Universitat Autònoma de Barcelona

Escola Técnica Superior d'Enginyeria

Departament d'Informàtica

Balanceo Distribuido del Encaminamiento en Redes de Interconexión de Computadores Paralelos

Memoria presentada por Daniel Franco Puntes para optar al grado de Doctor en Informàtica

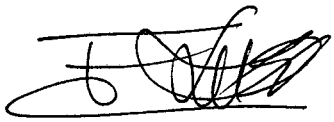
Barcelona, 11 de agosto de 2000

Balanceo Distribuido del Encaminamiento en Redes de Interconexión de Computadores Paralelos

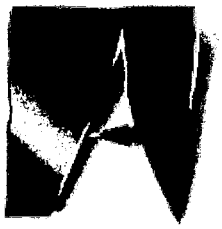
Memoria presentada por Daniel Franco Puntes para optar al grado de Doctor en Informàtica por la Universitat Autònoma de Barcelona. Trabajo realizado en el Departamento de Informática de la Escola Tècnica Superior d'Enginyeria de la Universitat Autònoma de Barcelona, bajo la dirección del Dr. Emilio Luque Fadón.

Barcelona, agosto de 2000

Vo. Bo. Director Tesis



Fdo. Emilio Luque Fadón



Universitat Autònoma de Barcelona

Escola Técnica Superior d'Enginyeria

Departament d'Informàtica

Balanceo Distribuido del Encaminamiento en Redes de Interconexión de Computadores Paralelos

Este trabajo ha sido financiado en parte por la Comisión Interministerial de Ciencia y Tecnología dentro del Programa de Tecnologías de la Información y de las Comunicaciones: CICYT TIC95/0868 y TIC98/0433

A mis hijas Carla y Berta,
ya que gracias a ellas...
A mi hijo Marc,
que es a quien más quiero
en este mundo.
A mi mujer Rut,
que es a quien más quiero
en este mundo y fuera de él...

GRACIAS a:

Emilio Luque por la dirección realizada a lo largo de todo este trabajo, y especialmente por la intuición en la temática elegida y en la búsqueda del tipo de solución requerido. También, por hacerme andar un paso y luego otro paso y luego otro, ... en la dirección correcta y, finalmente, por sus esfuerzos para dedicarme parte de su preciado tiempo necesario para la realización de este trabajo.

Ana Ripoll, porque a lo largo de todos estos años de trabajo en los que me he ido formando siempre he sentido su constante presencia cerca de mí, sobre todo al inicio durante el periodo en que me dirigió la Beca de Investigación que disfruté.

A Indira Garcés porque además de investigar nos lo pasábamos muy bien

A todos los compañeros de la Universidad, con los cuales comparto el barco en el que estamos subidos:

Joan Sorribes, Pofirio Hernández, Lola Rexachs, Tomás Margalef, Miguel Angel Senar, Remo Suppi, Tomás Díez, Juan Carlos Moure, Ana Cortés, Eduardo César, Elisa Heymann, Toni Espinosa, Fernando Cores, Josep Jorba, José Antonio Marco.

A los que ya no están cada día, pero que les guardo un gran recuerdo por los buenos ratos pasados juntos:

A María Serrano, Fina Parcerisa, Carlos Ortet.

A todos ellos, gracias sinceras por acompañarme durante este tiempo

Daniel Franco

Prólogo

Estas líneas quieren hacer una pequeña reflexión sobre lo que el autor entiende que es una tesis doctoral. Una tesis doctoral es un trabajo que visto desde algunos ámbitos alejados suscita respeto y admiración. Cuando yo era estudiante de Licenciatura y nunca había imaginado que realizaría una tesis doctoral, me parecía un trabajo inalcanzable por un ser humano normal. Ahora, que representa que estoy en la fase final de la realización de mi tesis doctoral, puedo afirmar que hace falta ser casi sobrehumano para realizar semejante trabajo. Esta claro que el doctorado es una carrera de fondo.

En esa carrera, la madurez intelectual es fundamental para realizar la inmersión en la investigación científica, porque el investigador debe conocer profundamente la ciencia que hay detrás de sus futuras exploraciones y sus métodos y procedimientos, y esto requiere esfuerzo y tiempo.

Según la opinión oficial de los estamentos universitarios, un programa de doctorado es un plan de estudios por el cual se pretende cualificar a un alumno postgraduado en las metodologías específicas y en las técnicas de investigación de una o más áreas asociadas al programa. Los resultados de cara a los doctorandos son el reconocimiento de la suficiencia investigadora y, finalmente, la obtención del grado de Doctor.

La tesis doctoral ha de consistir en un trabajo original de investigación que, a partir del estado de la técnica o el arte concreto objeto de su investigación, lo innova aportando aspectos históricos, metodológicos o empíricos.

Con los estudios de doctorado se pretende dotar al estudiante de las herramientas y la capacitación específica requerida para la investigación científica, que van orientadas a medio y largo plazo, confiriendo la madurez intelectual requerida para la investigación experimental o aplicada.

Para mí, la realización del presente trabajo ha sido una tarea ardua, ingente, emocionante, en la que debo confesar que ha habido más momentos de satisfacción que de frustración. Claramente puedo distinguir una serie de etapas en el avance del trabajo de la mano del director de la tesis. Brevemente, sin ánimo de ser exhaustivo, puedo citar la selección del área de la Ciencia a trabajar, la profundización en el tema elegido, la selección y definición de un problema o cuestión a resolver y la resolución de dicho problema, que formará la aportación principal del trabajo de tesis.

La primera etapa, en la que el doctorando elige una propuesta del director, me parece que se realiza más a ciegas que de otra manera, porque en ese momento no sabe muy bien que significa todo ello. Una fase crucial es la definición y especificación del problema a resolver. Como en muchos casos, el conocimiento de un problema y sus causas es la mitad del camino de su resolución. A partir de aquí, empieza un periodo impreciso y laberíntico de búsqueda de una solución original. Este periodo no se sabe cuándo acaba ni dónde. Pero una vez se obtiene un principio de solución coherente, se empieza a vislumbrar la luz al final del túnel. A partir de ahí, faltará la demostración de la validez del método o aportación desarrollados, un trabajo muy diferente al anterior, más metódico y preciso.

Los resultados visibles de todo este trabajo son una nueva aportación a la Ciencia pero además, y no menos importante, la formación del candidato que le capacita para investigar y para dirigir el trabajo de otras personas. Es por esta razón que, para mí, la realización de un trabajo de tesis tiene dos caras.

Por un lado, es el final de una etapa en la que se ha realizado un gran trabajo y de confirmación de una cierta capacidad ya mencionada. Pero no se acaba todo aquí, sino que, por el contrario, es el inicio de una etapa de producción de resultados de investigación en la que indudablemente se deberá tomar el testigo y dirigir a otras personas en este camino. Llega un momento en el que sientes que empiezas a "rodar" solo, en el que te das cuenta de que eres capaz de investigar por tu cuenta y de dirigir a otras personas, etapa que abordo con ímpetu e ilusión.

Creo que ante esta tarea, como muchas otras, se debe mostrar un gran interés por la investigación junto con gran paciencia y humildad, porque, como dice Umberto Eco en el libro "Cómo se hace una tesis doctoral": "el que pretende abarcarlo todo, al final no abarca nada".

Este autor, en cierta medida espera humildemente haber conseguido con este trabajo los aspectos antes mencionados,

El autor,

ÍNDICE GENERAL

Índice de figuras

Índice de tablas

Índice de ecuaciones

Capítulo 1 La computación y comunicación de altas prestaciones

1.1 Introducción

El mundo de la computación de altas prestaciones es una parcela de las ciencias de la computación cuyo objetivo es dar respuesta a las necesidades más exigentes de computación que se dan en muchas disciplinas. La computación de altas prestaciones no es un sector mayoritario dentro del mundo de los computadores en general, pero sí que es el que se plantea los mayores retos, asume las mayores demandas y es el sector que está en cabeza de la evolución tecnológica en la carrera por el aumento de prestaciones en los computadores. Sobre este aspecto, el sector de las altas prestaciones sirve, además, de motor o locomotora en la evolución de las prestaciones y/o características del resto de computadores y en las técnicas y tendencias del resto del mundo de la computación.

Muchas de las nuevas características inventadas y adoptadas en los computadores de altas prestaciones son heredadas en poco tiempo por los sistemas convencionales en la rápida evolución tecnológica que domina al mundo de los computadores, cuando la reducción de coste, tamaño y consumo lo hacen posible. Por hacer una metáfora gráfica, podríamos decir que la computación de altas prestaciones es al mundo de la computación en general, lo que la Formula 1 es al mundo de la automoción.

La computación y comunicación de altas prestaciones se puede definir de diferentes maneras dependiendo del aspecto en que se fije la atención. En general, se puede enunciar como la computación que es capaz de ofrecer potencia de cálculo y de comunicaciones arbitrariamente ilimitadas ante las demandas de los usuarios. Las comunicaciones son importantes entre los elementos de un computador paralelo y/o distribuido de altas prestaciones. Tradicionalmente, las características de este cómputo de altas prestaciones pasaban por grandes cantidades de cálculos matemáticos con números en punto flotante sobre matrices de múltiples dimensiones de gran tamaño.

Un factor importante, aunque no crítico, era el tiempo de respuesta en el que se querían obtener los resultados de los cálculos. Se usaban este tipo de computadores de altas prestaciones para reducir el tiempo de cálculo desde magnitudes totalmente inaceptables (meses o años) a cantidades aceptables (días o semanas), aunque el usuario podía esperar hasta obtener los resultados y no tenía restricciones de tiempo real que le hiciesen necesitar los resultados antes de un cierto tiempo y, si se daba el caso de que se superaba ese tiempo, los resultados ya no eran aprovechables.

Entre las áreas que tradicionalmente han requerido cómputo de altas prestaciones, podríamos nombrar como más emblemáticas, dentro de las aplicaciones científico-técnicas, la carrera por la conquista del espacio, la predicción meteorológica, la prospección geológica, el diseño y síntesis de nuevas moléculas químicas, descifrado de códigos genéticos, simulación de experimentos de física nuclear, etc. Algunos de los problemas identificados por el programa de "*High Performance Computing and Communications*" de los Estados Unidos, que se engloban bajo la designación de "*Grand Challenge Problems*" [136] son el genoma humano, la turbulencia de fluidos, la dinámica oceánica, de vehículos o de fluidos viscosos, así como el modelado de superconductores o la visión por computador. Las técnicas utilizadas por estas aplicaciones de la física, astronomía, geología, química, biología,... se basan en la resolución de sistemas de ecuaciones, modelado de sistemas lineales y no lineales, y, entre otras, la simulación de sistemas, siendo esta última área de gran importancia y trascendencia en el mundo de la computación actual.

Muchas son las disciplinas actualmente que demandan un cálculo de altas prestaciones, pues a medida que éste ha ido evolucionando, perfeccionándose, haciéndose más asequible económicamente y más fácil de utilizar, las diferentes áreas de aplicación han puesto al computo de altas prestaciones bajo su punto de mira. [109].

Las necesidades actuales del cómputo de altas prestaciones se han ampliado desde las aplicaciones puramente científico-técnicas, como las anteriormente mencionadas, a otras disciplinas como el procesamiento gráfico, la extracción y resumen de datos de grandes bases de datos para la ayuda a la toma de decisiones, la distribución de imágenes en movimiento, etc. Asimismo, 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. El progreso en aplicaciones y comunicaciones ha abierto el mundo de la computación y comunicación de altas prestaciones desde áreas tradicionales a otras como los negocios, la telemedicina, el teletrabajo, la teleenseñanza, teledistribución de películas bajo demanda, la estadística aplicada, etc.

Los requerimientos del computador de altas prestaciones inicial solía incluir, únicamente, la realización de un gran volumen de cálculos matemáticos, como ya se ha comentado, y el usuario esperaba hasta la finalización de dichos cálculos. Sin embargo, las aplicaciones actuales demandan, además, una serie de requerimientos nuevos como son: un gran volumen de comunicación de datos, una respuesta rápida o en tiempo real y/o la interactividad con el usuario. Las aplicaciones actuales que incluyen imágenes o gráficos de alta resolución en movimiento, por ejemplo, suelen requerir gran capacidad de almacenamiento, pues los datos ocupan una gran cantidad de espacio, gran capacidad de cómputo para procesarlos y gran capacidad de comunicaciones para moverlos o enviarlos de un sitio a otro bajo demanda del usuario.

Muchas de estas aplicaciones, frecuentemente englobadas bajo el nombre de multimedia, requieren, además, que la respuesta sea rápida, si hay un cliente esperando por un servicio, o en tiempo real, si se necesita que imagen y sonidos, comprimidos y descomprimidos, y enviados a los destinos lleguen sincronizados y a un ritmo constante, por ejemplo.

Otra de las aplicaciones, que ya se ha comentado que es de gran importancia, es la simulación. Muchos de los problemas del "*Grand Challenge*" son problemas de simulación. La simulación del comportamiento de cualquier sistema físico, biológico, sociológico,... suele requerir grandes prestaciones de cómputo cuando la fiabilidad que se requiere de los resultados es alta o el tamaño y/o resolución del sistema a simular crece aunque sea de manera moderada. Simulación conducida por eventos, conducida

por tiempo, simulación secuencial o distribuida, conservativa u optimista, son técnicas que se utilizan en la actualidad con éxito y que demandan gran capacidad de cómputo y de comunicaciones, si se ejecutan sobre un computador paralelo o distribuido.

Los computadores de altas prestaciones son los que son capaces de ofrecer los requerimientos de cómputo de altas prestaciones que necesitan las aplicaciones hasta ahora mencionadas. El siguiente punto describe las diferentes alternativas y tipos que existen de computadores paralelos.

1.2 Computadores de altas prestaciones

En este punto se hace un breve recorrido por las distintas alternativas de computadores de altas prestaciones existentes [72]. La búsqueda de arquitecturas cada vez más rápidas para manejar grandes aplicaciones de cálculo intensivo ha dado paso a los computadores paralelos, sistemas con gran capacidad de cómputo y con procesadores de alto rendimiento diseñados con la más alta tecnología. Los computadores paralelos son una evolución del computador convencional de Von Neumann de un único procesador [125].

Tales computadores se basan en la replicación de unidades funcionales tales como procesadores y memorias para aumentar la velocidad de cómputo manteniendo un costo aceptable. Los elementos replicados se conectan entre si para formar el computador de altas prestaciones mediante una red de interconexión, con el objetivo principal de ejecutar conjuntamente las tareas de cómputo de manera concurrente, lo que implica la operación simultánea de múltiples procesadores de cómputo, y paralela, cuando los procesadores ejecutan código de una misma aplicación.

Tradicionalmente, los computadores paralelos se han clasificado atendiendo al paralelismo existente entre flujos instrucciones y flujos de datos en cuatro categorías. Esta clasificación es del año 1972 y se debe a Flynn [47], que establece las siguientes categorías de computadores paralelos:

- ✓ Computadores SISD ("*Single Instruction, Single Data*"). Son los computadores tradicionales con una única unidad de procesamiento y arquitectura tipo Von Neumann. Esta arquitectura se ha mantenido a lo largo de la evolución de los computadores tradicionales y aún los más modernos procesadores actuales se pueden englobar en ella. La razón es que, aunque incorporen características de aprovechamiento del paralelismo como las memorias "*cache*" de instrucciones y datos separadas, la ejecución segmentada, la ejecución fuera de orden y la ejecución especulativa, todas estas características quedan ocultas al programador por la propia

arquitectura o por el compilador, de manera que el programador sigue viendo la máquina comportarse como un computador clásico Von Neumann.

- ✓ Computadores SIMD ("*Single Instruction, Multiple Data*"). Los computadores paralelos con un único flujo de instrucciones que opera sobre muchos datos a la vez se incluyen en esta categoría. En ellos existe una única unidad de control que gobierna todo el computador y múltiples unidades funcionales que operan sobre los datos simultáneamente. Suelen ser máquinas orientadas al cálculo matemático vectorial, cuando se precisa hacer la misma operación sobre un vector de elementos. Ejemplos de estos sistemas son el Illiac IV [10] o la Connection Machine CM-2 [128] o CM-5 [83].

- ✓ Computadores MISD ("*Multiple Instruction, Single Data*"). De este modelo, aunque conceptualmente válido, no se conoce ninguna implementación física realizada hasta la fecha.

- ✓ Computadores MIMD ("*Multiple Instruction, Multiple Data*"). En este tipo de computadores existen varios flujos de instrucciones y de datos operando simultáneamente de manera paralela. Ofrece muchas alternativas de configuración y es la categoría mas evolucionada y que ha dado lugar a la mayoría de los computadores paralelos actuales. Se pueden establecer varias categorías atendiendo a la disposición y utilización de los componentes que forman parte de ellos. A continuación se presentan y comentan cada una de las posibilidades

1.2.1 Computadores paralelos MIMD

Estos computadores se pueden dividir en dos tipos principales atendiendo al modelo de programación soportado: paso de mensajes o memoria común. Ambos tipos de computadores se forman por la replicación de las unidades básicas que aparecen en el computador clásico de Von Neumann: procesadores y memoria. Las características diferenciales de los modelos de programación son la visión y uso de la memoria, por un lado, y la sincronización y cooperación entre tareas, por el otro. Estas características han dado lugar tradicionalmente a los computadores de paso de mensajes y a los computadores de memoria común.

Actualmente, las diferencias entre uno y otro tipo son cada vez menores y existe una cierta convergencia motivada por razones de avance tecnológico, por un lado, y de mejor comprensión de los paradigmas de programación, por el otro. Típicamente, los computadores de paso de mensajes son más fácilmente escalables que los de memoria común y por ello son capaces de aprovechar mejor los avances tecnológicos en cuanto a

integración y multiplicación de componentes. Es por esto, que los computadores de memoria común han "importado" estrategias de diseño de otros tipos para mejorar su escalabilidad. Por otro lado, muchas de los computadores paralelos actuales soportan ambos modelos de programación, memoria compartida o paso de mensajes, aunque la implementación de cada uno de ellos difiera sustancialmente según el tipo de computador. La Tabla 1-1 muestra la clasificación de los computadores paralelos tipo MIMD.

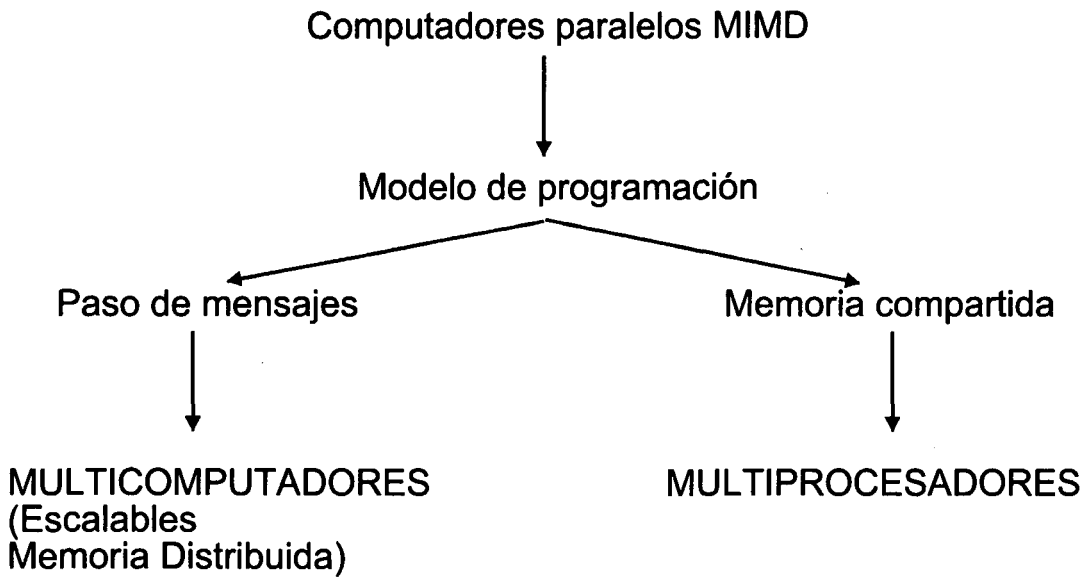


Tabla 1-1 Clasificación de los computadores paralelos MIMD

A continuación, se analizan cada uno de los dos tipos de computadores paralelos mencionados por separado presentando las diferentes alternativas posibles para cada uno de ellos.

1.2.1.1 Computadores paralelos de paso de mensajes

En el primer modelo, el paso de mensajes, se engloban los computadores paralelos llamados multicomputadores [7] [118]. Éstos están formados por un conjunto de nodos, donde cada uno de ellos es un computador completo con su unidad central de proceso y su memoria, unidos por una red de interconexión. Cada procesador tiene acceso directo a su memoria y para acceder a los datos colocados en la memoria de otro procesador debe hacerlo mediante la invocación explícita de una primitiva de comunicación para recibir y enviar mensajes. La Figura 1-1 representa el paso de mensajes en el que para que la comunicación entre dos procesos se haga efectiva, cada proceso debe realizar una acción coincidente sobre la otra, uno de enviar y el otro de recibir, mediante la especificación de un nombre de canal o del proceso correspondiente y un identificador o "*tag*".

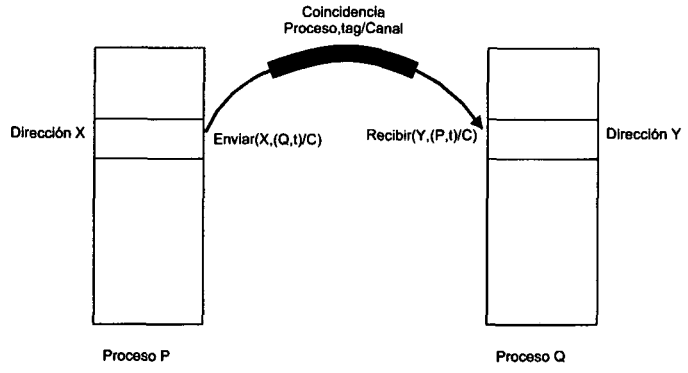
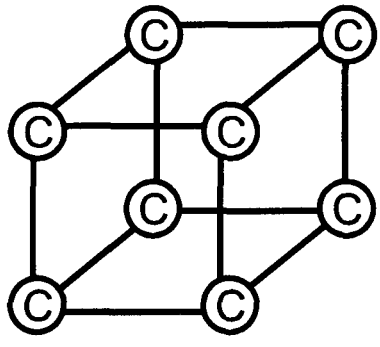


Figura 1-1 Modelo de programación de paso de mensajes

Estos computadores se llaman también computadores de memoria distribuida, en los cuales aparece de manera explícita el paralelismo, que, aunque es una característica natural del mundo real, rompen con la metodología tradicional secuencial de programar. Generalmente, estos computadores se programan mediante lenguajes con paralelismo explícito como CSP[70] u Occam[74] o mediante librerías de primitivas de creación de tareas paralelas y paso de mensajes explícitas como PVM ("*Parallel Virtual Machine*" [97]) o MPI ("*Message Passing Interface*" [63]).

La Figura 1-2 muestra una estructura típica de un multicomputador paralelo de primera generación, en la que los nodos de la red realizan tanto el cómputo como la gestión de las comunicaciones entre los nodos a través de los enlaces. A lo largo de la evolución de estos multicomputadores han aparecido procesadores especializados en las comunicaciones, frecuentemente llamados encaminadores o "*routers*", a los cuales se conectan los nodos que realizan el cómputo del programa de aplicación.



C:Computador (Procesador, Memoria, Sistema E/S)

Figura 1-2 Estructura típica de un multicomputador paralelo

Este tipo de computadores puede estar compuesto de componentes específicos o genéricos. En el caso de utilizar componentes específicos, es decir, especialmente diseñados para ese computador, las prestaciones que se pueden conseguir son muy altas, pero, a su vez, el coste también es muy alto. Frente a esta posibilidad, existe la

alternativa de utilizar componentes estándar de altas prestaciones, utilizando procesadores y componentes de red disponibles comercialmente, como la tecnología ATM [107]. Con esta aproximación se han desarrollado las llamadas redes de estaciones de trabajo (NOWs, "*Networks of Workstations*" [5]) que son capaces de proveer paralelismo eficaz a un bajo costo. Esta posibilidad se ha sugerido en [133], [14] y [48]. La Tabla 1-2 muestra la clasificación de los multicomputadores.

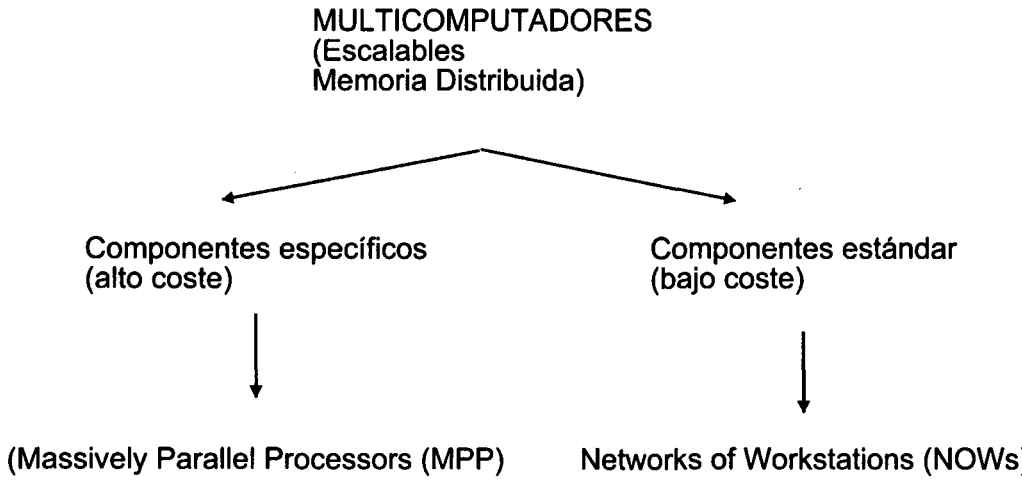


Tabla 1-2 Clasificación de los Multicomputadores

Ultimamente se está explotando la posibilidad de utilizar un gran número de estaciones de trabajo distribuidas sobre una gran área geográfica (que puede llegar a ser el mundo entero) para colaborar como sistema paralelo, utilizando una parte fija o variable según disponibilidad de la capacidad de cada estación de trabajo unida al sistema. Este tipo de computación paralela se llama "*High Performance Cluster Computing*" o "*Metacomputing*". Proyectos como "*Condor*" [84] y "*BeoWulf*" [123] ofrecen alternativas de este tipo.

Las principales ventajas de los multicomputadores son su buena escalabilidad y adecuado rendimiento siempre que se haga una programación adecuada de tales sistemas [101]. Debido a la capacidad de integrar gran cantidad de procesadores, a estos computadores paralelos, se les ha llamado Procesadores Paralelos Masivos (MPP- "*Massively Parallel Processors*"). Ejemplos de computadores de este tipo son el Intel iPSC, el IBM SP-2, basado en procesadores Power 2 RISC, o la iniciativa ASCI Red [87] del Gobierno de los Estados Unidos, sistema capaz de alcanzar un Teraflops (un trillón de operaciones en punto flotante por segundo) mediante 9632 procesadores Pentium-Pro.

1.2.1.2 Computadores paralelos de memoria común

El otro tipo de computadores paralelos son los computadores de memoria común o compartida, también llamados multiprocesadores [29]. Un multiprocesador [134] realiza las funciones de intercambio de datos y de sincronización a través de un espacio de direcciones global accesible a todos los procesadores. Estas funciones se realizan mediante el acceso a variables por parte del programa en ejecución. Estos computadores están formados por la replicación de los dos tipos de módulos mencionados anteriormente, procesadores y memorias, que se encuentran unidos por una red de interconexión. La red de interconexión permite que cualquier procesador pueda acceder a cualquier módulo de memoria. La Figura 1-3 muestra el modelo lógico de programación de memoria compartida en la que varios procesos poseen un área privada de memoria cada uno de ellos y además existe una zona común para el intercambio de información vía la lectura y escritura de variables compartidas.

Físicamente, la memoria compartida puede presentar dos tipos de arquitectura: Memoria compartida con acceso uniforme (Arquitecturas UMA: "*Uniform Memory Access*") y Memoria compartida de acceso no uniforme (Arquitecturas NUMA: "*Non-Uniform Memory Access*"). A continuación se presentan cada una de las alternativas [42].

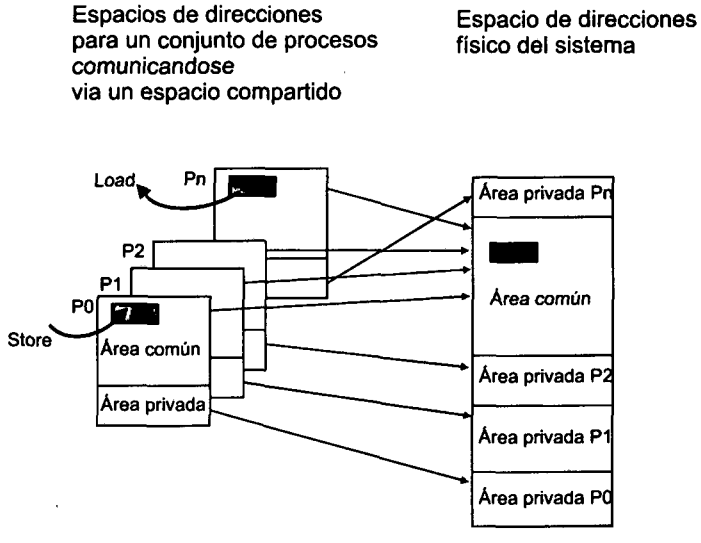


Figura 1-3 Modelo de programación de memoria compartida

1.2.1.2.1 Arquitecturas de memoria compartida con acceso uniforme UMA

En estas arquitecturas, toda la memoria global se accede a través de una interconexión común, de manera que el tiempo de acceso a cualquier posición de memoria compartida, en ausencia de conflictos, es uniforme para todos los procesadores porque todas las memorias se encuentran a la misma distancia. Esta arquitectura supone la implementación directa del principio de memoria compartida en el que existe una memoria centralizada compartida por varios procesadores. Estos sistemas suelen recibir el nombre de multiprocesadores simétricos (SMP- "*Symetric Multi Processors*") porque todos los procesadores están a la misma distancia de la memoria.

La Figura 1-4 muestra la estructura básica de un multiprocesador formada por un conjunto de procesadores, memorias y controladores de entrada y salida interconectados mediante una red de interconexión. En esta estructura, todos los módulos de memoria forman un espacio único y cualquier procesador puede acceder a cualquier posición de esa memoria.

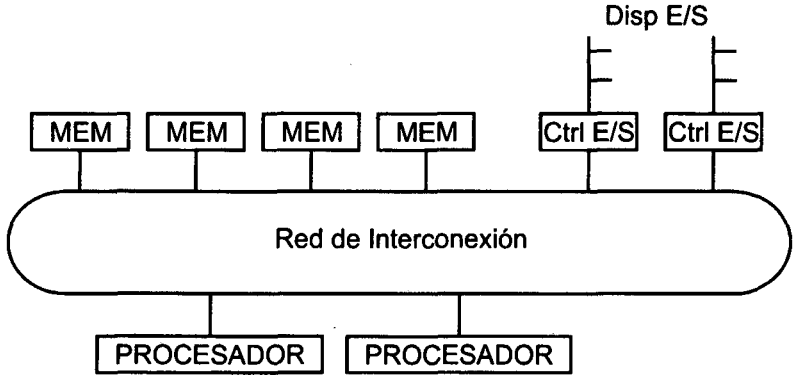


Figura 1-4 Estructura básica de un multiprocesador

La Figura 1-5 muestra los esquemas típicos de interconexión de multiprocesadores de memoria común, como son el "*crossbar*", la red multietapa o el "*bus*" único.

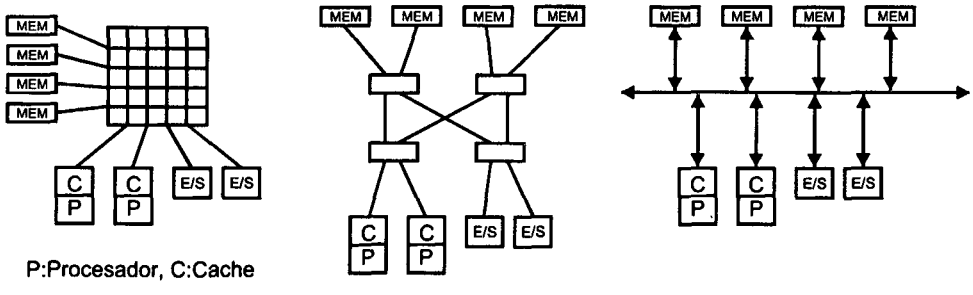


Figura 1-5 Esquemas típicos de interconexión de multiprocesadores de memoria común

1.2.1.2.2 Arquitecturas de memoria compartida distribuida de acceso no uniforme NUMA

Estas arquitecturas se llaman NUMA debido a que el tiempo de acceso a la memoria compartida tiene valores diferentes según los accesos son locales al procesador o remotos al mismo. En esta arquitectura, los módulos de memoria están distribuidos entre los procesadores de manera que cada procesador tiene una parte del total de la memoria compartida del sistema. Se puede decir que la memoria se encuentra físicamente distribuida, asociada a cada procesador, pero lógicamente compartida por todos ellos, ya que existe un espacio de direcciones único. Por esto a este tipo de computadores se les llama computadores de memoria compartida distribuida ("DSM-*Distributed Shared Memory*").

La Figura 1-6 muestra la organización de un multiprocesador de memoria compartida escalable de tiempo de acceso no uniforme a la memoria. En este caso existe una serie de nodos formados por un procesador, una memoria "*cache*" y una memoria principal, todos ellos interconectados por una red de interconexión. Aunque exista un espacio de memoria único y común a todos ellos, el tiempo de acceso a un dato variará respecto a en que módulo de memoria se encuentre.

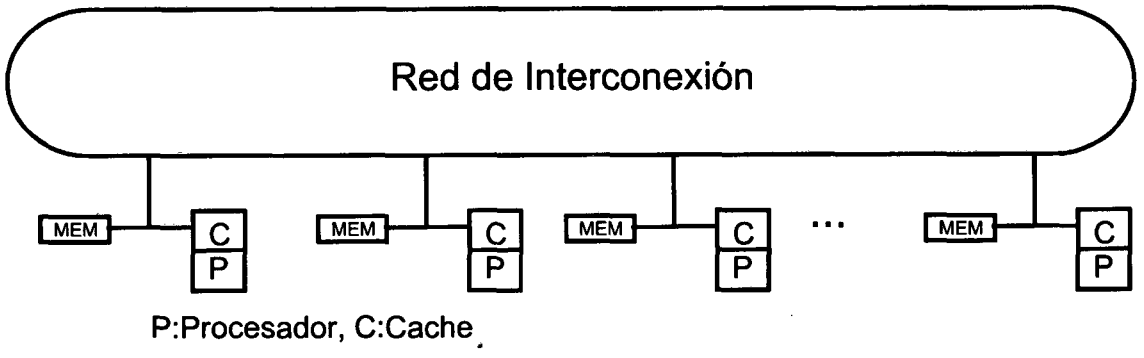


Figura 1-6 Organización de un multiprocesador de memoria compartida escalable de acceso no uniforme a memoria

Dentro de las arquitecturas NUMA, se pueden presentar dos variantes. El primer caso, son las arquitecturas COMA ("*Cache Only Memory Access*"), en las que cada procesador tiene una parte de la memoria compartida, pero esta memoria se comporta enteramente como una memoria cache. El segundo tipo, son las arquitecturas cc-NUMA ("*Cache Coherent Non-Uniform Access Memory*") que son capaces de mantener la coherencia de la memoria "cache" mediante mecanismos "*hardware*" [24]. Este último modelo es el más popular de multiprocesadores actuales, de los que el Origin 2000 de SGI es un buen ejemplo. Últimamente se han presentado desarrollos que pretenden agrupar lo mejor de varias alternativas como es "*Reactive NUMA*" [43].

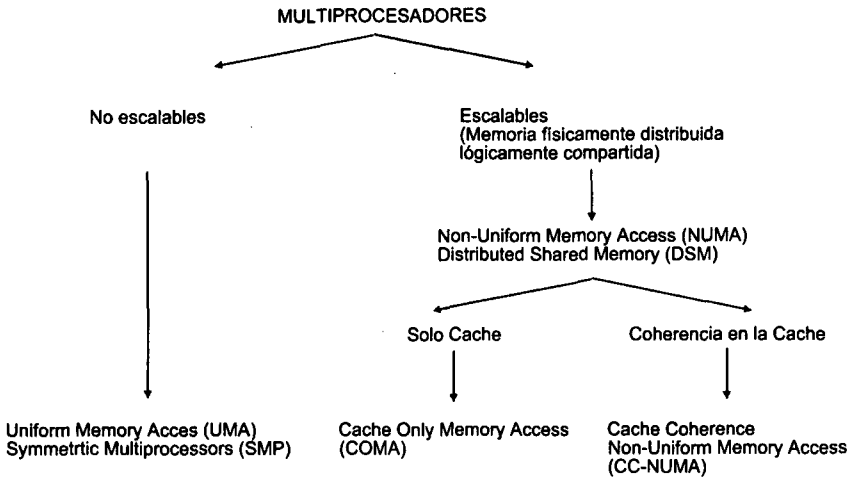


Tabla 1-3 Clasificación de los Multiprocesadores

La Tabla 1-3 muestra un resumen los tipos de multiprocesadores presentados. Hasta aquí se ha presentado una breve descripción de las arquitecturas de los computadores paralelos más habituales y sus principales características. Como se ha visto, los modelos de programación de paso de mensajes y de memoria común representan dos modelos claramente distintos, cada uno aportando un paradigma bien definido para la compartición de datos, la comunicación y la sincronización. Sin embargo, las estructuras de los sistemas sobre los que se soportan dichos modelos han convergido hacia una organización común, representada por una colección de computadores completos junto con un procesador de comunicaciones que conecta cada nodo del sistema a una red de comunicaciones escalable.

Como se ha visto, pues, las diferentes alternativas analizadas tienen en común algún sistema de interconexión de los elementos que se encuentran replicados, tanto si se muestra explícitamente, como si permanece oculto respecto al modelo de programación que implementa el computador paralelo. A continuación, se expondrán algunas consideraciones sobre la importancia de la red de interconexión y las razones de su elección como objeto de este trabajo de tesis.

1.3 La red de interconexión del computador de altas prestaciones

Actualmente, se están realizando muchos esfuerzos de investigación en el campo de los computadores de altas prestaciones. Estos esfuerzos están dirigidos en dos direcciones principales, las cuales no son totalmente independientes. La primera se centra en mejorar las prestaciones de tales supercomputadores mediante avances en

aspectos de su arquitectura y organización interna. La segunda se centra en la definición del modelo de computador paralelo que debería ofrecerse al usuario para permitirle usarlo eficientemente.

Un computador paralelo se puede entender como un conjunto de procesadores independientes unidos por algún tipo de red de interconexión, como se ha visto en la sección anterior [45]. Con un computador de tales características se dispone de una potencia distribuida que permite repartir el trabajo a realizar. Una actividad que surge de manera espontánea cuando se realiza un trabajo distribuido entre varias tareas es que éstas deben comunicarse para cooperar en la resolución de un problema común. Por lo tanto, existe la necesidad de que el computador asigne automáticamente las tareas a realizar entre los procesadores, por un lado, y ofrezca un sistema de comunicaciones que sea sencillo de utilizar y eficiente, por el otro.

Uno de los aspectos clave del computador de altas prestaciones, sea un computador paralelo o un sistema distribuido, que influye en las dos cuestiones anteriores en las que se centra la investigación actual, es la red de interconexión que conecta los diferentes nodos que lo componen. Como se ha dicho, el paralelismo en forma de replicación ha sido capaz de proporcionar mejoras de coste efectivo en prestaciones de los computadores. Un parámetro esencial de esta efectividad son las comunicaciones. En arquitecturas multicomputador, es fundamental la red de interconexión puesto que, respecto a las arquitecturas convencionales, es el elemento que proporciona el paralelismo, ya que la organización interna de los nodos de cómputo responde a la de un procesador actual convencional. La importancia de la red de interconexión es debida a una serie de razones que se agrupan en sus características de escalabilidad topológica, su comportamiento dinámico y la visión que presentan al usuario. A continuación se comentan cada una de ellas y su importancia en el computador paralelo.

- ✓ La red de interconexión es uno de los dos elementos centrales para formar un computador paralelo de altas prestaciones, junto con los nodos de cómputo. En este sentido, la escalabilidad de la red de interconexión es un concepto clave en la escalabilidad del sistema.

- ✓ Las prestaciones de la red de interconexión influyen directamente en el rendimiento del computador completo. Una red de interconexión ofrecerá un cierto ancho de banda, medido como la cantidad de información capaz de transmitir por unidad de tiempo. Generalmente, este ancho de banda será proporcionado por cada uno de los enlaces que componen la red. Las prestaciones de una red se miden el número de mensajes efectivamente enviados por unidad de tiempo. Este concepto se llama

"*throughput*". En la realidad, la transmisión de un mensaje comportará una serie de retardos debidos al propio sistema de interconexión ya que por razones tecnológicas y de coste no puede pensarse en una topología totalmente conectada. Este retardo, llamado latencia, limita el número de mensajes máximo que la red puede aceptar. Por otro lado, si pensamos ejecutar paralelismo de grado fino, que es el caso en el que la relación entre volúmenes de cómputo y comunicaciones del programa es pequeña respecto la relación de capacidad de cómputo y comunicaciones del computador paralelo, pediremos a la red que sea capaz de manejar un alto "*throughput*" de mensajes, manteniendo la latencia dentro de unos mínimos admisibles. Se debe hacer notar, además, que un sistema de comunicaciones puede hacer aumentar las prestaciones de un computador de altas prestaciones al aumentar el paralelismo entre cómputo y comunicación.

✓ El tercer aspecto se refiere a la visión que se ofrece al usuario de la red de interconexión. Porque según cómo se presenten sus características, tanto estáticas como dinámicas, al usuario, se puede conformar un modelo u otro de computador paralelo. Es decir, influirá en ese modelo, entre otras, las siguientes características de la red de interconexión:

- El hecho de que la estructura de la red de interconexión se presente oculta o no al usuario.
- El hecho que ofrezca conexión de todos con todos los nodos de cómputo, o sólo parcialmente conectados.
- El hecho de que permita diferentes formas de comunicación como sincrónica o asincrónica, tamaño fijo o variable del mensaje, comunicaciones múltiples o no, etc.
- Según sea el comportamiento de la latencia de los mensajes y el "*throughput*" máximo de la red.

Por lo tanto, según las características anteriormente enumeradas, se tendrá una visión u otra diferente del computador de altas prestaciones, lo que comportará un modelo u otro de programación. La simplicidad del modelo del computador paralelo es muy importante a la hora de facilitar su uso y programación. Creemos que esta es una cuestión abierta a la investigación dónde actualmente se están realizando múltiples aportaciones, por ejemplo, sobre si un computador paralelo debe ofrecer un modelo de programación de memoria común o distribuida con paso de mensajes [48].

Con respecto a este punto del modelo del computador paralelo, queremos hacer notar que los computadores convencionales gozan de un modelo de programación único. Este modelo, basado en el modelo de computador de Von Neumann, se basa en el concepto de programa almacenado y ejecución secuencial de las instrucciones que componen el programa. A partir de esta situación, la mayoría de aplicaciones se ajustan y se pueden describir usando este modelo. Es sobre la base de la existencia de este modelo unificador que ha sido posible ir haciendo avances y mejoras en prestaciones en los computadores que han permitido ser aprovechados eficazmente. Además, esos cambios, estructurales o en la organización, del computador, no han cambiado el modelo existente. Así, los nuevos avances podían aprovecharlos programas ya existentes, y programas nuevos podían desarrollarse con la seguridad de que se ejecutarían de manera adecuada sobre cualquier procesador. En este sentido, el modelo funciona como eficaz bisagra o intermediario entre "*hardware*" y "*software*".

Sirva como muestra el siguiente ejemplo sobre el modelo de Von Neumann. Del comportamiento de las instrucciones y el acceso a los datos de los programas actuales, se establecen los principios de localidad temporal y espacial en la ejecución de los programas secuenciales en un computador. Estos principios son los que han posibilitado que técnicas como los procesadores segmentados mediante "*pipeline*" o las memorias "cache" se desarrollasen y se aumentasen las prestaciones de los computadores sin cambiar el modelo de ejecución. Es cierto que no todos los programas se ajustan bien a las técnicas de memoria cache o procesadores segmentados, o que siempre se puede encontrar o escribir un programa que no funcione de manera eficiente, pero es verdad que la gran mayoría de programas sí que se ajusta. Además, la idoneidad de la memoria "cache" o la segmentación a los programas es un concepto ampliamente asumido por la comunidad de usuarios y diseñadores de programas y procesadores [121].

Aún las más modernas técnicas, como procesadores superescalares, ejecución especulativa y otras de las que últimamente se están introduciendo en los procesadores actuales, aunque parecieran deseosas de romper con el modelo de Von Neumann, lo respetan y ocultan sus efectos sobre el modelo al usuario o programador. Esta ocultación se realiza mediante sofisticadas técnicas implementadas en el propio procesador físicamente o dejadas bajo la responsabilidad del compilador, de manera que sólo se observan sus consecuencias positivas sobre el rendimiento del procesador [121].

Sin embargo, este panorama tan conveniente, que permite una evolución constante y unificada de los computadores secuenciales convencionales, no tiene parangón en el mundo de los computadores de altas prestaciones donde el paralelismo es un concepto

fundamental. La realidad es que no existe un modelo único ampliamente aceptado ni de funcionamiento de los programas paralelos, ni de arquitectura de altas prestaciones, ni de acoplo entre ambos mundos. Este hecho hace que surjan una serie de cuestiones o dificultades que no existen en la computación convencional secuencial que convierte el campo de la computación de altas prestaciones abierto a la experimentación y a la innovación constantes.

Algunas de estas cuestiones, que afectan a todo el ciclo del desarrollo de aplicaciones sobre computadores de altas prestaciones, son: cómo se debe abordar el desarrollo de algoritmos paralelos, cómo se deberían expresar los algoritmos paralelos en programas paralelos y qué herramientas "*software*" se dispone para ello, cómo deberían ser los computadores paralelos de altas prestaciones y cómo ejecutar eficientemente un programa paralelo sobre un computador de altas prestaciones [69] [126].

