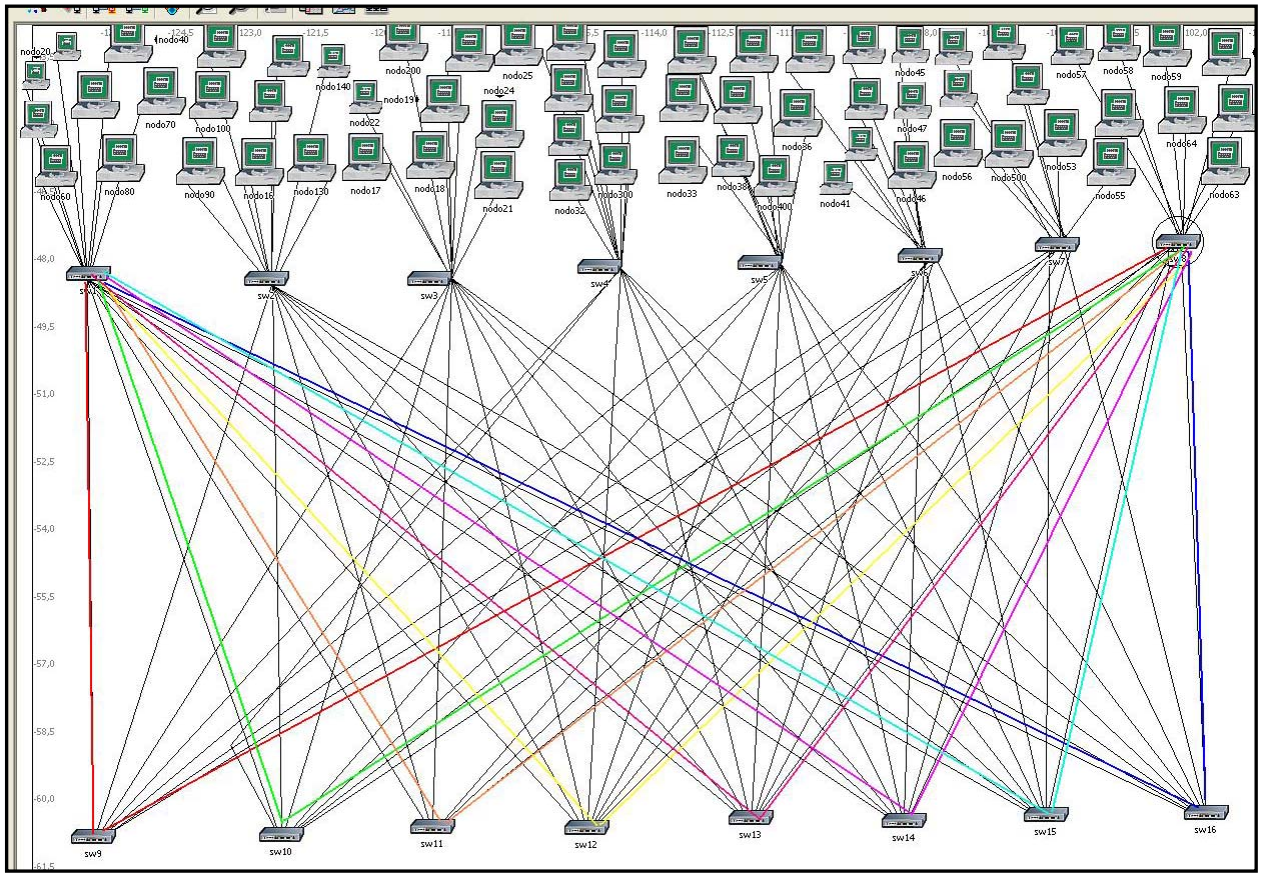


Figura 7.5. 8-ary 2-tree, los diferentes ocho caminos alternativos entre un par fuente-destino

Respecto al "Throughput" de la red de interconexión, en la Figura 7.6 observamos que la ganancia de DRB es menor que en los casos anteriores, principalmente por el patrón de tráfico Shuffle. DRB logra mejor utilización de la red a comparación con el agente de control de congestión de IB en un 38%. En este caso, IB tiene mejores niveles de carga recibida en función de la carga ofrecida en este tipo de red con poca profundidad, pero a pesar de esto DRB mantiene la red de interconexión con mejores prestaciones frente diferentes niveles de carga de tráfico.

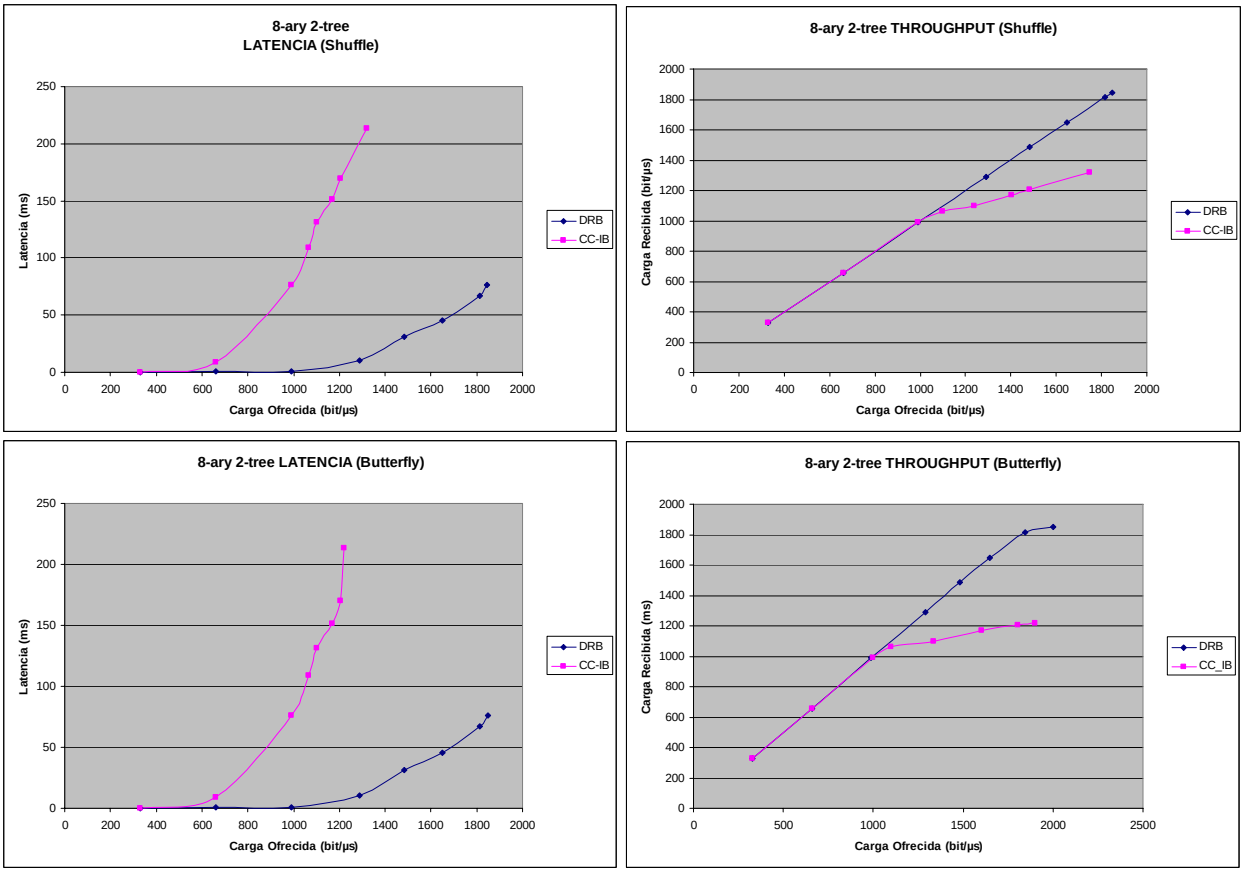


Figura 7.6. Resultados 8-ary 2-tree

A manera de resumen, se realizo una experimentación con tres tipos diferentes de redes, con diferentes patrones sintéticos de aplicaciones reales, para evaluar el desempeño y el impacto de DRB en topologías Fat-tree de una red de interconexión con tecnología Infiniband. Para ubicar a DRB frente a otras técnicas de control de congestión, se decidió comparar DRB con el mecanismo de control de congestión de Infiniband.