En este trabajo de tesis, nos centramos en alguna de las cuestiones de diseño que los computadores de altas prestaciones o paralelos deberían incorporar en el futuro. Concretamente, nos hemos fijado en la red de interconexión como componente clave y fundamental del computador de altas prestaciones. Creemos que este es un tema en el que se están haciendo muchas aportaciones y que tendrá una importancia creciente en el futuro. En este trabajo, pues, hacemos una propuesta sobre ciertos aspectos del diseño de las redes de interconexión vistas como componentes de la computación y comunicación de altas prestaciones, y proporcionamos los mecanismos que posibilitan esa propuesta.

A continuación se concretan los objetivos de este trabajo junto con la organización de la memoria.

1.4 Objetivos del trabajo y organización de la memoria

Como se ha señalado, nuestro trabajo se centra en el estudio de las redes de interconexión para computadores de altas prestaciones, y profundiza en la definición de las características deseables de los computadores de altas prestaciones y, concretamente, de las redes de interconexión como componente común a todos ellos. Nos centramos en el estudio de las redes de interconexión porque, como se verá más adelante a lo largo de la memoria, son un componente fundamental para conseguir los requerimientos que el tipo de aplicaciones actuales demanda sobre los sistemas de altas prestaciones. Aunque nosotros nos hemos centrado en el caso de topologías de red "densa", que son más propias de computadores paralelos y de aplicaciones tipo "*best effort*", hoy en día, las aplicaciones actuales multimedia como el "*Video on Demand*", telemedicina o

teletrabajo, son aplicaciones que están incrementando rápidamente su presencia en nuestra sociedad de manera generalizada, y exigen una respuesta muy alta de la red de interconexión sobre el sistema en que se están ejecutando. Además, actualmente, a la mayoría de aplicaciones de gran volumen de cómputo tradicionales se les están añadiendo características multimedia de visualización de datos de manera animada o de recolección y/o distribución de datos de zonas dispersas o de interactividad con el usuario. Por todo ello, podemos concretar los objetivos de este trabajo de tesis en:

- ✓ Modelado analítico de las redes de interconexión.
- ✓ Estudio de las características dinámicas de las redes de interconexión.
- ✓ Análisis de las problemáticas surgidas en el funcionamiento de las redes de interconexión y sus causantes.
- ✓ Propuesta de comportamiento deseable de una red de interconexión a partir de su respuesta en comportamiento de la latencia y el "*throughput*".
- ✓ Propuesta de un mecanismo de encaminamiento para conseguir el comportamiento establecido.
- ✓ Para poder llevar a cabo los puntos anteriores, hemos debido realizar un estudio e implementación de un simulador funcional que simule el comportamiento de las redes de interconexión.
- ✓ Estudio y análisis de las propuestas introducidas mediante la comparación vía simulación con otras técnicas existentes. Este es el estudio experimental que cuantifica y confirma la bondad de nuestras propuestas y modelos de los puntos anteriores.

A partir de estos objetivos y del trabajo realizado, hemos organizado la presente memoria de la forma siguiente:

El capítulo siguiente, capítulo 2, presenta un estudio de las características estáticas, estructurales y topológicas, de las redes de interconexión y hace un recorrido por las diferentes técnicas de diseño de redes existentes hasta la actualidad.

El capítulo 3 presenta la confección de dos modelos, uno matemático y el otro funcional, del comportamiento en latencia de las redes cuando sufren una cierta carga de mensajes. Estos modelos se utilizan como herramientas de simulación para evaluar nuestra propuesta presentada en el capítulo siguiente.

El capítulo 4 presenta nuestras aportaciones para eliminar o mitigar ese comportamiento problemático. Primeramente, se deduce una tendencia problemática del comportamiento cuando se dan unas ciertas características de carga elevada, y se analizan las razones causantes de dicho comportamiento problemático, y luego, se establecen cuáles deberían ser los objetivos de comportamiento deseables de una red de interconexión para facilitar el uso por parte del programador y ofrecer unas prestaciones elevadas. A continuación, se introduce la principal aportación de este trabajo de tesis consistente en un mecanismo de balanceo del encaminamiento que intenta uniformizar la carga en todos los enlaces de la red de interconexión. Este mecanismo se llama Balanceo Distribuido del Encaminamiento ("*Distributed Routing Balancing*", DRB por sus siglas en inglés) y se basa en la distribución uniforme de la carga en la red mediante la expansión de caminos. Esta expansión es dinámica y está controlada por el nivel de latencia existente en la red. El método establece nuevos caminos alternativos simultáneos entre cada par fuente y destino con objeto de mantener una latencia baja de los mensajes DRB define cómo crear los caminos alternativos para expandir los caminos simples originales y cuándo y cómo usarlos dependiendo de la carga de tráfico de la red de interconexión. Se produce un efecto colectivo, pues esta expansión se produce para todos los pares fuente-destino de la aplicación que también interaccionan entre sí.

El capítulo 5 presenta la primera parte de la experimentación realizada para cuantificar los parámetros estructurales de nuestra propuesta y el capítulo 6 presenta las simulaciones realizadas para evaluar dinámicamente las prestaciones de la propuesta frente a otras alternativas existentes. La evaluación muestra la validez de DRB frente a otros métodos de encaminamiento como el completamente adaptivo de caminos mínimos. Se ha evaluado para un conjunto de topologías y para un conjunto de patrones sintéticos de comunicación, lo que configura un amplio espacio de testeo. Además se han evaluado aspectos concretos como la influencia del tamaño del mensaje o la escalabilidad ante el crecimiento del tamaño de la red.

Finalmente, el capítulo 7 concluye la memoria con un resumen de las principales aportaciones realizadas y presenta las líneas abiertas del trabajo.

Capítulo 2 Redes de interconexión. Parámetros de diseño estáticos

2.1 Introducción

En el capítulo precedente, se ha enmarcado el ámbito del presente trabajo centrado en las redes de interconexión de los computadores paralelos y sistemas distribuidos. En dicho capítulo se ha comentado la creciente importancia de las comunicaciones en múltiples de sus aspectos tales como las prestaciones de rendimiento que ofrece el computador paralelo o como el modo de operación que presentan las redes de interconexión.

En este capítulo se realiza una exposición teórica de los aspectos estáticos estructurales que definen las redes de interconexión que se usan en multicomputadores o multiprocesadores. Una red de interconexión es un sistema físico que se compone de una serie de elementos como son enlaces y encaminadores que se comportan e interrelacionan entre ellos de manera específica y que sirven para facilitar la

comunicación de los nodos de cómputo del computador paralelo. Primero se realiza la definición y justificación de las redes de interconexión, y luego se introducen los diferentes elementos estructurales y funcionales y los aspectos de comportamiento a tener en cuenta cuando se construye o diseña una red de interconexión y, a continuación, se analizan cada uno de las diferentes alternativas existentes en la literatura para cada uno de los elementos y/o aspectos.

El punto 2 describe las necesidades de comunicación que surgen al utilizar computadores paralelos. Es un hecho conocido que el paralelismo en forma de replicación ha sido capaz de proporcionar mejoras de costo efectivo en prestaciones de los computadores. Un parámetro esencial de esta efectividad son las comunicaciones. Asimismo, a partir de las funciones que realiza la red de interconexión, el punto 2 introduce las características y parámetros que configuran la red de interconexión. Estos parámetros son la topología, el control del flujo y el algoritmo de encaminamiento.

En el punto 3 se enumeran las diferentes topologías mas usadas para construir computadores paralelos. Existen varias posibilidades desde la conexión completa hasta estructuras lineales. Se estudian topologías que permitan una conexión entre todos los nodos con un coste y escalabilidad adecuados. En el punto 4 se explican los mecanismos de paso de mensajes. Son mecanismos para transportar y gestionar los mensajes. Es deseable que estos mecanismos cumplan una serie de requerimientos, los cuales se enuncian en el punto 4.1. En el punto 4.2. se describen las formas de establecer el camino entre dos nodos y en el punto 4.3. se comentan los mecanismos de control de flujo de los mensajes. Éstos tratan de controlar qué se hace con un mensaje cuando llega a un nodo, y si debe enviarse al siguiente nodo en el camino hacia el destino.

El punto 5 se ocupa del encaminamiento de los mensajes, es decir, la determinación de cual es el siguiente enlace de comunicación que un mensaje debe tomar en el siguiente paso hacia su destino de entre todos los enlaces posibles de un nodo. Se presentan estrategias de encaminamiento estático, centralizado, aislado, distribuido y jerárquico. Este punto recoge las técnicas de cómo tratar el tema del bloqueo ("*deadlock*") en la comunicación, la inanición ("*starvation*") o el "*livelock*", problemas que surgen al asignar y reservar recursos de comunicación a los mensajes que los solicitan.

El punto 6, después de analizar todas las funciones y las alternativas de diseño de una red de interconexión, presenta la arquitectura física de un encaminador genérico que sea capaz de soportar dichas funciones. Se analiza también la naturaleza de los retardos

temporales que generan los elementos estructurales introducidos. Finalmente, el punto 7 finaliza el capítulo con las conclusiones.

2.2 Las redes de interconexión en los computadores de altas prestaciones.

En este punto se quieren introducir los aspectos principales que intervienen en el diseño de una red de interconexión. El resto de puntos de capítulo se dedicará a analizar con profundidad cada uno de los aspectos aquí presentados. Este punto comienza desde el punto de vista del programador que confecciona un programa paralelo y desea ejecutarlo sobre un computador paralelo. Desde este punto de vista, el sistema de comunicaciones del computador paralelo debe facilitar la tarea de comunicación y cooperación entre las tareas del programa paralelo. Para ello, la red de interconexión debe dar respuesta a una serie de cuestiones y cumplir una serie de requerimientos. A partir de estos requerimientos se identifican una serie de aspectos que definen las redes de interconexión.

Considérese una red de un computador paralelo con una topología dada donde cada nodo esta formado por un elemento de procesamiento simple, y un conjunto de procesos de usuario, asignados a varios nodos en el sistema, que intercambian datos en forma de mensajes.

Estos procesos y su asignación a los procesadores es el resultado de la partición del problema y de la asignación de procesos, realizados por el usuario y/o compilador en la fase de diseño del programa. Esta forma de ejecución se basa en una expresión del paralelismo según un modelo estático, en que debe conocerse en tiempo de compilación el número y tipo de los procesos y canales lógicos de comunicación, así como su asignación, que será fija a lo largo de la ejecución, a procesadores y a enlaces de comunicación, respectivamente. Existe otra posibilidad de paralelismo dinámico en la que se pueden crear y mover procesos en tiempo de ejecución y éstos se comunican con cualquier otro proceso especificando el nombre del proceso destino, calculado en tiempo de ejecución. En ambos casos, la necesidad de un soporte para las comunicaciones proviene de la necesidad de intercambiar mensajes entre procesos asignados en nodos no adyacentes por medio de encaminamiento/transporte del mensaje a través de nodos intermedios [20] [29]. La Figura 2-1 muestra un ejemplo de programa paralelo y su asignación sobre un computador paralelo.

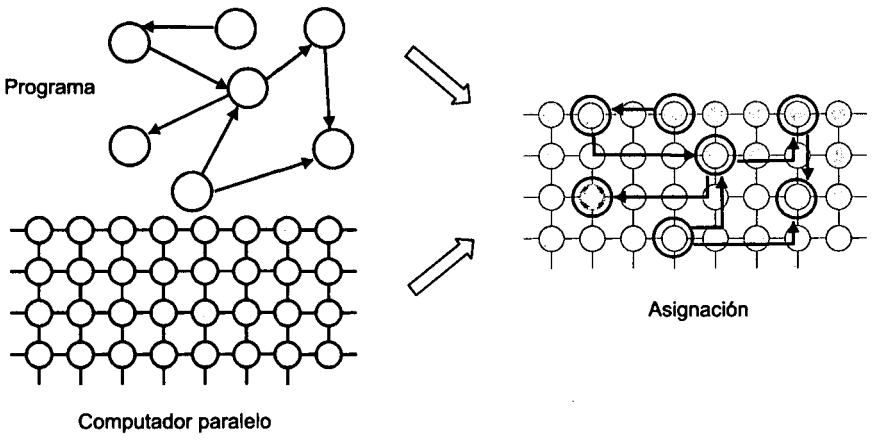


Figura 2-1 Asignación de procesos a nodos de cómputo

El sistema de comunicaciones debe hacer que las comunicaciones entre los procesos que el programador escribe en su programa sean independientes de la topología física donde se ejecutará su programa, de manera que el programa no deba cambiar si se ejecuta en otro computador con una topología diferente. La Figura 2-2 muestra cómo diferentes topologías implican cambios en la asignación de canales del programa paralelo sobre los enlaces de comunicación.

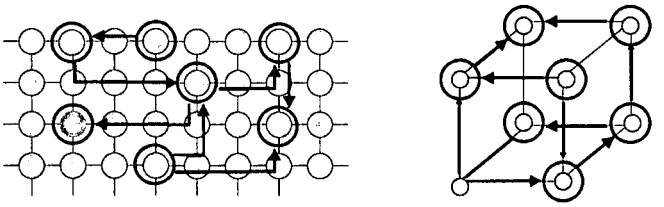


Figura 2-2 Asignación sobre diferentes computadores paralelos

El sistema de comunicaciones debe abstraer los detalles físicos del computador al programador y debe presentarle un modelo de comunicaciones de manera que el programador disponga de un modelo de lenguaje de programación donde se incluyan cuestiones de comunicación tales como:

- ✓ Especificación del destino: Cómo los procesos que desean enviar mensajes identifican el destino del mensaje. Existen dos alternativas:
  - Comunicación por nombre de proceso: Los procesos conocen los nombres de los otros procesos a los que envían mensajes y usan ese nombre para especificar el destino. El sistema debe traducir ese nombre de proceso destino al procesador en que se encuentra.
  - Comunicación por canales: Los procesos envían y reciben haciendo referencia a nombres lógicos llamados canales, que son las entidades que conectan

procesos. En este caso, el proceso emisor no necesita saber "quién" hay al otro lado del canal para recibir su mensaje.

- ✓ "Interface" con el programador: Es la manera cómo el programador hace uso de los servicios que le ofrece la red de interconexión. Existen dos posibilidades, dependiendo del nivel de integración del servicio con el lenguaje de programación:

  - Se proporcionan unas primitivas de comunicación que se insertan y se enlazan con el programa. Esta alternativa supone una mayor integración de las llamadas de comunicaciones con el lenguaje de programación, lo que facilita tareas de comprobación semántica en tiempo de compilación.

  - Se crean unas llamadas al sistema. En este caso, es el Sistema Operativo quién provee unas funciones de comunicaciones para ser usadas por cualquier programa en ejecución.

- ✓ Si se poseen diferentes esquemas para las comunicaciones intranodo e internodo o es el mismo para ambos casos. Este aspecto hace referencia a la uniformidad del programa paralelo. En el caso de tener un esquema único para ambos casos, el proceso emisor no distingue si comunica con otro proceso en el mismo procesador o no, lo cual puede facilitar la migración de procesos. En el caso contrario, el tener dos esquemas permite aplicar optimizaciones a cada uno de ellos por separado.

- ✓ Si se proveen mecanismos de comunicación síncrona y/o asíncrona. En el caso de comunicación síncrona, ambos emisor y receptor se esperan hasta que la comunicación esta lista para ser llevada a cabo. En el caso de comunicación asíncrona, el emisor envía independientemente del estado del receptor. Este aspecto es importante en la semántica de la comunicación. Hay sistemas que proveen sólo una de las posibilidades y otros que proveen simultáneamente de ambas a elección del programador.

- ✓ Si se dan facilidades solamente para la comunicación punto a punto o se proveen también mecanismos de comunicación colectiva: tipo "*multicast*" o "*broadcast*".

En un computador paralelo, cada procesador podría desear enviar un mensaje a cualquier otro. La red debe proveer, pues, un mecanismo de encaminamiento de los mensajes que los haga avanzar nodo a nodo por la red de interconexión desde el origen hasta el destino. Este mecanismo es el encargado de decidir qué ruta concreta siguen los mensajes.

Para proveer una conectividad completa, dinámica y arbitraria en una red en la que no existe una conexión física de todos con todos los nodos, es decir, que en cada instante cada nodo desee comunicar con cualquier otro sin estar el destino determinado con anterioridad, se debe implementar un mecanismo de transporte, el cual debe proporcionar la propagación de datos de procesador a procesador, basado en las direcciones de los destinos contenidas dentro del paquete de datos.

Tal mecanismo de encaminamiento de mensajes debe satisfacer un número de requerimientos dependiendo de la aplicación. Estos requerimientos son contradictorios en algunos casos, con lo que debe llegarse a un compromiso en el diseño de la red de interconexión [76] [20] [44] [41]. Estos requerimientos, que se pueden clasificar en dos ámbitos, desde el ámbito funcional y desde el ámbito de las prestaciones, se enuncian a continuación:

2.2.1 Requerimientos de un sistema de comunicaciones

2.2.1.1 Requerimientos funcionales

- El protocolo de encaminamiento debe ser libre de "*deadlock*", es decir, que no provoque una situación indefinida en la que ningún mensaje de la red pueda avanzar hacia su destino.

- Ningún paquete debe permanecer indefinidamente en la red, es decir, no debe darse "*livelock*", que es el estado en que un mensaje deambula eternamente por la red sin llegar nunca a su destino.

- Un nodo no debe rechazar indefinidamente la aceptación de un mensaje de un usuario que desea inyectar un mensaje en la red. Esta situación se llama "*starvation*".

2.2.1.2 Requerimientos de prestaciones

- Un paquete siempre debe tomar la ruta mas corta a su destino. La definición de ruta más corta puede entenderse como un concepto de distancia física topológica o de tiempo de comunicación.

- Se debe conseguir la mínima latencia posible. La latencia es el tiempo transcurrido desde que la cabecera del mensaje accede a la red hasta que la cola del mismo es entregada al destino. La latencia varía según las condiciones de tráfico en la red debido a problemas de contención sobre enlaces y nodos de encaminamiento.

- Se debe conseguir el mayor "*throughput*" posible, que solo estará limitado por el ancho de banda disponible. El "*throughput*" mide el número de mensajes por unidad de tiempo que la red es capaz de gestionar.

- El mecanismo de encaminamiento debe adaptarse a las condiciones de trafico y explotar el máximo ancho de banda de comunicaciones disponible en el sistema. Así será capaz de evitar los nodos saturados y presentará cierta tolerancia a fallos. Es decir, debe ser un mecanismo robusto y, además, que explote con eficacia los enlaces de la red.

Los parámetros principales que intervienen en la definición de una red de interconexión son la **topología**, que es la forma física como están interconectados los diferentes nodos del computador paralelo, el **control del flujo**, que es la manera cómo se administra el avance de la información de un nodo al siguiente, y el **encaminamiento**, que es la manera de determinar el camino que sigue un mensaje desde el nodo fuente hasta el destino a través de varios nodos intermedios.

Hay que señalar que la capacidad de gestionar un alto número de mensajes sin que se produzca un gran aumento de la latencia es fundamental en los casos de paralelismo de grano fino, en los que la relación de cómputo frente a comunicación es pequeña, y que por tanto hacen que haya un gran tráfico de mensajes entre los nodos de la red.

Cuando el número de paquetes que se inyectan en la red está dentro de los límites de capacidad de la red, dichos paquetes se entregan a sus respectivos destinos de manera que no hay paquetes rechazados en la inyección.

Sin embargo, cuando el tráfico aumenta de forma considerable, el rendimiento se degrada rápidamente porque la red se satura, se produce lo que denominamos congestión en la red, y existen rechazos de paquetes, es decir, los paquetes no son aceptados por la red y deben ser encolados en el nodo fuente y reenviados con posterioridad. El algoritmo de encaminamiento es un factor clave en el control de la congestión y, por lo tanto, en la tarea de asegurar que los paquetes son entregados rápidamente a sus destinos.

Por otro lado, el control del flujo se refiere al tráfico punto a punto entre un emisor y un receptor de datos. Su trabajo consiste en asegurar que, en caso de que haya un emisor muy rápido, éste no pueda enviar datos en forma continua, a una velocidad mayor que aquella con la cual se pueden recibir en el extremo receptor. Casi siempre el control de flujo implica alguna forma de realimentación directa del receptor al emisor, para indicarle al emisor lo que esta sucediendo en el otro extremo. El control de la

congestión y el encaminamiento están estrechamente relacionados. Las deficientes decisiones de encaminamiento son las principales responsables de la congestión que puede producir un cierto patrón de comunicaciones de una aplicación.

A continuación, se analizan cada uno de los aspectos siguientes de una red de interconexión: topología, establecimiento del camino y control del flujo, encaminamiento de los mensajes y resolución de anomalías en la comunicación como "*deadlock*", "*starvation*" o "*livelock*".

2.3 Topología

El primer aspecto a analizar de una red de interconexión es la disposición topológica de los nodos que la componen. El conjunto de nodos y enlaces se conectan de una cierta manera que definen una cierta estructura espacial. Esta estructura topológica de una red de interconexión se modela a través de un grafo cuyos vértices son los encaminadores de la red de interconexión y los arcos representan los enlaces físicos bidireccionales de comunicación entre nodos [120]. Un conjunto de parámetros define las características de la topología de una red de interconexión [44]:

- ✓ **Grado**: Es el número de enlaces que inciden en un nodo. Una red es de "grado regular" si el grado es constante para todos sus nodos. El grado esta en relación directa con el ancho de banda del sistema, porque más enlaces por nodo implican más capacidad de transmitir mensajes simultáneamente por unidad de tiempo. El coste asociado al grado de un nodo tiene una relación con las limitaciones tecnológicas, ya que el número de enlaces no puede crecer indefinidamente debido a la restricción de patillas o del área de cableado disponibles.

- ✓ **Diámetro**: Es la mayor de las mínimas distancias entre cualquier pareja de nodos, es decir, es el máximo número de enlaces de comunicación que deben ser atravesados por un mensaje en la red utilizando el camino más corto posible. Por lo tanto, es interesante tener redes con el menor diámetro posible, es decir, que contenga un número elevado de nodos manteniendo un diámetro reducido. Esta propiedad se llama "densidad". [15].

- ✓ **Distancia media**: Si definimos distancia entre dos nodos como el mínimo número de enlaces que hay que atravesar para ir de uno a otro, la distancia media es la media de las distancias entre cualquier pareja de nodos de la red, tal y como expresa la siguiente ecuación:

$$Distancia\ media\ de\ la\ red = \frac{\sum_{\forall i,j} distancia(nodo_i, nodo_j)}{N°\ de\ pares\ de\ nodos}$$

Ecuación 2-1 Distancia media de la red

Por otra parte, la distancia media recorrida por los mensajes viene influida por la distribución de los pares fuente-destino de los mensajes, definida por el programa de aplicación ejecutándose sobre el computador paralelo. Esta distribución se debe a la existencia de cierto grado de localidad en la comunicación. La siguiente ecuación muestra la distancia media recorrida por los mensajes de un programa:

$$Distancia\ media\ mensajes = \frac{\sum_{\forall i} distancia\ recorrida(mensaje_i)}{número\ de\ mensajes}$$

Ecuación 2-2 Distancia media recorrida por los mensajes de una aplicación

- ✓ **Conectividad**: Es el mínimo número de nodos o enlaces que deben fallar para provocar con ello que se parta la red en dos redes disjuntas. Son deseables esquemas que presenten alta conectividad.

- ✓ **Ancho de bisección**: Se define como el número mínimo de enlaces que hay que eliminar de la red de interconexión para dividirla en dos partes iguales. Es un parámetro que informa de manera indirecta de la capacidad de la red de interconexión porque un mayor ancho de bisección significa una mayor conectividad entre los nodos de la red. Este concepto representa el cuello de botella al dividir la red en dos partes iguales.

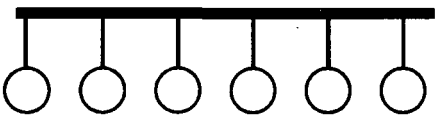
- ✓ **Simetría**: Una red es simétrica u homogénea si todos los nodos de la red tienen la misma visión del resto de la misma.

- ✓ **Expansibilidad**: Una red es fácilmente expansible si se puede incrementar de forma sencilla el tamaño de la red. Por ejemplo, hay redes simétricas en las que para aumentar el número de nodos es preciso aumentar el grado de cada uno de los nodos que la componen. En este caso, no es fácil añadir nodos a una red porque implica cambiar los nodos y enlaces ya existentes. Por otro lado, hay otras redes que aumentar el número de nodos simplemente implica añadir más nodos y conectarlos a algunos de los nodos existentes de manera simple.

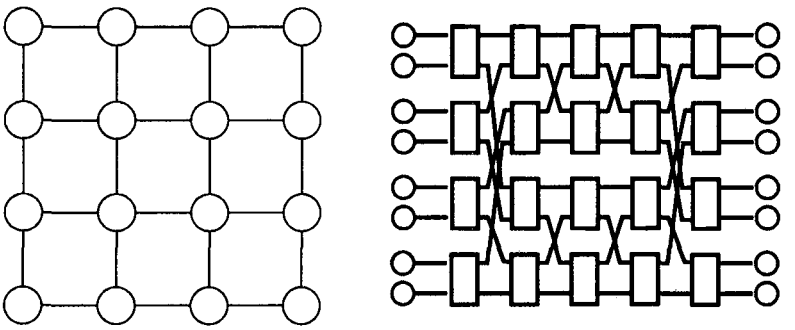
A continuación, se presentan las topologías más comúnmente utilizadas para configurar redes de interconexión.

2.3.1 Topologías más comunes

Las topologías que se utilizan para las redes de interconexión se pueden clasificar en tres categorías básicas [41] según su estructura tal y como se muestra a continuación. Las redes de medio compartido están formadas por un único elemento común de interconexión al que se conectan directamente todos los nodos procesadores de la red. Las redes directas están formadas por un conjunto de encaminadores cada uno de ellos con varios enlaces. Los enlaces se utilizan para conectarlos a otros encaminadores o nodos de procesamiento. Las redes indirectas, a diferencia de las directas, están basadas en conmutadores, cuya función es conectar los enlaces de la entrada con los de la salida. Los nodos de procesamiento se conectan a algunos de los conmutadores. Además de los tipos básicos, podemos considerar las redes híbridas que son combinaciones de los anteriores tipos. La Figura 2-3 muestra la filosofía de cada uno de los tipos básicos de redes mencionados.



(a)medio compartido



(b) red directa (c) red indirecta

Figura 2-3 Tipos básicos de topologias de red

2.3.1.1 Redes de medio compartido

Las redes de medio compartido están formadas por un único elemento común de interconexión al que se conectan directamente todos los nodos procesadores de la red. Estas redes no muy utilizadas en computadores paralelos por sus bajas prestaciones, se pueden clasificar en redes de área local tradicionales o en "*bus*", que es un único medio compartido al que se conectan todos los nodos de la red. La Figura 2-4 muestra dos ejemplos de redes de medio compartido.

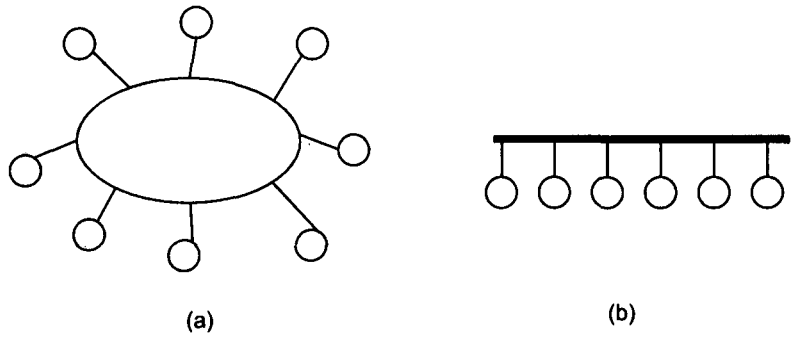


Figura 2-4 Topologias de red de medio compartido

(a) red de área local, (b) Red tipo "*Bus*"

2.3.1.2 Redes directas

Las redes directas están formadas por un conjunto de encaminadores cada uno de ellos con varios enlaces. Los enlaces se utilizan para conectarlos a otros encaminadores o nodos de procesamiento. Las redes directas se caracterizan por la topología que forman los encaminadores, un algoritmo de encaminamiento que determina el camino a seguir entre cada par de nodos y la técnica de control del flujo.

Las redes directas se pueden dividir en redes ortogonales, es decir, simétricas, y en redes no ortogonales [41]. Dentro de las redes ortogonales tenemos las mallas abiertas y los n-cubo k-ario, que son un tipo muy importante de mallas cerradas. Las primeras tienen una forma de cuadrícula rectangular, sea bidimensional o tridimensional, cuyos nodos periféricos no están conectados por el vértice exterior. En la Figura 2-5 se representan topologías tipo malla 2D y 3D. Las mallas 2D son de fácil expansión y de encaminamiento simple.

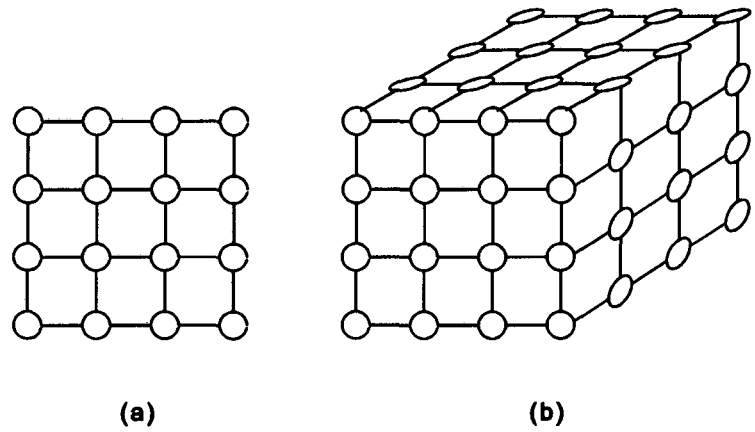


Figura 2-5 Topologias de red directas tipo Malla

(a) malla 2D (b) malla 3D

Las mallas cerradas son las redes en las que sí se conectan los enlaces externos de los nodos periféricos a nodos opuestos de la red de manera que se cierran sobre sí mismas. Estas topologías se suelen nombrar como "*n*-cubo *k*-ario", (o "*k-ary n-cube*", en inglés) donde la *n* hace referencia a la dimensión y la *k* al número de nodos por dimensión. En el caso de tomar constante el valor de n=2, las redes reciben el nombre de toros, y en el caso de fijar la k=2, se llaman hipercubos. Los toros son mallas 2D cerradas por los extremos. El hipercubo es una de las topologías mas empleada y estudiada y presenta unas ciertas propiedades de regularidad y simetría especiales.

En la Figura 2-6 se representan topologías tipo n-cubo k-ario como el anillo, el toro 2D, el hipercubo 3D y el hipercubo 4D.

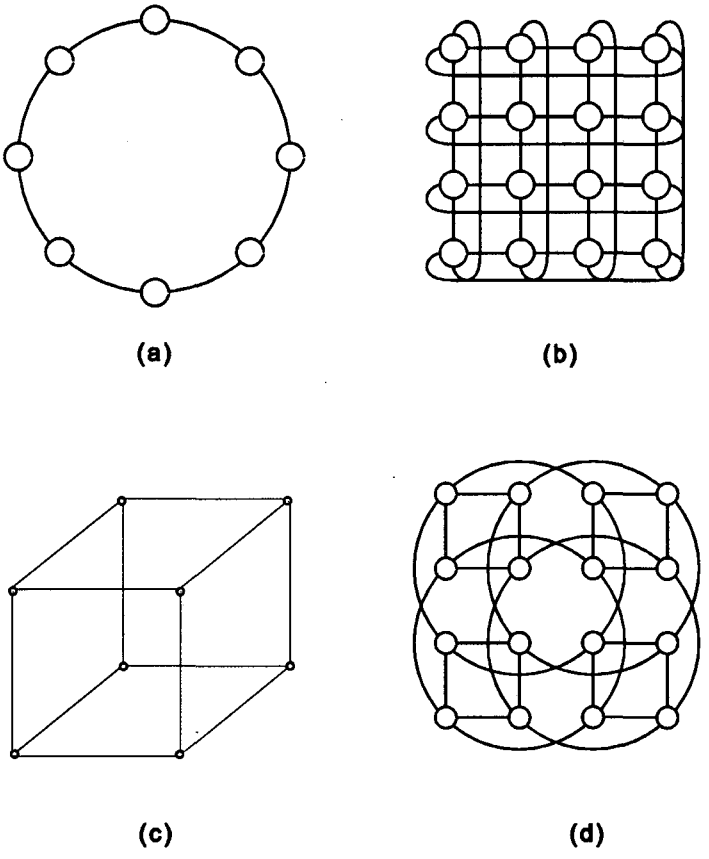


Figura 2-6 Topologias de red directas tipo *n*-cubo *k*-ario

(a) anillo (n=1, k=8), (b) Toro (n=2,k=4)
(c) cubo 3D (n=3,k=2) (d) cubo 4D (n=4,k=2)

Finalmente, dentro de las redes directas tenemos las redes no ortogonales. Existe una gran diversidad de alternativas desarrolladas en numerosos casos específicos que nos imposibilita nombrarlas todas aquí. La Figura 2-7 muestra algunas alternativas como son las topologías anillo con cuerdas, el cubo conectado en ciclos, árbol binario,

"*fat-tree*" y "*midimew*" [12]. El anillo con cuerdas establece enlaces extra entre algunos nodos para facilitar la comunicación entre ellos. El cubo conectado en ciclos es una combinación de anillos insertados en los vértices de un cubo. La topología "*fat-tree*" tiene forma de árbol donde se incrementa la anchura de los enlaces a medida que se acercan a la raíz. Las redes "*midimew*" son un caso especial de mallas cerradas con desplazamientos en los enlaces periféricos.

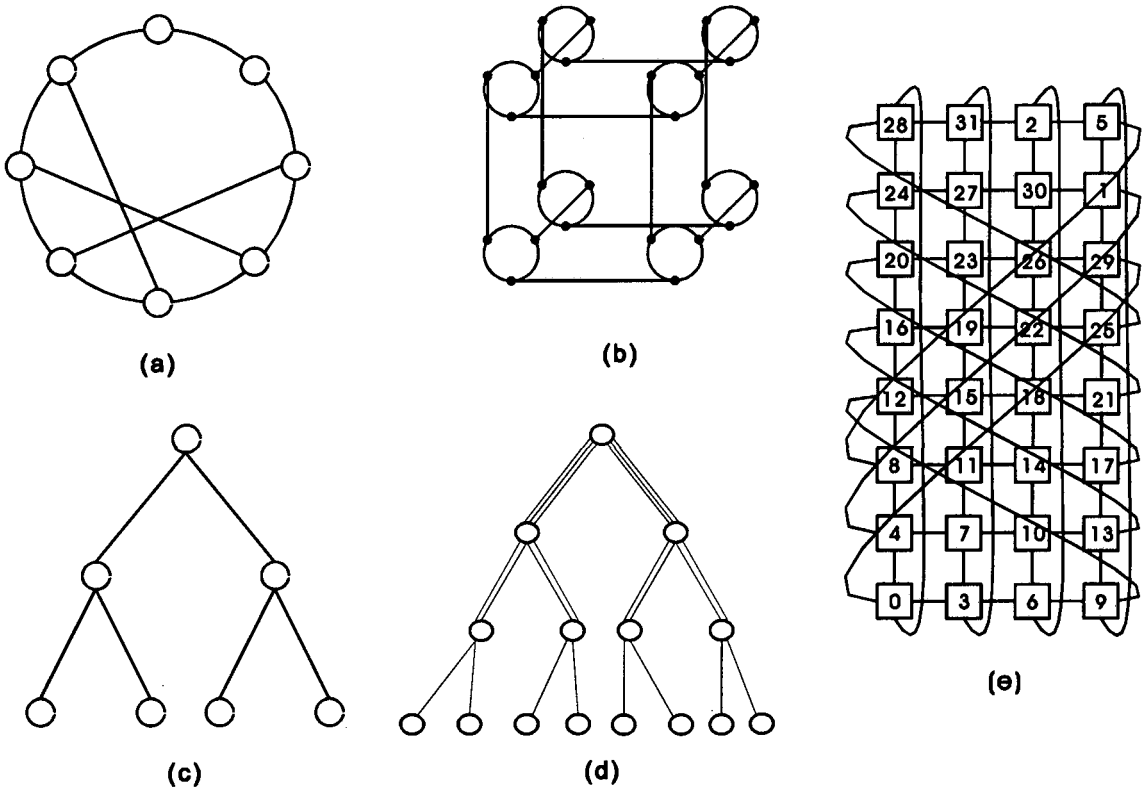


Figura 2-7 Topologias de red directas no ortogonales

(a) anillo con cuerdas, (b) cubo conectado con ciclos, (c) árbol, (d) "*fat-tree*", (e) "midimew"

2.3.1.3 Redes indirectas

Las redes indirectas están formadas por un conjunto de conmutadores, cada uno de ellos con varios enlaces. Los nodos de procesamiento se conectan a algunos de los conmutadores a través de los enlaces. A diferencia de las redes directas, que proveen conexiones directas entre pares de nodos, en las redes indirectas, la comunicación entre nodos de la red se establece a través de varios conmutadores. Cada nodo de la red tiene un adaptador que lo conecta a un conmutador de la red. Cada conmutador tiene una serie de "*ports*". Cada "*port*" esta formado por un enlace de entrada y otro de salida. Los

"*ports*" de cada conmutador se conectan a nodos de la red o a otros conmutadores. La interconexión de tales conmutadores forma la topología de la red. La Figura 2-8 muestra la estructura general de una red indirecta en la que los nodos se interconectan a través de varias etapas de conmutadores.

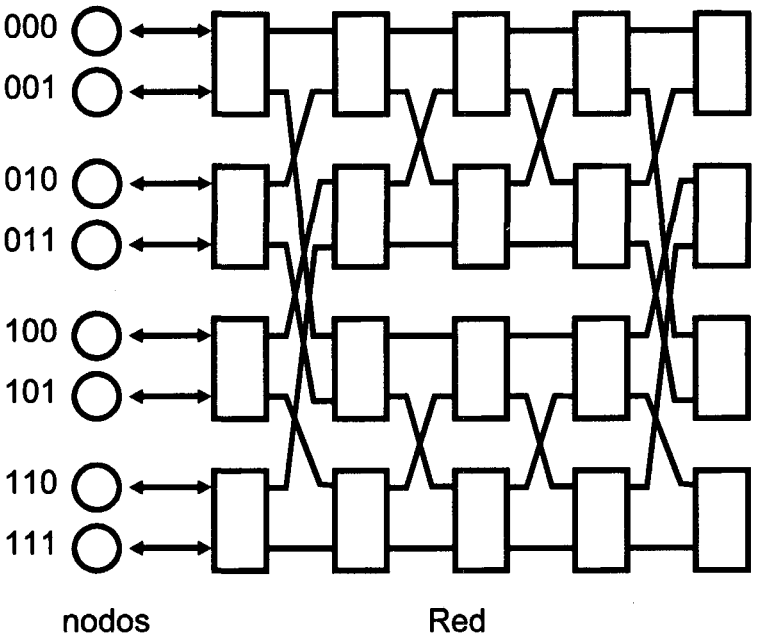


Figura 2-8 Esquema general de una red indirecta

Las redes indirectas han evolucionado considerablemente a lo largo del tiempo. Se han propuesto un gran número de topologías, desde las regulares usadas en "*arrays*" de procesadores y multiprocesadores de memoria compartida UMA, hasta topologías irregulares usadas actualmente en las NOWs. Las topologías regulares se llaman así por tener un patrón regular de conexión entre conmutadores mientras que las irregulares no siguen ningún patrón predefinido. Ambas clases de redes indirectas se pueden clasificar a su vez según el número de conmutadores que los mensajes tienen que atravesar para alcanzar su destino. Aunque este aspecto no es relevante en el caso de redes irregulares, sí que lo es en el caso de redes regulares porque de él se derivan importantes propiedades específicas de cada clase.

Cada conmutador de una red indirecta puede conectarse a cero, una o más procesadores. Obviamente, sólo los conmutadores conectados a algún procesador pueden ser fuente o destino de un mensaje.

Similarmente a las redes directas, una red indirecta se caracteriza por su topología, su algoritmo de encaminamiento y su técnica de control del flujo. La topología define cómo los conmutadores están interconectados mediante los enlaces, y se puede modelar mediante un grafo.

Para redes indirectas con N nodos, la topología ideal sería conectar todos los nodos a través de un único conmutador de N x N "*ports*". Tal conmutador se llama "*crossbar*" y ofrece una conexión completa entre los nodos de la red. Esta topología no es siempre posible debido a limitaciones tecnológicas lo que obliga a utilizar redes multietapa ("*Multi Stage Networks*" ó MINs), una alternativa de menor coste a los "*crossbar*". Aunque usar un "*crossbar*" de N x N es mucho más económico que usar una topología de red directa (que requiere N encaminadores, cada uno con un "*crossbar*" interno N x N), el coste es prohibitivo para redes grandes.

Similarmente a las redes directas, el número de conexiones físicas de un conmutador está limitado por condiciones físicas como el número de "*pins*" y de área de conexionado disponibles. Estas dificultades imposibilitan el uso de redes "*crossbar*" para redes de tamaño grande. Es por esta razón que se han propuesto topologías alternativas. En estas topologías, los mensajes han de atravesar diversos conmutadores antes de llegar al nodo destino. En redes regulares, estos conmutadores suelen ser todos iguales y tradicionalmente se han organizado en una serie de *etapas*. Cada etapa, excepto las de entrada y salida, está conectada a solamente a la etapa anterior y posterior siguiendo un patrón de conexión regular. Estas redes se llaman redes *multietapa ("Multistage Networks", MINs)*, y tienen diferentes propiedades dependiendo del número de etapas, y la conexión entre ellas. A continuación se comentan los "crossbar" y las MINs.

2.3.1.3.1 Las redes "crossbar"[41]

Las redes "crossbar" permiten a cualquier procesador del sistema conectarse a cualquier otro procesador o unidad de memoria de manera que cada procesador puede comunicar simultáneamente sin contención. Una nueva conexión se puede establecer en cualquier momento siempre que los "*ports*" de entrada y salida estén libres.

Las redes "crossbar" se utilizan en el diseño de multiprocesadores de altas prestaciones de pequeña escala, en el diseño de encaminadores para redes directas y como componentes básicos en el diseño de redes indirectas de gran tamaño. Un "*crossbar*" se define como una red de conmutación con N entradas y M salidas, la cual permite hasta $min\{N,M\}$ conexiones uno a uno simultáneas sin contención. La Figura 2-9 muestra una red "*crossbar*" N x M. Generalmente, N=M excepto en redes "crossbar" que se utilizan para conectar nodos procesadores y módulos de memoria, donde el número de cada uno de ellos puede variar.

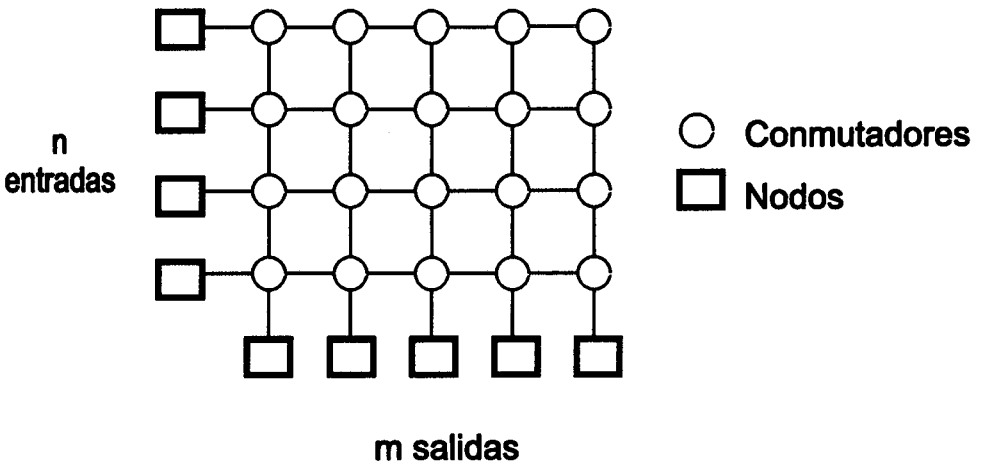


Figura 2-9 Topologias de red tipo "crossbar"

2.3.1.3.2 Redes Multietapa (MIN)

Las redes multietapa (MINs) conectan los dispositivos de entrada a los de salida a través de una serie de etapas de conmutadores, donde cada conmutador es un "*crossbar*". El número de etapas y los patrones de conexión entre las etapas determinan la capacidad de encaminamiento de estas redes. Dependiendo del esquema de interconexión utilizado entre dos etapas adyacentes, se han propuesto diversas MINs. Las MINs son adecuadas para construir computadores paralelos con cientos de procesadores y se han utilizado en algunos sistemas comerciales.

Las redes MIN pueden clasificarse a su vez en bloqueantes, no bloqueantes y reconfigurables dependiendo de la disponibilidad de caminos para establecer nuevas conexiones. En las redes bloqueantes, una conexión entre un "*port*" de entrada con otro de salida de un conmutador, ambos libres, no siempre se puede establecer según las conexiones existentes entre otros "*ports*" del conmutador. Típicamente, existe un único camino entre cada par de entrada y salida, de manera que se minimizan el número de etapas y de conmutadores.

En las redes no bloqueantes, cada entrada se puede conectar a cualquier salida de manera independiente de las demás. Tienen la misma funcionalidad que un "*crossbar*" en cada conmutador, lo que requiere múltiple caminos entre cada entrada y salida, lo que implica un incremento de las etapas. Las redes no bloqueantes tienen un coste elevado. Aunque, son más económicas que un "*crossbar*" del mismo tamaño, su coste es prohibitivo para tamaños grandes. El mejor ejemplo de red multietapa no bloqueante es la red de Clos [27], propuesta inicialmente para redes telefónicas.

En las redes reconfigurables, cualquier entrada se puede conectar a cualquier salida libre, pero ello puede requerir reconfigurar las conexiones existentes en el conmutador. Estas redes también requieren múltiples caminos entre cada entrada y salida de la red, pero el número de caminos y el coste es menor que el caso de redes no bloqueantes.

Dependiendo del tipo de enlaces y conmutadores utilizados, las redes MIN se pueden clasificar en unidireccionales y bidireccionales. En las redes MIN unidireccionales los enlaces y conmutadores son unidireccionales, es decir, sólo permiten la comunicación en un sentido. En las redes bidireccionales los enlaces y conmutadores son bidireccionales. Esto implica que la información se puede transmitir simultáneamente en direcciones opuestas entre conmutadores vecinos. Dentro de un conmutador permiten la conexión hacia un sentido, por ejemplo de derecha a izquierda, hacia el sentido contrario, de izquierda a derecha, o un cambio de sentido, entrando por la derecha y volviendo a salir por el mismo lado, por ejemplo.

Desde un punto de vista práctico, es interesante que todos los conmutadores sean idénticos para amortizar el coste del diseño. Las redes Banyan son una clase de redes MIN con la propiedad de que hay un único camino entre cada entrada y salida [64]. Una red Delta de N nodos ($N=k^n$) es una subclase de red Banyan, construida con conmutadores k x k idénticos en n etapas, donde cada etapa contiene N/k conmutadores. Muchas de las MINs más conocidas, tales como "*Omega*", "*cube*", "*butterfly*" y "*baseline*" pertenecen a las redes Delta [102] y se ha demostrado que son topológicamente y funcionalmente equivalentes [135]. Un buen informe de estas MIN se puede encontrar en [119].

2.3.1.3.2.1 Redes MINs Unidireccionales

Los bloques básicos que constituyen las redes MIN unidireccionales son los conmutadores unidireccionales. Un conmutador a x b es una red "crossbar" con a entradas y b salidas. Se puede demostrar que con N "*ports*" de entrada y salida, una red MIN unidireccional con conmutadores k x k requiere al menos $\log_k (N)$ etapas para permitir un camino de conexión entre cualquier "*port*" de entrada y de salida. Teniendo etapas adicionales, se permiten más caminos entre entradas y salidas a expensas de un coste extra. Cada camino de la MIN cruza todas las etapas, por lo tanto, todos los camino son de la misma longitud. La Figura 2-10 muestra las diferentes alternativas de conexión interna de un conmutador unidireccional.

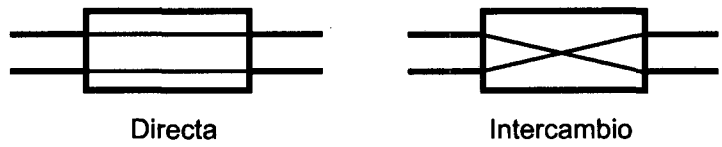


Figura 2-10 Conexiones del conmutador unidirecional

La Figura 2-11 muestra la topología de cuatro redes MIN de 16 x 16: (a) "*baseline*", (b) "*butterfly*", (c) "*cube*" y (d) "*Omega*", donde la topología viene definida por el patrón de enlace entre etapas de conmutadores.

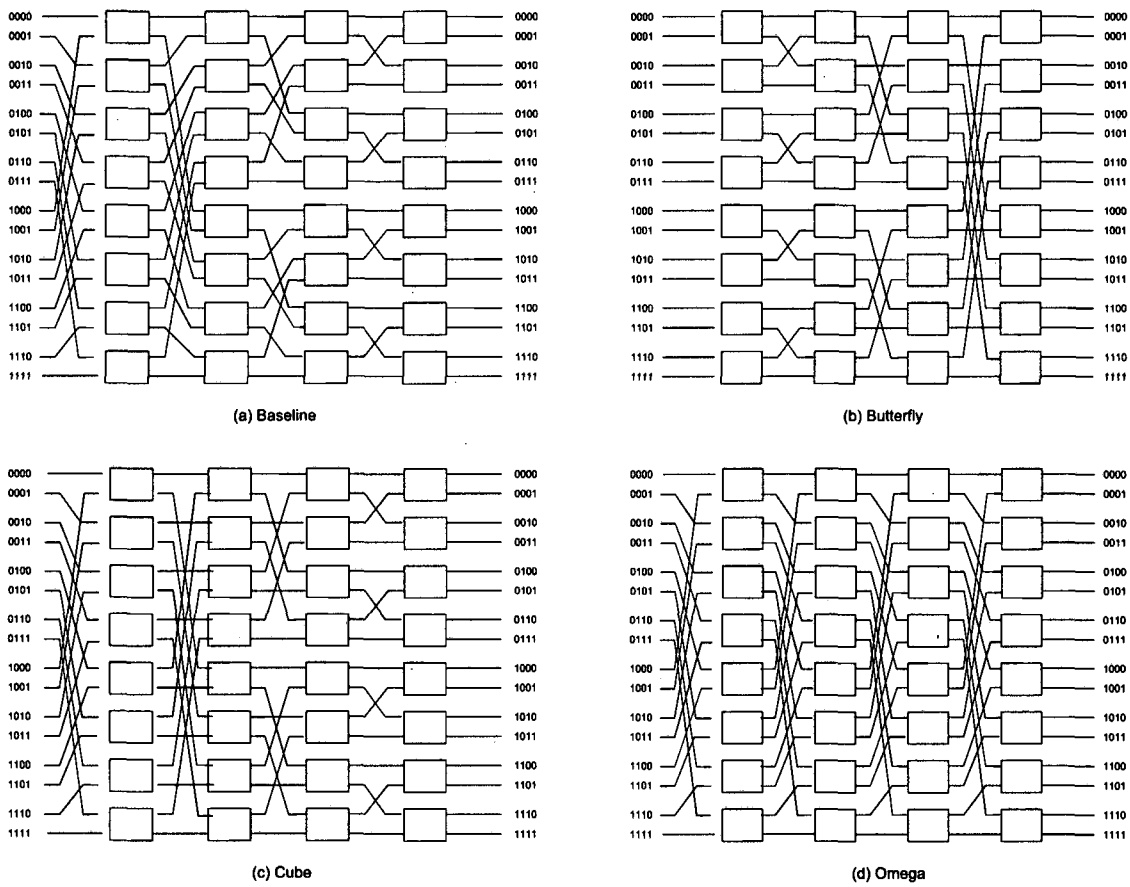


Figura 2-11 Topologias MINs unidireccionales

2.3.1.3.2.2 Redes MINs bidireccionales

Un conmutador bidireccional soporta tres tipos de conexiones: adelante, atrás y giro en redondo. La Figura 2-12 muestra las diferentes alternativas de conexión interna de un conmutador bidireccional.

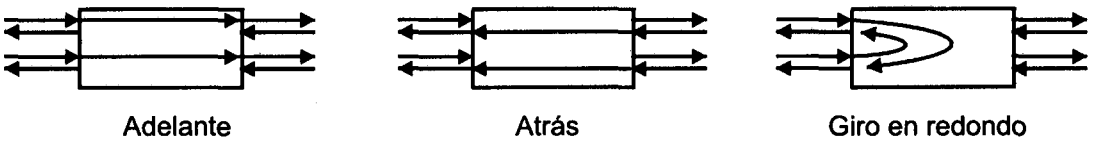


Figura 2-12 Conexiones del conmutador bidirecional

El hecho de permitir conexiones giro en redondo entre "*ports*" del mismo lado del mismo conmutador implica que los caminos no son todos de la misma longitud, ya que permite no atravesar todas las etapas de la red. La Figura 2-13 muestra una MIN bidireccional "*butterfly*" de ocho nodos.

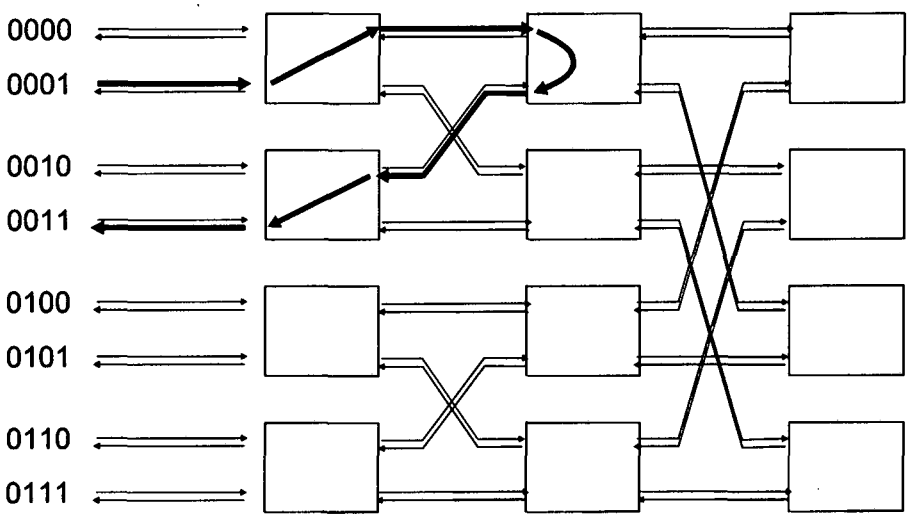
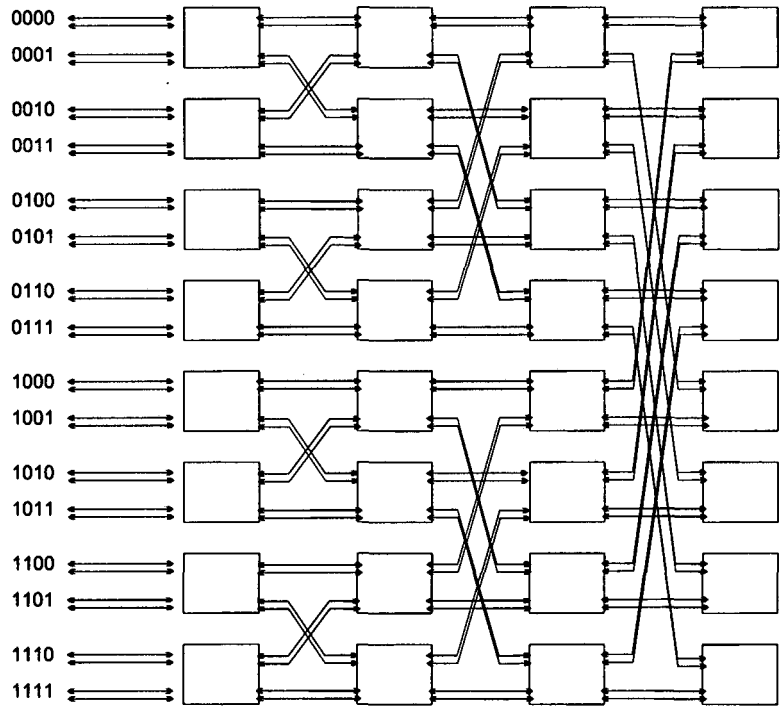
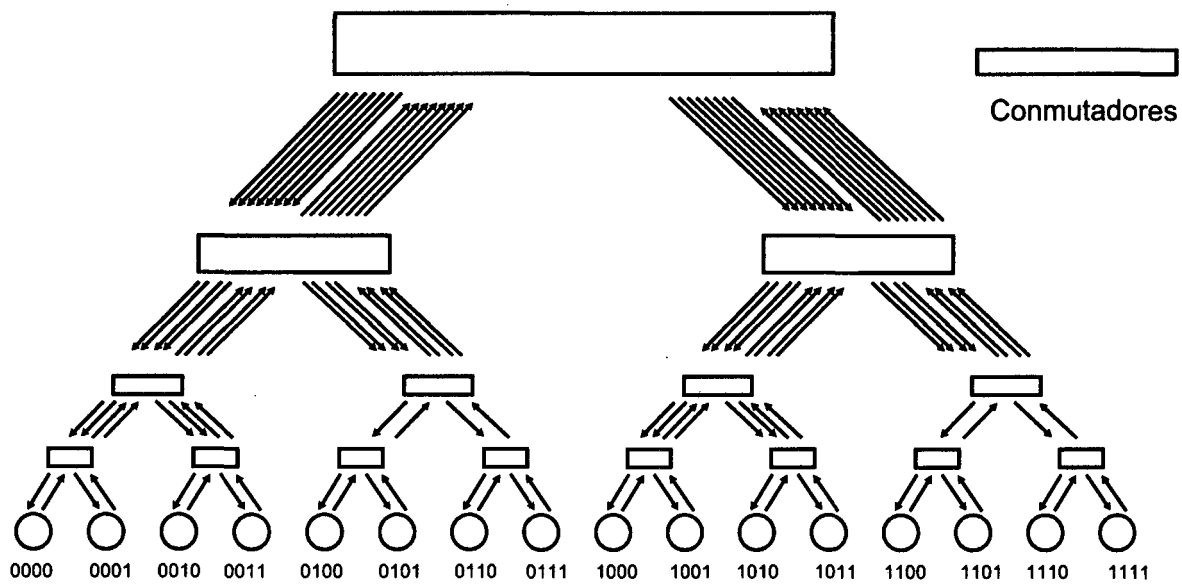


Figura 2-13 Topología MIN bidirecional tipo "*butterfly*"

Tal y como muestra la Figura 2-14, una MIN bidireccional "*butterfly*", con encaminamiento vuelta atrás puede verse como un "*fat tree*" [82]. En un "*fat tree*", los nodos están colocados en las hojas, y los vértices internos son los conmutadores. El ancho de banda de transmisión entre conmutadores se incrementa añadiendo más enlaces en paralelo a medida que los conmutadores están más cercanos a la raíz del árbol. Cuando un mensaje se encamina de un procesador a otro, se envía hacia arriba (en dirección hacia delante) en el árbol hasta el antecesor común más cercano de ambos procesadores, y entonces se envía hacia abajo (en dirección hacia atrás) hasta el destino. Tal encaminamiento en árbol explica el encaminamiento vuelta atrás mencionado.



(a) Red MIN bidireccional "butterfly" de 16 nodos construida con conmutadores 2x2



(b) Fat tree de 16 nodos

Figura 2-14 Equivalencia entre MIN butterfly bidirecional y "*fat tree*"

Hasta aquí se ha presentado la clasificación clásica de las redes de interconexión en las dos clases más importantes para los computadores paralelos que son las redes directas e indirectas, aparte de las de medio compartido y las híbridas. Actualmente, sin embargo, los avances tecnológicos han hecho converger ambos tipos de redes, directo e indirecto, de manera que la diferencia se basa en una cuestión topológica.

Tradicionalmente, las redes directas usaban encaminadores con capacidad de tomar decisiones de encaminamiento en cada nodo de la red, mientras que las redes indirectas usaban encaminamiento determinado en la fuente. Sin embargo, los avances actuales en arquitectura de los encaminadores, de los algoritmos de encaminamiento y en el diseño de los conmutadores hacen que los encaminadores y los conmutadores converjan a un mismo dispositivo y sea la manera cómo lo usa el diseñador de la red, el que determine se construye una red directa o indirecta. Ejemplos de tales dispositivos son el C104 de Inmos [88] o el SGI SPIDER [56]. La diferencia estriba, entonces, en si se conecta al menos un nodo de cómputo a cada encaminador/conmutador, en cuyo caso tendríamos una configuración de red directa, o, por el contrario, se configuran algunos encaminadores/conmutadores a los que sólo se les conectan otros encaminadores/conmutadores de manera que existen diversas etapas antes de alcanzar algún nodo de cómputo, en cuyo caso se tiene una red indirecta.

Finalmente, la Tabla 2-1 muestra, sobre la base de las clasificaciones ya descritas, algunas de las diferentes topologías más utilizadas en sistemas comerciales de entre los tipos analizados en los puntos anteriores.

2.4 Control del flujo

Como ya se señaló en la introducción, el control del flujo de la información en la red es uno de los aspectos que la definen. El control del flujo determina cómo se hace avanzar la información desde el nodo origen hasta el nodo destino. Concretamente, se ocupa de definir los dos aspectos siguientes:

1. Cómo se establece el camino entre el nodo fuente y el destino.
2. Cómo se regula el avance de la información en cada nodo de la red (Control de flujo).

A continuación se analizan estos dos aspectos mencionados.

2.4.1 Establecimiento del camino entre dos nodos a través de la red

Este aspecto hace referencia a la formación y establecimiento del camino entre nodo fuente y destino. Para ello existen dos técnicas posibles, que presentan aproximaciones opuestas. La primera es conmutación de circuitos, en la que hay un preestablecimiento del camino anterior al envío de información, y la segunda es conmutación de mensajes o paquetes, donde el camino se construye a medida que la información avanza sobre la red de interconexión.

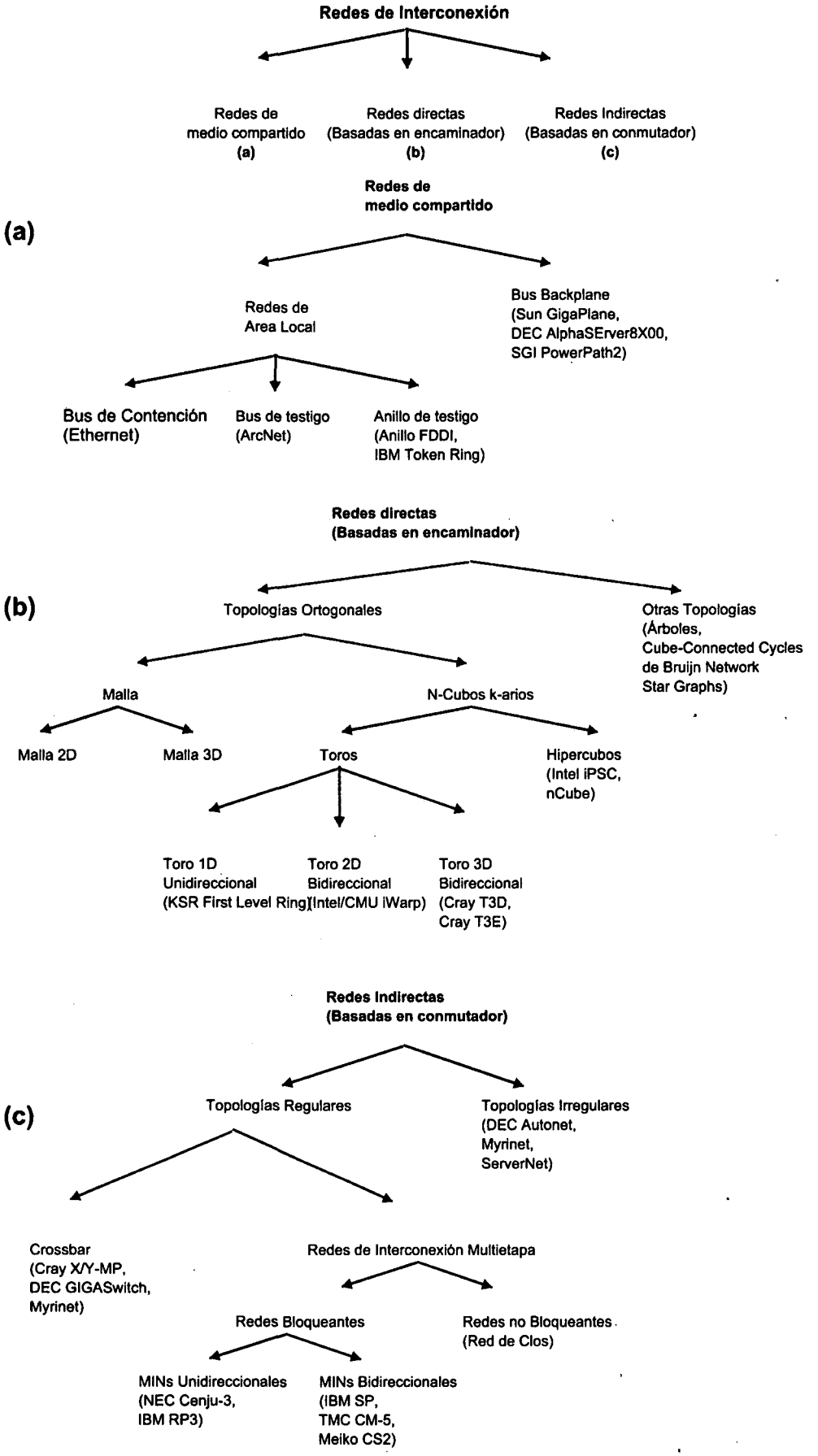


Tabla 2-1 Clasificación y ejemplos de topologías de redes de interconexión

2.4.1.1 Conmutación de circuitos (Circuit Switching)

En esta técnica, para enviar un mensaje, un nodo debe establecer previamente un camino a lo largo de la red de comunicación que lo conecte con el nodo receptor. Una vez formado este camino, que durante la transmisión quedará exclusivamente utilizado por la pareja de nodos en cuestión, la transmisión de datos se produce a la máxima velocidad que permita el ancho de banda. Para establecer el camino, se envía una sonda, llamada cabecera, que se va abriendo paso y va reservando los enlaces hasta llegar al nodo destino.

Una propiedad importante de la conmutación de circuitos es la necesidad de establecer una ruta de fuente a destino antes de que cualquier conjunto de datos pueda ser enviado. El tiempo transcurrido desde que se inicia la comunicación hasta que se puede enviar el primer dato puede ser considerable. Durante este intervalo de tiempo, el sistema de comunicaciones se encuentra en la etapa de búsqueda de un camino físico mediante la reserva de una serie de enlaces. Este tiempo se llama retardo de la cabecera. Estos enlaces se usarán de manera exclusiva durante todo el tiempo de comunicación, llamado retardo de la red.

Como consecuencia de la búsqueda de la ruta entre los nodos fuente y destino, una vez que se ha completado el establecimiento de la conexión, el único retardo en la transmisión de los datos es el tiempo de propagación de la señal. La Figura 2-15 muestra un diagrama de tiempos de los eventos que suceden durante la transmisión de un mensaje mediante conmutación de circuitos.

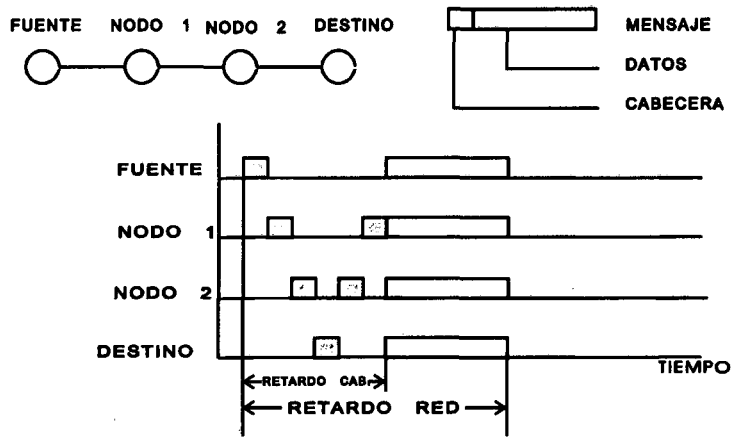


Figura 2-15 Conmutación de circuitos

Otra consecuencia de dicho establecimiento es que no hay peligro de colisión con otros mensajes una vez abierto el camino, es decir, una vez que se lleva a cabo la conexión, nunca se tendrán conflictos, aunque podría aparecer congestión antes de que

se llegue a establecer la conexión, debido a la falta de capacidad del sistema de comunicaciones. Esta es la técnica clásica que se usa en redes telefónicas. No obstante, no es muy utilizada en comunicación en redes de ordenadores ya que ofrece bajas prestaciones [44] [127].

2.4.1.2 Conmutación de mensajes y de paquetes (Message y Packet Switching):

Existe otra estrategia llamada conmutación de mensajes. Cuando se utiliza esta forma de conmutación, no hay un establecimiento anticipado de la ruta entre el que envía y el que recibe. En su lugar, cuando el nodo que envía tiene listo un bloque de datos, éste se almacena en el primer nodo, para reexpedirse después dándose sólo un paso cada vez. Cada bloque se recibe íntegramente, se revisa en busca de errores, y se retransmite con posterioridad. Los enlaces solo se usan en el momento físico de paso de la información de un nodo al siguiente. El avance de un mensaje a través de cuatro nodos usando conmutación de mensajes se muestra en la Figura 2-16. El tiempo de comunicación o retardo en la red es el producto del tiempo de transmitir el mensaje de un nodo a otro por el número de nodos que atraviesa.

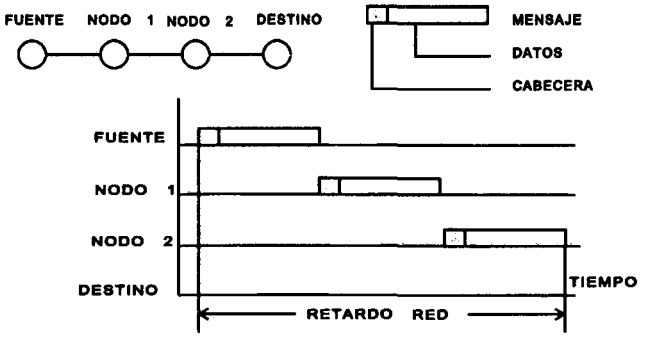


Figura 2-16 Conmutación de mensajes

Todavía hay otra posibilidad para realizar la conmutación, y es la llamada conmutación de paquetes. Con la conmutación de mensaje no existe ningún límite para el tamaño del bloque, lo que significa que los nodos deben tener espacio para almacenar temporalmente bloques grandes. Esto también significa que sólo un bloque de tamaño muy grande puede ocupar una línea entre dos nodos durante un tiempo grande, inutilizando la conmutación para otros mensajes, los cuales sufrirían esperas muy altas.

A diferencia de esto, las redes de conmutación de paquetes fijan un límite superior en el tamaño del bloque, permitiendo que los paquetes sean almacenados en "*buffers*" en el nodo. Por lo tanto, se mejora el tiempo de respuesta por mensaje si éstos se fraccionan en un efecto similar a la planificación de tareas en un Sistema Operativo.

Otra ventaja es que el primer paquete de un mensaje multipaquete puede reexpedirse antes de que el segundo haya llegado por completo, reduciendo la latencia y mejorando el rendimiento. La Figura 2-17 muestra un diagrama de tiempos para la conmutación de paquetes.

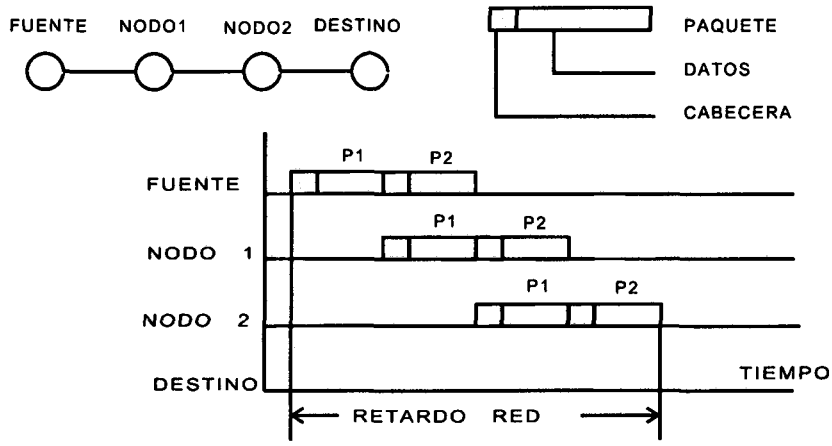


Figura 2-17 Conmutación de paquetes

El paquete va abriéndose camino entre el nodo origen y destino por la red a medida que avanza por ella, es decir, que la comunicación progresa nodo a nodo sin que exista simultáneamente una conexión abierta entre todos los nodos del camino que recorre el mensaje. Esto posibilita que otros paquetes se crucen simultáneamente por el camino que esta recorriendo el mensaje.

Dado que el paquete no tiene asegurado el camino, se debe implementar algún tipo de control de flujo que permita almacenar temporalmente el mensaje cuando un enlace de salida del nodo se encuentre ocupado [44] [127].

La conmutación de circuitos y la conmutación de paquetes difieren en muchos aspectos. La diferencia principal es que la conmutación de circuitos reserva, de forma estática y anticipada, el ancho de banda necesario, en tanto que la conmutación de paquetes lo adquiere según se necesita. Con la conmutación de circuitos, cualquier ancho de banda, que no se utilice en un circuito asignado, se desperdicia.

En la conmutación de paquetes, y debido a que los circuitos nunca están dedicados a una tarea especial, éstos pueden ser utilizados por paquetes de diferentes orígenes o con diferentes destinos. Sin embargo, y precisamente porque los circuitos no están dedicados a una tarea especial, la aparición de una sobrecarga repentina en el tráfico de entrada puede llegar a saturar un nodo, por ejemplo, excediendo su capacidad

de almacenamiento y ocasionando la pérdida de paquetes. Para que esto no ocurra se necesita realizar algún tipo de control del flujo de paquetes.

La conmutación de paquetes, debido a sus características, es la técnica mas utilizada en redes de interconexión de computadores paralelos por lo que ahora nos vamos a centrar en las técnicas de control del flujo para esta técnica.

2.4.1.2.1 Control de flujo de los paquetes en la conmutación de paquetes

El control de flujo debe asegurar que un nodo no quede desbordado por los mensajes que recibe. Es necesario establecer una política de control de flujo que especifique el modo en que los mensajes fluyen hacia sus destinos evitando el desbordamiento de las colas de mensajes de los nodos intermedios. Las técnicas más habituales son: "*store-and-forward*", "*wormhole*", "*virtual cut-through*" y "*mad-postman*" [20].

Antes de analizar individualmente cada una de estas técnicas, vamos a definir los parámetros comunes que nos permitirán evaluar la latencia en cada una de ellas [76]:

- L es la longitud de la información a transmitir en bits
- W es la longitud de un "*flit*" en bits. Un "*flit*" es la menor unidad de un mensaje que una cola o un enlace puede aceptar o rechazar y, por lo tanto, la cantidad de información que debe ser almacenada en un nodo intermedio antes de ser transmitido al siguiente nodo.
- D es el número de enlaces atravesados
- tr es el tiempo para realizar una decisión de encaminamiento
- ts es el tiempo que tarda un "*flit*" en atravesar un encaminador
- tw es el tiempo que tarda un "*flit*" en atravesar un enlace

En el caso de que la información de cabecera ocupe un "*flit*", entonces, la longitud del paquete es la suma de la longitud de la información más la cabecera, es decir, L+W.

2.4.1.2.1.1 "Store-and-forward" [35]

En esta técnica, cada paquete se almacena completamente en cada nodo y entonces se transmite al siguiente nodo cuando el enlace de salida está libre. La Figura 2-18 muestra el avance de un paquete a lo largo de un camino mediante la técnica de

control de flujo "*Store-and-forward*". La principal desventaja de un método como éste es su alta latencia en el transporte de un paquete individual de un nodo a otro. Esta latencia es proporcional al producto de la longitud del paquete por el número de enlaces atravesados. La latencia sin colisiones es, por lo tanto,

$$\text{Latencia Store-and-forward (sin colisiones)} = D*(tr+(ts+tw)*\left\lceil \frac{L+W}{W} \right\rceil),$$

dónde $\left\lceil \frac{L+W}{W} \right\rceil$ es la longitud de paquete en "*flits*". Esta técnica impone altos requerimientos sobre el mínimo espacio de "*buffer*" en cada nodo, ya que precisa almacenar como mínimo un paquete, es decir, L+W, siendo necesario aumentar el número de paquetes almacenados para aumentar las prestaciones o eliminar problemas de "deadlock", como se verá más adelante.

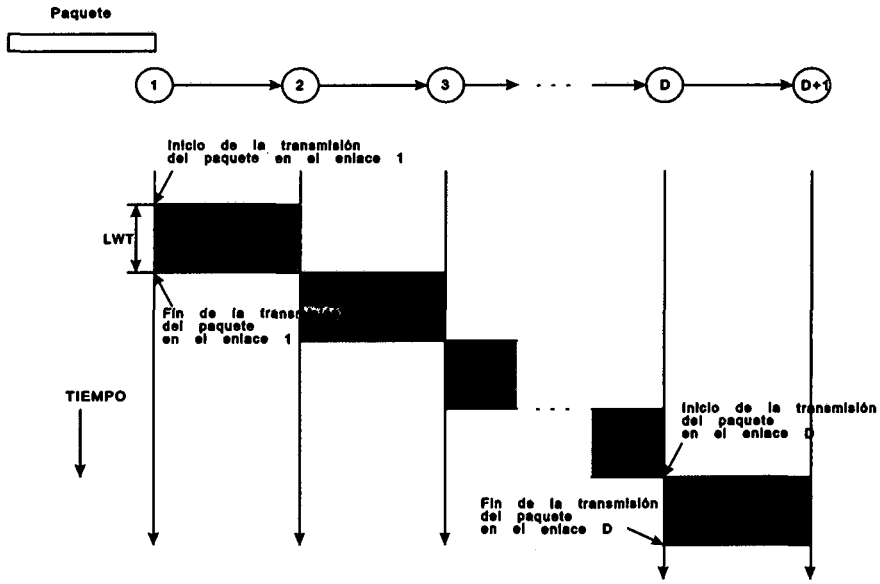


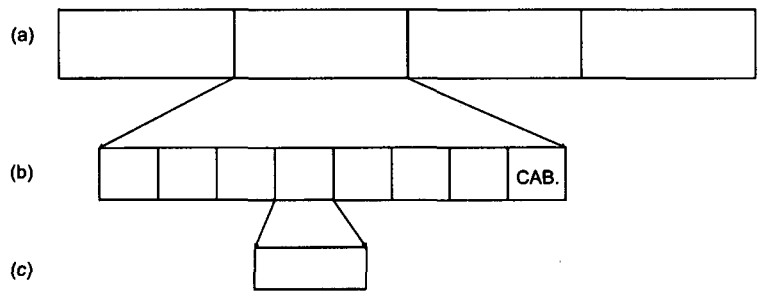
Figura 2-18 Control de flujo "*Store and Forward*"

2.4.1.2.1.2 "Wormhole" [30]

En este tipo de control del flujo, el mensaje se descompone en una secuencia de "*flits*" ("*flow control digits*"), es decir, el mensaje total (cabecera mas cuerpo del mensaje) se divide en unidades más pequeñas y cada una de éstas es un "*flit*". La cabecera, donde se incluye la información de encaminamiento puede ocupar uno o varios "*flits*".

En la Figura 2-19 se muestra la división del mensaje en paquetes y estos en "*flits*". Los mensajes son estructuras de datos de longitud arbitraria. Los paquetes son estructuras de longitud fija y cada uno de ellos tiene información completa de cabecera

(cab.) sobre el destino. Los "*flits*" son pequeñas particiones de tamaño fijo de los paquetes sin información de encaminamiento incluida.



(a) mensaje, (b) paquete, (c) "*flit*"

Figura 2-19 División del mensaje

La Figura 2-20 muestra la técnica de "*wormhole*". La técnica opera de la siguiente forma. Tan pronto como un nodo recibe el "*flit*" de cabecera de un mensaje, se identifica el enlace de salida y los "*flits*" son enviados tan pronto como son recibidos. El camino creado por el "*flit*" cabecera será seguido por todos los "*flits*" restantes del paquete. Por lo tanto, en un momento dado, el mensaje esta repartido por los enlaces que unen fuente y destino. De hecho, se esta haciendo un "*pipeline*" de la comunicación. Es posible que los primeros "*flits*" del mensaje lleguen antes de que la cola haya abandonado el nodo fuente.

A causa de que los "*flits*" que componen un mensaje, excepto el "*flit*" de cabecera, no contienen información del destino, no pueden ser intercalados con los "*flits*" de otros mensajes. Así, cuando un "*flit*" de cabecera se bloquea, todos los "*flits*" del mensaje dejan de avanzar y bloquean el progreso de todos los mensajes que necesitan los enlaces que están ocupando.

Esta técnica es capaz de reducir significativamente la latencia de los mensajes. La latencia sin mensajes en la red es:

$$\text{Latencia Wormhole (sin colisiones)} = D^*(tr+ts+tw)+\max(ts,tw)^* \left\lceil \frac{L}{W} \right\rceil,$$

donde D*(tr+ts+tw) es el tiempo de transmitir el "*flit*" de cabecera por todos por enlaces, y $\max(ts, tw)^* \left\lceil \frac{L}{W} \right\rceil$ es el tiempo de transmitir el resto del paquete, que es independiente del número de enlaces que atraviese. Si la longitud del paquete en "*flits*", $\left\lceil \frac{L}{W} \right\rceil$, es grande con relación a la distancia recorrida D, la latencia en el caso de

"*wormhole*" se podría aproximar por el segundo sumando de la expresión presentada, resultando un valor independiente de la distancia recorrida.

La necesidad de almacenamiento temporal es mínima, debido a que los enlaces sólo deben ser capaces de almacenar un "*flit*" de W bits, que suele ser una cantidad de unos pocos bits. Por otro lado, el hecho de partir el mensaje en "*flits*" aumenta el numero de transferencias por la red lo que incrementa el "*overhead*" de la comunicación, ya que en un momento dado se están usando más enlaces que en el caso del "*store-and-forward*", y cada transmisión requiere unos tiempos de inicialización y sincronización añadidos a la pura transmisión de la información.

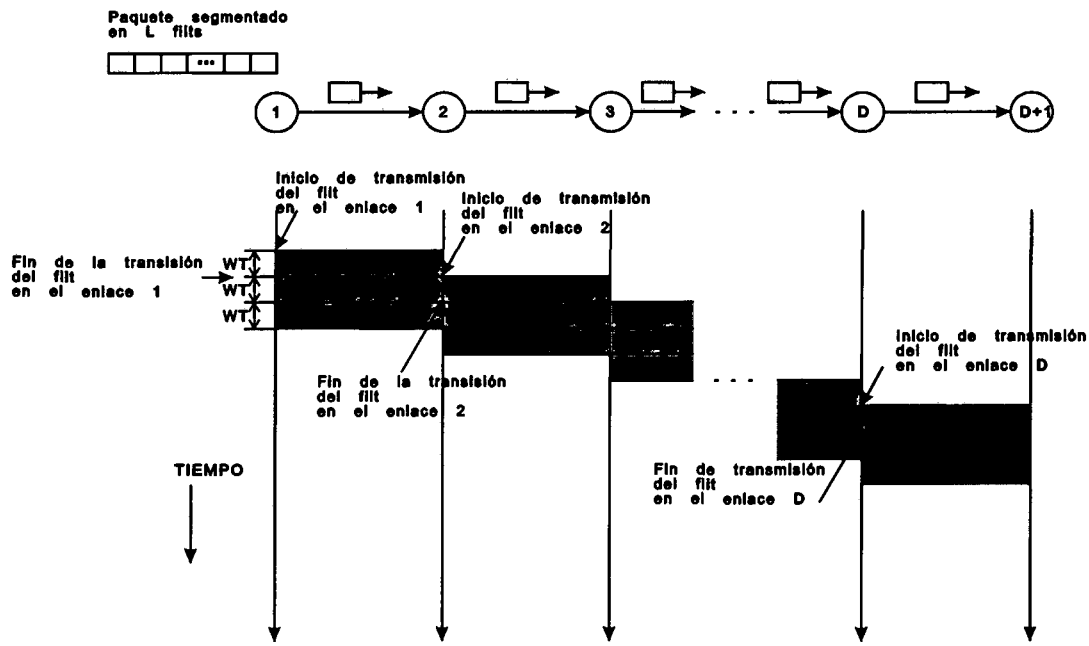


Figura 2-20 Control de flujo "*wormhole*"

2.4.1.2.1.3 "Virtual cut-through" [78]

Es un método similar al de "*Wormhole*". El mensaje se hace avanzar dividido en "*flits*" a lo largo de los enlaces de la red de interconexión de la misma manera como lo hace "*Wormhole*". Difiere del método "*Wormhole*" en el caso en que la cabecera de un mensaje se bloquea en un nodo. En este caso, el resto del mensaje no queda repartido por la red ocupando los "*buffers*" en los que se encontraba cada "*flit*", si no que los "*flits*" se hacen avanzar hasta el nodo en que se encuentra el "*flit*" cabecera bloqueado y se almacenan sobre ese nodo, quitando el mensaje de la red. El mensaje es almacenado, liberando los enlaces, hasta que puede continuar su camino. Así se mejora el rendimiento respecto a los nodos saturados y, en general, incrementa el "*throughput*". La latencia en ausencia de colisiones es, por tanto, la misma que en el caso de "*wormhole*",

Latencia Virtual cut-through (sin colisiones) = D*(tr+ts+tw)+max(ts,tw)* $\left\lceil \frac{L}{W} \right\rceil$.

La Figura 2-21 muestra una comparación de las técnicas "*Wormhole*" y "*virtual cut-through*" cuando un paquete se bloquea. El "*flit*" cabecera se bloquea en el nodo 3, en el caso de "*Wormhole*" todos los "*flits*" restantes del paquete se bloquean en el mismo instante en el nodo en el que se encontraban en ese momento, ocupando los enlaces de los nodos 1 y 2 también; en el control de flujo "*virtual cut-through*" los restantes "*flits*" avanzan hacia el nodo 3, almacenándose en este nodo, hasta que desaparece el bloqueo y el "*flit*" de cabecera prosigue su camino.

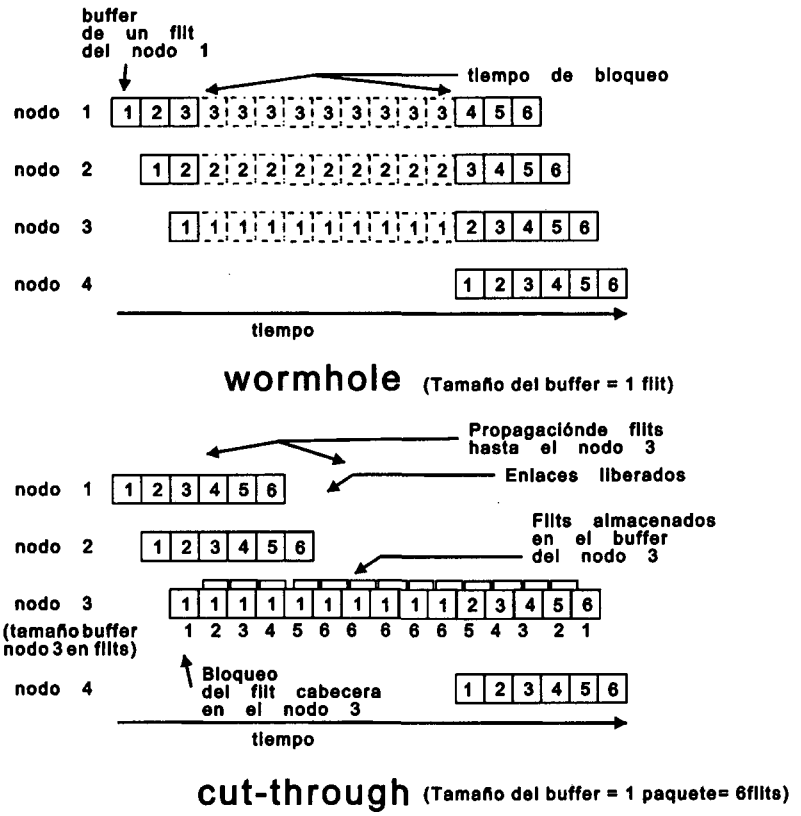


Figura 2-21 Comparación entre "*wormhole*" y "*virtual cut-through*"

2.4.1.2.1.4 "Mad postman" [137]

En este tipo de control del flujo, cada paquete se transmite también en una serie de "*flits*", como en el caso de "*wormhole*". La diferencia con este método es que el primer "*flit*" de un mensaje que entra en un nodo intermedio es enviado por la misma dirección por la que está entrando tan pronto como su primer bit se recibe, y la decisión de encaminamiento se hace después. Es un método de predicción especulativa en la que se hace una suposición en avance sobre la dirección que tomarán los mensajes.

Por ejemplo, supongamos un nodo con cuatro enlaces de salida distribuidos geográficamente en los cuatro puntos cardinales. Entonces, un paquete entrando por el Sur sería enviado por el enlace del Norte. Esto requiere obviamente un "*hardware*" de encaminamiento adecuado. Una vez que la dirección de destino ha sido completamente almacenada, es decir, se han recibido el "*flit*" o "*flits*" de cabecera (en el caso de que la cabecera ocupe más de un "flit"), se calcula el enlace por el cual el paquete tiene que ser enviado y pueden presentarse tres situaciones:

1. La transmisión tiene que seguir por el mismo enlace ya seleccionado, enviando los "*flits*" que siguen.

2. La transmisión tiene que ser parada puesto que el mensaje ya ha alcanzado su destino.

3. El mensaje tiene que ser encaminado en otra dirección.

En los casos 2 y 3 anteriores, en los que la transmisión del resto del paquete se aborta, los "*flits*" de dirección que ya han sido enviados continúan corriendo por la red hasta que son descargados (eliminados) por el propio sistema de comunicaciones cuando son detectados como una cabecera de paquete sin cuerpo. Esto ocurre cuando alcanzan el límite de la red o se bloquean por trafico de mensajes en algún nodo intermedio de la red, en el caso de redes toroidales.

La latencia en el caso *MadPostman* es

Latencia MadPostman (sin colisiones)= D*(ts+tw)+max(ts,tw)*W+max(ts,tw)*L,

donde D*(ts+tw)+max(ts,tw)*W es el tiempo para transmitir el "*flit*" cabecera y max(ts,tw)*L es el tiempo para transmitir el resto del paquete. Esta expresión asume que el enlace es de anchura un solo bit, que es el caso más favorable para esta técnica. El tiempo de encaminamiento *tr* no aparece en la fórmula ya que la decisión de encaminamiento se realiza en paralelo junto con la transmisión.

La desventaja de este método es la posible transmisión de los primeros "*flits*" del mensaje en direcciones equivocadas, lo que incrementa el número de "flits" en la red y, por lo tanto, aumenta la probabilidad de bloqueos entre mensajes.

Hasta aquí se han comentado las principales alternativas en cuanto al control del flujo de la información se refiere. Las técnicas "*store-and-forward*", "*wormhole*" y "*virtual cut-through*" han sido las más utilizadas en sistemas comerciales. La técnica

"*store-and-forward*" es la más antigua y clásica y se usaba en la primera generación de redes de interconexión para computadores paralelos como Cosmic Cube [117] e Intel iPSC/1 [75]. Como se ha comentado, sus desventajas son la alta latencia sin colisiones y la gran necesidad de espacio de almacenamiento para guardar los paquetes enteros. Frente a esto, la técnica mas utilizada en la actualidad es la de "*wormhole*", que presenta menores latencias y necesidades de espacio de almacenamiento en los nodos intermedios. Esta técnica se utiliza en computadores paralelos como el Cray T3D [28]. A medida que la tecnología permita aumentar el espacio de almacenamiento de los nodos, se permitirá utilizar la técnica de "*virtual cut-through*", que tiene la misma latencia base que "*wormhole*", pero cuando el paquete se bloquea no permanece en la red ocupando un alto número de enlaces, ya que se almacena en el nodo bloqueado, como haría el "*store-and-forward*").