Los datos obtenidos de la experimentación determinan que los aspectos de diseño de DRB frente a situaciones de gran cantidad de carga de comunicaciones, son efectivos, logra balancear la carga de tráfico de la red, previniendo así que se colapse la red de interconexión. Esto lo logra abriendo nuevas trayectorias entre cualquier par fuente-destino, monitorea constantemente los niveles de ocupación de los buffers dentro de los switchs, y distribuye los paquetes de los enlaces mas utilizados a los menos utilizados, consiguiendo así el uso uniforme del ancho de banda.

Las gráficas de latencia muestran como DRB tiene mejor desempeño con diferentes niveles de carga, el encaminamiento de los mensajes de DRB se realiza en menores fracciones de tiempo a comparación con el agente de control de congestión de Infiniband. Se consigue una mejora hasta en el mejor de los casos en un 81% frente a Infiniband.

Con los resultados obtenidos, podemos deducir que DRB es escalable hasta redes de interconexión Fat-tree de 64 nodos, y poder mantener las prestaciones de la red con diferentes niveles de carga. En los resultados de throughput anteriores, DRB tiene una

ganancia entre un 38% y un 81% a comparación con el mecanismo de control de congestión de Infiniband.

Los resultados obtenidos, nos llevan a continuar por esta vía para que DRB cumpla con el objetivo de ser una solución en los problemas que presentan las redes de interconexión.

CONCLUSIONES Y LÍNEAS ABIERTAS

Las redes de interconexión son un factor importante dentro de la arquitectura de la computación de altas prestaciones. A pesar de que actualmente existen tecnologías que ofrecen un gran número de recursos y mecanismos para controlar problemas de congestión, los problemas de saturación aún continúan presentándose en estas tecnologías, degradando las prestaciones del sistema. Es por esto que nos planteamos abordar un mecanismo capaz de controlar los niveles de carga de comunicaciones a las que son sometidas las redes de interconexión actualmente.

Se realizo un estudio de las redes de interconexión para conocer su comportamiento, las características con las que son diseñadas actualmente, los elementos que la componen, y abordar los aspectos que definen una red de interconexión como son la topología, el encaminamiento, control de flujo. También se analizo las complicaciones que se presentan en las redes de interconexión, y se planteo continuar con una técnica (DRB) que mantiene las prestaciones de la red, que detecta-notifica-actúa ante situaciones de congestión.

Se realizo con éxito el análisis de la topología Fat-tree, y qué problemas de encaminamiento presentan. Se planteo la solución de los inconvenientes de encaminamiento en los Fat-tree con el algoritmo de encaminamiento adaptativo "DRB". Las modificaciones a DRB con respecto a esta topología fueron las de como abrir trayectorias alternativas de acuerdo a los niveles del árbol, como crear caminos cortos y disjuntos en un red multinivel.

El análisis de la especificación Infiniband nos permitió detectar las características de DRB que coincidían con el estándar de IBA, y que puntos de DRB modificar, pero respetando su filosofía, para adaptarlo a Infiniband.

La plataforma de simulación Opnet Modeler, fue de gran ayuda para cumplir con los objetivos y demostrar los resultados primeramente planteados.

El algoritmo estudiado durante toda la investigación necesitaba ser demostrado que cumplía con los principios de balancear grandes niveles de carga inyectados en red de interconexión fat-tree. Por esto es que realizamos la experimentación con diferentes tipos de redes como 2-ary 3-tree, 2-ary-4-tree, 8-ary 2-tree, con diferentes patrones de comunicación ("benchmarks" de aplicaciones reales) como el *"Butterfly" y "Perfect*

Shuffle", con diferentes niveles de inyección de paquetes por nodos. La comparación con el mecanismo de control de congestión de Infiniband nos daría un punto de referencia de DRB con técnicas similares en la actualidad.