2.5 Encaminamiento

Esta función se ocupa de encaminar los mensajes desde el nodo fuente y, durante todo el camino, hasta alcanzar su destino. Para alcanzar su destino puede surgir la necesidad de hacer varios saltos en nodos intermedios a lo largo del recorrido.

Para realizar este trabajo, la función de encaminamiento deberá conocer la topología de la red y seleccionar trayectorias apropiadas a través de ella. También deberá tener cuidado, al seleccionar las rutas, de evitar las sobrecargas en algunas de las líneas de comunicación, mientras deja a otras inactivas.

El algoritmo de encaminamiento en una red de interconexión es el mecanismo mediante el cual se guían los paquetes a sus destinos a través de la red. El principal objetivo del algoritmo de encaminamiento es seleccionar caminos con el mínimo retardo total para cada paquete [13]

Las propiedades que se le piden a un algoritmo de encaminamiento son corrección, simplicidad, robustez, estabilidad, justicia y optimalidad [127].

Los conceptos de corrección y simplicidad significan que el algoritmo debe realizar decisiones de encaminamiento correctas en un tiempo limitado, por lo que no puede ser un algoritmo computacionalmente complejo. La robustez del algoritmo significa que sea capaz de resistir ante cambios de topología debido a fallos de nodos o enlaces. La estabilidad significa que los algoritmos converjan a una situación estática, si se utiliza una técnica que se va adaptando a las condiciones del tráfico. La propiedad de justicia implica garantizar igualdad de oportunidades en el acceso a los enlaces por parte de los paquetes en la red. El criterio de optimalidad puede ser aplicado considerando

dos estrategias diferentes, minimizar el retardo medio de los paquetes o maximizar la eficiencia total de la red. Estos dos objetivos pueden estar en conflicto, ya que aumentar el uso de la red mediante la inyección de más paquetes, puede implicar mantener los paquetes en colas de espera un tiempo mayor si no se utilizan políticas de gestión del tráfico adecuadas.

Justicia y optimalidad pueden estar enfrentados por el hecho de que el querer optimizar un parámetro puede dejar algún nodo con pocas posibilidades de uso de la red. Por ejemplo, si hay mucha comunicación entre dos nodos de manera que saturan un camino, y un tercer nodo necesita enviar un solo paquete por un camino que se cruza con el que está saturado, el criterio de maximizar el uso de la red indicaría no frenar la comunicación y no permitir el paso del paquete aislado. Por lo tanto, se deberá llegar a un punto de equilibrio entre los dos conceptos.

2.5.1 Evitación de anomalías: "Starvation", "livelock" y Bloqueo de la comunicación.

Además de las propiedades antes expuestas, los diferentes algoritmos de encaminamiento deben asegurar que no presentan anomalías de funcionamiento. Las anomalías mas frecuentes o típicas son tres. La primera es que no se asegure que todos los paquetes tienen capacidad de avanzar en algún momento. Si se produce la situación de que a un paquete no se le permite avanzar durante un tiempo indefinido, se dice que la red sufre de "*starvation*". La segunda anomalía hace referencia al tiempo que los paquetes permanecen en la red. Si algún paquete, debido a que el algoritmo de encaminamiento no lo acerca a su destino, permanece indefinidamente viajando por la red y nunca es entregado a su destino, se dice que la red sufre de "*livelock*". Finalmente, si se da una situación de interbloqueo de manera que un conjunto de paquetes de la red se encuentran bloqueándose unos a otros de manera que ningún paquete puede avanzar, se dice que la red sufre de "*deadlock*".

2.5.1.1 Evitación de "starvation" y "livelock"

Los problemas de "*starvation*" y "*livelock*" son fáciles de evitar o prevenir. En el caso de "*starvation*", este problema se circunscribe a un enlace único y simplemente aplicando políticas ecuánimes entre los paquetes que lo solicitan, es suficiente para eliminarlo. Por ejemplo, una política de "round-robin" es un ejemplo de política que distribuye a todos por igual el recurso solicitado. En el caso de que existan paquetes con diferentes prioridades, no se debe permitir que los paquetes de mayor prioridad copen totalmente el recurso compartido y se debe reservar alguna parte del ancho de banda para los paquetes de menor prioridad.

El caso de "*livelock*" se puede solucionar simplemente usando un encaminamiento mínimo de manera que los paquetes se acerquen siempre a su destino. En el caso de usar un encaminamiento no mínimo, se debe asegurar que los caminos que genere dicho algoritmo no sean de longitud infinita. Esto se puede asegurar por dos medios: primero, que por propia definición de construcción de los caminos no mínimos, el algoritmo de encaminamiento no genere caminos de longitud infinita, o la segunda alternativa es no permitir caminos no mínimos para los paquetes que superan un cierto tiempo de permanencia en la red [41].

2.5.1.2 Evitación de los bloqueos de comunicación

El tercero de los problemas que surge en la comunicación entre procesos en sistemas basados en paso de mensajes es el bloqueo en la comunicación. Este problema es una manifestación particular del problema del "*deadlock*" (bloqueo), el cual es bien conocido por los diseñadores de Sistemas Operativos. Este es un problema de difícil solución porque implica a un conjunto de elementos distribuidos por la red y esta característica hace difícil tanto su detección como su eliminación.

Un bloqueo puede surgir cuando un conjunto de agentes activos envía peticiones para solicitar el uso exclusivo de recursos, mientras retienen otros recursos que ya poseen hasta que el nuevo recurso se recibe, sin posibilidad de preliberación de uno o todos los recursos que poseen. En tales condiciones, el bloqueo se manifiesta cuando hay una cadena circular de peticiones, de tal manera que cada agente retiene uno o más recursos pedidos por el próximo agente en la cadena. Esta situación se muestra en la Figura 2-22. En ella hay cuatro nodos formando un círculo. Cada uno de ellos posee sus "*buffers*" llenos de paquetes destinados a otros de los nodos en el círculo de manera que ninguno puede avanzar. El nodo 1 desea enviar al nodo 3 a través del nodo 2, que a su vez desea enviar al nodo 4 a través del 3, y éste desea enviar al nodo 1 a través del 4 y éste último desea enviar al nodo 2 a través del 1, de manera que nadie puede avanzar.

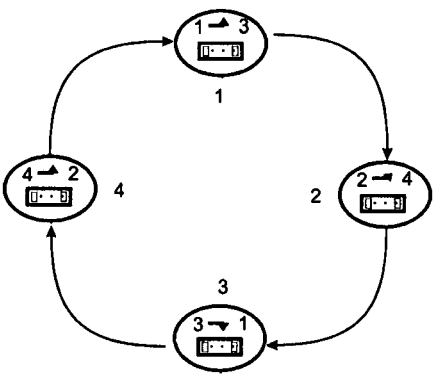


Figura 2-22 Bloqueo en un ciclo de cuatro nodos

Existen tres alternativas para tratar el caso del "*deadlock*". La primera es aplicar técnicas de prevención, la segunda son técnicas de evitación y la tercera técnicas de recuperación [41].

Las técnicas de prevención sólo asignan un recurso a un solicitante si se asegura que la petición no podrá producir "*deadlock*". Esta premisa en el caso de las redes de interconexión se puede conseguir si se reserva el camino en su totalidad con antelación.

Las técnicas de evitación de los bloqueos asignan recursos a medida que los paquetes avanzan por la red. Se asegura, mediante un diseño preciso, que esas asignaciones nunca pueden producir "*deadlock*". La manera de conseguirlo es ordenando los recursos y asignándolos por estricto orden o limitando los caminos que los paquetes pueden tomar.

Las técnicas de recuperación de bloqueos no tratan de evitar que se produzcan los bloqueos y se centran en detectarlos y eliminarlos, removiendo un mensaje del bloqueo y reinyectándolo con posterioridad.

Antes de presentar las técnicas de evitación de bloqueos más comunes en el campo de las redes de interconexión utilizadas por los algoritmos de encaminamiento, vamos a hacer un análisis de las características del problema del bloqueo, centradas en la comunicación de computadores paralelos.

Se parte de la situación en la que no existen "*buffers*" en los encaminadores de la red y, por lo tanto, el método más obvio para intentar evitar el bloqueo en la comunicación es la inserción de "*buffers*" en los procesos que se comunican a nivel del sistema de comunicaciones. Los "*buffers*" se usan para almacenar temporalmente los mensajes dirigidos a otros procesos que no están disponibles para comunicar en ese momento.

De hecho, el uso de "*buffers*" reduce la probabilidad de tener bloqueo pero no resuelve el problema completamente. Es posible que, como resultado de peticiones de comunicación recibidas por un servidor de comunicaciones, uno o más "*buffers*" acaben llenándose. Un servidor que agota sus "*buffers*" es exactamente como un servidor sin "*buffers*" y el bloqueo es nuevamente posible.

Las soluciones de carácter general que se dan a este problema están basadas siempre en evitar que los mensajes formen bucles en sus trayectorias por la red, con lo que el problema se evitaría de raíz. Las características ideales de un algoritmo de prevención de bloqueo serían:

- ✓ No limitar el encaminamiento de mensajes, pudiéndose escoger cualquier camino para alcanzar el nodo destino (encaminamiento adaptativo).
- ✓ Utilizar un número de "*buffers*" por nodo independiente del tamaño de la red.
- ✓ Usar los "*buffers*" de manera óptima, de forma que no se deje de recibir ningún mensaje mientras existan "*buffers*" libres.

Muchos de los métodos conocidos para eliminar los bloqueos imponen restricciones sobre la utilización de los "*buffers*" para almacenar mensajes o sobre los caminos usados por los mensajes entre los nodos fuente y destino. Los métodos que adoptan esta opción no se pueden aplicar junto con las técnicas de comunicación de tipo "*wormhole*". En esta técnica, puesto que sólo el "*flit*" de cabecera contiene información de encaminamiento, no es posible intercalar los "*flits*" pertenecientes a diferentes mensajes sobre el mismo enlace físico. Por lo tanto cuando un "*flit*" ha sido aceptado y asignado a un "*buffer*", los restantes "*flits*" deben ser aceptados antes que los "*flits*" de cualquier otro mensaje. Así pues, no se puede restringir la asignación de "*buffers*". Para el control del flujo "*wormhole*", se utilizan técnicas que limitan los caminos que toman los mensajes para evitar la formación de ciclos.

A continuación, vamos a describir las técnicas más comunes para resolver el problema del bloqueo: Asignación de "*buffers*" [65], canales virtuales [31], redes virtuales [76]. Junto con cada método de eliminación del "*deadlock*" se indica, entre paréntesis, cual es la técnica de control del flujo que se puede aplicar.

2.5.1.2.1 Asignación de "buffers" ("Store-and-forward")

De los métodos de "*buffers*", son mayoría los algoritmos basados en el concepto de estructuración de "*buffers*" ("*structured buffer pool*") [65]. En esta técnica, se particionan los "*buffers*" de mensajes en *dmax* diferentes clases desde *0* hasta *dmax-1*, siendo *dmax* el diámetro de la red. Se establece una política de gestión de esas clases de "*buffers*" de manera que a un paquete que ya ha realizado un determinado número de pasos se le permite sólo usar "*buffers*" de clases menor o igual al número de pasos que ha dado. Cada clase puede contener espacio para guardar un solo paquete o más.

Bajo condiciones de tráfico normal, los mensajes se almacenan sólo en "*buffers*" de clase 0; cuando la carga de mensajes se incrementa, se van usando progresivamente los "*buffers*" de clase 0 hasta clase *dmax-1*. Esto significa que si se llenan los "*buffers*" de las clases menor o igual que *k*, sólo se aceptan los mensajes que vienen de una distancia de al menos *k* pasos desde su origen.

Si se adopta esta estrategia, una petición no concedida de un "*buffer*", sólo puede venir de un "*buffer*" de clase i a un "*buffer*" de clase *(i+1)* Así pues, no se puede formar una cadena circular de peticiones no concedidas y no puede producirse el bloqueo. La Figura 2-23 muestra un conjunto de "*buffers*" clasificados en clases y las peticiones de los "*buffers*" realizadas por los paquetes en orden ascendente.

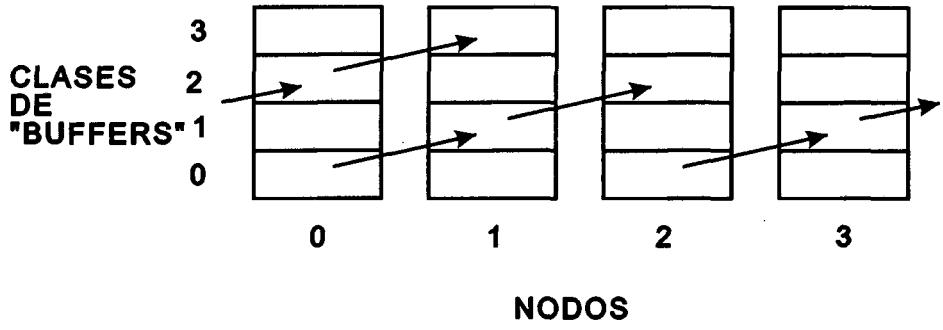


Figura 2-23 Evitación de "*deadlock*": "*Structured Buffer Pool*"

Este método tiene la desventaja de no aprovechar óptimamente los "*buffers*" disponibles y no poderse utilizar en técnicas como "*wormhole*"

2.5.1.2.2 Restricción de los caminos. Canales virtuales ("*Wormhole*")

La segunda alternativa para evitar el "*deadlock*" es utilizar mecanismos de restricción de los caminos. La técnica "*wormhole*" se debería usar si se tiene como requisito esencial una latencia baja. En este caso, se debe adoptar una aproximación que de algún modo imponga restricciones sobre los caminos por los cuales pueden fluir los mensajes ya que no se pueden imponer restricciones sobre el uso de "*buffers*". (Nótese que esta técnica también se puede aplicar para el control de flujo de tipo "*store-and-forward*").

El algoritmo de evitación de "*deadlock*" mediante restricciones sobre los caminos se basa en la construcción de un grafo de dependencias de canales en el que los nodos del grafo representan los canales de la red y los arcos unen los canales según la función de encaminamiento, que hace corresponder canales de entrada con canales de salida en función del nodo destino. Este grafo es la representación de los intercambios de datos entre nodos y se determina según la topología de interconexión de la red y el algoritmo de encaminamiento elegido. A partir de este punto, el grafo construido se examina en busca de ciclos. Si el grafo de dependencias no presenta ciclos, el algoritmo esta libre de "*deadlock*" [29].

Las restricciones que se imponen sobre el encaminamiento de mensajes pueden ser más o menos restrictivas en el sentido que para cada par nodo fuente-nodo destino

sea posible identificar un único camino o se puedan seguir varios caminos alternativos. La principal desventaja de un único camino es que no es posible usar encaminamiento adaptativo.

En la practica, si el uso de un algoritmo de encaminamiento determinístico es satisfactorio para las características de la aplicación, la primera cuestión a escoger es identificar para la red en cuestión una estrategia de encaminamiento que sea libre de bloqueo.

Cuando no sea posible identificar un algoritmo de encaminamiento libre de bloqueo, es decir, que elimine los ciclos en el grafo de dependencias, se puede adoptar el método de los **canales virtuales** [31]. Un canal virtual es una entidad lógica asociada con un enlace físico usado para distinguir múltiples grupos de datos atravesando el mismo enlace físico.

Por ejemplo, veamos el problema de asignar cuatro canales virtuales sobre la conexión bidireccional de un enlace (2 enlaces físicos). Para evitar la introducción de bloqueo debido a la intercalación de mensajes, se debe adoptar un protocolo adecuado, para prevenir bloqueos temporales en una red virtual que causarían bloqueos en la comunicación de otras redes.

El protocolo permite comenzar una transmisión sobre un enlace solo si es cierto que el proceso receptor en la red virtual tiene suficiente espacio de "*buffer*" para el mensaje. Este mecanismo puede ser implementado con un protocolo que requiere que el proceso receptor (que por supuesto tiene un área de "*buffer*") transmita un reconocimiento indicando que esta disponible para recibir otro mensaje. Por mensaje se entiende el paquete de datos entero en el caso de "*store-and-forward*", y un "*flit*" para la técnica "*wormhole*".

La estructura final de los procesos y mensajes enviados por los enlaces que los interconectan se muestra en la Figura 2-24. Esto también muestra que no sólo los datos transmitidos por redes virtuales separadas, sino también los mensajes de reconocimiento correspondientes se intercalan sobre los enlaces físicos.

La técnica de eliminación de "*deadlock*" mediante canales virtuales se basa en la eliminación de los ciclos del grafo de dependencias de canales de la red. Esta eliminación se consigue multiplexando los enlaces físicos, donde sea necesario para evitar la formación de ciclos, en grupos de canales virtuales, cada uno con su propio "*buffer*" independiente. Cada grupo de canales virtuales comparte el enlace de comunicación pero cada canal requiere su propia cola de almacenamiento.

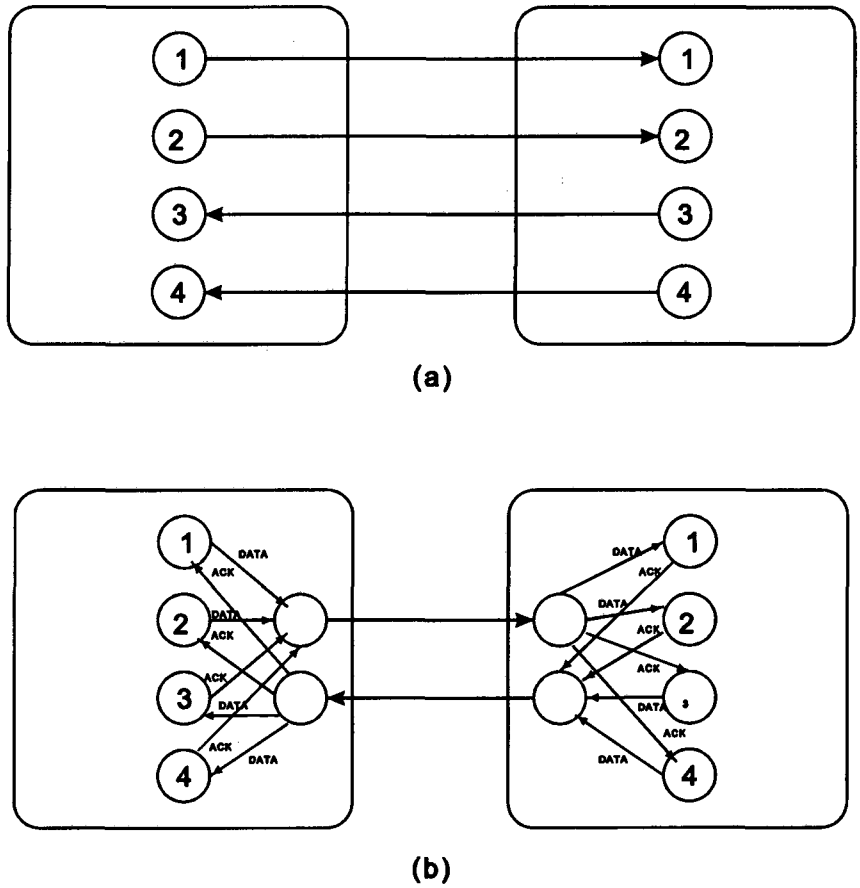


Figura 2-24 "Mapping" de canales virtuales

(a) Cuatro canales virtuales; (b) "Mapping" sobre el enlace físico bidirecional

La Figura 2-25 muestra un ejemplo en el que el canal B se bloquea sin poder llegar a su destino porque se encuentra el camino bloqueado por el canal A, ya que existe un único recurso de comunicación que es el enlace físico. En la Figura 2-26 se elimina el bloqueo mediante canales virtuales y el mensaje del canal B puede avanzar hacia su destino. En este caso, se crean diferentes canales virtuales que existen de forma concurrente mediante la multiplexación del enlace físico en el tiempo. Cada canal virtual tiene, por tanto, menor ancho de banda y "*buffers*" independientes y propios.

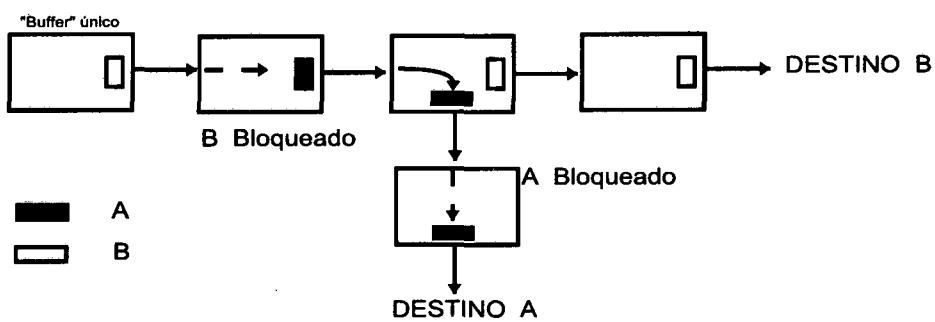


Figura 2-25 Bloqueo de un canal sin usar canales virtuales

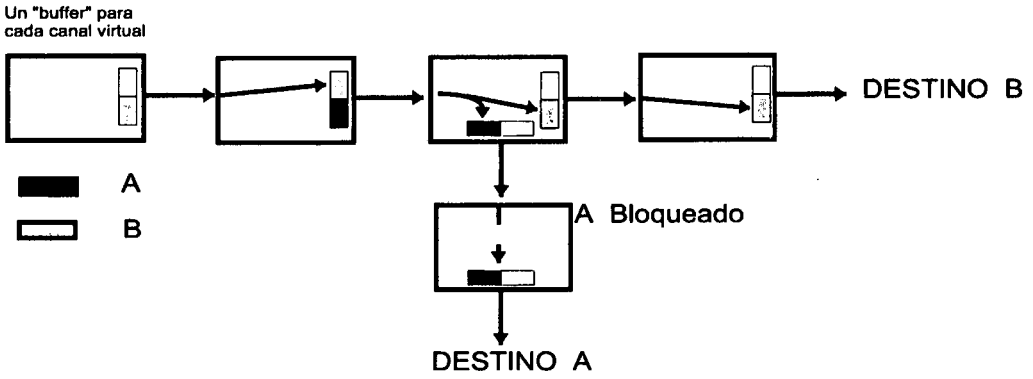


Figura 2-26 Eliminación del bloqueo mediante canales virtuales

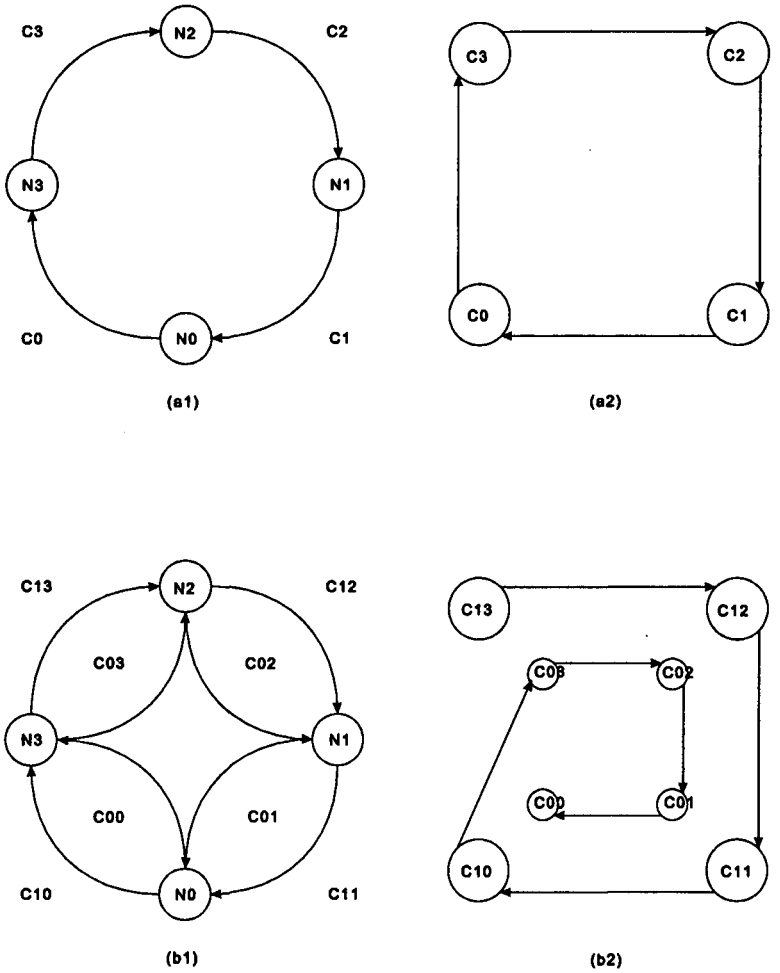


Figura 2-27 Uso de canales virtuales para romper el ciclo de comunicaciones

Veamos a continuación un ejemplo de eliminación de ciclos mediante canales virtuales. En el ejemplo de la Figura 2-27, tenemos un ciclo unidireccional de 4 nodos. El grafo de interconexión se muestra en (a1) y el grafo de dependencia de canales en (a2), que, como se puede observar, muestra un ciclo cerrado. Para eliminar este ciclo, se divide cada enlace físico en dos canales virtuales llamados *superior* (c10,c11,c12,c13) e *inferior* (c00, c01, c02, c03). Con ello el grafo de dependencias de canales queda

como en la Figura 2-27 b1. A continuación, se elige el canal 0 para dividir el ciclo de manera que el grafo de dependencia de canales usando canales virtuales se muestra en Figura 2-27 b2.

La política de asignación de los canales virtuales es la siguiente. Los mensajes iniciados en un nodo cuyo identificador numérico es menor que el de su nodo destino son encaminados por los canales superiores, y los mensajes generados en un nodo mas grande que su destino se encaminan por los canales inferiores. El canal c00 no se usa. Con esto se tiene un ordenamiento de los canales virtuales según sus subíndices: C13 >> C12 >> C11 >> C10 >> C03 >> C02 >> C01 >> C00. (Figura 2-27 b2) Con este diseño, no existen ciclos en el grafo de dependencias y la función de encaminamiento es libre de "*deadlock*".

2.5.1.2.3 Redes virtuales (Mad Postman)

Este método impone menos restricciones que el método de canales virtuales sobre las rutas que los mensajes pueden seguir y, por lo tanto, permite usar algoritmos de encaminamiento con mayor adaptatividad. La idea básica es descomponer la red de interconexión del sistema paralelo en un conjunto de planos de comunicación independientes, llamados redes virtuales, cada uno de los cuales, cuando es analizado individualmente, es libre de bloqueo. Se puede, entonces, demostrar que la red completa es libre de bloqueo.

Esta descomposición se puede conseguir fácilmente para muchas de las topologías de interconexión mejor conocidas. Dados el nodo fuente y destino de un mensaje a transmitir, el primer paso es elegir una red virtual simple sobre la cual el mensaje será transportado a su destino. Dentro de esta red virtual el mensaje puede seguir cualquier camino disponible, permitiendo así un encaminamiento que se adapte a las condiciones de trafico actuales sobre la red. Se prohibe a los mensajes atravesar varias redes virtuales, puesto que esto podría causar de nuevo bloqueos.

Como se ha indicado arriba, la identificación de un conjunto de redes virtuales libres de bloqueo es una operación directa. Por ejemplo, en una topología de malla cuadrada los mensajes se pueden transportar usando cuatro redes virtuales que son libres de ciclos porque transportan mensajes en solo una dirección (Figura 2-28). Todos los paquetes se pueden dividir en cuatro clases según la dirección en que necesiten ser encaminados. Estas cuatro clases corresponden a los cuatro cuadrantes de plano 2D:

Clase I: +X,+Y , Clase II: +X,-Y ,

Clase III: -X,+Y , Clase IV: -X,- Y.

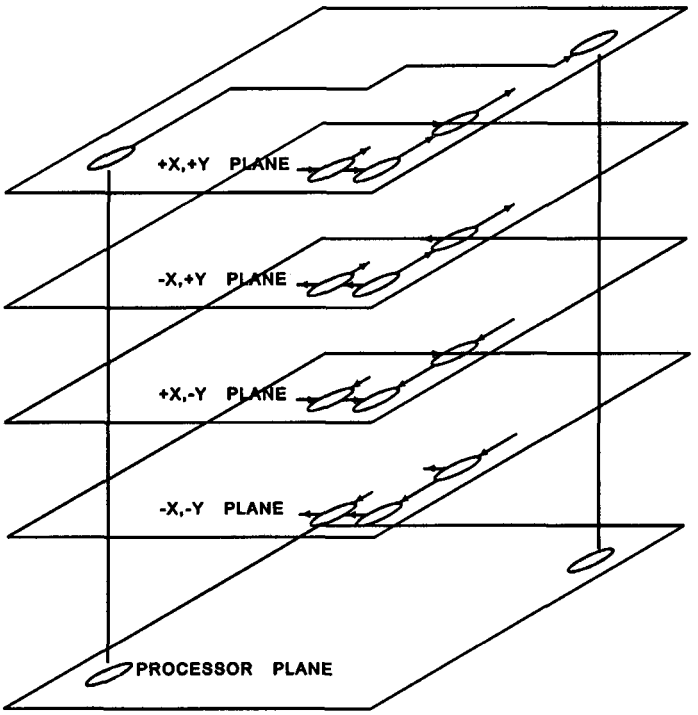


Figura 2-28 Redes virtuales para una topología 2D

Los caminos necesarios para el encaminamiento pueden ser proporcionados por cuatro redes virtuales, cada una encaminando paquetes en uno de los cuatro cuadrantes, supuesto que cada paquete se inyecte inicialmente en la red adecuada.

Esto significa, como se puede probar fácilmente, que el grafo de canales correspondiente a una implementación genérica de una de esas redes virtuales es libre de ciclos, sin importar qué algoritmo de encaminamiento se use. Se debe hacer notar que, a pesar de que a cada mensaje se le restringe a viajar solo en una red virtual, aún preserva cierta libertad de movimiento.

La identificación de un conjunto de redes virtuales libre de ciclos, para otras topologías mas complicadas, es más compleja que en el caso de la red 2D. El procedimiento a seguir es subdividir los mensajes en clases según su dirección, y partir la red física en redes virtuales libres de ciclos que tienen toda la conectividad necesaria para los mensajes de cada clase.

En general, el número de redes virtuales necesario para transmitir mensajes en cualquier dirección es bajo, pero puede llegarse a situaciones en las que, para algunas

topologías, sea necesario tener tantas redes virtuales como parejas fuente-destino haya, lo cual haría el método prácticamente imposible de usar.

El conjunto de redes virtuales identificado debe ser asignado sobre la red física, asignando "*buffers*" separados en cada nodo para cada red virtual, e intercalando los mensajes pertenecientes a diferentes planos de comunicación sobre los mismos enlaces físicos.

Después de la descripción de las propiedades que deben incorporar los algoritmos de encaminamiento, vamos ahora a realizar una clasificación. Los algoritmos de encaminamiento se pueden clasificar en diferentes categorías según una serie de criterios [41]. La Tabla 2-2 muestra, para cada criterio considerado, las diferentes alternativas posibles. El primer criterio es el **número de destinos** a los que se envía el mensaje, que puede ser uno solo, en cuyo caso tenemos comunicación "*unicast*", o varios, en cuyo caso hablaremos de comunicaciones "*multicast*" o "*broadcast*", según se destine a un subconjunto de varios nodos o a todos los nodos de la red.

El segundo criterio es quién y dónde se toman las **decisiones de encaminamiento**, es decir, la determinación del camino que seguirán los mensajes. Se puede hacer de manera *centralizada*, en el caso de que exista un nodo especial que centralice las decisiones, o *no centralizado*, donde las decisiones se deciden localmente entre todos los nodos de la red. En este caso de encaminamiento no centralizado, existen dos posibilidades, dependiendo de si la determinación del camino a seguir la toma el *nodo fuente* que envía el mensaje o si son los nodos que recorre el mensaje los que lo deciden de manera *distribuida*. La alternativa *multifase* es una combinación de las dos anteriores. En ella el camino se divide en diversas fases determinadas por destinos intermedios decididos por el nodo fuente y cada una de las fases se utiliza encaminamiento distribuido. La **implementación** de la determinación del camino puede realizarse mediante una *tabla* que almacena los caminos o mediante un *algoritmo* que calcula el camino a recorrer.

La categoría más importante de clasificación de los algoritmos de encaminamiento es la **adaptabilidad**. Este aspecto hace referencia a si se tiene en cuenta el tráfico presente en la red y/o la ocupación de los enlaces para determinar los caminos o no se tiene en cuenta dicho estado. En el primer caso tenemos algoritmos *adaptativos* y en el segundo caso, tenemos algoritmos *deterministas o estáticos*, en los cuales, las decisiones de encaminamiento no se basan en mediciones o estimaciones del trafico existente en un instante dado, sino sólo dependen de los nodos fuente y destino, en el sentido de que dado un par de nodos se determina el camino de manera estática según una topología determinada.

Los algoritmos adaptativos son capaces de cambiar o adaptar las rutas a las condiciones de tráfico de la red. Dentro de los algoritmos *adaptativos*, pueden darse varias alternativas respecto de *información de la red*, la *progresividad*, la *minimalidad* y el *número de caminos* que utilizan. Si la determinación del tráfico de la red se realiza utilizando información únicamente del nodo en tránsito, tenemos encaminamiento aislado; en caso de que se use información de una vecindad de nodos, tenemos el caso de encaminamiento con información local. En los algoritmos *progresivos*, la cabecera siempre avanza reservando nuevos enlaces, en los algoritmos "*backtracking*" es posible que la cabecera "desande" parte del camino realizado para tomar nuevos caminos. Respecto la *minimalidad*, los algoritmos adaptativos *provechosos* siempre utilizan caminos de longitud mínima de manera que cada paso acerca los mensajes al destino, mientras que los algoritmos *no mínimos* permiten alargar los caminos mediante un alejamiento del destino. Finalmente, existen algoritmos que permiten configurar caminos utilizando *todos los enlaces* de la red, mientras que otros sólo permiten utilizar un cierto *subconjunto parcial de los enlaces* de la red.

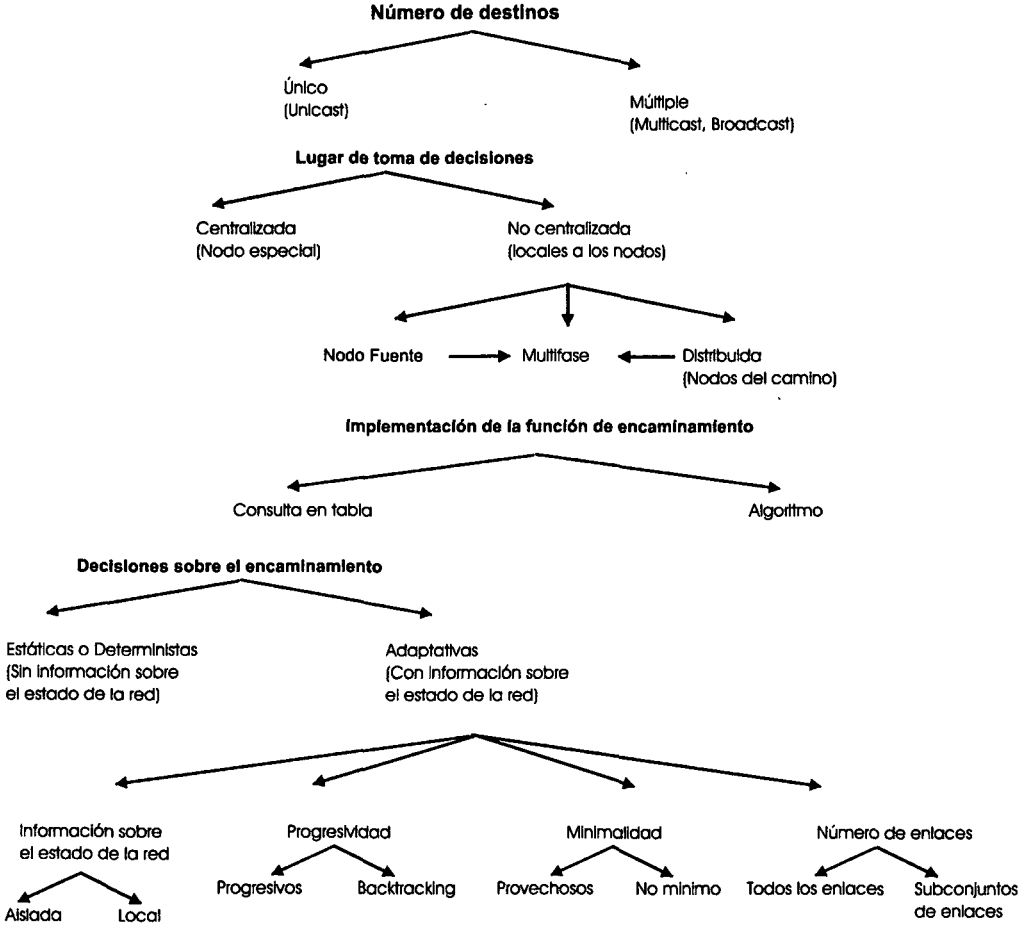


Tabla 2-2 Clasificación de los algoritmos de encaminamiento

A continuación, vamos a describir las diferentes alternativas que se pueden presentar a la hora de encaminar mensajes tanto de forma estática como adaptativa, siguiendo el criterio de cuándo se toman las decisiones de encaminamiento.

2.5.2 Encaminamiento determinista (estático) de camino mínimo

En este tipo de encaminamiento, se define un único camino entre cada par fuente-destino de manera fija y no se cambia en función del tráfico de la red o posibles fallos físicos. Esta definición suele hacerse de manera "off-line" y almacenarse en la red antes de ejecutar el programa de aplicación. Para cada envío se utiliza el camino previamente determinado. Para redes regulares, se han desarrollado algoritmos de encaminamiento estáticos simples como es el encaminamiento por orden de dimensión (DOR) para n-cubos k-arios [41]. En este tipo de encaminamiento se encaminan los paquetes primero en una de las dimensiones hasta llegar a la posición del destino en esa dimensión, a continuación se elige otra dimensión, y se opera de la misma manera, hasta consumir todas las dimensiones posibles de la red. La Figura 2-29 muestra un ejemplo de encaminamiento DOR para una malla 2D, malla 3D e hipercubo.

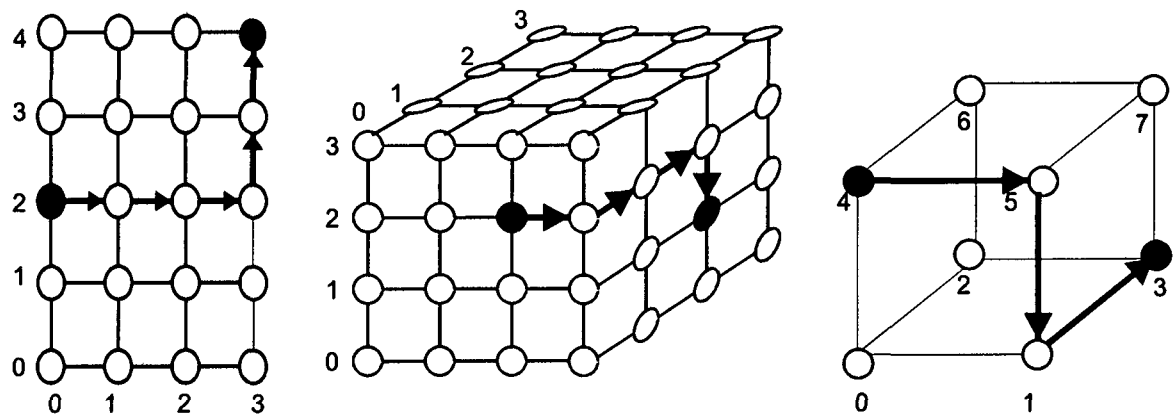


Figura 2-29 Encaminamiento mínimo estático DOR

2.5.3 Encaminamiento adaptativo

Como se ha señalado en la clasificación de los algoritmos de encaminamiento, las decisiones de encaminamiento que realizan los algoritmos de encaminamiento adaptativos pueden tomarse de manera centralizada o no centralizada. Como se ha comentado, en el encaminamiento centralizado existe un nodo central que toma las decisiones de encaminamiento. En el encaminamiento no centralizado las decisiones de encaminamiento las pueden tomar sólo los nodos fuentes o todos los nodos que intervienen en la ruta. Respecto de la cantidad de información que utilizan para tomar las decisiones, si solo utilizan información del propio nodo para tomar la decisión,

tendremos el caso de encaminamiento aislado, si utilizan más información de nodos vecinos tendremos información local. A continuación se presentan las técnicas de encaminamiento centralizada, aislada y distribuida.

2.5.3.1 Encaminamiento centralizado

En un algoritmo de encaminamiento centralizado se elige un nodo de la red como centro de control del encaminamiento. Periódicamente cada nodo envía cierta información sobre su estado al centro de control (por ejemplo, una lista de sus vecinos activos, las longitudes actuales de las colas de espera, la cantidad de tráfico procesado por enlace desde el último informe, etc.)

El centro de control recoge toda esta información y en función del conocimiento total de la red completa, calcula todas las rutas óptimas de todos los pares de nodos, construye las nuevas tablas de encaminamiento y las distribuye a cada nodo. Todo esto se realiza periódicamente cada cierto intervalo de tiempo.

Este método presenta graves problemas como son el hecho de parar todo el computador paralelo cada intervalo para transmitir la información de la comunicación de todos los nodos al centro de control y del centro de control a cada nodo. Otro problema es el hecho de la vulnerabilidad: si se desactiva el centro de control se para todo el sistema de actualización de la información de encaminamiento y el sistema se convierte en un encaminamiento estático. Este tipo de encaminamiento se ha utilizado en sistemas tipo SIMD [73]

2.5.3.2 Encaminamiento aislado

Los algoritmos de encaminamiento aislado [67] son una forma de encaminamiento descentralizado más sencillos en que los nodos llevan a cabo decisiones de encaminamiento cuando deben enviar un paquete, únicamente basándose en la información que ellos mismos hayan reunido. No intercambian información de rutas con otros nodos.

Un algoritmo de este tipo es el algoritmo de "*hot potato*" ("patata caliente") [9]. En este algoritmo, cada nodo mantiene unas colas de espera asociadas a cada enlace de salida. Estas colas guardan los mensajes que deben ser enviados por el correspondiente enlace. En el momento en que llega un paquete a un nodo, éste trata de deshacerse de él lo más rápido posible poniéndolo en la cola de espera más corta.

2.5.3.3 Encaminamiento distribuido

En los algoritmos de encaminamiento adaptativos distribuidos, la selección del camino a seguir por parte del mensaje se basa en la información obtenida sobre el estado de la red. Estos algoritmos pueden ser mínimos o no mínimos, según generen caminos de longitud mínima o no, y total o parcialmente adaptativos, según consideren todos los caminos físicos para encaminar los mensajes o no.

Este tipo de algoritmos se ha estudiado mucho en la literatura y se han hecho muchas propuestas. Algunas de ellas son las de Dally [31], Duato [38], Chien&Kim (Planar Adaptive) [26], Glass&Ni (Turn model) [99], Kostantantindou&Snyder (Chaos Routing) [17], Kim&Liu&Chien (Comprenssionless Routing) [80], Schwiebert&Jayasimha [115] y métodos para hipercubos basados en "*backtracking*" presentados por Gaughan&Yalamanchili en [62]. La mayoría de estas técnicas de encaminamiento utilizan la técnica de canales virtuales para evitar el "*deadlock*".

J. Duato ha presentado en [38] una técnica generalizada para el diseño de algoritmos de encaminamiento libres de "*deadlock*" utilizando canales virtuales. Esta teoría se basa en el análisis del *grafo de dependencia de canales extendido* para una red de interconexión y una subfunción de encaminamiento R_1 de una función de encaminamiento R. A partir de este grafo se concluye que una función de encaminamiento adaptativa y conectada R es libre de "*deadlock*" si y solo si existe un subconjunto de canales, el cual define una subfunción R_1 de encaminamiento, conectada y que no posee ciclos en el grafo de canales extendido. Este resultado está basado en que aún permitiéndose la adaptabilidad se provean canales de escape que conduzcan los paquetes de manera segura hasta su destino.

Muchos autores han propuesto algoritmos de encaminamiento cuyo diseño se fija especialmente en la evitación de "*deadlock*" y en la provisión de adaptabilidad y que utilizan canales virtuales. De los algoritmos desarrollados hasta la actualidad, los que ofrecen mayores prestaciones son los algoritmos de camino mínimo que proveen una adaptatividad completa, es decir, que permiten utilizar cualquiera de los caminos mínimos existentes entre un par fuente-destino y no restringen el uso de caminos. En este caso, cuando un paquete llega a un nodo, elegirá el enlace de salida que primero esté disponible de entre los que le acercan al destino.

Ejemplos de algoritmos adaptativos son el Planar-adaptive [26], el Turn model [99], Chaos Routing [17] [18] y los métodos basados en la limitación de la inyección [21] [23]. A continuación se comentan cada uno de estos métodos.

2.5.3.3.1 Encaminamiento "*Planar-adaptive*"

El encaminamiento "*Planar-adaptive*", desarrollado por Chien y Kim, se basa en restringir los posibles caminos que un mensaje puede tomar a un plano bidimensional. Así, un paquete se encamina adaptativamente en series de planos 2D. Las dimensiones del encaminamiento cambian a medida que el paquete avanza hacia su destino. En cada plano se permiten todos los caminos posibles y se evita el "*deadlock*" mediante canales virtuales. En la Figura 2-30 se muestra los planos de movimiento permitidos para el caso de redes de 3 y 4 dimensiones.

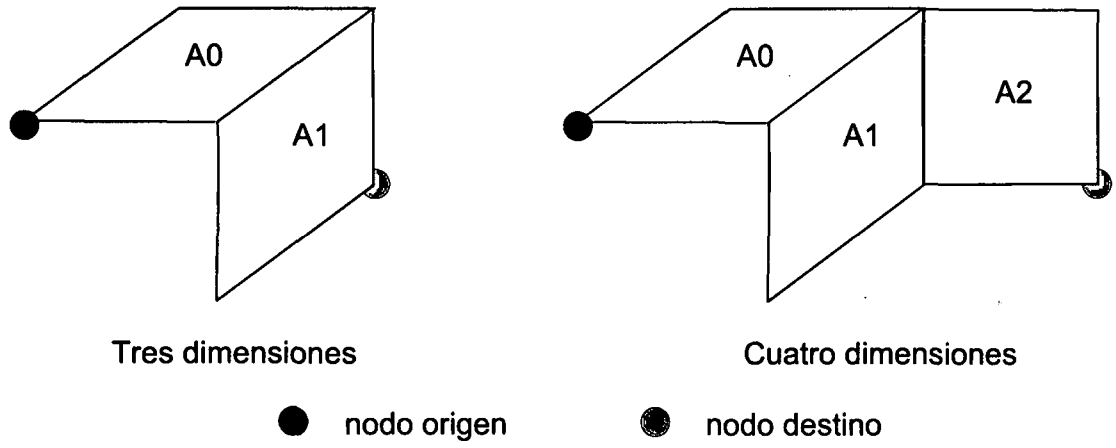


Figura 2-30 Encaminamiento "Planar-Adaptive"

El objetivo de este mecanismo es minimizar el número de recursos necesarios para implementar un cierto grado de adaptatividad. Según su definición, el encaminamiento Planar-adaptive requiere tres canales virtuales por enlace físico para una red de tipo malla y seis canales virtuales por enlace físico para una red tipo toro.

2.5.3.3.2 Encaminamiento "*Turn model*"

El encaminamiento "*Turn model*", desarrollado por Glass y Ni, es una técnica para evitar bloqueos restringiendo los posibles caminos del paquete. Es un mecanismo sistemático para diseñar algoritmos de encaminamiento parcialmente adaptativos. Se basa en limitar los posibles "giros" que los paquetes pueden dar en la topología con objeto de eliminar los ciclos causantes de "*deadlock*".

En la Figura 2-31 se puede ver un ejemplo para una malla 2D. En la parte (a) se muestra los ocho posibles giros de los paquetes, los cuales son susceptibles de provocar "*deadlock*". En la parte (b) se muestra el conocido algoritmo estático de DOR que elimina cuatro de los giros y solamente permite los otros cuatro giros. En la parte (c) se

muestra los seis giros permitidos por Turn model, con los cuales se permite una cierta adaptatividad, pero se elimina la posibilidad de "*deadlock*", porque no se permiten crearse ciclos al eliminarse uno de los giros posibles.

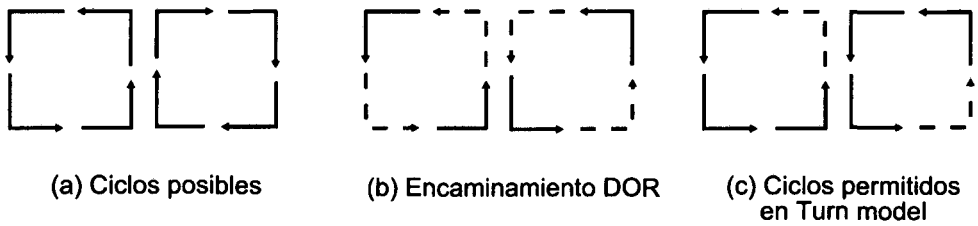


Figura 2-31 Encaminamiento "Turn Model"

2.5.3.3.3 Encaminamiento "*Chaotic*"

El encaminador "*Chaotic*" introduce aleatorización para producir un algoritmo de encaminamiento tolerante a fallos no mínimo usando control del flujo "*virtual cut-through*". En el encaminamiento "Chaotic", los mensajes se encaminan normalmente siguiendo cualquier enlace que proporciona un camino mínimo. Cuando un mensaje permanece en un "*buffer*" de entrada demasiado tiempo, esperando por el enlace de salida, se almacena en un "*buffer*" local. Los mensajes de este "*buffer*" local tienen prioridad sobre los enlaces libres frente a otros mensajes. Esta técnica es suficiente para evitar el "*deadlock*". Cuando este "*buffer*" local, que es finito, está lleno y se precisa un espacio, se elige aleatoriamente un mensaje del "*buffer*" y se envía por cualquier enlace libre, aunque lo aleje del destino. El análisis del encaminamiento "*chaotic*" muestra que la probabilidad de generar caminos de longitud infinita se acerca a cero por el hecho de elegir aleatoriamente un paquete cualquiera para desencaminarlo. Esta característica evita el problema del "*livelock*".

2.5.3.3.4 Limitación de la inyección

Otros algoritmos de encaminamiento cuyo mecanismo de evitación de "*deadlock*" está basado en la limitación de la inyección de nuevos mensajes han sido presentados, por R. Beivide y otros autores, en [21] y en [23] basados en los trabajos de Roscoe [113]. Estos algoritmos son válidos para el control del flujo "*cut-through*". El primero de estos algoritmos, llamado "*Bubble Router*", se basa en garantizar que no todos los recursos de la red involucrados en dependencias cíclicas de canales se saturen. Para ello, la función de control del flujo "*Bubble*" solo permitirá que un paquete se incorpore a un ciclo de la red si y solo si con esa incorporación no se agotan los recursos de almacenamiento de ese ciclo. De esta forma, siempre existirá en ese ciclo, un paquete que pueda avanzar hacia su destino, eliminando la posibilidad de "*deadlock*". El segundo algoritmo, llamado "*ghost-packet*", se aplica a redes n-cubos k-arios y elimina

la necesidad de canales virtuales para evitar el "*deadlock*". En este algoritmo, aparece un nuevo elemento en el control del flujo que se llama "*ghost-packet*". La movilidad asignada a los "*ghost-packets*" es contraria al flujo de los paquetes normales. Los "*ghost-packets*" se caracterizan por tendencia a la inmovilidad, viéndose forzados a un cambio de posición cuando el recurso que ellos ocupan está siendo requerido por algún paquete, intercambiando su posición con este último, como si el "*ghost*" no existiera. Ahora bien, en el caso de que el paquete deba incorporarse a una nueva dirección, deberá esperar a que haya algún "*buffer*" libre no ocupado por ningún paquete, normal o "*Ghost*". Ante esta situación, el "*Ghost*" intentará desplazarse hacia el nodo vecino para dejar espacio para el paquete que quiere incorporarse. La idea clave para evitar el "*deadlock*" es incorporar un "*ghost packet*" para cada uno de los anillos que existan en la red de interconexión.

2.6 Estructura de los encaminadores

Los encaminadores físicos son los componentes de la red de interconexión que implementan las funciones hasta ahora descritas de control del flujo y encaminamiento. Estos encaminadores cuentan con un conjunto de enlaces de entrada y otro conjunto de enlaces de salida. Su función principal es aceptar mensajes desde sus enlaces de entrada, paquetes o "*flits*" (según la técnica de control del flujo que implemente el encaminador) y dirigirlos hacia los enlaces de salida. Para realizar sus funciones, los encaminadores deben ser capaces de almacenar la información si es necesario, decidir el enlace de salida y arbitrar el uso de los recursos cuando son solicitados por varios peticionarios simultáneamente. A continuación se presentan las estructuras básicas de un encaminador sin canales virtuales, y con canales virtuales y el análisis del comportamiento temporal de un encaminador básico [29] [41].

2.6.1 Estructura básica de un encaminador

Para realizar sus funciones, los encaminadores cuentan con una serie de elementos como se muestra en la Figura 2-32. Estos elementos son un conjunto de controladores de enlaces de entrada (CEE) y otro de salida (CES), una serie de "*buffers*" de almacenamiento, un elemento de interconexión entre las entradas y las salidas o conmutador, que suele ser un "crossbar", y una lógica de encaminamiento y arbitraje [41].

Los enlaces, tanto de entrada y de salida, son las conexiones físicas entre encaminadores y pueden tener diferentes anchuras, desde 1 solo bit, en cuyo caso tenemos un enlace serie, hasta varios bits, como una palabra, en cuyo caso permite transmitir varios bits en paralelo simultáneamente.

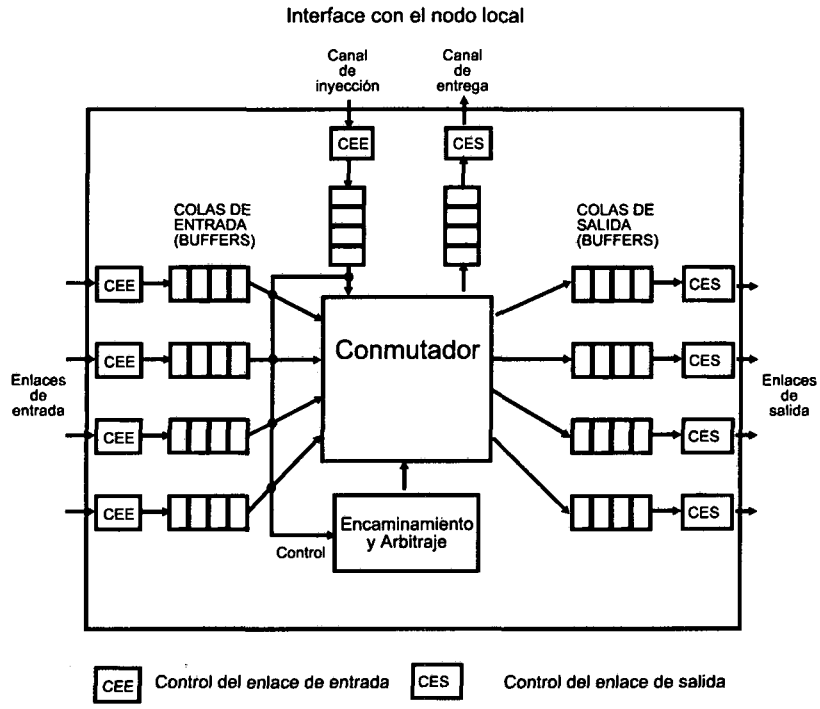


Figura 2-32 Estructura básica de un encaminador

Los "*buffers*" son utilizados para almacenar los mensajes en tránsito. Son colas de funcionamiento FIFO "el primero en entrar, el primero en salir" y cada elemento del "*buffer*" es capaz de almacenar un "*flit*". Como estructuras alternativas a la presentada, los "*buffers*" pueden existir asociados sólo a los enlaces de entrada o sólo a los enlaces de salida.

Los controladores del enlace son los circuitos receptores (CEE) y transmisores (CES) que se encargan de controlar el enlace físico entre dos encaminadores y de transferir los "*flits*" entre ambos.

El conmutador es el elemento encargado de conectar los "*buffers*" de entrada con los de salida y de hacer avanzar los *"flits"* de unos a otros. Suele ser un "crossbar" de alta velocidad que permite conectar cualquier entrada con cualquier salida.

El módulo de encaminamiento y arbitraje es el encargado de implementar la función de encaminamiento, sea basada en un algoritmo o en una tabla, de seleccionar el enlace de salida para cada paquete de entrada y de configurar el conmutador. En el caso de que varios paquetes soliciten simultáneamente un mismo enlace de salida que se encuentre libre, se encarga de realizar el correspondiente arbitraje para seleccionar un paquete para asignarlo al enlace de salida y de almacenar en el "*buffer*" de salida los paquetes no inmediatamente seleccionados para continuar su camino.

Los canales de inyección y de entrega forman la "*interface*" con el procesador en el nodo local, y son similares a los enlaces de entrada y salida, respectivamente. Se utilizan para inyectar o consumir mensajes enviados o recibidos por el nodo local en la red de interconexión.

2.6.2 Estructura de un encaminador con canales virtuales

En el caso de utilizar canales virtuales, éstos tienen una implicación sobre la arquitectura del encaminador [41]. Como ya se ha comentado, cuando se han introducido los canales virtuales, cada canal virtual perteneciente a un enlace físico se implementa con una cola de almacenamiento separada. La gestión de los canales virtuales implica la multiplexación del enlace físico con lo que se precisa un identificador de canal virtual y la lógica y el protocolo necesarios para copiar cada paquete en su cola correspondiente. Por lo tanto, además de las diferentes colas, se necesita modificar los módulos de control de los enlaces de entrada (*CEEV*) para que decodifiquen a qué canal virtual pertenece el paquete entrante para copiarlo sobre la cola adecuada. Asimismo, se debe incluir un controlador del canal virtual *CV* para multiplexar los enlaces de salida entre las diferentes colas de salida de los diferentes canales virtuales. Los paquetes que usen canales virtuales deben incluir una identificación del canal virtual por el que circulan para poder manejarlo adecuadamente. Esta estructura modificada de un encaminador con canales virtuales puede verse en la Figura 2-33.

2.6.3 Modelo de prestaciones del encaminador básico

Para analizar el comportamiento dinámico de un encaminador con esta estructura básica, hace falta fijarse en los elementos que retardan el avance de los mensajes por la red de interconexión. Para ello debemos tener en cuenta las diferentes acciones que se llevan acabo para procesar un paquete. La Figura 2-34 muestra la serie de acciones realizadas desde que en paquete llega a un encaminador. Primeramente, debe decidirse el enlace de salida mediante una decisión de *encaminamiento*, a continuación debe configurarse el conmutador interno y copiar el paquete desde los "buffers" de entrada a los de salida del enlace seleccionado. Finalmente, el paquete se transmite al nodo siguiente a través del enlace.

Suponemos que tenemos un paquete almacenado en una de las colas de entrada y queremos calcular el tiempo transcurrido para atravesar el encaminador y transmitirlo hasta los "*buffers*" de entrada del siguiente encaminador. Este tiempo puede dividirse en tres partes: El tiempo para encaminar el paquete, el tiempo para atravesar el

encaminador y el tiempo para atravesar un enlace hasta el próximo encaminador. A continuación, se explican cada uno de ellos.

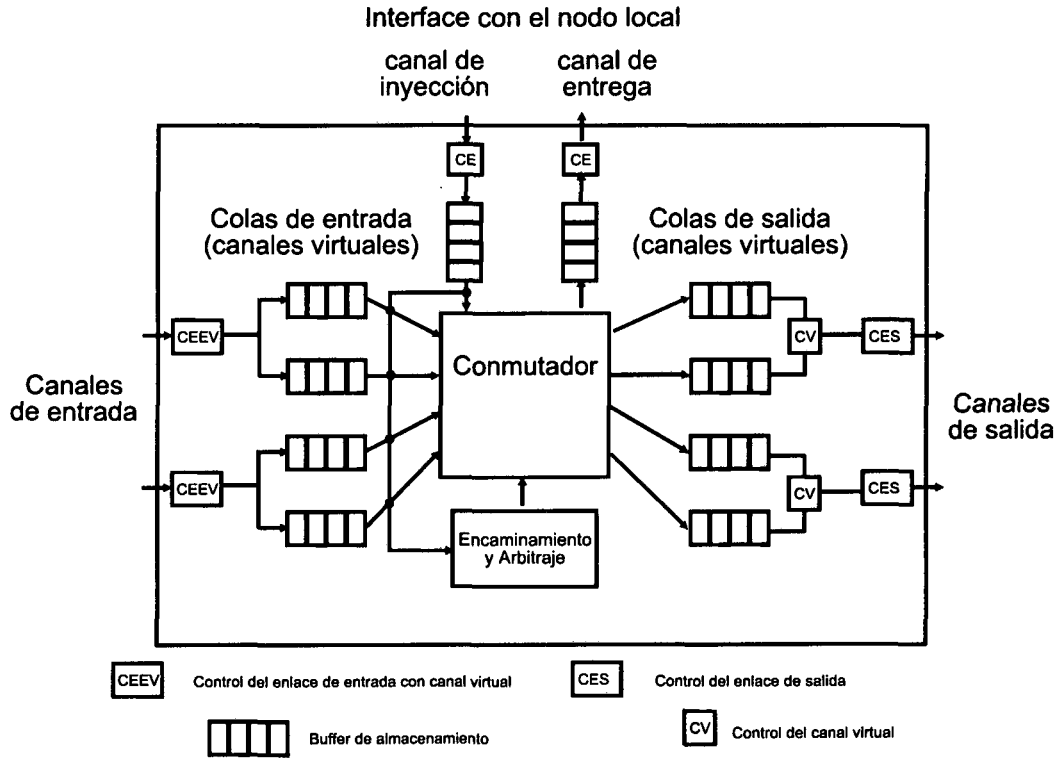


Figura 2-33 Estructura básica de un encaminador con canales virtuales

Primeramente, cuando un paquete llega a un encaminador debe tomarse una decisión de encaminamiento para determinar el enlace de salida. El tiempo dedicado para ello se llama **retardo de encaminamiento (tr)**. Suele incluir el tiempo de configurar el conmutador con la decisión tomada.

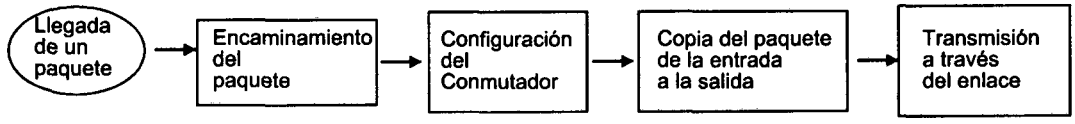


Figura 2-34 Acciones realizadas para transmitir un paquete a través de un encaminador

A continuación, el paquete progresa desde los "*buffers*" de entrada hacia los de salida. El retardo en este avance se llama retardo **intraencaminador (ts)** y está compuesto de dos retardos: el retardo de **propagación de la señal** a través del conmutador y el retardo necesario para **sincronizar** la transferencia de datos entre el "*buffer*" de entrada y el de salida. Además, en el caso de que el enlace se encuentre ocupado por otro paquete, el mensaje debe almacenarse hasta que el enlace esté disponible para ser usado. Este retardo se llama retardo de **encolamiento** por contención.

Finalmente, el paquete debe transmitirse por el enlace hasta los "*buffers*" de entrada del siguiente encaminador. El retardo necesario para ello se llama retardo **interencaminador (tw)** y se compone de dos retardos: el tiempo necesario para **inicializar** la transmisión y el tiempo puro de **transmisión** de la información por el enlace. La Figura 2-35 muestra el análisis de retardos existentes en un encaminador típico.

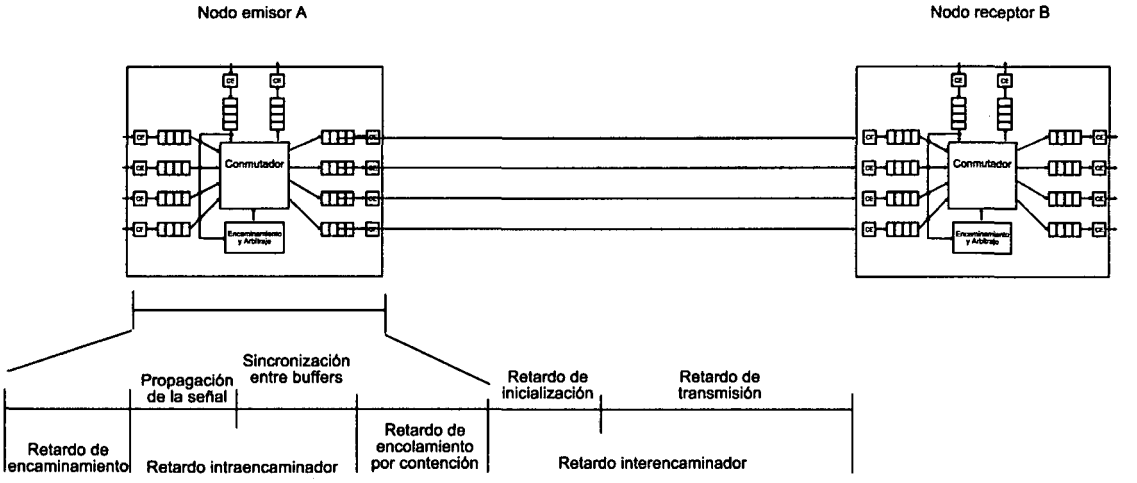


Figura 2-35 Análisis de los retardos en una red con encaminadores básicos para un mensaje que se traslada desde el nodo A al nodo B

En la actualidad, existen algunos encaminadores disponibles comercialmente que presentan diversas alternativas para las características expuestas anteriormente. Los ejemplos más representativos de ellos son el *Intel Cavallino* con un tamaño de "*flit*" de 16 bits y un tiempo de ciclo de 5 ns [19], el encaminador del *Cray T3E* con un tamaño de "*flit*" de 70 bits y un tiempo de ciclo de 13.5 ns [116], el *SPIDER* de *SGI* con un tamaño de "*flit*" de 160 bits y un tiempo de ciclo de 2.5 ns [56], Myrinet con un tamaño de "*flit*" de 16 bits y un tiempo de ciclo de 6.25 ns [14] o el encaminador del IBM SP2 con un tamaño de "*flit*" de 16 bits y un tiempo de ciclo de 25 ns [124].

2.7 Conclusiones.

En este capítulo se ha hecho un recorrido por todos los aspectos que configuran el tema de las redes de interconexión para computadores de altas prestaciones, presentando las técnicas existentes en la literatura. El estudio se ha centrado en una serie de aspectos: topología, establecimiento del camino y control del flujo, encaminamiento de los mensajes, resolución de anomalías en la comunicación, y estructura y análisis temporal de los encaminadores.

Respecto la topología, hemos visto que, tradicionalmente, existen dos grandes familias: redes directas y redes indirectas, aunque actualmente la tecnología las está haciendo converger a un mismo tipo de redes de alta velocidad. Dentro de las técnicas de control del flujo utilizadas actualmente, "*wormhole*" es la más utilizada en computadores comerciales aunque existen otras alternativas como "*virtual cut-through*" que también son posibles y que precisarán de un avance tecnológico. Dentro de los algoritmos de encaminamiento, hemos visto que existen un gran número de alternativas, aunque las posibilidades tecnológicas hacen del encaminamiento estático el más utilizado, mientras que alternativas adaptativas no son muy utilizadas en sistemas actuales, aunque empiezan a utilizarse. Finalmente, hemos visto que el sistema de comunicaciones de la red de interconexión debe proveer una comunicación libre de anomalías como "*livelock*", "*starvation*", y "*deadlock*". Hemos estudiado cómo eliminar esas anomalías y hemos visto que el tema del "*deadlock*" ha ocupado un gran volumen de estudio en la literatura.

A continuación, el próximo capítulo 3 se centra en el estudio del comportamiento dinámico temporal de las redes de interconexión cuando una cierta carga de mensajes existe sobre ella. Para ello, se realizan dos modelos, uno matemático y el otro funcional. Estos modelos son las herramientas con las cuales se estudia, en el capítulo 4, la respuesta en latencia que sufren los mensajes en la red debido a la contención por el uso de recursos comunes como son los enlaces de comunicación. De este estudio del comportamiento de la red de interconexión, se deducen una serie de problemas intrínsecos a dicho funcionamiento y se analiza la razón de aparición de dichos problemas.

Capítulo 3 Modelado de una red de interconexión

3.1 Introducción

En el capítulo 1, se introdujeron las redes de interconexión como componentes que forman parte de un computador de altas prestaciones a través del análisis de los diferentes tipos de sistemas que pueden ofrecer alta potencia de cálculo en la actualidad. En el capítulo 2, se ha presentado un estudio de las características estructurales que definen una red de interconexión tales como topología, control del flujo, encaminamiento, etc. y que son parámetros que se deben decidir a la hora de configurar la red de interconexión de un computador de altas prestaciones del tipo presentado.

Este capítulo, después del estudio realizado en el capítulo precedente de los parámetros estáticos de una red de interconexión, tiene como objetivo principal el desarrollo de modelos de comportamiento de las redes de interconexión. La descripción de la problemática que existe en el funcionamiento dinámico que tienen las redes de

interconexión y el estudio del impacto que tiene este comportamiento dinámico en el rendimiento y funcionamiento de las aplicaciones que se ejecutan sobre el computador de altas prestaciones debe hacerse contando con las herramientas adecuadas.

Para poner de manifiesto los problemas de funcionamiento de las redes de interconexión, se deben analizar las características de su comportamiento funcional. Por esta razón, para poder analizar el funcionamiento de las redes de interconexión y comprender el porqué de ese comportamiento dinámico, hemos realizado un trabajo de modelado de las redes de interconexión. Creemos que sólo un modelo de representación es la única herramienta que nos posibilita profundizar en las razones y las causas de un comportamiento u otro. Los modelos nos permitirán diseñar propuestas de solución o cambio de dicho comportamiento con cierto conocimiento de causa y con posibilidades de hacer una predicción estimada de los resultados esperados. Con todo ello, estaremos en disposición de entender la importancia de las redes de interconexión en los computadores de altas prestaciones.

Hemos abordado el modelado de las redes de interconexión desde dos puntos de vista radicalmente diferentes. El primero es la de modelado analítico y el segundo el modelado funcional. El primero se concreta en un modelo matemático y el segundo en un simulador del comportamiento de la red de interconexión

El "modelo matemático" analítico que hemos desarrollado en este trabajo resume el comportamiento temporal de una red de interconexión y explica la influencia en dicho comportamiento de los parámetros de diseño de la red de interconexión y de la carga de mensajes sobre la red. De una manera muy breve, se puede describir diciendo que es un modelo que, mediante técnicas matemáticas estadísticas y analíticas, plantea un sistema de ecuaciones a partir de cierta información sobre la red de interconexión y la configuración de mensajes que circularán sobre ella. Este sistema de ecuaciones, una vez resuelto, informa de los retardos que sufren los mensajes en la red de interconexión.

La otra posibilidad para el modelado y simulación de una red de interconexión es la simulación funcional del sistema. En este caso, el modelo busca reflejar el comportamiento funcional de la red de interconexión mediante un programa de ordenador que simula la operatividad o modo de operación de la red. En este trabajo de tesis, se ha desarrollado también el simulador funcional de redes de interconexión que implementa el modelo funcional desarrollado.

Las razones de disponer de dos modelos, uno analítico y otro funcional, son diversas. Por un lado, el modelo analítico ofrece unos resultados de comportamiento en estado estacionario con un esfuerzo de cómputo relativamente pequeño. El modelo

funcional exige unos recursos de cálculo mucho más elevados y, a cambio, ofrece el comportamiento temporal completo, incluyendo el transitorio. Además, por ser dos modelos que describen la red de interconexión desde puntos de vista tan diferente pueden ser utilizados para la validación mutua del uno respecto del otro.

El resto del capítulo esta organizado como sigue: El siguiente punto se dedica a introducir las técnicas de modelado y simulación como herramientas de estudio de sistemas reales. El punto 3 introduce los parámetros de entrada y de salida de los modelos desarrollados. En este punto se consideran los modelos de la red de interconexión como una caja negra y se describen sus entradas y salidas.

El punto 4 presenta el modelo analítico dando una descripción detallada del modelo internamente, es decir, cómo se construye el sistema de ecuaciones de representación a partir del funcionamiento conceptual de la red de interconexión y, asimismo, cómo se resuelve dicho sistema de ecuaciones planteado.

El punto 5 introduce y desarrolla el modelo funcional. Primeramente describe las posibles alternativas de diseño de estos simuladores para centrarse luego en la simulación funcional de redes de interconexión. A continuación, describe la estructura del simulador funcional desarrollado en este trabajo de tesis, mediante el detalle de las estructuras de datos y el algoritmo de funcionamiento del simulador.

El punto 6 muestra la metodología seguida, y los resultados obtenidos en el proceso de validación realizado de los modelos aquí presentados, mediante la comparación del uno contra el otro. Finalmente el punto 7, presenta las conclusiones.

3.2 Análisis, modelado y simulación

En la tarea de modelado y simulación de sistemas aparecen tres elementos principales (Figura 3-1). El primero es el sistema real a analizar o estudiar, el segundo el modelo, y el tercero el simulador. Estas tres entidades no pueden entenderse por separado sino que deben ser vistas interelacionadas entre ellas. El sistema real puede verse como una fuente de datos de comportamiento de una cierta variable física que se quiere estudiar. Un modelo es básicamente un conjunto de reglas, ecuaciones, algoritmos o tablas que nos permiten generar, a partir de los datos de entrada, los valores de las variables cuyo comportamiento queremos modelar. En nuestro caso, un simulador es un programa de computador que implementa un modelo. Entre estos tres elementos, sistema real, modelo y simulador, se establecen una serie de relaciones como son el modelado y la simulación. El término modelado hace referencia a las relaciones

entre el sistema real y el modelo. El término simulación hace referencia a las relaciones entre el simulador y el modelo.

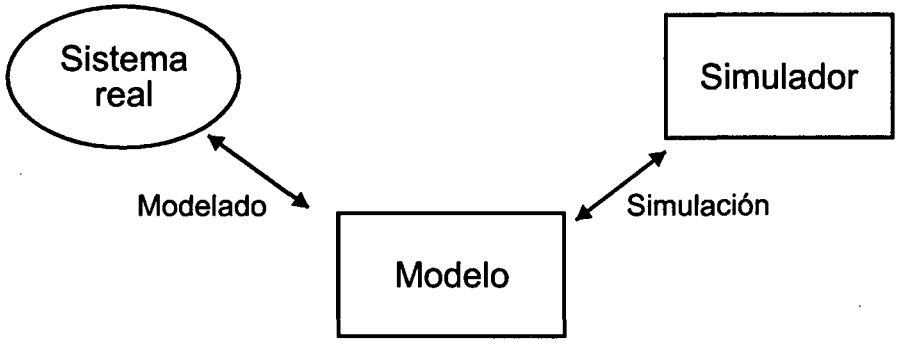


Figura 3-1 Modelado y simulación de sistemas reales

Las técnicas de modelado y simulación pueden ser utilizadas como una herramienta de análisis para predecir los efectos de los cambios en sistemas existentes, y como una herramienta de diseño para predecir las prestaciones de nuevos sistemas bajo diversas circunstancias. Concretamente, la simulación realizada en ordenadores ofrece un método para analizar el comportamiento del sistema, entendiéndose por sistema una colección de entidades [86] relacionadas, cada una de las cuales se caracteriza por atributos que pueden estar relacionados entre sí [46]. Una característica importante de la simulación es que permite especular con diferentes alternativas de configuración del sistema dando respuesta a preguntas del tipo "¿qué pasaría si...?".

El primer paso para estudiar un sistema es elaborar un modelo. Un modelo puede ser una representación formal de la teoría o una explicación formal de la observación empírica. El realismo y exactitud es una característica importante de un modelo, por lo que normalmente se combinan ambos enfoques. Los modelos de simulación permiten la experimentación donde los experimentos reales no son factibles, pero es necesario controlar la validez del modelo cuidadosamente para garantizar que representa adecuadamente el sistema bajo todas las condiciones de interés. El beneficio principal que aportan los modelos es aumentar la capacidad de toma de decisiones [138].

Al verificar la fiabilidad de un modelo se debe buscar que sea:

- ✓ **Exacto**: El modelo debe tener la habilidad de reproducir o predecir resultados reales.
- ✓ **Preciso**: Significa que no debe ser ambiguo, es decir, que el nivel de detalles del modelo debe ser alto.

- ✓ **Completo**: Esta característica asegura que contiene todos los elementos necesarios para satisfacer el propósito del estudio.

- ✓ **Útil**: Esta propiedad se cumple si el modelo consigue proporcionar la información necesaria, en función de los objetivos propuestos del estudio.

Tradicionalmente, los estudios de modelado y simulación han estado típicamente orientados hacia una serie de objetivos [36], los cuales enumeramos a continuación:

- ✓ El entendimiento de la naturaleza de un sistema por medio de la identificación de sus variables significativas, con una modelización apropiada para obtener el modelo antes mencionado y los valores de sus parámetros asociados.

- ✓ El entendimiento del rendimiento del sistema, evidenciando características cualitativas de su conducta, por ejemplo, el crecimiento/decrecimiento en el tiempo, aumentos/disminuciones exponenciales, etc. El uso de gráficas y animación en este tipo de simulación son de gran ayuda.

- ✓ Evaluación de políticas alternativas. Este tipo de estudio está también enfocado hacia un mejor entendimiento del sistema pero el interés, en este caso, se centra en medir específicamente un aspecto controlado como por ejemplo, la actividad, relación, ciertas restricciones, etc.

- ✓ Reorganización/diseño del sistema, que es el caso del diseño de ordenadores y sistemas de control donde se disponen de ciertos subsistemas con características conocidas y se reorganizan de manera que se crea un nuevo sistema que funciona integralmente. En este enfoque se utilizan simulaciones debido a que el gran número de subsistemas y/o a la complejidad de sus interacciones no permiten establecer un modelo matemático que pueda ser resuelto analíticamente.

- ✓ Optimización del rendimiento del sistema, donde, mediante condiciones relevantes para un aspecto del sistema, que pueden ser tanto valores de entrada como parámetros controlables, se maximiza el rendimiento según las necesidades del usuario.

Como se desprende del estudio anterior, la ventaja básica del modelado y de la simulación es su gran aplicabilidad por lo que se ha convertido en una herramienta básica. No obstante, también tiene las siguientes desventajas [36]:

- ✓ El tiempo y costo en la realización y utilización de los modelos de simulación.

- ✓ La dificultad de implementar y validar correctamente los modelos.

✓ La imprecisión de los resultados de simulación.

A continuación se presenta un breve resumen de los pasos a seguir en un estudio de modelado y simulación extraído de [8] y cuya interrelación se puede observar en la Figura 3-2. Esquemas similares se pueden encontrar en [81], [66] y [36].

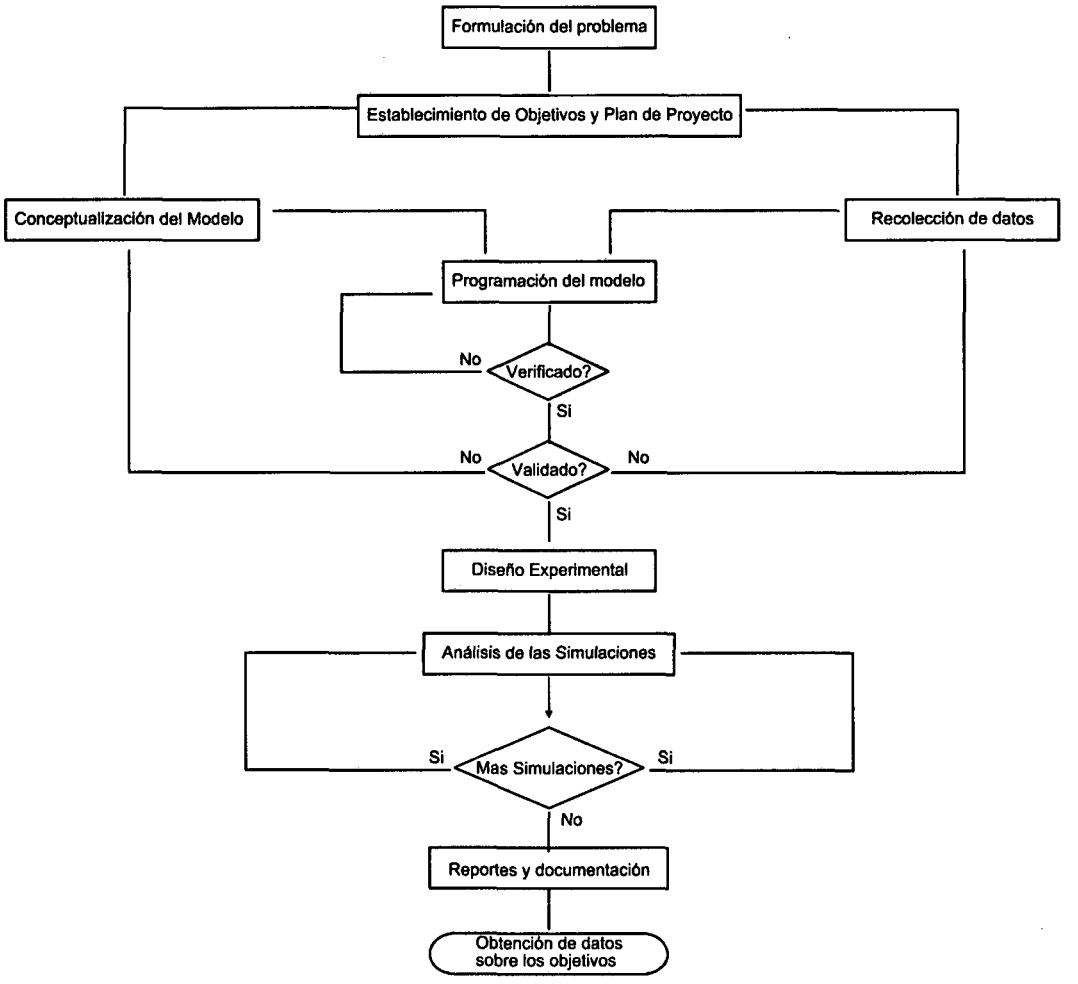


Figura 3-2 Pasos en el estudio de la simulación.

- **Formulación del Problema**: Consiste en describir el problema que se quiere resolver claramente.

- **Establecimiento de los objetivos**: Implica la especificación de aquellos problemas que se quieren resolver con la simulación. Es en este paso donde se debe decidir si un simulador es la mejor alternativa para lograr los objetivos que se han propuesto.

- **Conceptualización del Modelo**: Es importante abstraer correctamente las características esenciales del problema, elaborando y enriqueciendo el modelo hasta obtener resultados útiles.

- **Recolección de Datos**: A medida que se construye el modelo cambian las necesidades de información que retroalimentan su diseño. La selección de la clase de datos relacionados con los objetivos del estudio es un proceso largo, y se debe empezar conjuntamente con las primeras etapas del estudio. También se deben obtener datos estadísticos que nos permitan validar el simulador.

- **Programación del Modelo**: A partir de la conceptualización del modelo y de la obtención de datos sobre el mismo, procedemos a su programación. La programación del modelo puede ser hecha con lenguajes de simulación o utilizando "*software*" de simulación de propósito general.

- **Verificación**: Consiste en comprobar que el programa de computación desarrollado para el modelo de simulación funcione correctamente.

- **Validación**: Determina si el modelo tiene una representación adecuada del sistema real. Se logra mediante el proceso iterativo de comparar el modelo con el comportamiento real del sistema y una vez detectadas las discrepancias utilizarlas para mejorar el modelo.

- **Diseño Experimental**: En este paso se determina el conjunto de experimentos que se simularán. Con frecuencia, éstas decisiones dependen de las ejecuciones que han sido completadas y analizadas, porque proporcionan datos sobre los parámetros de simulación más adecuados a nuestro programa, como por ejemplo, la longitud de los ciclos de simulación, el número de veces que se van a replicar las mismas simulaciones, etc.

- **Simulaciones y Análisis**: El análisis de las simulaciones se hace con el fin de estimar medidas del rendimiento del diseño del sistema que está siendo simulado.

- **Continuar con las simulaciones**: En esta etapa se debe determinar si, basado en el análisis de las simulaciones, se necesitan agregar experimentos adicionales. También se harán en esta etapa los nuevo diseños sobre esos experimentos, si fueran necesarios.

- **Informes y Documentación**: Incluye la documentación del programa y los informes de progreso para tener una cronología de lo que se ha hecho y las decisiones que se han tomado.

- **Implementación del prototipo**: La implementación del sistema diseñado es la etapa final y depende completamente del buen desarrollo de las etapas anteriores.

Estos pasos en el estudio del modelado y la simulación tienen el propósito final de lograr una implementación real del modelo desarrollado, que es posible a partir del programa de simulación, validado y verificado.

Actualmente hay numerosos estudios de simuladores específicamente para el área de computación. El *"software"* de modelado y simulación se está desarrollando principalmente sobre multicomputadores por ser el medio más atractivo para la simulación distribuida debido a que a partir de un modelo dado, se pueden explotar todas las ventajas que ofrecen estas arquitecturas [138].

En este trabajo de tesis hemos abordado dos posibilidades de modelado. La primera es el modelado matemático con resolución analítica donde se intenta capturar el comportamiento del sistema mediante una serie de ecuaciones matemáticas. En nuestro caso nos basamos en un modelo estocástico o probabilístico el cual ofrece unos valores medios de los parámetros a medir del sistema. La segunda posibilidad es el modelado funcional mediante la simulación directa del comportamiento de los diferentes elementos que constituyen el sistema. En este caso, un programa de ordenador "simula" o se comporta como los elementos de la red de interconexión con objeto de representar su comportamiento.

Estos dos modelos utilizan representaciones complementarias del sistema real, tal y como se define en el libro de B. Zeigler [138]. Las representaciones complementarias comprenden las mismas hipótesis sobre el funcionamiento del sistema real, pero las representan de manera diferente. La razón de usar más de una representación es que cada una de ellas puede ser mejor que otra en determinadas aplicaciones. En nuestro caso, veremos como el modelado analítico es más abstracto y rápido que el funcional, mientras que este último es capaz de ofrecer la respuesta temporal completa, incluyendo la transitoria, de la red de interconexión.

En nuestro caso, queremos modelar cómo un mensaje generado por un procesador es inyectado en la red de interconexión, viaja a través de ella atravesando enlaces y encaminadores como los descritos en el capítulo 2, colisiona con otros mensajes y finalmente llega al nodo destino.

A continuación se describen ambos modelos los cuales se ofrecen como herramientas útiles a la hora de estudiar el comportamiento de las redes de interconexión.

3.3 Parámetros de los modelos

Tal y como se ha comentado en el punto de introducción de este capítulo, los modelos de la red de interconexión desarrollados ofrecen el comportamiento dinámico de la red de interconexión cuando una cierta carga de mensajes circula por la red. Los modelos tienen en cuenta la red de interconexión descrita a través de una serie de parámetros, por un lado, y el programa de aplicación, que supone una cierta carga de mensajes sobre la red, por el otro.

El comportamiento de la red, para una carga de mensajes, lo representamos mediante un modelado del sistema, en el que tenemos en cuenta una serie de parámetros reales. A partir de la realidad, elaboramos un modelo simplificado de ella. Este modelo nos permite entender la realidad, simularla y predecir su comportamiento bajo unas ciertas condiciones y unos límites.

De hecho, la resolución del modelo ofrece una foto fija del estado de la red de interconexión para una cierta carga de comunicaciones concreta. De esta foto fija, se puede realizar un análisis del funcionamiento de la red donde se pueden identificar problemas como la aparición de "*hot-spots*" o zonas de máxima contención de mensajes [106].

En este subapartado, se definen los parámetros que los modelos, comunes a ambos, necesitan como entrada. Estos parámetros describen, por un lado, el programa a ejecutar, y, por el otro, la red de interconexión a modelar. El modelo de la red de interconexión se describe mediante su topología, el algoritmo de encaminamiento y las características del encaminador. Todos estos conceptos ya se introdujeron en el capítulo 2. El modelo del programa paralelo se especifica mediante un grafo de comunicación de tareas junto con unos valores de volumen de cómputo y de comunicaciones. Asimismo, para determinar el conjunto de mensajes y por qué caminos circulan por la red, es necesario definir la asignación de tareas a nodos de computo de la red de interconexión.

Finalmente, también se describen los resultados de salida que ofrece el modelo de la red en estudio. Básicamente, los resultados son la latencia que sufren los mensajes en la red bajo una carga de mensajes determinada. A partir de estos resultados se puede extraer otra información como el retardo medio en la red, la curva de latencia/carga, la identificación de "hot-spots" y, en suma, servir como herramienta tanto para el análisis como el diseño de redes de interconexión. La Figura 3-3 muestra las entradas y salidas del modelo de la red de interconexión.

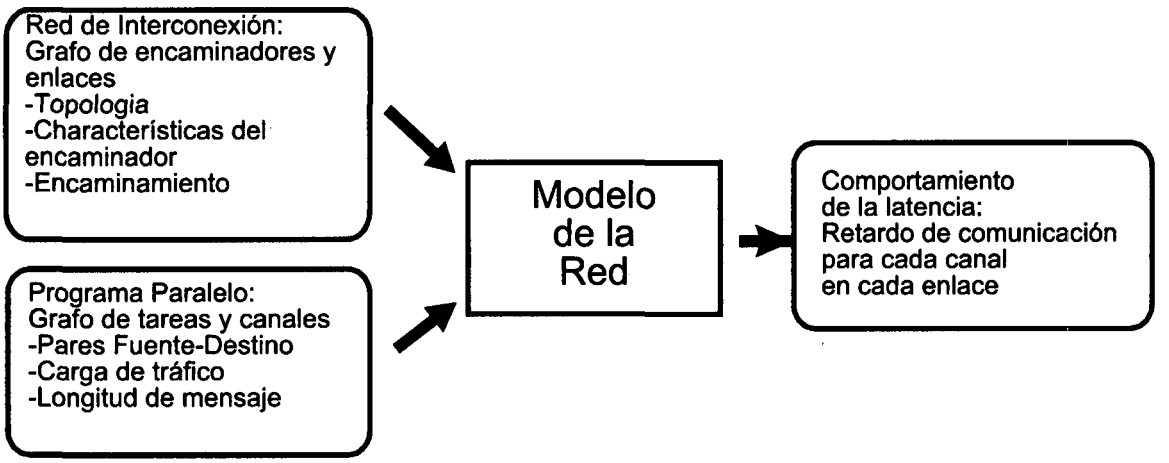


Figura 3-3 Entradas y salidas de los modelos

3.3.1 Parámetros de entrada

3.3.1.1 Red de interconexión

Tal y como se ha definido en el capítulo 2, suponemos una red de interconexión representada como una colección de enlaces y de encaminadores. Suponemos, asimismo, sin que suponga una pérdida de generalidad, que cada nodo de cómputo está enlazado con un único encaminador. Pueden existir encaminadores que tengan enlazados uno, ninguno o más de un nodo de cómputo. Las conexiones entre los diferentes encaminadores forman una cierta topología.

Los modelos consideran cualquier topología con cualquier numero arbitrariamente grande de nodos encaminadores modelados por los parámetros temporales introducidos en el capítulo 2 y que son recordados a continuación. Estos encaminadores pueden utilizarse para formar tanto una red directa como indirecta. El modelo supone que la red esta operada bajo el control del flujo tipo "*wormhole*", porque es la técnica que, cómo se ha mostrado en el capítulo 2, presenta las mejores prestaciones con un costo menor de almacenamiento de mensajes y porque además es la técnica más utilizada hoy en día en los encaminadores comerciales disponibles tales como lo son el *Intel Cavallino* [19], el encaminador del *Cray T3E* [116] o el *SPIDER* de *SGI* [56].

El **algoritmo de encaminamiento** utilizado, definido estáticamente, es también un parámetro de entrada del modelo, pudiéndose especificar cualquier esquema estático de encaminamiento. Esta especificación se puede realizar mediante las técnicas tradicionales de consulta en tabla o la ejecución de un algoritmo. En ambos casos, estas técnicas establecen, para un mensaje en tránsito en un encaminador intermedio dirigido a un nodo destino, el siguiente enlace del encaminador a utilizar.