Dentro de la experimentación obtuvimos unos resultados muy favorables, donde mostramos con las curvas de latencia que DRB se mantiene paralelo al eje X desde niveles de carga pequeños hasta niveles de saturación; en los resultados se observa que el mecanismo de control de congestión de Infiniband no es capaz de mantener medidas de latencia similares a DRB, la curva de latencia de Infiniband se prolonga hacia arriba con niveles de carga considerablemente medianos. A resumen DRB consigue una mejora en los tiempos de latencia en más de un 100% con respecto a Infiniband.

La utilización de la red de interconexión es mejor por parte de DRB, como se comento anteriormente, DRB presenta mas carga aceptada en función con la carga inyectada en la red, por otro lado, el mecanismo de control de congestión es incapaz de controlar la saturación de la red de interconexión, y por consecuencia la red se colapsa. DRB con respecto a utilización de la red de interconexión mejora hasta un máximo de 70% las prestaciones de la red a comparación con el mecanismo de control de congestión de Infiniband.

En el principio del trabajo de investigación se plantearon unos objetivos, hasta este punto consideramos que se han cumplido cada uno. Los resultados preliminares obtenidos, nos incentivan a trabajar para que DRB sea un algoritmo de encaminamiento adaptivo que puede seguir extendiéndose y abarcando diferentes tipos de redes de interconexión de los sistemas actuales, y seguir evolucionando en sus funcionalidades de algoritmo de encaminamiento.

LÍNEAS ABIERTAS

Durante el desarrollo de este proyecto, se fueron precisando algunos aspectos que pueden ser desarrollados a futuro, Entre ellas podemos citar:

- Experimentar con un mayor número de nodos en la topología Fat-tree, utilizar patrones de comunicación de aplicaciones reales, variar el tamaño de los paquetes. Buscando con esto comprobar la escalabilidad de DRB,

- Comparar DRB con otras técnicas actuales que tengan impacto sobre la topología Fat-tree, unas opciones de comparación serían las técnicas mencionadas en el estado del arte del capítulo 2.

- Evaluar el desempeño de DRB frente a otros mecanismos de control de congestión que los coloquen como los mejores y que respeten los estándares de la tecnología Infiniband.

- Estudiar las posibilidades que existen de acoplar DRB a otras tecnologías de redes de interconexión, siempre y cuando se siga respetando los estándares de estas tecnologías y la ideología de DRB (Monitorización-Notificación-Selección).

- Mejorar la actuación de DRB dentro del switch cuando se detecta saturación de uno de los buffers; es decir, en el momento que el primer paquete de un mensaje llega a un switch y el buffer de salida asignado presenta niveles de ocupación muy altos, DRB actuaría mandando una notificación al nodo origen de esta congestión, y en el switch este primer paquete cambiara su trayectoria original para ser inyectado por otro puerto de salida. De esta manera se elude que el paquete tenga que esperar en un buffer de salida saturado, mientras que los otros paquetes son encaminados por caminos alternativos.

- Aplicar la creación de caminos alternativos de DRB para conseguir una cierta tolerancia a fallos. En la fase donde DRB encuentra congestión en uno de los puertos de salida, podría trabajarse para que DRB detecte el caso de un enlace caído, notificar al nodo origen que usa la trayectoria fallada, entonces DRB abre nuevas trayectorias para enviar los paquetes hacia el destino, pero de igual manera DRB monitorizaría el fallo considerando tiempos máximos de espera para considerar un camino inutilizado.

BIBLIOGRAFÍA

[1] "Top500 Supercomputers Sites", http://www.top500.org

[2] William Dally, Brian Towles. *Principales and practices of interconnection networks*. Morgan Kaufmann publishers, 2004.

[3] J. Duato, S. Yalamanchili, L. M. Ni, *Interconnection Networks An Engineering Approach (Revised printing)*, Morgan Kaufmann publishers. 2003.

[4] Franco D. "*Balanceo Distribuido del Encaminamiento en Redes de interconexión de computadores paralelos*", tesis doctoral, Dept. Arquitectura de Computadores y Sistemas Operativos (DACSO-CAOS), Universitat Autónoma de Barcelona., 2000.