Asimismo, también se debe especificar para la red en cuestión un conjunto de valores numéricos de ciertos parámetros temporales que definen las operaciones básicas del encaminador. Estos parámetros ya fueron introducidos en el capítulo 2 y se repiten aquí para mayor claridad:

a) Retardo de encaminamiento:

Es el tiempo necesario para tomar una decisión de encaminamiento, es decir, para decidir el enlace de salida del mensaje en función del nodo destino. Esta decisión suele implicar una consulta en una tabla o la ejecución de un cierto cálculo algorítmico.

b) Retardo intraencaminador:

Es el tiempo de atravesar un encaminador. Típicamente, es el tiempo que tarda el mensaje en atravesar el "crossbar" que incorpora el encaminador para conectar enlaces de entrada con enlaces de salida.

c) Retardo interencaminador o de propagación a través del enlace:

Es el tiempo que el mensaje tarda en atravesar el enlace que une dos encaminadores. Viene determinado por las características físicas del cable y los protocolos de transmisión.

d) Anchura del enlace:

Es el número de bits del mensaje que se pueden transmitir simultáneamente en paralelo por el enlace

3.3.1.2 Programa de aplicación

Debido a que suponemos un computador de altas prestaciones con varias unidades de cómputo independientes, la mejor manera de aprovechar esas unidades de cómputo para resolver en el menor tiempo posible un cierto programa de aplicación es dividir el problema en diversas tareas independientes que se comunican y sincronizan entre ellas y distribuirlas por las diferentes unidades de cómputo.

Independientemente del modelo de programación (memoria común, paso de mensajes, etc.) y del nivel de especificación del programa por parte del programador (paralelismo implícito o explícito), siempre tendremos que, finalmente, el programa a

ejecutar será un conjunto de tareas a repartir entre las unidades de cómputo. Esta partición del trabajo total en tareas vendrá especificada por parte del programador de manera explícita o será el propio sistema de desarrollo del programa que la realice a partir de una cierta especificación por parte del programador.

No es objetivo de esta tesis entrar en detalle en discusión en los temas de la concepción, diseño y especificación de los programas de aplicación, pero tanto en el modelo de programación de paso de mensajes como de memoria compartida, el programa paralelo se encuentra dividido en una serie de tareas que se sincronizan y comunican mediante algún mecanismo a través de la red de interconexión.

Consecuentemente, consideramos que el programa de aplicación se modela como un conjunto de tareas que se ejecutan independientemente unas de otras y que se comunican y se sincronizan en función del trabajo a desarrollar entre sí para la resolución de la tarea global por medio del intercambio de mensajes a través de la red de interconexión. Se supone que las tareas realizan un cierto cálculo sobre un conjunto de datos durante un cierto tiempo determinado por el algoritmo en cuestión y entonces realizan alguna operación de envío o recepción de mensajes con otra tarea, y vuelven a realizar otra serie de cómputos hasta la próxima acción de comunicación en una forma cíclica de computo sobre los datos y envío-recepción de mensajes de datos. Esta sincronización e intercambio de mensajes puede venir expresada mediante un lenguaje de programación de paso de mensajes o un lenguaje de mayor nivel de abstracción de memoria común.

Cuando dos tareas se comunican mediante el intercambio de mensajes, decimos que lo realizan a través de un **canal lógico**. Definimos canal lógico, pues, como la entidad a través de la cual se comunican dos tareas. Decimos que una de las tareas envía un mensaje, siendo ésta la tarea origen o fuente del mensaje, y la otra tarea lo recibe, siendo ésta la tarea receptora o destino. El modelo acepta programas en los que el número total de tareas es conocido al inicio del programa, aunque no todas estén activas en todo momento, y que los pares de tareas que se comunican también es conocido.

Definimos el **patrón lógico** de comunicaciones como el conjunto de canales lógicos de comunicación que forman las tareas. Este concepto, patrón lógico de comunicaciones, hace referencia a la combinación de tareas fuente con tareas destino para un conjunto de tareas dado. Aunque en la literatura hay veces que se utiliza el término canal para referenciar la conexión entre elementos, por ejemplo, encaminadores, y otras para denominar la entidad lógica que conecta procesos, aquí reservamos el término de **canal** para la denominar la conexión entre procesos y llamamos **enlace** a las conexiones físicas entre encaminadores de la red de

interconexión. Un canal es una entidad lógica entre la tarea fuente y destino a nivel del programa de aplicación.

Por lo tanto, para representar el programa de aplicación se utiliza un grafo de nodos unidos mediante arcos dirigidos. Los nodos del grafo representan las tareas del programa que se comunican y los arcos representan la comunicación entre las tareas, es decir, los canales lógicos. Cada arco une un par de nodos, estando definidos el nodo origen del arco y el nodo destino, lo que configura la dirección del arco. No hay ninguna restricción por parte de los modelos que hemos desarrollado en el número de tareas ni en el patrón de comunicación de las tareas que puede representar este grafo. Para el modelo analítico el número de tareas debe ser fijo, mientras que para el modelo funcional puede variar a lo largo de la ejecución del programa.

Además de la definición topológica del grafo, se definen una serie de parámetros numéricos para nodos y para arcos con objeto de modelar sus principales características. Estos parámetros son el volumen de cómputo y el volumen de comunicaciones, este último representado por el tamaño del mensaje. A continuación, se comentan cada uno de ellos.

3.3.1.2.1 Volumen de cómputo:

Para cada uno de los nodos de cómputo, que alojan las tareas, se define el volumen de cómputo como el tiempo medio de duración de los cálculos entre dos acciones de comunicación. Haciendo la suposición de que, cada vez que una tarea computa unos ciertos cálculos, lo hace de manera independiente de las demás veces, la duración de esos cálculos se modela mediante un proceso de Poisson con una función estadística exponencialmente distribuida con una cierta media. Esta duración media de los cálculos es precisamente el intervalo entre dos envíos. A este parámetro lo denominamos *MI* ("Message Interval"). De hecho, como el intervalo entre mensajes se define para cada canal lógico de cada tarea, hablaremos del *MI* de canal *C*: "MI^C". Este parámetro también suele recibir el nombre de tráfico o carga de tráfico.

3.3.1.2.2 Tamaño del mensaje:

Es la cantidad de información intercambiada por cada canal en cada acción de comunicación medida en bytes y que nosotros denominaremos "*L*". Cada canal lógico de la aplicación puede tener un tamaño diferente.

La Figura 3-4 muestra un ejemplo de grafo dirigido que modela un programa de aplicación.

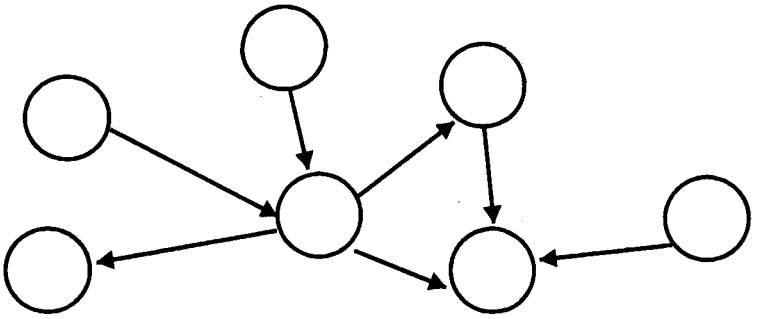


Figura 3-4 Ejemplo de grafo de programa de aplicación

3.3.1.3 Asignación de tareas a nodos de cómputo

Para poder ejecutar el programa, cada una de las tareas que lo componen se asigna a un nodo de cómputo. Suponemos que esta asignación se realiza de manera estática antes de ejecutar el programa para el modelo analítico. Para el modelo funcional esta asignación puede ser dinámica, es decir, puede variar a lo largo de la ejecución del programa. Esta asignación de tareas implica, teniendo en cuenta la topología de la red de interconexión, la asignación de los canales lógicos del programa a enlaces físicos de la red conformando una serie de caminos físicos. Esta asignación es el resultado de que dos tareas que comparten un canal no sean asignadas a nodos de cómputo directamente conectados mediante un único enlace sino que entre ellos halla varios enlaces de manera que los canales lógicos se asignen sobre los enlaces que existan desde el nodo donde se halla la tarea fuente hasta el nodo donde se halla la tarea destino.

Con esta asignación, y junto con el algoritmo de encaminamiento, se consigue determinar qué caminos utilizarán los mensajes por la red. A esta configuración de caminos le llamamos el **patrón físico** de comunicaciones. El patrón físico es la asignación del patrón lógico a enlaces físicos de la red de interconexión. El patrón físico define la carga de comunicaciones sobre la red de interconexión, es decir, la cantidad de mensajes que circulan y los enlaces por los que lo hacen.

La Figura 3-5 muestra la asignación del modelo de un programa al modelo de una red de interconexión.

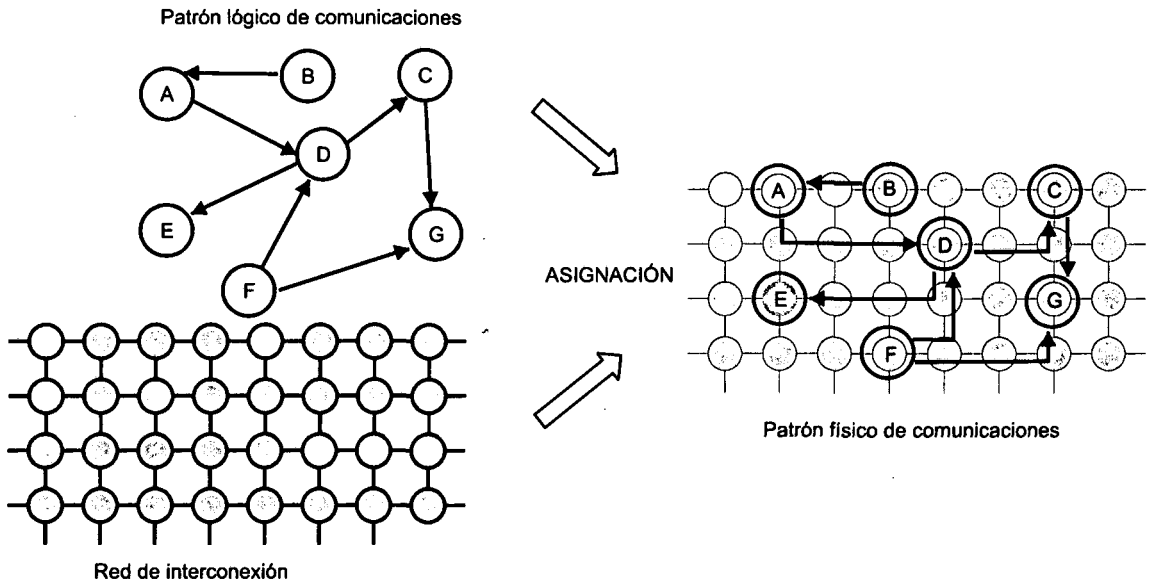


Figura 3-5 Asignación de tareas a procesadores

A partir de esta asignación de canales lógicos a enlaces, el modelo se encarga de caracterizar cuantitativamente cada uno de los canales que aparecen en el sistema para determinar la carga de tráfico de la red. Dos son los parámetros que caracterizan o definen a un canal lógico una vez asignado sobre un conjunto de enlaces desde el punto de vista de carga de tráfico: primero, la frecuencia con que genera los mensajes, o lo que es lo mismo, su inversa, el intervalo de tiempo entre dos mensajes consecutivos, y, segundo, el tiempo de ocupación del canal o retardo de transmisión en ausencia de contención. Estos dos parámetros de la carga de tráfico, que se asocian a cada canal lógico en ausencia de otros canales en la red, los llamaremos **parámetros en origen**. Más adelante veremos cómo modificamos la definición de los parámetros en origen para incluir la influencia de otros canales en la red.

El primero de estos parámetros es un dato de entrada al modelo y ya lo hemos introducido un poco más arriba, MI^C. El segundo, el "tiempo de ocupación del canal" es el tiempo de transmisión del mensaje TD^C en ausencia de otros mensajes en la red y depende del número de enlaces que atraviese, de los parámetros de velocidad definidos para el encaminador, de la anchura y velocidad de los enlaces y de la longitud del mensaje. Algunos de estos parámetros dependen de la asignación de canales a enlaces (el número de enlaces), otros de la red de interconexión (parámetros del encaminador y enlaces) y otros del programa de aplicación (longitud del mensaje).

Para una red operada bajo control del flujo "*wormhole*" el tiempo de transmisión del mensaje para el canal TD^C viene dado por la Ecuación 3-1, tal y como se introdujo en el capítulo 2, donde el primer término de la expresión corresponde a la apertura del camino a través de los n enlaces por parte del "*flit*" cabecera y el segundo a la

transmisión del resto del mensaje en forma de "*pipeline*" desde el nodo origen hasta el nodo destino una vez el camino ya ha sido abierto.

$$TD^C = n * (tr + ts + tw) + max(ts, tw) * \lceil L / w \rceil$$

tr = retardo de encaminamiento *n = Número de enlaces*

ts = retardo intraencaminador *L = Longitud del mensaje*

tw = retardo interencaminador *w = Anchura del enlace*

Ecuación 3-1 Retardo de transmisión de un canal

En nuestros modelos, como en el caso real, supondremos que la contención se produce durante el primer tiempo de apertura del camino y que, una vez abierto, el mensaje tarda un tiempo en transmitirse expresado por la segunda parte de la expresión de TD^C en la Ecuación 3-1. La Figura 3-6 representa estos parámetros.

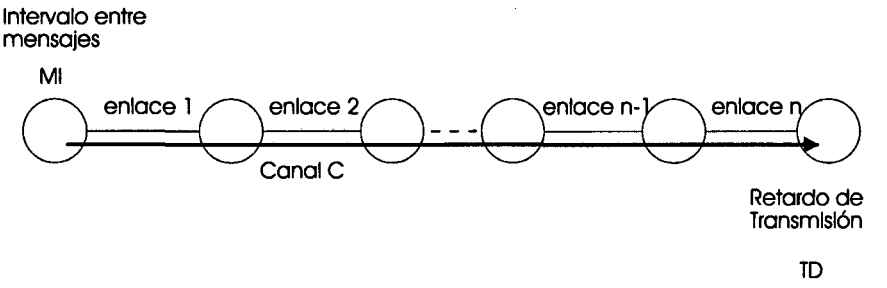


Figura 3-6 Parámetros que caracterizan un canal

Resumiendo, la topología de la red, el algoritmo de encaminamiento, los parámetros de los encaminadores, el conjunto de canales lógicos, la asignación de tareas a nodos de cómputo y la carga de tráfico y el tamaño del mensaje para cada canal son los **parámetros de entrada** al modelo.

3.3.2 Resultados de salida de los modelos

Los modelos evalúan los tiempos de transmisión de los mensajes en presencia de colisiones por la circulación de otros mensajes. Los **resultados de salida** de los modelos son los retardos de comunicación de cada canal lógico en cada uno de los enlaces físicos que atraviesa. A cada uno de los retardos del canal C en el enlace l lo llamaremos D_l^C. Estos retardos son la media de todos los retardos que sufrieron cada uno de los mensajes que pasaron por el enlace l pertenecientes al canal C.

A continuación, el retardo total sobre un canal se obtendrá sumando todos los retardos en cada uno de los enlaces que atraviese el canal según se expresa en la Ecuación 3-2, suponiendo que atraviesa n enlaces numerados desde 1 hasta n.

$$D^C = \sum_{l=1}^{n} D_l^C$$

Ecuación 3-2 Latencia de un canal D^C

Los resultados de salida aparecen en la Figura 3-7, donde se muestran los puntos de colisión de los mensajes en un ejemplo de una red de interconexión. La suma de todos estos retardos es el tiempo que los mensajes gastan en viajar desde el nodo fuente hasta el nodo destino, incluyendo el retardo puro de transmisión TD^C (retardo en ausencia de carga) más el tiempo de contención debido a colisiones con otros mensajes. Al retardo por contención se le suele llamar latencia del mensaje. La contención se produce en cada uno de los puntos de colisión, que son los enlaces de salida compartidos por más de un canal por el que compiten los mensajes. Esta figura resalta, para uno de los canales de comunicación que atraviesa 7 enlaces de comunicación, cada uno de los puntos posibles de colisión, y el tiempo de comunicación formado por la suma del tiempo de transmisión en ausencia de contención más la contención sufrida a lo largo del camino.

Algunos de los índices de comportamiento del sistema que pueden evaluarse a partir de los valores de retardo de los canales calculados son los siguientes:

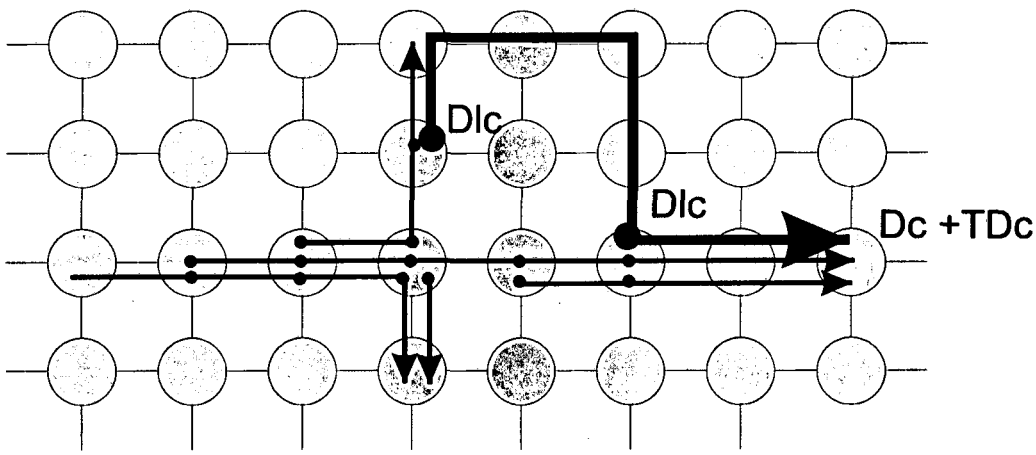


Figura 3-7 Puntos de colisión de los mensajes en la red de interconexión.

3.3.2.1.1 Latencia media

Obtener la latencia media de la red promediando los tiempos de espera de cada canal según muestra la Ecuación 3-3.

$$NetD = \frac{\sum_{\forall c} D^C}{\#\, canales}$$

Ecuación 3-3 Latencia media de la red *NetD*

3.3.2.1.2 Curva latencia media v. carga

Evaluando el modelo para una serie de valores de carga de tráfico de entrada se puede obtener una gráfica latencia v. carga de tráfico.

3.3.2.1.3 Identificación de "*Hot-spots*"

A partir de los retardos calculados en cada enlace, se puede confeccionar un "mapa" de retardos de la red. Con este mapa es posible identificar que puntos o zonas de la red tienen una mayor concentración de mensajes y las razones causantes de tal condición de funcionamiento.

El objetivo final es que con estas herramientas se pueda realizar el análisis y diseño de redes de interconexión y poder evaluar la influencia que tienen los parámetros de diseño de la red junto con las características del patrón de comunicaciones de entrada en el comportamiento dinámico de la red de interconexión.

La Figura 3-8 muestra de manera unificada las entradas y salidas de los modelos desarrollados.

3.4 Modelo Analítico

Como se ha indicado en la sección anterior, los modelos se ocupan de calcular los retardos por colisiones de los mensajes que se envían por los canales en cada uno de los enlaces que atraviesan. A cada uno de los retardos por colisión del canal C en el enlace l se llama D_l^C, tal y como se definió en el punto anterior y se mostró en la Figura 3-7, donde se muestran los puntos de colisión de los mensajes en un ejemplo de una red de interconexión.

El cálculo se basa en el desarrollo de unas ecuaciones estocásticas que representan el comportamiento de los mensajes en la red de interconexión. En la literatura existen

múltiples desarrollos de modelado y simulación matemática de redes de interconexión. Las diferentes aproximaciones se basan desde los modelos de prestaciones [32], el análisis del valor medio [1], la teoría de colas [79], las redes de Petri [68], y el modelo de contención multicamino [94].

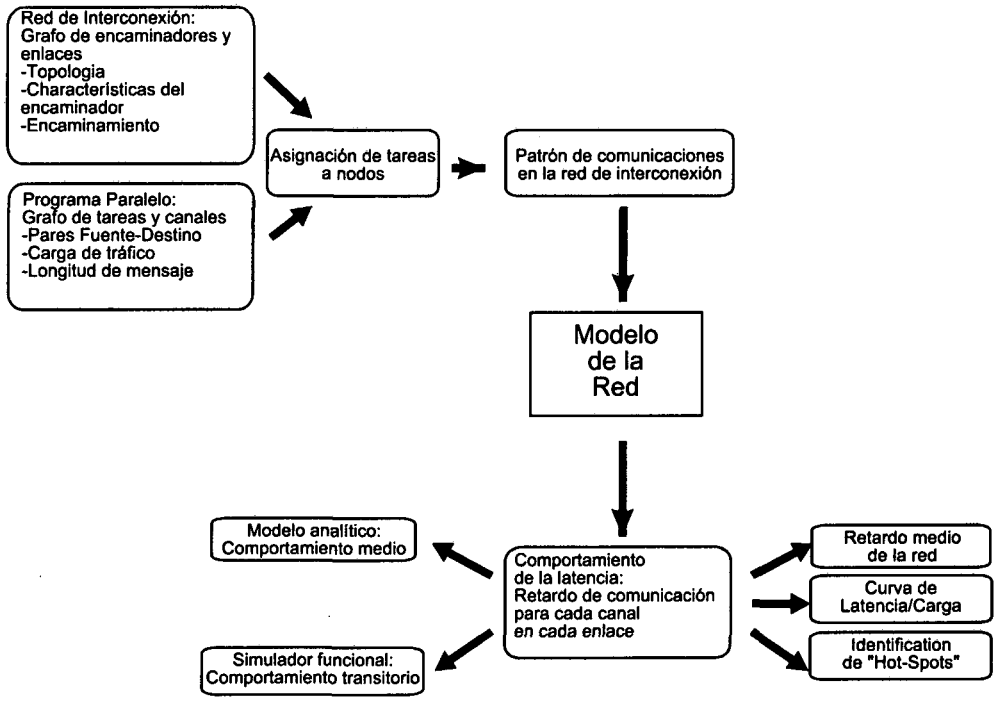


Figura 3-8 Detalle de las entradas y salidas de los modelos

Esta sección se dividirá en dos partes. Primero se deducirán las ecuaciones para calcular el D_l^C para un canal C y un enlace l determinados y a continuación se explicará cómo calcular todos los retardos para todos los canales en todos los enlaces, a partir del cálculo de uno de ellos. Esta estrategia responde a la utilización de una técnica del tipo "divide-y-vencerás" para abordar la resolución de problemas.

3.4.1.1 Cálculo de D_l^C para el canal C en el enlace l

Después de tener cada uno de los canales lógicos, que están asignados sobre los enlaces físicos, caracterizados en origen, supongamos que tomamos un cierto canal C y un enlace al que llamaremos l de los mostrados en la Figura 3-7 y analicemos la naturaleza del retardo de C en el enlace l. Cuando hablamos del enlace l nos referimos asimismo al buffer asociado a ese enlace de salida, donde se esperan los mensajes que no pueden utilizar el enlace por estar éste ocupado por otro mensaje.

En primera vista, el retardo del canal C en el enlace l es debido a la presencia de otros mensajes pertenecientes a otros canales en el mismo enlace, que son con los que tiene que competir por conseguir el enlace. Físicamente, corresponde al hecho de la

colisión entre dos o más mensajes en el encaminador, lo cual provoca que los mensajes se "encolen" en "*buffers*" con el consiguiente aumento del retardo. El cálculo de D_l^C, por tanto, se hace calculando cuánto retardan a C los otros canales en el enlace l. Es cierto que el resto de canales en otros enlaces de la red de interconexión y sus colisiones pueden afectar de manera indirecta a los retardos en el enlace l, porque todas las comunicaciones forman un sistema en que se afectan unos a otros de manera recíproca y transitiva. Por lo tanto, el proceso de cálculo será un proceso global en el que se realizará el cálculo para todos conjuntamente.

Este efecto de influencia indirecta hace que los parámetros en origen que caracterizan a los canales, intervalo entre mensajes y tiempo de transmisión, no se conserven a lo largo de toda su trayectoria y se vean modificados por la acción de otros canales en la red. Esta modificación es debida a que las colisiones aumentan el tiempo de transmisión de cada mensaje por espera en los "*buffers*", por un lado, y limitan la inyección de nuevos mensajes, por el otro ya que los "*buffers*" de inyección no son de tamaño ilimitado.

Para calcular el retardo que provocan los canales presentes en el enlace l sobre el canal C vamos a utilizar una fórmula de cálculo del retardo que provoca un mensaje sobre otro cuando son los dos únicos mensajes presentes en el sistema y sólo existe un punto de colisión entre ambos como muestra la Figura 3-9. Esta fórmula fue desarrollada por Min-Mutka en [94] y su deducción se presenta en un apéndice al final de la memoria.

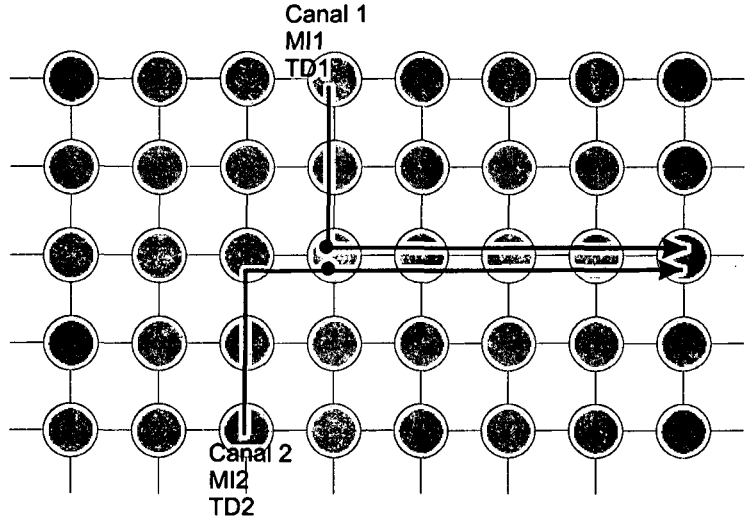


Figura 3-9 Sistema de dos canales

La fórmula Min-Mutka calcula el retardo de transmisión de un canal cuando comparte el camino con otro a partir de los parámetros que los caracterizan a ambos,

intervalo entre mensajes y retardo de transmisión, es decir, devuelve la suma de la contención o tiempo de espera en el enlace de colisión más el tiempo puro de transmisión de uno de los dos canales según se especifique, k=1 o k=2 (Ecuación 3-4)

Retardo de transmisión para el canal 1 ó el canal 2 =
Min-Mutka (MI1, TD1, MI2, TD2,k)

Ecuación 3-4 Latencia entre dos canales

Basándose en el caso simple de dos canales, nosotros queremos extenderlo al encaminador donde colisionan n canales sobre un enlace como el caso general que muestra la Figura 3-10, para una red de grado 4, donde una serie de canales colisionan con el canal *C* en el enlace *l*. El canal *C* proviene de un nodo vecino a través del enlace *l-1* y los demás canales convergen sobre el enlace *l* provenientes del mismo nodo del enlace *l*, en el caso de la inyección desde el procesador, u otros nodos vecinos.

Por lo tanto, la estrategia del cálculo de D_l^C se basa en dos puntos: Determinación de los parámetros de los canales cuando se encuentran en el enlace *l* y reducción de los canales presentes en el enlace *l* que afectan a *C* a un único canal equivalente, para poder aplicar la fórmula de Min-Mutka. Las ideas de calcular un canal equivalente están basadas en las propuestas de los mismos autores desarrolladas en [95] donde se aplican a casos restringidos de patrones simples en mallas 2D.

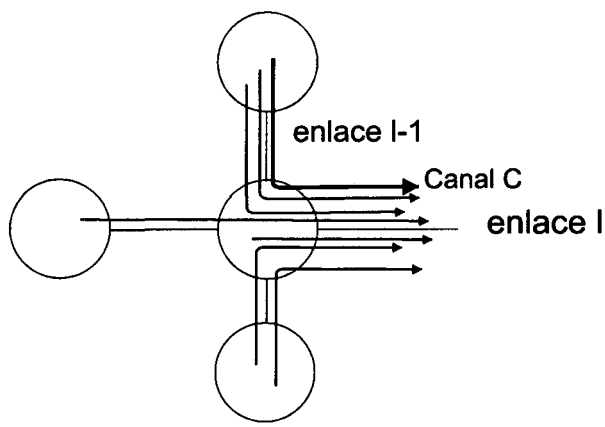


Figura 3-10 Sistema de canales múltiples

Primeramente, haremos dos definiciones sobre los enlaces que atraviesan los canales:

3.4.1.1.1.1 Enlace predecesor del canal C en el enlace l Predecesor (C, l)

Es el enlace que el canal *C* usa antes del enlace *l*, es decir, *l-1*.

3.4.1.1.1.2 Enlace sucesor del canal C en el enlace l, Sucesor (C, l)

Es el enlace que el canal *C* usa después del enlace *l*, es decir, *l+1*.

3.4.1.1.2 Caracterización de los canales en el enlace l

Para determinar los parámetros de un canal *C* en un enlace *l*, debemos recoger sus retardos anteriores y posteriores a ese enlace. La existencia de un canal en un enlace la caracterizaremos mediante dos parámetros similares a los parámetros nominales que ya hemos definido que caracterizan un canal: el número de veces que pasa un mensaje del canal *C* por ese enlace, es decir, el número de mensajes que llegan al enlace por unidad de tiempo y la cantidad de tiempo que lo va a utilizar, es decir, la suma del tiempo en transmitirlo más las colisiones que se encontrará desde ese enlace hasta el destino. Así definimos dos parámetros para cada canal que está presente en el enlace *l*:

3.4.1.1.2.1 Intervalo de mensajes MI_l^C *:*

Es el tiempo transcurrido entre la llegada de dos mensajes consecutivos en el enlace *l* por el canal *C*. Para el primer enlace, el del nodo origen, es simplemente el parámetro en origen MI^C definido como parámetro de entrada del programa al modelo. Para cada uno de los demás enlaces que atraviesa el canal se define como la suma del *MI* original más los retardos sufridos desde el inicio hasta el nodo en cuestión (Ecuación 3-5).

$$MI_l^C = MI^C + \sum_{i=1}^{l-1} D_i^C$$

Ecuación 3-5 Intervalo entre mensajes para el canal *C* en el enlace *l*

Esta definición responde al hecho de que los retardos sufridos retrasan la inyección de nuevos mensajes desde el mismo fuente, porque un mensaje no es inyectado hasta que el mensaje anterior del mismo canal no ha llegado a su destino. Para el caso *l*=1, la definición coincide con el intervalo entre mensajes nominal.

Definimos específicamente el *intervalo entre mensajes total* MI_t^C como el intervalo entre mensajes en el último enlace antes del nodo destino más el tiempo de transmisión en ausencia de carga en la red TD^C tal y como muestra la Ecuación 3-6.

$$MI_t^C = MI^C + \sum_{l=1}^{n} D_l^C + TD^C$$

Ecuación 3-6 Intervalo entre mensajes total para el canal *C*

3.4.1.1.2.2 Retardo de transmisión TD_l^C *:*

Es la suma del retardo de transmisión nominal más todos los retardos desde el enlace *l* hasta el destino. Es, como se ha dicho, el tiempo que tardará el mensaje en atravesar, y por tanto, en liberar el enlace *l*.

$$TD_l^C = \sum_{i=l}^{n} D_i^C + TD^C$$

Ecuación 3-7 Retardo de transmisión del canal *C* en el enlace *l*

La Figura 3-11 muestra los conceptos de intervalo entre mensajes y retardo de transmisión del canal *C* en el enlace *l*.

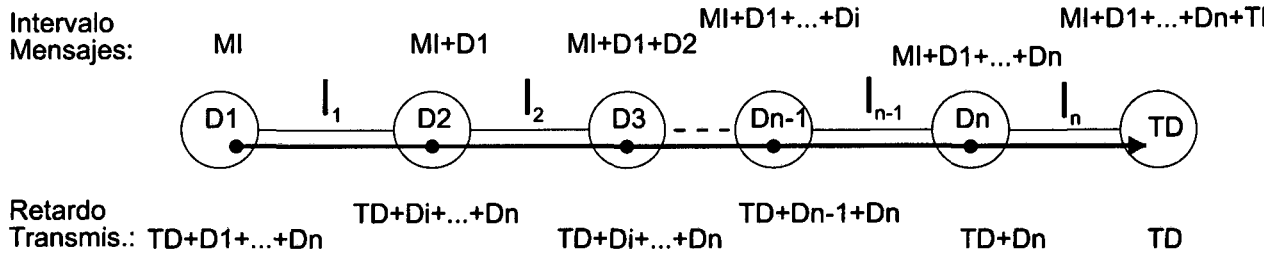


Figura 3-11 Intervalo entre mensajes y retardo de transmisión en el enlace *l*

3.4.1.1.3 Cálculo del canal equivalente:

Para reducir varios canales presentes en un mismo enlace a un canal equivalente, debemos tomar en consideración qué significa el hecho de que varios canales coincidan sobre un mismo punto con respecto de los parámetros de intervalo entre mensajes y tiempo de transmisión. En concreto, lo que buscamos es calcular el Intervalo entre mensajes, *MIeq*, y retardo de transmisión, *TDeq*, del canal equivalente a partir de los *MIs* y *TDs* de los canales en cuestión que forman el equivalente.

La idea intuitiva es considerar que los canales se comportan como un sistema físico, por ejemplo, un circuito de resistencias eléctricas sometido a un cierto voltaje, y asociarlos aplicando los mismos principios que se utilizan en esos sistemas físicos.

En este sentido, es evidente que con respecto del intervalo entre mensajes, cuanto mayor número de canales se asocien, menor será el intervalo entre mensajes resultante. Este hecho puede asimilarse a la asociación de resistencias en paralelo, donde la resultante equivalente es el inverso de la suma de los inversos de los valores de los parámetros.

Por otro lado, cuanto mayor sea el número de canales, mayor será el tiempo de ocupación del canal resultante. Este fenómeno es similar a la asociación de resistencias en serie, donde cada resistencia aumenta la resistencia total. De todas formas, en este caso, no todos los canales contribuyen por igual con su tiempo de transmisión al tiempo de transmisión equivalente, si no que la contribución viene ponderada por el número de mensajes inyectados por unidad de tiempo, es decir, pesa mas el tiempo de transmisión de los mensajes del canal con mayor tasa de inyección, porque contribuye más veces con ese tiempo de ocupación. Por lo tanto, en este caso deberemos sumar los tiempos de ocupación de los canales a asociar, pero dándole a cada uno un peso dependiendo del número de mensajes que cada canal aporta.

Finalmente, el intervalo entre mensajes en el enlace l MI_l^{Eq} se calcula como el intervalo entre mensajes total menos el retardo de transmisión desde el enlace l hasta el destino final.

A partir de esta explicación, podemos deducir las expresiones para el intervalo entre mensajes total MI_t^{Eq}, el retardo de transmisión en el enlace l TD_l^{Eq} y el intervalo entre mensajes en el enlace l MI_l^{Eq} correspondientes al canal equivalente (Ecuación 3-8)

$$MI_t^{Eq} = \left(\sum_{C \in Eq} {}^{1}\!/_{MI_t^C}\right)^{-1}$$

$$TD_l^{Eq} = \frac{\sum_{C \in Eq} \left({}^{1}\!/_{MI_t^C} * TD_l^C\right)}{\sum_{C \in Eq} \left({}^{1}\!/_{MI_t^C}\right)}$$

$$MI_l^{Eq} = MI_t^{Eq} - TD_l^{Eq}$$

Ecuación 3-8 Parámetros de caracterización del canal equivalente

Veamos estos conceptos aplicados a un ejemplo sencillo para una mejor comprensión. Supongamos que tenemos tres canales *C1*,*C2* y *C3* cuyos parámetros en el enlace *l* son:

Canal *C1*: *MI1*=20; *TD1*=5;

Canal *C2*: *MI2*=50; *TD2*=4;

Canal *C3*: *MI3*=18; *TD3*=6;

Entonces, aplicando la Ecuación 3-8 obtenemos:

$MI_l^C\ eq = (1/20+1/50+1/18)^{-1} = 7{,}96$

$TD_l^{Eq}\ eq = (1/20*5+1/50*4+1/18*6) / (1/20+1/50+1/18) = 5{,}28$

$MI_l^{Eq} = MI_l^C\ eq - TD_l^{Eq}\ eq = 7{,}96 - 5{,}28 = 2{,}68$

A partir de ahora, ya contamos con las ecuaciones necesarias (caracterización del canal en el enlace *l* y cálculo del canal equivalente) para calcular la influencia del canal equivalente de los canales presentes en el enlace *l* sobre el canal *C*, al que se le dedica el punto siguiente.

3.4.1.1.4 Ecuaciones de cálculo de D_l^C:

Suponemos que hemos tomado un cierto canal *C* y un cierto enlace *l* y que tenemos *C* bloqueado en *l* debido a la competencia con otros canales que inciden sobre *C*, los cuales proceden de otros caminos, comunes o diferentes y continúan por otros caminos, parte común y parte diferente. Esta es la situación de un sistema de múltiples canales mostrada en la Figura 3-10. Ahora, debemos estudiar las relaciones entre los canales que inciden sobre el enlace *l*.

Para ello, clasificaremos los canales en el enlace *l* respecto al canal *C* de la siguiente manera, según las definiciones de enlace predecesor y sucesor dadas anteriormente:

3.4.1.1.4.1 Canales hermanos

Son aquellos canales cuyo predecesor es igual al del canal *C*, es decir,

D es hermano de *C* en el enlace *l* <=> Predecesor(D, *l*) = Predecesor (*C*, *l*)

3.4.1.1.4.2 Canales primos

Son aquellos canales cuyo predecesor no es igual al del canal *C*, es decir,

D es primo de *C* en el enlace *l* <=> Predecesor (D, *l*) <> Predecesor (*C*, *l*)

La Figura 3-12 muestra los canales primos y hermanos para el canal *C* del ejemplo.

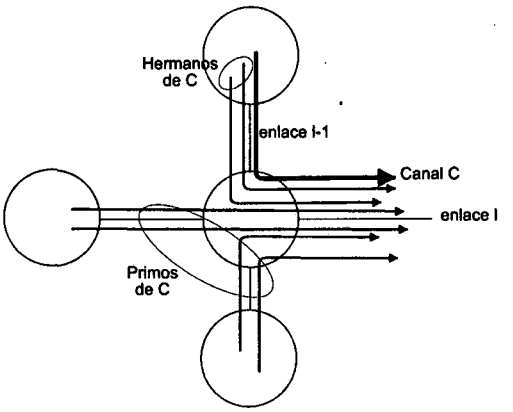


Figura 3-12 Relaciones entre los canales

Con esta clasificación hacemos dos observaciones con respecto a las relaciones de los canales en el enlace *l*, para ver como influencian los canales sobre *C* en el enlace *l*.

- ✓ La primera observación hace referencia a los enlaces previos al enlace *l* y es la siguiente: el retardo D_l^C del canal *C* es debido sólo a la competencia del canal *C* contra sus canales primos. El canal *C* no compite con sus canales hermanos en el enlace *l* porque estos canales hermanos están bloqueados antes del enlace *l*, digamos, *l-1* por el propio canal *C* presente en el enlace *l* (Figura 3-13) Esto es así porque estamos suponiendo un control del flujo de tipo "*wormhole*" para la red y esto implica que una vez un canal ocupa un enlace, lo hace de manera exclusiva todo el tiempo que tardan en pasar todos y cada uno de los "*flits*" que lo componen. La consecuencia de esto es que sólo se debe calcular la influencia sobre *C* de sus canales primos en el enlace *l*.

- ✓ La segunda observación hace referencia a los enlaces posteriores al enlace *l*. Después del enlace *l*, suponemos que no hay mensajes de *C* ya que éste esta bloqueado en el enlace *l*. Por la observación anterior, se deduce, pues, que el enlace *l* estará ocupado por un canal primo de *C*, que está pasando hacia *l+1* y los enlaces posteriores. La segunda observación es, entonces, que los canales hermanos de *C*,

sin embargo, sí que pueden tener mensajes anteriormente inyectados que ya habían pasado por el enlace *l* antes que el canal *C* se bloquease en *l* y que ahora están ocupando una parte posterior del camino, por ejemplo, el enlace *l+1* (Figura 3-14) Estos mensajes de los canales hermanos de *C* pueden colisionar con los mensajes de los canales primos de *C* después del enlace *l*, provocando que cambien los parámetros de los canales primos de *C* en el enlace *l*, concretamente, aumentando su retardo de transmisión.

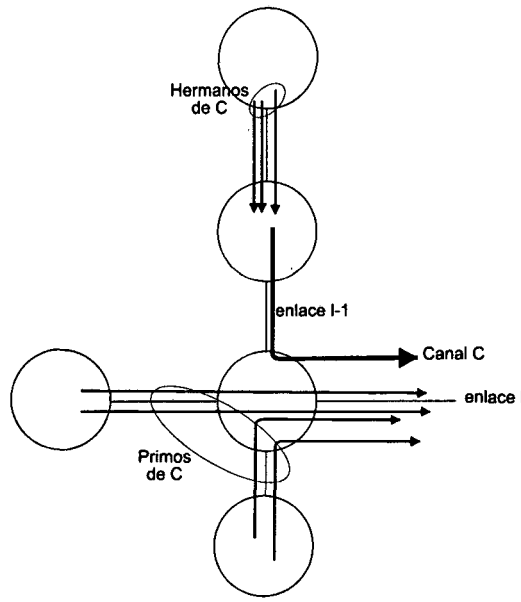


Figura 3-13 Canales hermanos bloqueados antes del enlace *l*

Por lo tanto, la estrategia de cálculo se basará en los siguientes cuatro puntos:

4. Reducir los canales primos de *C* a un canal primo equivalente calculando MI_l^{cs} y TD_{l+1}^{cs} con las fórmulas de canal equivalente aplicadas a los canales primos. (Figura 3-15)

5. Reducir los canales hermanos de *C* a un canal hermano equivalente calculando MI_l^{br} y TD_{l+1}^{br} con las mismas fórmulas que en el caso anterior aplicadas a los canales hermanos. (Figura 3-15)

6. Calcular la influencia, usando la formula de Min-Mutka, en el retardo de transmisión de los hermanos sobre los primos **después** del enlace *l*. Con este cálculo obtendremos lo que llamamos la influencia de la "familia" de *C* (primos directamente y hermanos indirectamente) sobre *C* en el enlace *l* (Figura 3-16) Con este cálculo hallamos MI_l^f y TD_l^f. MI_l^f es simplemente MI_l^{cs} ya que la tasa de generación de mensajes de los primos de *C* en el canal *l* no se ve afecta

por los hermanos de *C*. TD_l^f es el resultado de la fórmula de Min-Mutka (Ecuación 3-9).

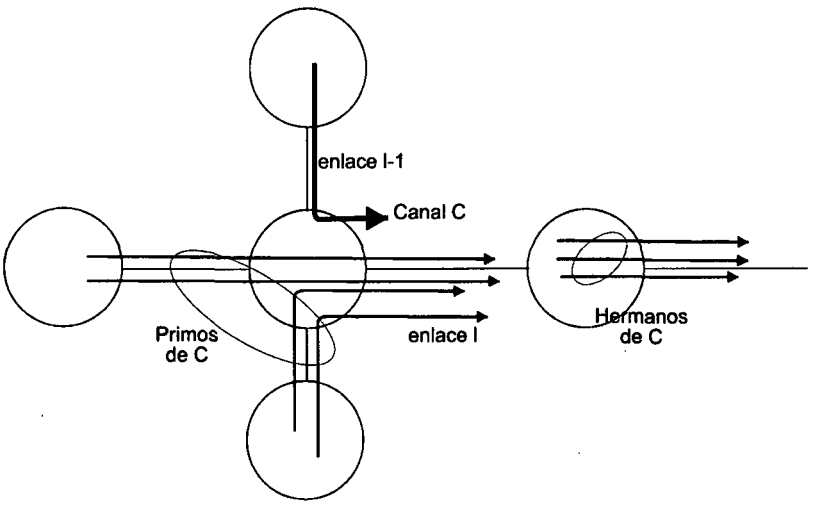


Figura 3-14 Canales hermanos presentes después del enlace *l*

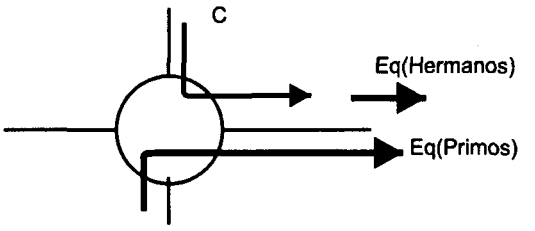


Figura 3-15 Reducción a canales equivalentes

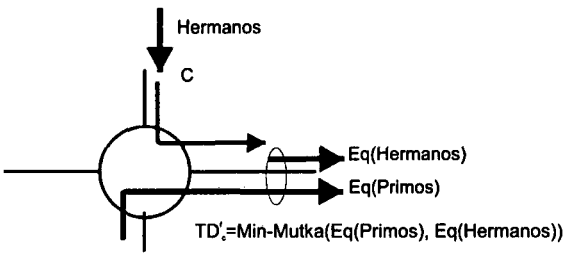


Figura 3-16 Cálculo del canal "familia" de C

$$MI_l^f = MI_l^{cs}$$

$$TD_l^f = Min - Mutka((MI_l^{cs}, TD_{l+1}^{cs})(MI_l^{br}, TD_{l+1}^{br}),2)$$

Ecuación 3-9 Cálculo de la familia equivalente

7. Calcular la influencia usando la formula de Min-Mutka en el retardo de transmisión del canal equivalente "familia de *C*" sobre *C* en el enlace *l* (Figura 3-17) Este resultado final, que utiliza los parámetros de *C*, MI_l^C y TD_{l+1}^C, y los parámetros de su "familia", MI_l^f y TD_l^f, es el D_l^C final que estábamos buscando (Ecuación 3-10)

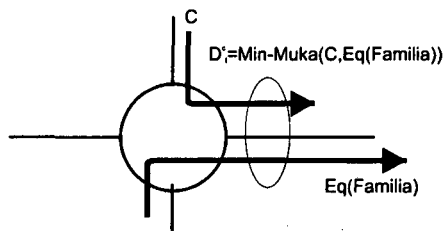


Figura 3-17 Estrategia de cálculo de D_l^C

$$D_l^C = Min - Mutka((MI_l^C, TD_{l+1}^C)(MI_l^f, TD_l^f),1)$$

Ecuación 3-10 Retardo del canal *C* en el enlace *l*

La Figura 3-18 muestra un esquema del procedimiento general de cálculo. En este procedimiento hemos visto dos herramientas: la caracterización de cada uno de los canales en cada enlace y el cálculo del canal equivalente; unas definiciones: enlace predecesor, enlace sucesor, canal hermano y canal primo y una serie de cálculos para obtener el retardo D_l^C a partir de la influencia de los demás canales presentes en el enlace *l*: Canal primo equivalente, canale hermano equivalente, canal familia equivalentey cálculo de D_l^C final.