[5] Diego Lugones. "*Control de congestión adaptivo en redes Infiniband*", tesis predoctoral, Dept. Arquitectura de Computadores y Sistemas Operativos (DACSO-CAOS), Universitat Autónoma de Barcelona., 2007.

[6] F. Gilabert, M.E. Gómez, P. López, J. Duato. "On the influence of the selection fuction on the performance of fat-trees", Proc. *Euro-Par 2006*, LNCS 4128, Springer, 2006, pp. 864-873.

[7] Baydal E., "A family of Mechanisms for Congestion Control in Wormhole Networks", Proc. *IEEE Trans. Parallel Distrib. Syst. 2005,* pp. 772-784.

[8] Rajkumar Buyya, *High Performance Cluster Computing*, Prentice Hall, 2000.

[9] F. Petrini, M. Vanneschi, "K-ary n-trees: High Performance Networks for Massively Parallel Architectures", Proc. *International Parallel Processing Symp.*, IEEE, 1997. pp. 87-93.

[10] J. Ferrer, Baydal E., A. Robles, P. López, J. Duato, "Congestion Management in MINs through Marked & Validated", Proc. *15th EURMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP07),* IEEE, 2007. pp. 254-261.

[11] M. Gusat, D. Craddock, W. Denzel, T. Engbersen, N. Ni, G. Pfister, W. Rooney, J. Duato, "Congestion Control in Infiniband Networks", Proc. *13th Symposium on High Performance Interconnects (HOTI 05),* IEEE, 2005. pp. 158-159.

[12] G. Gómez, F. Gilabert, M.E. Gómez, P. López, J. Duato, "Deterministic versus Adaptive Routing in Fat-Trees", Proc. *International Parallel and Distributed Processing Symposium (IPDPS 2007, Workshop on Communication Architecture for Clusters*, IEEE, 2007. pp. 1-8.

[13] X. Yuan, W. Nienaber, Z. Duan, "Oblivious Routing for Fat-tree Based SystemArea Networks with Uncertain Traffic Demands", Proc. *SIGMETRCIS Conference 07*, ACM, 2007. pp. 337-348.

[14] C. Gómez, M.E. Gómez, P. López, J. Duato, "An Efficient Fault-Tolerant Routing Methodology for Fat-Tree Interconnection Network", Proc. *International Symposium Parallel Applications (ISPA 2007),* LNCS 4742, Springer, 2007. pp. 509-522.

[15] F. O. Sem-Jacobsen, T. Skeie, O. Lysne, J. Duato, "Dynamic Fault Tolerance with Misrouting in Fat Tress", Proc. *International Conference on Parallel Processing (ICPP 06),* IEEE, 2006. pp. 33-45.

[16] X. Y. Lin, Y. C. Chung, T. Y. Huang, "A Multiple LID Routing Scheme for Fat-Tree-Based Infiniband Networks", Proc. *18th International Parallel and Distributed Processing Symposium (IPDPS 04),* IEEE, 2004. p. 11a.

[17] J. Zhou, X. Y. Lin, C. H. Wu, Y. C. Chung, "Multicast in Fat-Tree-Based Infiniband Networks", Proc. *4th International Symposium on Network Computing and Applications (NCA 05),* IEEE, 2005. pp. 239-242.

[18] D. Franco, I. Garcés, E. Luque, "Distributed routing balancing for interconnection network communication", Proc. *5th International Conference On High Performance Computing (HIPC 98),* 1998, pp. 253–261.

[19] D. Franco, I. Garcés, E. Luque, "Avoiding Communication Hot-Spot in Interconnection Network", Proc. *32th Annual Hawaii International Conference on System Sciences (HICSS 99),* IEEE, 1999, pp. 80-40.

[20] D. Franco, I. Garcés, E. Luque, "A new method to make communication latency uniform: distributed routing balancing", Proc. *13th international conference on Super-computing (ICS '99),* ACM, 1999, pp. 210–219.

[21] I. Garcés, D. Franco, "Analysis of distributed routing balancing behavior", in SAC'02:Proceedings of the 2002 ACM symposium on Applied computing. New York, NY, USA: ACM, 2002, pp. 817–824.