3.4.1.2 Cálculo de D_l^C para todos los canales en todos los enlaces

Hasta ahora hemos visto, en el punto anterior, cómo calcular el retardo que sufría un canal en un enlace. No obstante, recordemos que nuestro objetivo era calcular todos los retardos en todos los enlaces. Para realizarlo, la estrategia de cálculo será plantear todas las ecuaciones pertenecientes a todos los canales en todos los enlaces y resolverlas de manera secuencial una detrás de otra.

Figura 3-18 Esquema resumen del cálculo de D_l^C

El problema que aparece es que, como hemos visto en el apartado anterior, el cálculo del D_l^C para un canal en un enlace determinados implica usar los intervalos entre mensajes y retardos de transmisión de otros canales en ese y/o otros enlaces. Pero, precisamente, estos parámetros dependen de los D_l^C de los canales vecinos, que aún no hemos calculado

Por lo tanto, se generan dependencias circulares donde unos se necesitan a otros para calcularse de manera recíproca. Hace falta adquirir una estrategia de cálculo para resolver las dependencias circulares. La solución adoptada es, entonces, repetir los cálculos para todos los D_l^C iterativamente hasta que la diferencia de los resultados entre una iteración y la siguiente se considera suficientemente pequeña como para parar de iterar. De este modo, se obtienen unos valores finales para los retardos.

Esta estrategia se considera válida porque supone calcular unos valores de retardos usando unos datos de entrada que no son adecuados, para obtener un primer resultado. Este resultado representa en parte el comportamiento del sistema, sin tener en cuenta la influencia real de los vecinos, que aún no se conoce, sino sólo una influencia nominal. A continuación, el segundo cálculo se basa en los resultados anteriores, que si representan unos retardos aproximados, para hallar unos nuevos resultados mas aproximados. De esta manera sucesiva, se va construyendo la solución hasta que se llega a los resultados correctos. En este punto, una nueva iteración no crea unos nuevos valores diferentes a la anterior porque ya se tiene en cuenta toda la influencia real de los vecinos.

Para la primera iteración se toman valores de los intervalos entre mensajes y retardos de transmisión sin tener en cuenta ningún retardo de contención, es decir, usando los valores nominales en todas las fórmulas.

A continuación, se muestra un resumen de los símbolos utilizados (Tabla 3-1), las ecuaciones presentadas (Tabla 3-2) y el pseudocódigo (Tabla 3-3) para el cálculo del modelo analítico.

Símbolos utilizados en el modelo analítico

MI^C **Intervalo entre mensajes del canal C en ausencia de contención**

TD^C **Retardo de transmisión del canal C en ausencia de contención**

D^C **Retardo del canal C**

D_l^C **Retardo del canal C en el enlace l**

MI_l^C **Intervalo entre mensajes del canal C en el enlace l**

MI_t^C **Intervalo entre mensajes total del canal C**

TD_l^C **Retardo de transmisión del canal C en el enlace l**

MI_t^{Eq} **Intervalo entre mensajes total del canal equivalente**

TD_l^{Eq} **Retardo de transmisión del canal equivalente en el enlace l**

MI_l^{Eq} **Intervalo entre mensajes del canal equivalente en el enlace l**

Canal equivalente = canales hermanos/canales primos/canal familia

Tabla 3-1 Símbolos utilizados en el modelo analítico

Ecuaciones del Modelo Analítico

Caracterización del canal C en el enlace l:

$$MI_l^C = MI^C + \sum_{i=1}^{l-1} D_i^C$$

$$MI_l^C = MI^C + \sum_{l=1}^{n} D_l^C + TD^C$$

$$TD_l^C = \sum_{i=l}^{n} D_i^C + TD^C$$

Cálculo del canal equivalente

$$MI_l^{Eq} = \left(\sum_{C \in Eq} {}^{1}\!/_{MI_l^C}\right)^{-1}$$

$$TD_l^{Eq} = \frac{\sum_{C \in Eq} \left({}^{1}\!/_{MI_l^C} * TD_l^C\right)}{\sum_{C \in Eq} \left({}^{1}\!/_{MI_l^C}\right)}$$

$$MI_l^{Eq} = MI_t^{Eq} - TD_l^{Eq}$$

Cálculo del canal familia

$$MI_l^f = MI_l^{cs}$$

$$TD_l^f = Min - Mutka((MI_l^{cs}, TD_{l+1}^{cs})(MI_l^{br}, TD_{l+1}^{br}),2)$$

Cálculo del retardo D_l^C

$$D_l^C = Min - Mutka((MI_l^C, TD_{l+1}^C)(MI_l^f, TD_l^f),1)$$

Tabla 3-2 Ecuaciones del Modelo Analítico

Algoritmo Modelo Analítico

Mientras la diferencia entre dos iteraciones sea mayor que la tolerancia hacer

Para cada enlace en el que haya canales presentes hacer

1.Caracterizar los canales presentes en el enlace

2.Para cada canal presente en el enlace hacer

2.1.Calcular canal hermano equivalente

2.1.1.Determinar canales hermanos

2.1.2.Calcular canal hermano con la formula del canal equivalente

2.2.Calcular canal primo equivalente

2.2.1.Determinar canales primos

2.2.2.Calcular canal primo con la formula del canal equivalente

2.3.Calcular canal familia equivalente

2.4.Calcular retardo D_l^C

Tabla 3-3 Algoritmo Modelo Analítico

El programa que simula este modelo acepta un fichero de tipo texto que describe los parámetros de entrada. En este fichero existe una línea por cada canal de comunicación. Cada línea contiene un identificador del canal, la frecuencia de generación de mensajes, el tiempo de ocupación del canal y los enlaces que ocupa. La Tabla 3-4 muestra un ejemplo para el caso del patrón "*ring*" mostrado en la Figura 3-19.

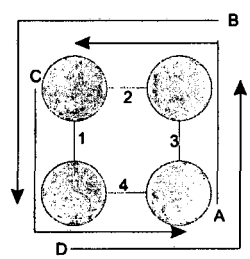


Figura 3-19 Patrones de comunicaciones Ring

Canal	MI^C	TD^C	*Enlaces*	
A	50	25	3	2
B	50	25	2	1
C	50	25	1	4
D	50	25	4	3

Tabla 3-4 Fichero de entrada de datos para el patrón "ring"

Los resultados de salida obtenidos por la ejecución del modelo para el patrón de ejemplo "*ring*" se muestran en la .En este fichero se informa del nombre del fichero y fecha de ejecución, del número de canales ("*paths*"), del número de iteraciones necesarias para converger ("n"), y se incluye una línea para cada canal con el retardo D^C *("L")* que sufre.

```
#----------file: ring2.aout; date: Sun Jul 16 23:47:14 2000

# number of paths: 4; n= 10

# path  1 (MTBS=  50.00 Rd=  25.00 Pth=  0 3 2          L=   16.102

# path  2 (MTBS=  50.00 Rd=  25.00 Pth=  0 2 1          L=   16.102

# path  3 (MTBS=  50.00 Rd=  25.00 Pth=  0 1 4          L=   16.100

# path  4 (MTBS=  50.00 Rd=  25.00 Pth=  0 4 3          L=   16.100
```

Tabla 3-5 Fichero de salida de datos para el patrón "ring"

3.5 Modelo funcional

En este punto se introduce el modelo y el simulador funcionales desarrollados. Primeramente se hace un recorrido por las diversas técnicas de implementación de un modelo funcional mediante un simulador como son la simulación conducida por tiempo y la simulación conducida por eventos. A continuación se estudia el trabajo existente en la literatura sobre la simulación de redes de interconexión y finalmente se presenta el simulador desarrollado en este trabajo.

3.5.1 Simulación funcional

En nuestra área de trabajo, las técnicas de simulación pueden dividirse en simulaciones conducidas por tiempo y simulaciones conducidas por eventos. A continuación se comentan brevemente cada una de ellas.

3.5.1.1 Simulaciones conducidas por tiempo:

El elemento básico de estas simulaciones es el reloj. El cambio del estado de las entidades del sistema modelado se realiza controlado por el reloj central. En cada iteración del simulador se avanza el tiempo de simulación y se ejecutan las acciones correspondientes ciclo a ciclo. Este tipo de simulador es adecuado si el número de eventos que se simulan por ciclo es alto.

3.5.1.2 Simulaciones conducidas por eventos:

En este tipo de simulador los eventos son los que hacen avanzar el tiempo de simulación y determinan las acciones que se van a realizar y el cambio del estado de las entidades del sistema mediante una "lista de eventos futuros." Esta lista se ordena de acuerdo al tiempo de los eventos. Si el intervalo de tiempo entre los eventos que se simulan es grande, este tipo de simulador avanza de manera más rápida que el anterior, que es totalmente independiente de las tareas a realizar.

Otra clasificación interesante, dependiente del computador donde se realizan los experimentos, es dividir las simulaciones en **secuenciales** o **paralelas**, según su forma de ejecución. La simulación secuencial es la forma tradicional en la que toda la simulación es asumida por un único procesador.

En la simulación paralela, se pretende dividir la tarea de simular en un conjunto de subtareas paralelas y distribuirlas entre un conjunto de procesadores. La tarea de paralelización no es trivial. Una simulador conducido por tiempos se puede paralelizar como procesos que avanzan de forma síncrona un ciclo de tiempo a la vez. El problema principal que surge es que se debe sincronizar el reloj de este conjunto de procesos ejecutados en paralelo. En el caso de los algoritmos de simulación conducidos por eventos, la idea es paralelizar la lista de eventos a simular y distribuirla entre los procesadores.

La cuestión que surge en este caso es que no es posible paralelizar la lista de eventos sin tomar en cuenta los errores de causalidad, provocados cuando el procesamiento de un mensaje genera nuevos eventos que debieron ser simulados con anterioridad a otro mensaje ya procesado. En lugar de mantener una estructura global

(como un reloj o una lista de eventos futuros), en este tipo de simulación, se asignan "procesos lógicos" a los elementos que componen el sistema, cada uno de ellos con sus propias listas locales de eventos futuros y su propio reloj.

Dependiendo de la forma como se elige el evento próximo a ejecutar en cada proceso lógico, los simuladores paralelos se dividen a su vez, en **conservadores** y **optimistas**. Los enfoques conservadores evitan estrictamente la posibilidad de que cualquier error de causalidad ocurra. Se basan en estrategias que determinan la seguridad de procesar un evento, es decir, que no pueda ser afectado por ningún otro evento. Los enfoques optimistas utilizan una aproximación de "detección y recuperación", permiten un mayor avance de la simulación en cada proceso lógico, y si se detectan errores de causalidad, se invoca un mecanismo de marcha atrás para recuperar el estado anterior del sistema, almacenado previamente, y continuar desde este punto teniendo en cuenta la nueva situación.

Sobre la simulación paralela de eventos discretos PDES (*Parallel Discret-Event Simulation*), en [55] se hace un estudio de los enfoques y técnicas existentes. En este trabajo se examinan las características de métodos optimistas y conservadores, y también se comenta el alcance de la PDES en los diferentes tipos de simulación de aplicaciones paralelas de propósito general en tiempo real. De manera práctica, en [96] se explica una simulación distribuida por eventos, que se lleva a cabo en una red de procesadores con capacidades de comunicación de mensajes asíncronas. En [111] también se hace una simulación paralela de eventos. Se presentan un conjunto de experimentos y sus resultados, usando el algoritmo de simulación distribuida de Chandy-Misra [25] para simular modelos de redes de colas. Los parámetros de este estudio incluyen probabilidades de encaminamiento y topología, número de procesadores y la asignación de servidores a la red de colas en los procesadores.

También son importantes los estudios orientados a la comparación de simuladores secuenciales y paralelos. En [91] se evalúan tres simuladores: secuencial conducido por eventos, secuencial conducido por tiempo y un simulador paralelo conservador conducido por eventos enfocados hacia arquitecturas masivamente paralelas, resaltando sus diferencias, ventajas e inconvenientes.

Por último, pero no menos importante, también encontramos el desarrollo de facilidades para desarrollar el trabajo de simulación, ya que proveen nuevas herramientas e integran herramientas ya existentes en un único ambiente. Permiten trabajar al usuario con interfaces gráficas para la creación de ficheros de entradas y para el manejo de ficheros de salida. Un ejemplo de este tipo de herramientas es OpenSim [98].

3.5.2 Los simuladores funcionales de redes de interconexión

Al optar por confeccionar un simulador para medir el comportamiento de los mensajes en una red de interconexión, es necesario revisar los trabajos que se han hecho actualmente sobre el tema, donde encontramos simuladores sencillos como SNS [104], un simulador conducido por tiempo que simula mallas de dimensión *d* y raíz *R*, con encaminamiento estático, y que con pequeñas modificaciones permite la simulación de caminos múltiples utilizando un encaminamiento adaptativo y el manejo de *"buffers"* finitos.

Actualmente, encontramos diferentes trabajos relacionados con el análisis y evaluación de redes de interconexión enfocados a arquitecturas paralelas, apropiados para estudiar el comportamiento de los mecanismos de encaminamiento de mensajes (adaptativos frente a estáticos), sobre diferentes topologías (malla, toro y *"square midimew"*) y variando el control de flujo (*"cut-through"*, *"store-and-forward"* y *"wormhole"*).

Ejemplos de estos trabajos son los simuladores secuenciales conducidos por tiempo como en [6], con el propósito de lograr las medidas de rendimiento, o bien implementando simuladores conducidos por eventos orientados a objetos que permiten el modelado de diferentes combinaciones de estrategias de conmutación, topologías, modelos de tráfico y variaciones en la arquitectura como *Multisim* [90] y PP-MESS-SIM [112], un simulador diseñado para evaluar las redes de multicomputadores. En *SMART (Simulator of Massive Architectures and Topologies)* [105] se amplía el rango de topologías estudiadas, por ejemplo, *"k-ary n-trees"*, *"k-ary n-butterflies"* y *"crossbars"* completos, y los algoritmos de encaminamiento (se han probado *Chaos* [17] y la metodología de Duato[39] con dos, cuatro y seis canales virtuales). Todos estos simuladores proveen diferentes herramientas de análisis.

En [93] se presenta una simulación paralela de una red de encaminamiento de mensajes con técnica de control de flujo *"cut-through"*, encaminamiento estático y topología toroidal para computadores paralelos de memoria distribuida. Para su análisis, los autores usan un simulador por eventos secuencial clásico y un simulador paralelo con una estrategia de sincronización conservadora que permite reducir el tiempo de ejecución de las simulaciones. En otro trabajo de los mismos autores, [92], se analizan los factores que pueden ayudar a mejorar las prestaciones.

Además, encontramos herramientas para el estudio de multiprocesadores que evalúan trazas generadas por programas paralelos generando estimaciones de rendimiento y medidas de calidad para la redes de interconexión cuyas topologías

pueden ser reconfigurables, como el simulador PEPE *(Programming Environment for Parallel Execution)* desarrollado en [61].

También se han realizado experimentaciones para estudiar los problemas de simulación en el ámbito funcional de las necesidades de almacenamiento de los encaminadores y del tipo de control de flujo, como en [22]. En este trabajo se utilizan redes tipo malla, encaminamiento estático, control de flujo *"cut-through"* y *"wormhole"* y se concluye que para obtener medidas de rendimiento confiables se deben utilizar simuladores que reflejen la arquitectura específica de manera que no sea necesario extraer datos temporales de implementaciones concretas para trasladarlas al simulador. También se señalan los inconvenientes de hacer simulaciones a nivel *"hardware"* de redes de tamaño medio o alto.

3.5.3 Diseño del simulador netsim

Después de la explicación teórica de las técnicas de simulación y de los trabajos relacionados en la literatura, abordamos en este punto el diseño del simulador realizado para evaluar la técnica DRB. Una versión preliminar de este simulador fue desarrollada como parte del trabajo experimental para optar al grado de Master en Arquitectura de Ordenadores y Procesamiento Paralelo de I. Garcés Botacio. Parte de la información de este capítulo esta extractada de la memoria de ese trabajo realizado [57].

La primera versión no incorporaba la técnica de encaminamiento adaptativa ni la técnica de detección y recuperación del *"deadlock"*, aspectos que sí que se incorporan en la versión actual, además de múltiples detalles en la flexibilización de la simulación y la extracción de resultados. En este punto se muestra una versión más evolucionada y versátil del mismo. Primeramente, se presentan las necesidades y los objetivos que se buscan satisfacer, y a continuación, se describe en detalle el diseño del simulador en sí. Entre los objetivos principales para el simulador se debe considerar la flexibilidad, velocidad y portabilidad.

El simulador tiene como objetivo fundamental el estudio del comportamiento de una red de interconexión de paso de mensajes. Concretamente, nos hemos centrado en la medida de la latencia media de los mensajes en función del tráfico y en la capacidad máxima (*"throughput"* máximo) de gestión de mensajes en la misma.

La red que se quiere simular es un sistema físico. Como tal, esta red de interconexión está formada por nodos que se comunican entre sí por medio de canales. Cada nodo consta de un encaminador con canales bidireccionales.

El encaminador de mensajes consta de los siguientes elementos básicos: Un número de canales bidireccionales, un canal de inyección, un canal de recepción de mensajes, *"buffers"* con capacidad de almacenamiento de un *"flit"* en los cuatro puertos de entrada para recibir los paquetes que son enviados al nodo por los nodos vecinos y colas de *"buffers"* en los puertos de salida para almacenar los paquetes que son encaminados hacia otro nodo. La Figura 3-20 muestra un ejemplo para el caso de tener 4 canales por nodo (no se muestran los canales de inyección y consumo).

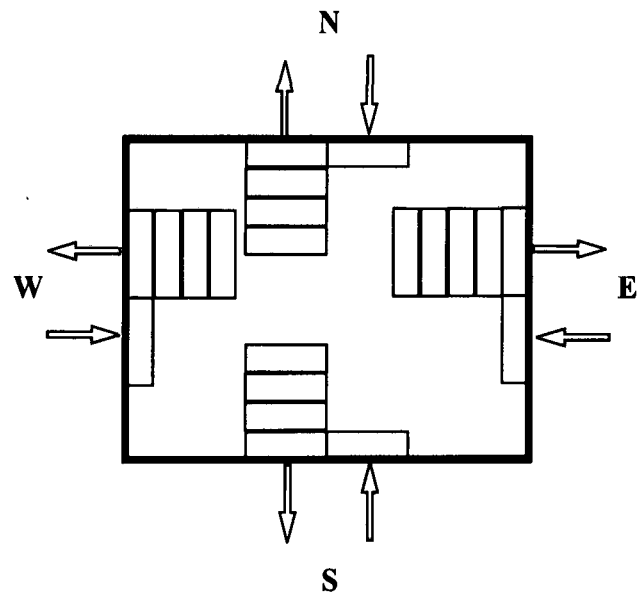


Figura 3-20 Encaminador de mensajes

Se requiere que la red se comporte según las técnicas de control de flujo *"Store-and-forward"*, *"Wormhole"* y *"Cut-Through"*, con diversos algoritmos de encaminamiento estático, adaptativo, y concretamente, los propuestos por DRB.

Las características del comportamiento son seleccionables por el usuario variando los siguientes parámetros de entrada que conforman la carga de trabajo del simulador:

- Patrón de comunicaciones
- Longitud de los mensajes (n *"flits"*)
- Carga de comunicaciones.
- Topología (n-cubos k-arios: toro, hipercubo, *"midimew"*)
- Tamaño de la red
- Control de flujo (*"wormhole"*, *"store-and-forward"*, *"cut-through"*)

- Encaminamiento (estático, adaptativo completo de camino mínimo, DRB)
- Número de Ciclos de Simulación (Tiempo de simulación)

Entre las medidas que se van a obtener, tenemos los siguientes indicadores estadísticos de prestaciones:

- Distancia media recorrida
- Latencia promedio, máxima y desviación de la latencia por canal
- Retardo máximo y medio y desviación del retardo de la red
- "*Through-put*" por canal y "*through-put*" de la red
- Tamaño de las colas de los enlaces
- Ocupación media de los enlaces
- Número de mensajes generados y consumidos

Respecto a la técnica utilizada para implementar el simulador, es necesario señalar que después del estudio de las características del sistema a simular y los resultados a obtener, se ha decidido hacer una simulación secuencial por tiempo dada la alta cantidad de eventos por ciclo que se espera simular y se estima que se obtendrán mejores resultados con un simulador de este tipo. Otros autores confirman esta posibilidad en [104] y [6].

A continuación se describe el diseño del simulador, detallando tanto las estructuras de datos utilizadas como el algoritmo de simulación.

3.5.3.1 Estructuras de datos del simulador

Las estructuras de datos de un simulador soportan la representación del sistema real, describiendo y relacionando las propiedades del mismo. Sobre estas estructuras trabaja el programa mientras transcurre el tiempo simulado para proporcionar los resultados al finalizar la simulación.

Las estructuras de datos representan los encaminadores de la red y sus colas de mensajes. Existirá una estructura por cada nodo que, a la vez, contendrá una estructura por cada enlace de entrada o de salida. Estas estructuras son capaces de almacenar los paquetes en tránsito. La estructura de los paquetes consta de una serie de campos donde

se almacena la información necesaria para gestionar la simulación y hacer las medidas de rendimiento posteriores.

Dentro del paquete distinguimos dos clases de información importantes: el registro de encaminamiento y los datos de control. El registro de encaminamiento almacena el destino o destinos a que va dirigido el paquete. Los datos de control son información relativa al comportamiento del paquete a través de la red, la información registrada consiste en parámetros como la distancia recorrida, el tiempo que ha permanecido en cada nodo, el tiempo que ha estado en colas de espera, el tiempo que tardó en inyectarse en el nodo origen, etc.

Las estructuras que se van a utilizar en el simulador se pueden observar en la Figura 3-21, para una red de grado 4.

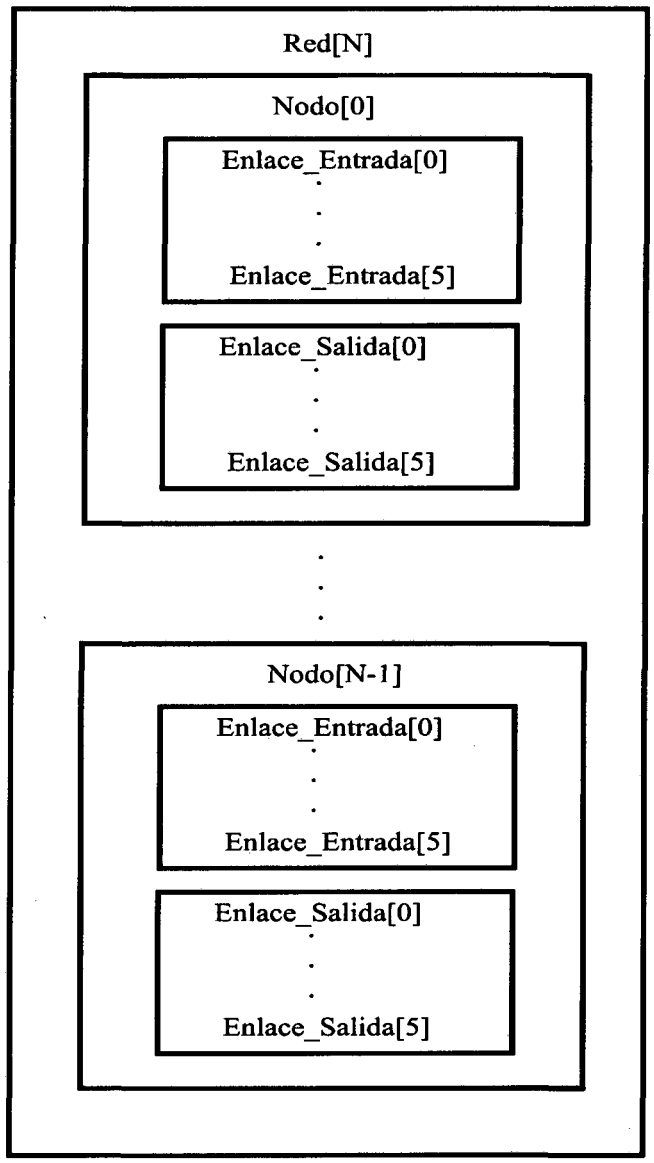


Figura 3-21. Estructuras de datos del simulador

3.5.3.2 Esquema de funcionamiento

Al ser un simulador secuencial conducido por tiempo, el simulador hace una revisión secuencial de todos los nodos de la red, procesando los mensajes que encuentra, encaminándolos a su nodo destino, haciendo que avancen hacia allí o consumiéndolos si lo han alcanzado. En cada ciclo de tiempo dado, el simulador realiza la siguiente secuencia de funciones que tienen en cuenta las diferentes técnicas de control de flujo (*"wormhole", "store-and-forward"* y *"cut-through"*) y los patrones de comunicaciones:

- ✓ **Inyección de mensajes**: Esta función es la que produce el tráfico de mensajes, emulando a los procesos de cálculo que envían mensajes a la red como si se estuviera ejecutando una aplicación paralela en el multicomputador. Se inyectan en la red *n* paquetes de información por unidad de tiempo considerando el patrón especificado de destinos de mensajes.

- ✓ **Encaminamiento de los paquetes**: Éste encaminamiento depende de los parámetros de entrada (como por ejemplo, la topología que se va a utilizar). Esta función determina el enlace de salida que debe seguir un paquete según su destino. El *"flit"* cabecera es el único que lleva la información del encaminamiento.

- ✓ **Transmisión de los *"flits"***: o sea, hacer avanzar hacia el destino cada uno de los paquetes. Se realiza sobre los *"flits"* que le siguen al *"flit"* cabecera, avanzando en forma de *"pipeline"* una vez que se ha abierto el camino, en el caso de *"Wormhole" y "cut-through"*, o avanzando el paquete completo en el caso de *"Store-and-Forward"*. Se trata de determinar, a partir del enlace de salida de nodo en el que se encuentre el *"flit"*, el enlace de entrada del siguiente nodo según determina la topología.

- ✓ **Entrega de mensajes**: Esta función permite que los mensajes sean consumidos cuando llegan a su destino. En ella se toman las estadísticas del simulador.

- ✓ **Monitorización**: se hace avanzar el simulador en cada transmisión y al finalizar se realizan las medidas de monitorización del rendimiento para comprobar los resultados de las ejecuciones.

Este funcionamiento se muestra en la Figura 3-22, donde se detallan el algoritmo principal y el de la función más importante que es la de simular.

```
Algoritmo Simulador ()
{
Introducir parámetros (topología, tipo de encaminamiento,
                       tipo de control de flujo, número de procesadores)
 Para todos los nodos de la red {
 inyectar ( ),
 simular ( ),
 entregar ( ),
 monitorizar ( )
} /*simulador*/

simular ( )
{
 para todos los enlaces de entrada{
        si es cabecera y no ha sido procesado{
                        encaminamiento ( );
                        asignación ( );
        }
 }
 para todos los enlaces de salida{
        si es cabecera {
                si es único y  no ha sido procesado:  transmitir ( );
                si existen varios y ninguno transmite{
                                escoger uno de ellos ( );
                                transmitirlo ( );
                                bloquear los demás ( );
        }
                }
        } /*simular*/
```

Figura 3-22 ***Esquema de Funcionamiento del Simulador***

Por otra parte, existen una serie de consideraciones adicionales en el diseño que también se deben tener en cuenta, tales como la gestión de los bloqueos y la llegada de múltiples paquetes a un mismo nodo. A continuación, se comentan cada una de ellas.

3.5.3.2.1 Tratamiento del "*deadlock*"

Respecto al primer aspecto, se debe asegurar que los mensajes lleguen a su destino, evitando que queden bloqueados entre ellos en situación de "*deadlock*". Se produce un "*deadlock*" en la red cuando se forma un ciclo cerrado de peticiones en los que cada mensaje quiere avanzar a la posición ocupada por el siguiente mensaje en el ciclo. En el simulador se ha implementado una técnica de detección y recuperación de situaciones de "*deadlock*", consistente en la detección de ciclos cerrados de paquetes en la red.

Para la detección de ciclos en la red, periódicamente se va examinando cada uno de los nodos por si se ha formado algún ciclo de comunicaciones. En el caso que así sea se selecciona un mensaje del ciclo, se extrae de la red inyectándolo en uno de los encaminadores para romper el ciclo y se reinyecta con posterioridad.

3.5.3.2.2 Llegada de paquetes simultáneamente

El segundo aspecto a considerar es la llegada de múltiples paquetes a un mismo nodo. Mientras un paquete atraviesa la red, puede ser retrasado teniendo que esperar detrás de otros paquetes en las colas de entrada. Cuando llegan varios paquetes que quieren utilizar el mismo enlace de salida, se debe asegurar que no siempre se concede el enlace al mismo paquete porque sesgaría el resultado de la simulación y podría llegar a provocar "*starvation*". En el caso real de un encaminador asíncrono, la llegada de paquetes a los nodos es asimismo asíncrona y no se puede considerar que dos o más paquetes lleguen simultáneamente y no siempre se decide a favor del mismo paquete. Por lo tanto, en el caso que varios paquetes lleguen al mismo tiempo a un puerto de entrada para ser atendidos, al hacer el análisis se escogerá aleatoriamente uno de ellos, para no predeterminar cuál de los paquetes se escoge y dar más prioridad a uno sobre los otros. La gestión de estos "*buffers*" comprende el almacenamiento de mensajes que encuentran sus canales de salida ocupados y el envío de éstos una vez que el canal de salida se libere.

Para finalizar, en este punto hemos visto la fase de análisis y diseño del simulador funcional utilizado para la evaluación de las prestaciones dinámicas de las propuestas de este trabajo de tesis. En el apéndice B se incluye un documento descriptivo del simulador funcional donde se especifican sus entradas, sus salidas, su "*interface*" de "*front-end*" con el usuario para controlar interactivamente las simulaciones y aspectos internos de implementación. También se describen algunas utilidades anexas para la creación de ficheros de entrada al simulador y de visualización de los resultados.

3.6 Validación de los modelos analítico y funcional de redes de interconexión.

Estos dos modelos, el simulador analítico y el simulador funcional, se han validado con una serie de ejemplos representativos comparando los resultados de uno contra otro. El modelo analítico junto con la evaluación aquí incluida han sido presentados y publicados en [54]. A continuación, se muestra la experimentación realizada para validar los modelos.

El modelo analítico presentado ha sido implementado mediante un programa de ordenador y evaluado mediante una serie de tests experimentales. Para verificar los resultados del modelo analítico, éstos se han comparado con los resultados del simulador funcional de redes de interconexión dirigido por tiempo que también ha sido desarrollado para evaluar las propuestas principales de este trabajo de tesis. De estos simuladores analítico y funcional se presentan mas detalles en los apéndices al final de la memoria. En este caso, nos centramos en los resultados de latencia y también en las prestaciones temporales.

Los experimentos consisten en el envío de paquetes a través de los enlaces de la red para varios patrones de tráfico. Las simulaciones se realizaron para varias topologías de diversos tamaños. Las topologías aquí presentadas son n-cubos k-arios. Para los experimentos se mantiene fija la longitud de los mensajes en 10 *"flits"*, y se varia la carga de tráfico a lo largo de un rango mediante incrementos constantes desde baja carga hasta la saturación.

Los resultados del simulador funcional se ejecutaron varias veces con diferentes semillas del generador de números aleatorios para simular la llegada de paquetes y se observó que eran consistentes. EL simulador ejecuta 200.000 unidades de tiempo y las 20.000 primeras iteraciones no se incluyen en los resultados para despreciar los efectos transitorios en la simulación.

Presentamos resultados del retardo medio en la red que demuestran que el modelo analítico se asemeja al simulador funcional para varios patrones. Las gráficas siguientes muestran un par de curvas para cada patrón, una para los resultados del modelo analítico y la otra para el simulador funcional. Además, para cada par de curvas hemos calculado un factor de correlación. Este factor de correlación se calcula correlando los resultados para ambos experimentos y da un valor entre 0 y 1 que indica cuánto se parecen las curvas, cuanto más cercano a 1, mayor similitud. A continuación se muestran los resultados para dos grupos de patrones de tráfico: específicos y regulares, y también se muestra un análisis de complejidad y de tiempos de ejecución.

3.6.1 Resultados para patrones de comunicación específicos

Para testar todas las posibilidades del modelo analítico, es necesario diseñar algunos patrones de comunicación específicos. Aquí hemos elegido tres patrones a los que hemos llamado "*hot-spot*", anillo y "*complex*", respectivamente.

Bajo los patrones "*hot-spot*" y "*complex*", algunos destinos se fijan para incrementar el tráfico en un área particular de la red y causar zonas saturadas. Estos dos patrones se muestran en la Figura 3-23 (a) y (b), respectivamente. En el patrón anillo, se tiene un anillo formado por n nodos donde cada nodo *i* envía mensajes al nodo *(i+2) mod n*, con $i=\{0,1,2,\ldots,n-1\}$. Este patrón se muestra en la Figura 3-23 (c). Estos tres patrones representan la nueva funcionalidad del modelo porque debido a su complejidad no pueden ser representados con los modelos de Min-Mutka.

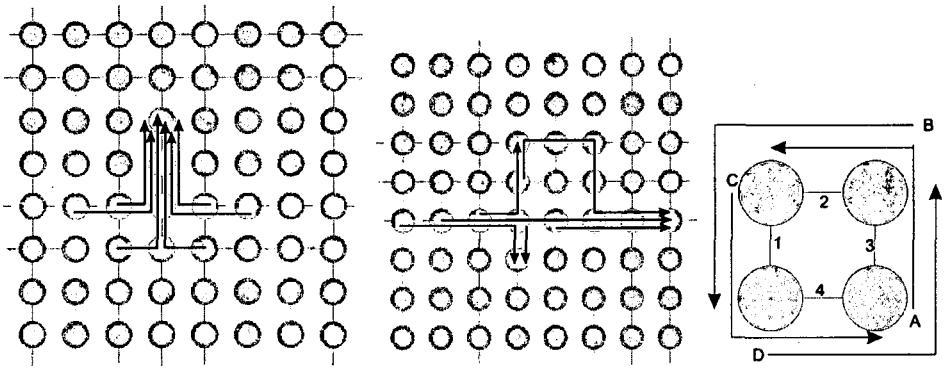


Figura 3-23 Patrones específicos: *Hot-spot*, Complex, Ring

En la Figura 3-24 se pueden observar los resultados para cada uno de los experimentos. Estas gráficas muestran la latencia media de los paquetes para los tres patrones respecto a la carga aceptada de tráfico. Se presentan dos curvas para cada experimento: Una que corresponde al modelo analítico, denominada con el prefijo 'A', y la otra para el simulador funcional, denominada con el prefijo 'F'. En la Tabla 3-6 se muestra el factor de correlación antes explicado para cada uno de los experimentos.

Los tres pares de curvas presentan la misma tendencia y se observa que son muy similares en ambos casos, el analítico y el funcional. Las diferencias entre cada par de curvas se acentúa cuando la saturación crece. En el caso del experimento Ring, no existe una saturación muy acusada, y ambas curvas permanecen juntas para todo el rango de carga aceptada. El factor de correlación es muy alto llegando al 0,99. Para el experimento Complex, podemos observar que para tasas de carga bajas (carga < 0,5), las dos curvas son prácticamente iguales.

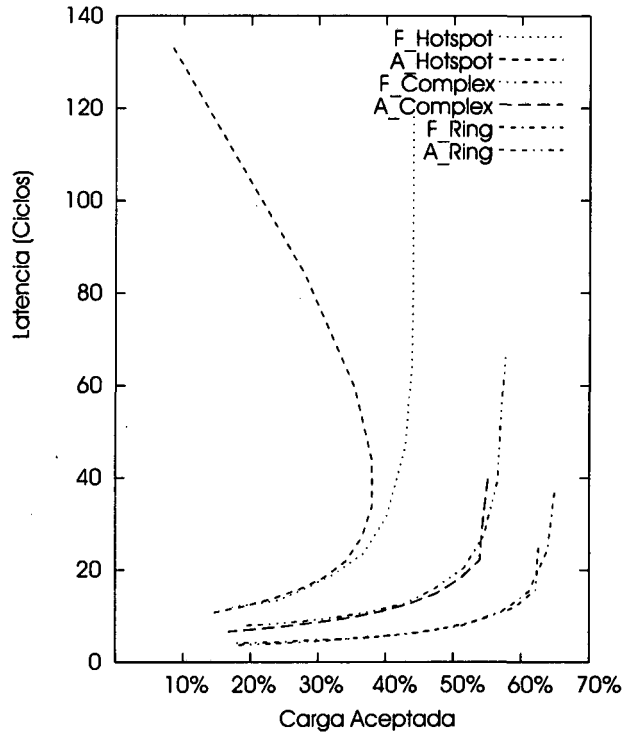


Figura 3-24 Resultados comparativos para los patrones específicos

Se puede ver asimismo que como el simulador funcional tiende a mostrar la saturación cuando se entra en zonas de carga alta (carga > 0,7), aunque ambas gráficas muestran un factor de correlación muy bueno del 0,99. El experimento de "*hot-spot*" fue diseñado para presentar una gran área saturada, y los resultados muestran mayor diferencia debido a la saturación cuando la carga está por encima del 0,5. En este caso el factor de correlación es 0,96, un poco menor que en los casos anteriores. Esta diferencia se puede explicar porque el simulador funcional tiene en cuenta en la latencia el tiempo de espera por inyección del mensaje en la red, es decir, el tiempo desde que el mensaje se inyecta hasta que accede al primer enlace de la red, y este es un factor que el modelo analítico no tiene en cuenta.

Experimento	Factor de Correlación
Hot-Spot	0.96
Ring	0.99
Complex	0.99

Tabla 3-6 Factor de correlación para los patrones específicos

3.6.2 Resultados para patrones de comunicación regulares

Para testar un amplio rango de casos, hemos elegido algunos de los patrones de comunicación comúnmente usado para evaluar redes de interconexión [41]. "*Bit-*

reversal", "*Butterfly*", "*Perfect Shuffle*", "*Matix Transpose*" y "*Complement*". Estos patrones representan las permutaciones que generalmente aparecen en muchos algoritmos paralelos de aplicaciones numéricas y por esta razón son un buen ejemplo de carga en la red de interconexión. Además, son unos patrones que implementan una permutación entre fuentes y destinos donde todos los nodos envían a algún otro nodo mensajes con lo que demandan un alto grado de comunicaciones de la red de interconexión. En general, estos patrones tampoco pueden ser representados por el modelo básico de Min-Mutka.

Bajo el patrón de tráfico "*Bit-reversal*", el nodo con coordenadas binarias a_{n-1}, a_{n-2}, ..., a_1, a_0 se comunica con el nodo a_0, $a_{1,...}$, a_{n-2}, a_{n-1}. El patrón de tráfico "*Butterfly*" se forma intercambiando los bits mas y menos significativos del número del nodo: el nodo con coordenadas binarias a_{n-1}, a_{n-2}, ..., a_1, a_0 se comunica con el nodo a_0, a_{n-2}, ..., a_1, a_{n-1}. En el patrón "*Matrix Transpose*", el nodo con coordenadas binarias a_{n-1}, a_{n-2}, ..., a_1, a_0 se comunica con el nodo $a_{n/2-1}$, ..., a_0, a_{n-1}, ..., $a_{n/2}$,. El patrón "*Perfect Shuffle*" rota un bit a la izquierda, es decir, el nodo con coordenadas binarias a_{n-1}, a_{n-2}, ..., a_1, a_0 se comunica con el nodo a_{n-2}, a_{n-3}, ..., a_0, a_{n-1}. En el patrón "*Complement*" cada nodo envía al nodo resultante de complementar cada uno de los bits del número de nodo.

A continuación, presentamos y comparamos los resultados de la latencia media para cada uno de los cinco patrones simulados para dos topologías tipo n-cubos k-arios: el toro 2D 4x4 y el Hipercubo 4D. La Figura 3-25 muestra los resultados de latencia para los patrones mencionados sobre el toro y la Figura 3-26 para el hipercubo. Las tablas Tabla 3-7 y Tabla 3-8 muestran los factores de correlación para cada uno de los experimentos. Los resultados son muy similares para todos los patrones y topologías. Se puede observar que, a baja carga (carga menor de 0,5), ambas gráficas, la del modelo analítico y del simulador funcional, son casi iguales. A cargas altas (carga mayor de 0,7), las gráficas se separan ligeramente debido al efecto de saturación del simulador funcional antes mencionado. Los factores de correlación muestran buenos resultados por encima de 0,96 para todos los pares de curvas.

Factor de Correlación para el Toro				
Butt.	**Comp.**	**Trans.**	**Bit R.**	**Per.S.**
0.96	*0.98*	*0.97*	*0.97*	*0.97*

Tabla 3-7 Factor de correlación para los patrones regulares en el toro

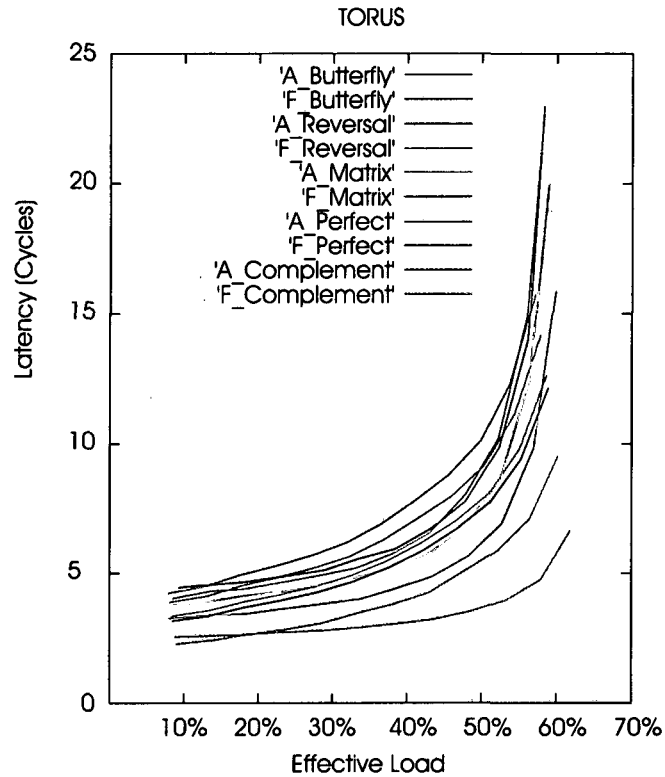


Figura 3-25 Resultados comparativos para los patrones regulares en el toro

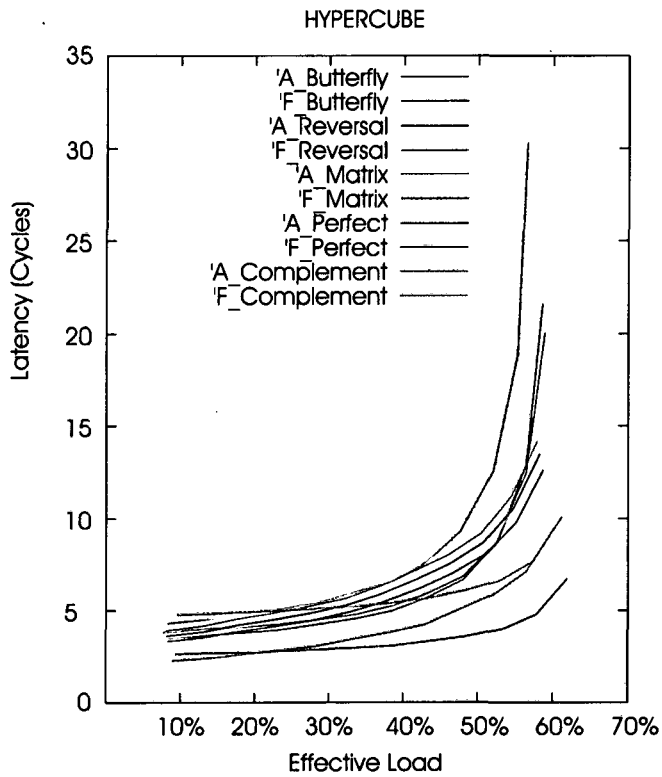


Figura 3-26 Resultados comparativos para los patrones regulares en el hipercubo

Factor de Correlación para el Hipercubo				
Butt.	**Comp.**	**Transp.**	**Bit R.**	**Perf. S.**
0.98	*0.98*	*0.97*	*0.97*	*0.96*

Tabla 3-8 Factor de correlación para los patrones regulares en el hipercubo

3.6.3 Análisis de complejidad y tiempos de ejecución

Después de haber visto los resultados ofrecidos por el modelo analítico frente al funcional, vamos ahora a comentar y comparar algunas de las ventajas del modelo analítico frente al simulador funcional. Cómo se comentó al principio de este capítulo, un modelo matemático analítico permite conocer las razones del porque del funcionamiento de un sistema al variar algunos de sus parámetros. Esta es una característica muy importante para comprender la naturaleza intrínseca del sistema a modelar.

Por otro lado, y dadas las características del modelo desarrollado en el que se plantea un conjunto de ecuaciones a resolver, los tiempos de ejecución, es decir, de resolución del sistema de ecuaciones y obtención de resultados, son mucho menores que los del simulador funcional. Para ver esto, debemos hacer un análisis de la complejidad de ambos algoritmos y su rendimiento. Para ello debemos analizar el número de operaciones que se realizan y el número de veces que se repiten esas operaciones. En ambos casos, las operaciones realizadas son las involucradas para determinar la latencia producida en cada enlace de cada nodo activo de la red de interconexión. Este procedimiento se repite hasta que se llega a un resultado estable.

La Ecuación 3-11 muestra el tiempo de cómputo, *Tc*, para ambas alternativas. La Tabla 3-9 muestra los parámetros de la fórmula y los valores típicos para cada uno de los dos casos en estudio, el modelo y el simulador. Como puede verse el modelo analítico converge en muchas menos iteraciones que el simulador funcional con lo que, aunque sus cálculos sean más complejos, porque implican operaciones costosas en punto flotante, es del orden de entre 100 y 1000 veces más rápido que el simulador funcional.

$$Tc = i * N * L * k$$

Ecuación 3-11 Tiempo de cómputo

	Analítico	Funcional
i= Factor de Convergencia (# de iteraciones)	[15:235]	200,000
N= Numero de nodos de la red	n	n
L= numero de enlaces por nodo	m	m
k= tiempo para procesar un enlace	10	1
Relación Funcional/Analítico	**[85:1314]**	

Tabla 3-9 Parámetros del Tiempo de Cómputo

La Tabla 3-10 muestra los tiempos de ejecución para cada uno de los experimentos donde se observan las diferencias existentes. Nótese que el modelo analítico presenta resultados en pocos segundos, mientras que el simulador funcional necesita varios minutos de ejecución.

Tc: Tiempos de Ejecución (minutos:segundos)				
Experimento		**Analítico**	**Funcional**	**Relación**
Específicos	*Hot-spot*	0:03	15:38	312
	Ring	0:03	15:38	312
	Complex	0:01	15:25	925
Toro	Butterfly	0:04	11:53	178
	Complement	0:01	20:32	1232
	Transpose	0:04	16:50	252
	Reversal	0:06	8:33	**85**
	P.Shuffle	0:04	18:47	281
Hipercubo	Butterfly	0:01	12:22	742
	Complement	0:01	21:54	**1314**
	Transpose	0:06	16:55	169
	Reversal	0:04	16:47	251
	P.Shuffle	0:06	18:36	186

Tabla 3-10 Tiempos de ejecución del modelo analítico y el simulador funcional

Finalmente, debemos comentar algunas otras ventajas del modelo analítico frente al simulador funcional. La primera es que además de menor tiempo de ejecución, necesita menos memoria para almacenar los datos, pues sólo necesita los valores que está calculando mientras que el simulador funcional necesita representar todos los encaminadores y mensajes que están circulando por la red. Otra de las ventajas, intrínseca a su funcionamiento, es que nunca puede producirse la aparición de "*deadlock*" de comunicaciones, ya que no simula el avance de los mensajes por la red de interconexión. Esta característica puede verse en el ejemplo Ring mostrado, que es un caso típico de "*deadlock*".

3.7 Conclusiones

En este capítulo se han presentado las herramientas desarrolladas para el estudio de las redes de interconexión. Estas herramientas son dos modelos de las redes de interconexión. Un modelo es matemático analítico y el otro funcional. Ambos modelos representan el comportamiento de la red de interconexión cuando existe un conjunto de mensajes viajando sobre la red y aceptan los mismos parámetros de entrada. Estos son una descripción de la red de interconexión (topología, control de flujo, algoritmo de encaminamiento, parámetros temporales del encaminador), el programa de aplicación paralelo (conjunto de tareas, canales de comunicación entre las tareas, volúmenes de cómputo y de comunicación) y la asignación de tareas a nodos de la red. Los resultados de salida obtenidos son la latencia de cada uno de los canales de la red. De hecho, ofrecen una visión del estado de la red. El modelo analítico ofrece el resultado estacionario final de una configuración de canales y el funcional es capaz de ofrecer todo el transitorio a lo largo del avance del tiempo.

El primero de los modelos, el analítico, es un modelo matemático analítico que plantea una serie de ecuaciones estocásticas que representan la actividad en la red de interconexión y cuya resolución ofrece los resultados de latencia. El segundo modelo, el funcional, se implementa mediante un programa de computador que simula el funcionamiento real de la red de interconexión a medida que avanza el tiempo. Estos dos modelos utilizan aproximaciones totalmente diferentes para representar la misma entidad física. Las razones de desarrollar dos modelos se basan en la profundización del estudio de las redes de interconexión y en tener la posibilidad de validar el uno contra el otro, ya que representan lo mismo desde dos visiones totalmente diferente.

En el próximo capítulo, se utilizan estas herramientas para analizar las razones que provocan la problemática en el comportamiento temporal de las redes de interconexión y establecer cuál debería ser el comportamiento ideal de una red de interconexión desde el punto de vista del diseñador o programador de un computador de

altas prestaciones que incluya una red de este tipo, y se establecen, por tanto, los objetivos en cuanto a diseño de redes de interconexión. A continuación, se realiza la introducción del tipo de solución que hemos diseñado con objeto de hacer frente a la problemática de funcionamiento presentada y cumplir los objetivos establecidos. La definición del comportamiento ideal o esperado de una red de interconexión y el mecanismo necesario para conseguirlo son la finalidad principal del presente trabajo de tesis.

Capítulo 4 Balanceo distribuido del encaminamiento

4.1 Introducción

En este capítulo se exponen los criterios de diseño de las redes de interconexión de los computadores de altas prestaciones aportados en este trabajo de tesis y se presentan los mecanismos necesarios para cumplir dichos criterios.

Por lo tanto, este capítulo presenta la técnica desarrollada en este trabajo de tesis para conseguir el balanceo del encaminamiento de los mensajes acorde con los objetivos que se presentan en el capítulo. El método persigue distribuir los caminos que usan los canales en la red de interconexión. Para conseguirlo, usa una técnica basada en la expansión de caminos controlada por la propia carga de la red de interconexión. El método se llama Balanceo Distribuido del Encaminamiento ("*Distributed Routing Balancing*", DRB según sus siglas en inglés)

El principal objetivo del método es balancear uniformemente el tráfico de comunicaciones sobre todos los caminos de la red de interconexión completa. El método se basa en la creación caminos alternativos simultáneos entre cada par fuente y destino con objeto de mantener una baja latencia de los mensajes. DRB define cómo crear los caminos alternativos para expandir los caminos simples originales (Definición de caminos "multicarril") y cuándo y cómo usarlos dependiendo de la carga de tráfico de la red de interconexión (Selección del camino multicarril). Se produce un efecto colectivo, pues esta expansión se produce para todos los pares fuente-destino que también interaccionan entre si.

El resto del capítulo está organizado de la siguiente manera. El punto 2 muestra, a través de una colección de ejemplos seleccionados, la especificidad de las características de comportamiento de las redes de interconexión y la problemática de este comportamiento. El punto 3 analiza las implicaciones que esta problemática tiene en las aplicaciones típicas que se ejecutan sobre los computadores de altas prestaciones, y que no son sólo de tipo temporal, sino que también tiene implicaciones en el diseño y el funcionamiento correcto de dichas aplicaciones.

El punto 4 analiza los criterios que deben acompañar el diseño de redes de interconexión. El punto 5 presenta las ideas y técnicas generales para conseguir los objetivos de diseño. Los antecedentes que existen de nuestro trabajo y sobre los que se basa se exponen en el punto 6. El punto 7 presenta el mecanismo diseñado en este trabajo dentro del marco establecido para cumplir los objetivos de diseño llamado DRB. En este punto 7 analizan con detenimiento los diferentes aspectos de la propuesta presentada basada en la expansión de caminos. Se muestra un ejemplo básico de funcionamiento del método y se estudia la implementación del método mediante el diseño de un encaminador. También se analizan diferentes aspectos del coste de la propuesta y, finalmente, las conclusiones del capítulo se encuentran en el punto 8.

4.2 Comportamiento de las redes de interconexión frente a "hot-spots"

Después de la descripción de los modelos de la red de interconexión expuesta en el capítulo anterior, en esta sección vamos a analizar las características típicas de funcionamiento de las redes de interconexión mediante el modelo analítico. Para ello vamos a presentar una serie de casos prácticos representativos que comprenderán diferentes aspectos de la carga de la red. En los ejemplos se modificará cada vez un solo parámetro de la carga de comunicaciones en el que se está interesado en estudiar, mientras los demás parámetros se mantienen constantes con objeto de que no enmascaren el efecto del parámetro en estudio. Los parámetros que se consideran más

relevantes son: el patrón de comunicaciones, el número de canales, el tráfico de los canales y la longitud de los mensajes.

Por otro lado, lo que estamos interesados en estudiar es el resultado temporal con respecto a la latencia de la red de interconexión y extraer una serie de conclusiones que describan unas características típicas del comportamiento de las redes de interconexión. Analizaremos la respuesta en latencia de cada canal y las zonas de la red con mayor concentración de mensajes, entre otros resultados. A partir de ahí, deduciremos una serie de problemáticas en el funcionamiento y analizar las causas que provocan ese mal funcionamiento.

El conjunto de ejemplos seleccionados para mostrar el funcionamiento de las redes de interconexión va a estar compuesto por cuatro ejemplos. Estos ejemplos están seleccionados de manera que colocan la red en situación límite variando la carga desde valores de carga baja 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, lo cual se observa en que la pendiente de la curva de latencia es muy acusada. Los ejemplos seleccionados son:

- ✓ "*Hot-spot*" por frecuencia de mensajes
- ✓ "*Hot-spot*" por número de canales
- ✓ Variación de la longitud del mensaje
- ✓ Propagación de "*hot-spots*"

A continuación describimos el experimento en detalle y los resultados que obtenemos usando el modelo analítico de la red de interconexión, para cada ejemplo.

4.2.1 Ejemplo 1: "*Hot-spot*" por frecuencia de mensajes

Este es un primer ejemplo sencillo en el que se muestra un sistema formado por dos canales *m1* y *m2* que colisionan en un punto en común como muestra la Figura 4-1. Los canales están caracterizados, como se ha definido en el capítulo anterior, por una tasa de generación de mensajes MI y por una longitud L. En el experimento se evalúa la latencia que sufren los canales para un cierto intervalo de frecuencia de generación de mensajes igual para ambos mientras que la longitud del mensaje se mantiene contante.

La tasa de generación de mensajes por unidad de tiempo varia desde 10 ciclos por mensaje hasta 120 ciclos variando de 10 en 10 unidades y se expresa como [10,120].

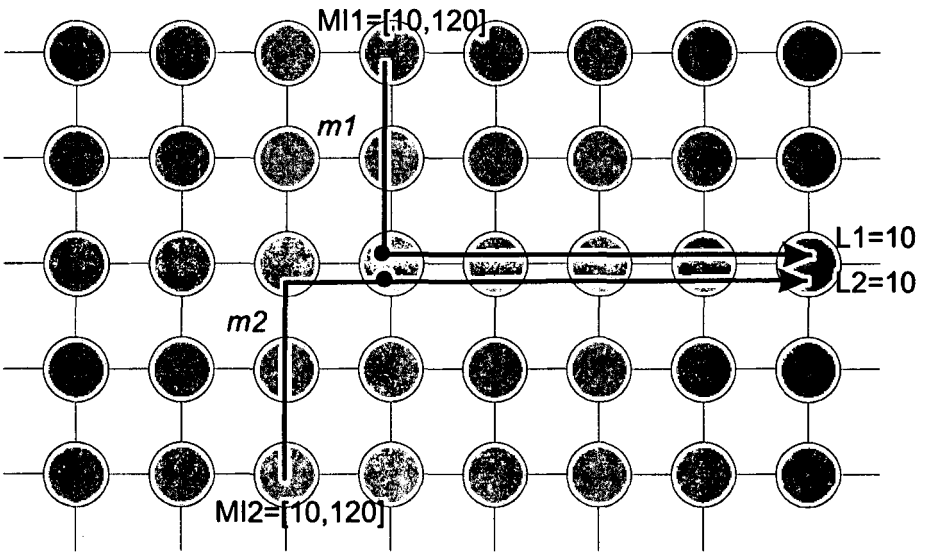


Figura 4-1 Experimento con dos canales

El objetivo de este experimento es mostrar la influencia de la frecuencia de generación de mensajes. Para ello, a estos dos canales se les va cambiando el intervalo de generación de mensajes desde baja carga hasta alta carga. La carga la definimos como el número de mensajes por unidad de tiempo, es decir, cada cuantos ciclos se genera un mensaje. El inverso de este valor da el número de mensajes por unidad de tiempo y, conocido el tamaño del mensaje en "*flits*", se conoce el número de "*flits*"/ciclo inyectados en la red. Se juega con todas las posibilidades de manera que se va variando la carga de uno de los canales y, para cada valor de carga de este canal, se varía la carga del otro canal para todos los valores desde baja carga hasta carga de saturación. Se entiende carga de saturación cuando el intervalo de generación llega a ser una inyección de mensajes continua uno tras otro sin interrupción. En este momento se produce la aparición de grandes latencias sufridas por los mensajes, lo cual se llama "*hot-spot*". En nuestro ejemplo, esta tasa de inyección es de un mensaje cada 10 unidades de tiempo, ya que la longitud del mensaje es 10 "*flits*", y suponemos que la transmisión de un "*flit*" tarda una unidad de tiempo.

Se muestran dos gráficas de resultados para este ejemplo. La Figura 4-2 muestra la latencia del mensaje *m1* respecto la variación de la carga de él mismo mientras la carga del mensaje *m2* se mantiene constante para cada uno de los valores 10, 50, 100, 250 y 500 ciclos por mensaje. En esta gráfica se ve que cuando mayor es la carga de *m2*, mayor es la latencia. Cuando la carga de *m2* es muy baja, 500 ciclos por mensaje, la influencia es muy pequeña y constante. Se observa, para cada una de las curvas, que la

carga de *m1* influye poco resultando una latencia prácticamente constante hasta que se llega a una carga de entre los 20 y 10 ciclos por mensaje, en que la latencia inicia una curva de crecimiento exponencial muy pronunciada. En esta situación tenemos que se ha producido el "*hot-spot*" y la red no es capaz de asimilar más carga.

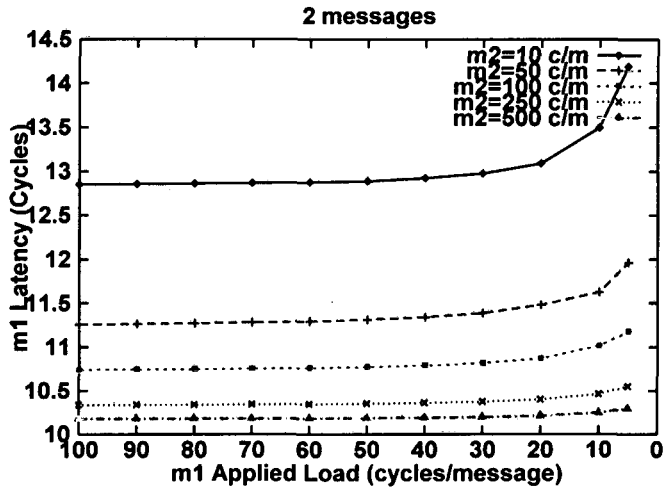


Figura 4-2 Ejemplo 1: Latencia de m1 respecto a la carga de m1

La Figura 4-3 muestra la influencia de la carga del mensaje *m2* sobre la latencia del mensaje *m1*. Se presenta la latencia de *m1* respecto de la carga de *m2* para varios valores constantes de la carga de *m1*: 10, 50, 100, 250 y 500 ciclos por mensaje. Se puede observar que la latencia del mensaje *m1* crece exponencialmente cuando la carga de *m2* aumenta linealmente, por lo que la red no muestra un comportamiento linealmente proporcional. Además, como todas las curvas son muy parecidas, se observa que la influencia de la carga del mensaje *m1* es poco relevante, siendo la causa de la latencia de *m1* la presencia de otro mensaje *m2* en el camino que comparten.

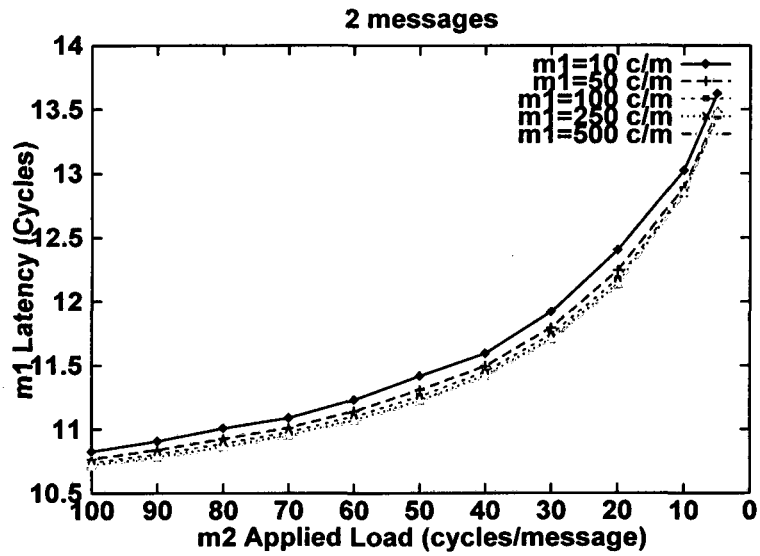


Figura 4-3 Ejemplo 1: Latencia de m1 respecto a la carga de m2

4.2.2 Ejemplo 2: "*Hot-Spot*" por número de canales

En este ejemplo, se tienen una serie de canales que parten de puntos diferentes pero que tienen un mismo destino de manera que inciden sobre una parte del camino común. Lo que hacemos es variar el número de canales desde 2 hasta 5 para ver la influencia de este parámetro en la latencia. Para que otros parámetros no nos influencien en este caso, se fijan a unos valores determinados iguales para todos los mensajes. Estos parámetros son el intervalo entre mensajes y el tiempo de transmisión, de manera que el intervalo entre mensajes es el mismo para todos los canales en cada punto de evaluación. El experimento se evalúa para un rango de carga desde baja carga hasta la saturación, como en el ejemplo anterior. La Figura 4-4 muestra el patrón de "*hot-spot*" elegido.

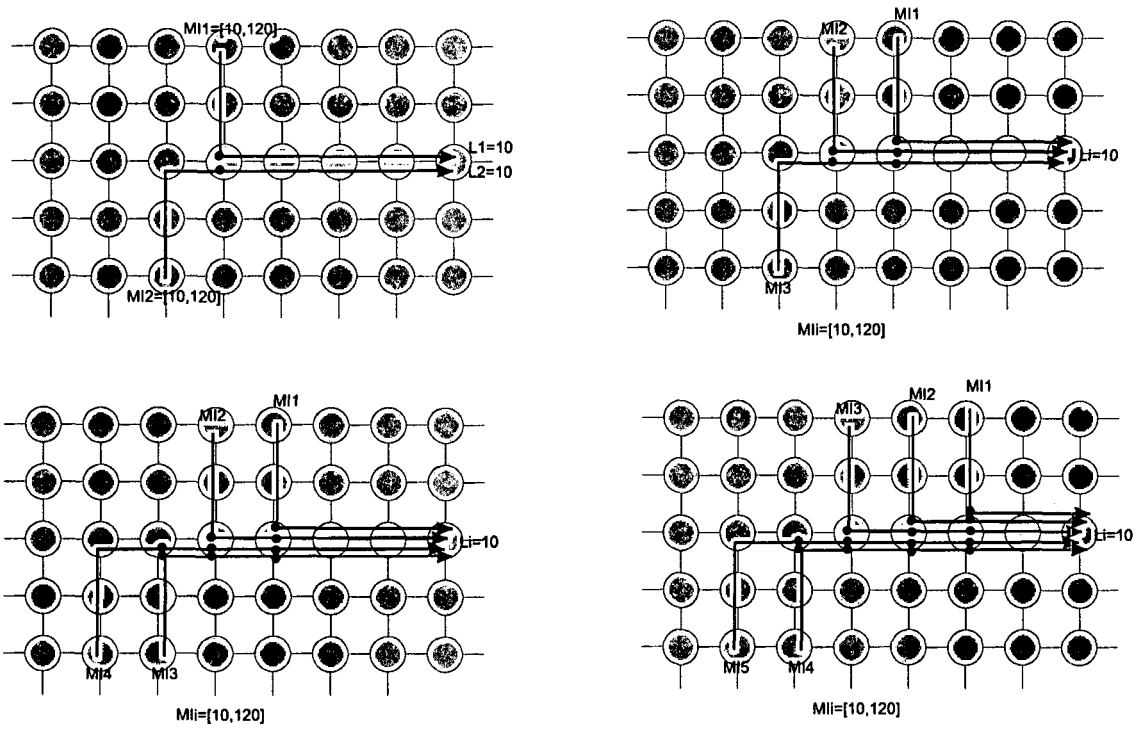


Figura 4-4 Experimento cambiando el número de canales

Se observa, en la Figura 4-5, que la latencia media de la red crece al aumentar el número de mensajes, como era de esperar, pero este crecimiento no es lineal. A baja carga (intervalo entre mensajes mayor que 60 ciclos), el crecimiento es lineal y la latencia resultante es baja. Pero a partir de este punto y sobre todo al llegar a cargas altas (10 y 20 ciclos por mensaje), el crecimiento de la latencia se dispara de manera desproporcionada, mostrando que la red de interconexión ha llegado al límite de saturación.

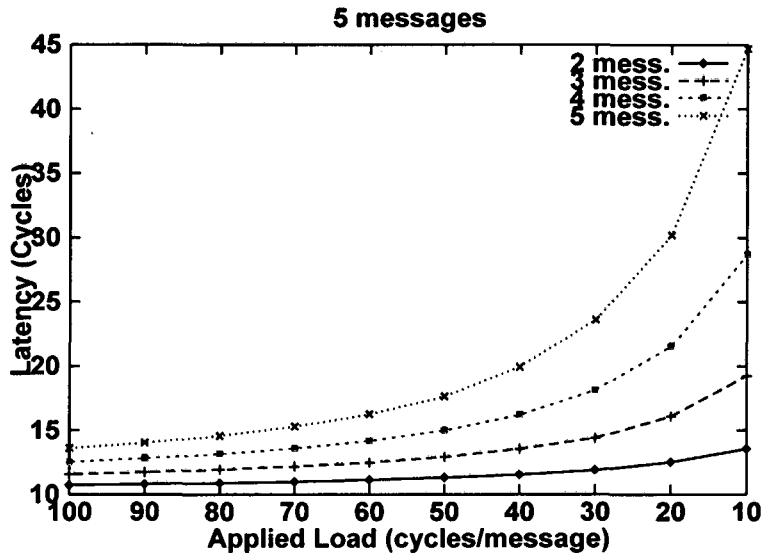


Figura 4-5 Ejemplo 2: Latencia media de la red cambiando el número de canales

La Figura 4-6 muestra la latencia para cada uno de los canales en el caso de que existan cinco canales sobre el camino común. Se observa que no todos los canales sufren la misma latencia, sino que a medida que se incrementa el número de puntos de colisión por canal, mayor es la latencia. Este efecto es debido a que el canal que tiene que atravesar más puntos de colisión tiene menos oportunidades de conseguir el canal libre, y eso hace que se incremente su latencia. Este es un resultado muy importante sobre el comportamiento de las redes de interconexión.

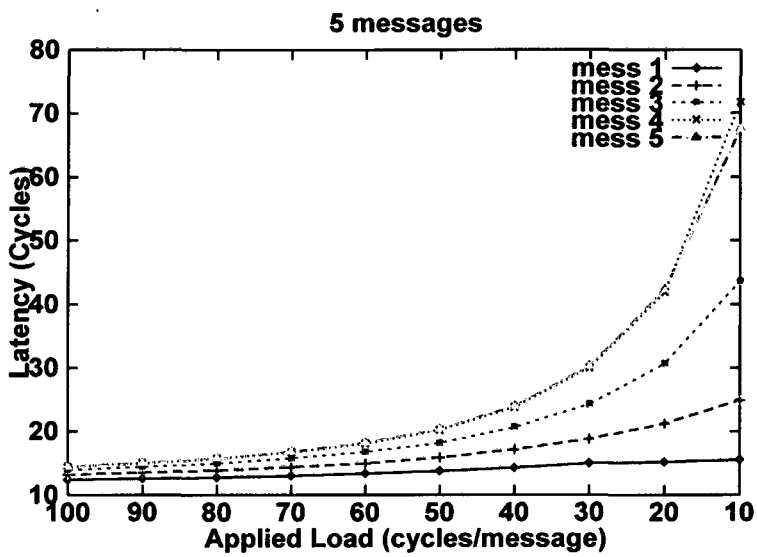


Figura 4-6 Ejemplo 2: Latencia para cada uno de los canales

4.2.3 Ejemplo 3: Variación de la longitud del mensaje

En este experimento se quiere ver la influencia de la longitud de los mensajes en la carga de comunicaciones sobre la red de interconexión. Para ello, se tiene un sistema de 5 canales similar al sistema del ejemplo anterior pero ahora se mantienen fijos los intervalos entre mensajes a un valor fijo y se va variando la longitud de los mensajes desde 10 a 1000 "*flits*", tal y como se muestra en la Figura 4-7. El experimento se evalúa para diferentes valores del intervalo entre mensajes desde 10 hasta 500 ciclos por mensaje.

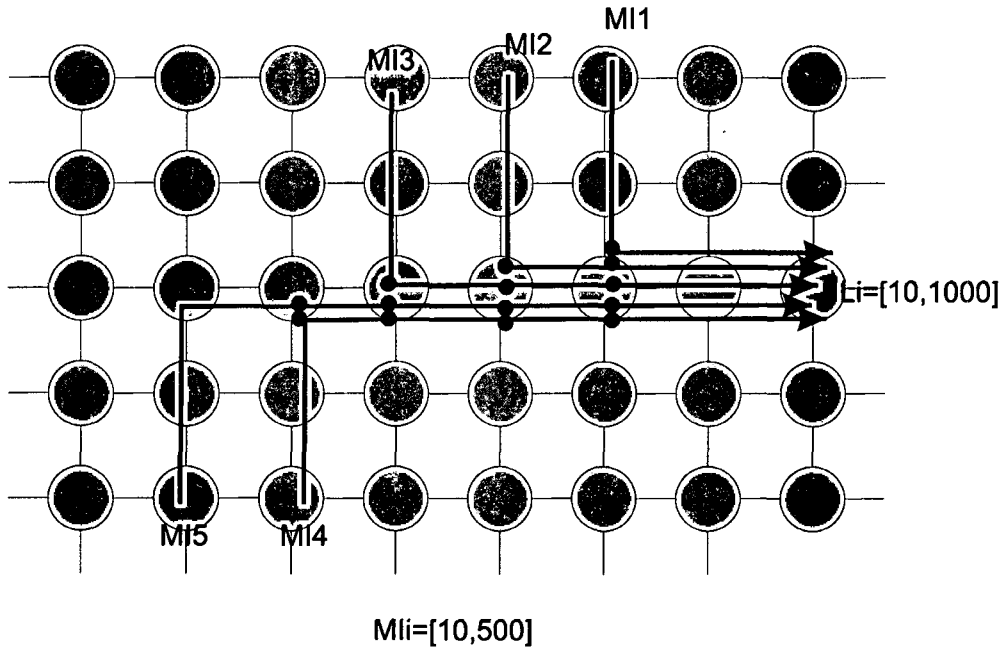


Figura 4-7 Experimento variando la longitud del mensaje

En la Figura 4-8 se observa el resultado del experimento para varias longitudes de mensajes entre 10 y 1000 *"flits"* cuando se evalúa para diferentes cargas de tráfico de los mensajes entre 10 y 500. Como puede verse, al aumentar el número de *"flits"*, la latencia crece casi linealmente, lo que muestra un crecimiento proporcional de la latencia con el incremento de carga que significa aumentar la longitud del mensaje. Sólo a cargas muy bajas, de 250 y 500 ciclos por mensaje, y mensajes pequeños, menor de 250 *"flits"*, el crecimiento de la latencia es menor que en otros casos.

4.2.4 Ejemplo 4: Propagación de "*hot-spots*"

Finalmente, este ejemplo quiere mostrar un efecto pernicioso que se dan en las redes de interconexión. Este efecto es la propagación de situaciones problemáticas a zonas cada vez mayores a consecuencia del efecto colectivo de influencia de unos canales en otros. Para ello vemos un ejemplo con un sistema de canales (*m2* a *m5*)en los

que no hay saturación (Figura 4-9 a) A continuación, se genera un "*hot-spot*" o zona saturada en una zona localizada como consecuencia de la aparición de un nuevo canal (*m1*) cuya tasa de generación de mensajes es muy alta (10 ciclos/mensaje) (Figura 4-9 b). El efecto observado es que la red entera se colapsa por culpa de un único canal de alta carga.

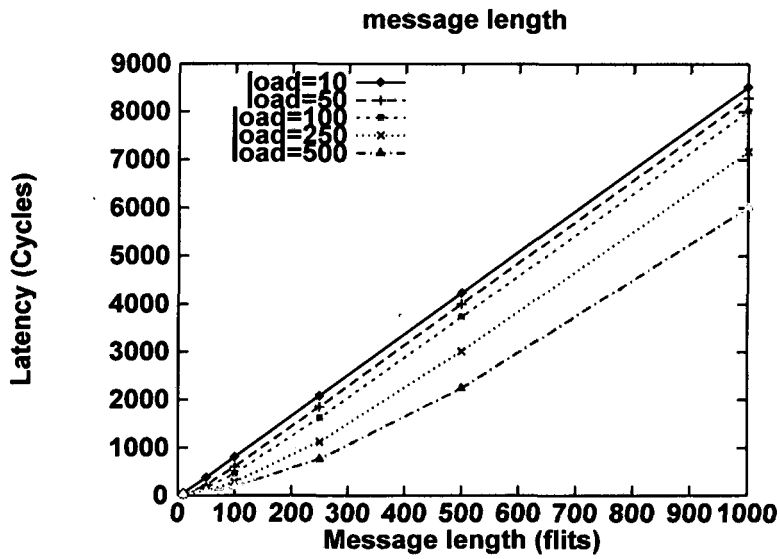


Figura 4-8 Ejemplo 3: Latencia variando la longitud del mensaje

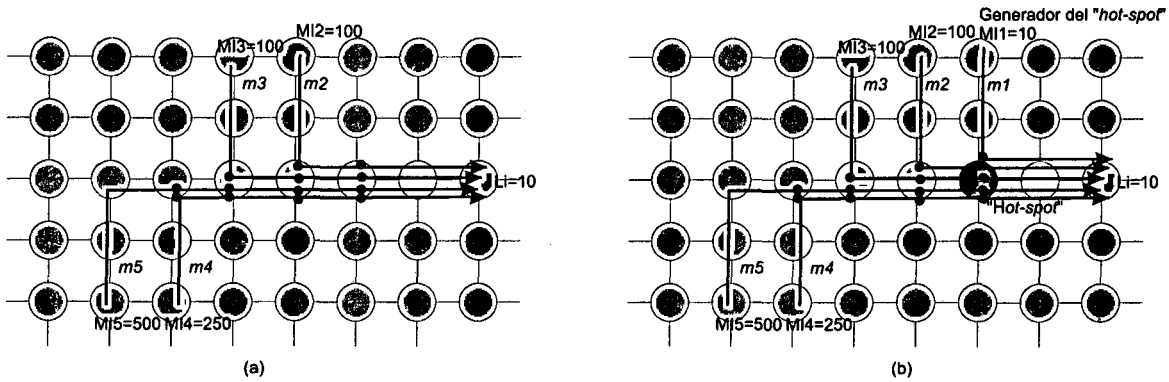


Figura 4-9 Experimento de "*hot-spot*".

(a) sin "*hot-spot*", (b) con "*hot-spot*"

En la Figura 4-10 se puede observar que la latencia en el caso que no hay *"hot-spot"* se mantiene por valores bajos, creciendo en pequeña medida hasta que no se llega a una carga realmente alta de menos de 30 ciclos por mensaje. Por el contrario, en el caso de añadir un mensaje de alta carga potencial saturador de algunos caminos y por tanto generador de un "*hot-spot*", en una zona del camino común, hace que la latencia crezca en gran proporción llegando casi a duplicarse en los peores casos de mayor carga. Se puede observar, asimismo, que la separación entre las curvas que representan

el caso sin y con "*hot-spot*" se va incrementando a medida que se incrementa la carga de los mensajes.

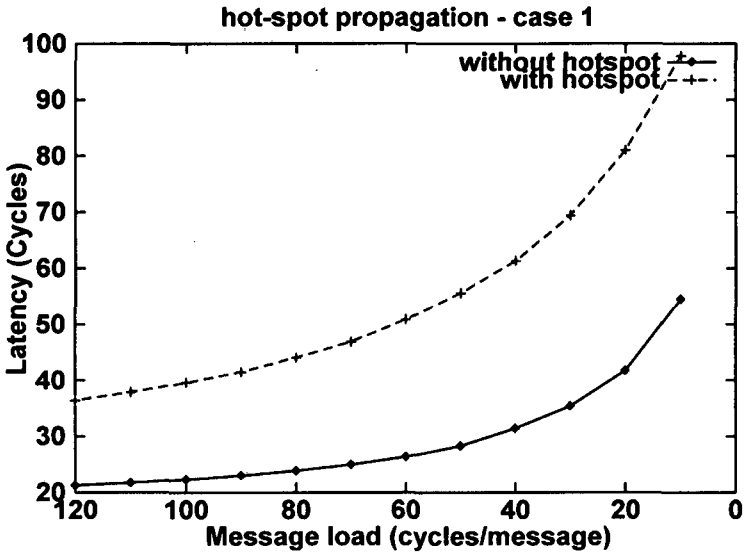


Figura 4-10 Ejemplo 4: Latencia con y sin "*hot-spot*".

4.2.5 Análisis de la respuesta de las redes de interconexión frente a "*hot-spots*"

A partir de estos ejemplos se pueden extraer algunas conclusiones sobre el comportamiento de las redes de interconexión con respecto a la latencia de los mensajes. La principal conclusión es que, 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 cuasi lineal de la latencia en la cual grandes cambios de la carga provocan nulos o pequeños cambios en la latencia. En esta zona, la red de interconexión no esta saturada y no se usa su capacidad al cien por cien 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 lo llamaremos "*latencia umbral*", a partir de la cual, si la carga sigue aumentando, la latencia de la red entra en la zona de saturación.

La parte de pendiente acusada de la curva de latencia es una zona de comportamiento no deseable porque la latencia no es estable y sufre grandes cambios con relación a pequeños cambios del tráfico de la red. Este comportamiento, como hemos comentado, es debido a que la red presenta una saturación exponencial, donde a partir de un punto no se acepta mas tráfico y la latencia crece de manera descontrolada.

También se puede extraer de los experimentos mostrados, qué factores son los que determinan la carga de la red. Hemos visto que tanto la frecuencia de inyección de mensajes (ejemplo 1) como el número de canales (ejemplo 2) que inciden sobre un enlace, son muy importantes en la influencia de la latencia. Asimismo, del ejemplo 2 se puede ver que el número de puntos de colisión de un canal con otros canales influye de manera muy directa en la latencia que experimentan los mensajes que circulan por ese canal, de manera que, a más puntos de colisión, mayor crecimiento de la latencia en un factor que no es lineal con el número de puntos de colisión.

Por otro lado, como era de esperar, la longitud del mensaje también influye en la ocupación de los enlaces de la red y por lo tanto tiene una influencia directa en la latencia, de manera que los tres factores: frecuencia de inyección, número de canales y longitud de los mensajes son los que influyen en la latencia de la red.

Finalmente, el último experimento muestra cómo se produce un efecto "dominó" de propagación de la latencia. 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 pronto colapsará toda la red de interconexión aunque la situación problemática estuviese restringida a una pequeña zona inicialmente.

En este caso, como se observa en las gráficas de la latencia, la red no se encuentra de manera global saturada, sino que existe una carga baja. Pero al introducir un solo canal de carga alta en la red, ésta se comporta como si toda ella estuviese altamente cargada. Se produce un fenómeno de saturación exponencial aún no habiendo altas demandas generalizadas de ancho de banda por parte de los canales. Esta situación de saturación a baja carga de la red se debe a una incorrecta distribución de la carga de comunicaciones en la red, de manera que ciertos enlaces están muy cargados mientras otros tienen recursos libres.

Concluyendo, dos son los aspectos problemáticos que presenta la respuesta en latencia de una red de interconexión: por un lado, la saturación exponencial a partir de una latencia umbral y, por el otro, el efecto "dominó" de propagación de los "*hot-spots*". Estos dos factores, de hecho, impiden aprovechar al máximo la capacidad total de la red de interconexión.

Veamos ahora qué implicaciones tiene este comportamiento en las aplicaciones que se ejecutan sobre el computador de altas prestaciones.

4.3 Implicaciones de la problemática de comportamiento de las redes de interconexión en las aplicaciones

El comportamiento, y las características de ese comportamiento, hasta ahora descritos, de las redes de interconexión tienen una serie de implicaciones sobre las aplicaciones que se ejecutan sobre el computador paralelo. Cómo se ha comentado en el capítulo anterior, el programa se compondrá de una serie de tareas independientes que se ejecutan en paralelo sobre los nodos de cómputo y que intercambian mensajes por la red de interconexión.

Evidentemente, el retardo sobre los mensajes tendrá una influencia directa sobre el retardo de ejecución total del programa porque las tareas deberán esperar a que les lleguen sus mensajes para poder operar. En este sentido, en una primera aproximación el tiempo de ejecución de un programa se puede determinar por la suma del tiempo de ejecución de las tareas sobre los procesadores más el tiempo de comunicar todos los mensajes que genera el programa. Este tiempo, si se conoce el volumen de comunicaciones y la carga que genera sobre la red, se puede estimar calculando la latencia que sufren los mensajes en la red de interconexión. De este modo, a mayor carga, mayor latencia y el usuario debería esperar un tiempo de ejecución mayor proporcionalmente. Otra posibilidad es que se produzca un solapamiento entre cómputo y comunicaciones, de manera que la latencia de la comunicación no tenga influencia en el cómputo. Pero para que este solapamiento sea eficaz, se debe tener controlada y conocer la latencia, pues sino también fallaría y aparecerían los tiempos muertos de espera.

Sin embargo, existen otros factores que hacen que la influencia de la latencia tenga importantes consecuencias sobre la ejecución de la aplicación. Para analizar estos factores vamos a clasificar las aplicaciones en dos tipos, pues para cada tipo de aplicación, tendremos un caso diferente.

El primer grupo de aplicaciones serán las de cálculo intensivo donde las tareas se distribuyen por los nodos de cómputo con objeto de maximizar su uso y minimizar, por lo tanto, el tiempo de ejecución total de la aplicación. Este es el único índice a mejorar y se pretende hacer "lo mejor que se puede" (denominado "best-effort" en inglés). El segundo grupo estará formado por aplicaciones de tipo multimedia que tienen otro tipo de requerimientos diferentes a las de cálculo intensivo. En estas aplicaciones, como la distribución de vídeo bajo demanda, teletrabajo o telemedicina, aparecen una serie de requerimientos mucho más estrictos que el puramente de minimizar el tiempo total de ejecución de la aplicación. Vamos a analizar con profundidad las implicaciones de los retardos en la red de interconexión en cada uno de estos tipos de aplicaciones.

4.3.1 Aplicaciones de cálculo intensivo

En estas aplicaciones, como se vio en el punto 3.1.3 del capítulo 3, se realiza 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 estén ejecutando tareas y ningún procesador se quede ocioso sin ninguna tarea disponible mientras otros procesadores tienen tareas que están esperando a ser ejecutadas. Esta distribución no es fácil de realizar y depende de una serie de factores el que se realice correctamente o no. Los factores principales de los que depende son las dependencias funcionales de las tareas, es decir, si una tarea debe esperar a que acabe otra precedente porque usará el dato producido por esta última, y los volúmenes de cómputo y de comunicación de las tareas. La relación volumen de cómputo/volumen de comunicación se llama granularidad.

Con la información de los volúmenes de cómputo y las características de los nodos de cómputo, por un lado, y los volúmenes de comunicación y las características de la red de interconexión, por el otro, la fase de asignación pretende estimar los tiempos de ejecución y de comunicaciones de las tareas, respectivamente, para conocer la granularidad. Con estos tiempos se realiza la asignación de tareas a nodos para cumplir los objetivos antes mencionados de minimizar el tiempo de ejecución. El problema principal es que las estimaciones de los tiempos de cómputo y comunicación no son todo lo exactas que sería necesario para hacer una asignación óptima. Con respecto al tiempo de comunicación, que es el que nos ocupa en este caso, el tema es especialmente complicado por el comportamiento de la latencia de las redes de interconexión antes expuesto.

Con respecto al tiempo total de ejecución del programa, la latencia de comunicaciones debe ser minimizada con objeto de minimizar tal tiempo de ejecución. Pero también es necesario conocer este dato, el tiempo de comunicación de cada tarea, para realizar una asignación de tareas a nodos de cómputo adecuada. Por esta razón, el comportamiento de la latencia en la zona de la pendiente vertical es indeseable porque la latencia experimenta grandes variaciones y toma valores impredecibles. La cuestión es que una cierta cantidad de latencia es posible tolerarla mediante el método de asignar un "exceso" de paralelismo, es decir, teniendo suficientes tareas por nodo de cómputo, de manera que si alguna o algunas están esperando por sus mensajes, siempre haya otras tareas disponibles para ser ejecutadas.

Este principio, sin embargo es opuesto al de distribuir al máximo las tareas entre el mayor número de nodos de cómputo para conseguir el máximo paralelismo y minimizar el tiempo de ejecución. Por esto, se debe llegar a un cierto compromiso y balancear adecuadamente la carga de cómputo. Para ello debe conocerse el tiempo de comunicación de las tareas. Pero si sucede, como hemos visto en los ejemplos, que cuando la red se comporta en la zona de saturación, pequeños cambios imprevistos del comportamiento de las tareas pueden dar lugar a grandes cambios en la latencia, 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 incrementará el tiempo total de ejecución del programa porque no se hará una utilización adecuada del sistema. Por esta razón, lo que se debe evitar, más que la latencia, es que la misma experimente grandes variaciones, es decir que se trabaje en la zona de la curva de pendiente acusada.

4.3.2 Aplicaciones multimedia

En este tipo de aplicaciones, distribución de vídeo bajo demanda, videoconferencia, etc., se dan requerimientos de tiempo real, por un lado, y de sincronismo, por el otro. Para el buen funcionamiento de estas aplicaciones es deseable que la latencia sea baja, como en el caso anterior. Pero en este caso, los efectos de cambios bruscos e inesperados en la latencia todavía son peores que en el caso anterior, porque, en ese caso, el efecto pernicioso provocado era el aumento del tiempo de ejecución. Ahora, con aplicaciones con requerimientos de tipo multimedia, el efecto de la latencia puede incluso impedir el correcto funcionamiento de dichas aplicaciones. Estas aplicaciones tienen necesidades especiales de tiempo real, sincronismo o interactividad, que hacen que la latencia sea mucho peor tolerada. Por ejemplo, en la recepción de una película de vídeo distribuida por la red, es necesario que las imágenes

lleguen a los receptores de destino a un ritmo constante y, si imagen y sonido se envían por separado, se necesita que lleguen sincronizados al mismo tiempo. En el caso de aplicaciones cooperativas e interactivas por parte de los usuarios, como videoconferencia, es necesario que la latencia sea baja y constante para permitir un trabajo cooperativo eficiente y una interacción real. Nuevamente, se observa que la latencia de comunicaciones tiene grandes implicaciones en la ejecución de las aplicaciones, no sólo en el tiempo de duración de la ejecución, sino también en el desarrollo mismo de dichas aplicaciones.

Hasta aquí hemos visto el comportamiento de las redes de interconexión y la problemática asociada de ese comportamiento sobre las aplicaciones a ejecutar. En el próximo punto se introducen los requerimientos que una red de interconexión ideal desde el punto de vista del usuario, programador o diseñador de un computador de altas prestaciones, debería ofrecer según nuestro punto de vista. A continuación, presentamos con detalle el mecanismo diseñado para conseguir dichos requerimientos.

4.4 Diseño de la red de interconexión

En este punto, se abordan primeramente los requisitos que debería cumplir una red de interconexión de un computador paralelo. En el punto siguiente, se dan unas ideas de cómo se pueden conseguir esos requisitos.

Después del análisis del comportamiento que hemos hecho sobre las redes de interconexión y las implicaciones que provocan en el funcionamiento de las aplicaciones que se ejecutan, y de la constatación de la importancia de la red de interconexión tanto desde el punto de vista de las prestaciones como desde el punto de vista del modelo del computador de altas prestaciones, vamos ahora a definir cuál debería ser el comportamiento de una red ideal desde el punto de vista del programador o el diseñador de un computador de altas prestaciones.

La forma más amplía de definir el objetivo básico que debe cumplir una red de interconexión es decir que la red debe ser un subsistema dentro del computador capaz de interconectar de manera lo más eficiente posible los nodos que componen el computador de altas prestaciones.

Podemos concretar las palabras "de manera lo más eficiente posible" concretando los aspectos que una red de interconexión ideal debería ofrecer:

- ✓ Un bajo coste de interconexión que crezca linealmente con el número de nodos de procesamiento.

- ✓ Una capacidad de comunicación, o ancho de banda, que crezca linealmente con el número de nodos de procesamiento.

- ✓ Un retardo de transmisión de mensajes mínimo, e independiente del número de nodos de procesamiento, o al menos que dependiese logarítmicamente de él. Idealmente, pediríamos a una red de interconexión un "overhead" mínimo en la ejecución de las tareas paralelas, es decir, un tiempo de comunicación nulo, o por lo menos, comparable al de un acceso a la memoria del propio nodo, ya que gracias al solapamiento entre cómputo y comunicaciones lo importante es la predicibilidad y estabilidad del tiempo de comunicación.

Para poder ofrecer los aspectos anteriores, una red de interconexión debe presentar una serie de características que las podemos agrupar en tres categorías: Escalabilidad, características estáticas y características dinámicas.

4.4.1 Escalabilidad

Respecto a la escalabilidad, existirán dos aspectos importantes: Primero, que el coste de los enlaces, memoria, y demás elementos físicos necesarios para construir la red se mantenga proporcional con el número de nodos de procesamiento y, segundo, que las prestaciones de la red de interconexión no se degraden drásticamente con respecto al incremento del número de nodos de cómputo. Este aspecto significa que la capacidad de la red debe crecer proporcionalmente respecto al número de nodos de procesamiento. Evidentemente, algunos aspectos de la escalabilidad de la red (el coste, que sea factible físicamente, etc.) dependen de la tecnología, pero otros (grado de la red, ancho de bisección, etc.) dependen del diseño.

Los dos aspectos mencionados coste y prestaciones son contradictorios por lo que se deberá llegar a algún compromiso. Por ejemplo, en una red de grado no constante como el hipercubo, el ancho de banda se mantiene constante con el número de nodos de cómputo, pero el coste se dispara, porque el número de enlaces por nodo no es constante.

4.4.2 Características estáticas

Respecto a las características estáticas, una red de interconexión debe ofrecer posibilidad de que cualquier nodo pueda comunicar con cualquier otro de la red. Además esta conexión debe ser libre de "*deadlock*", "*livelock*" y "*starvation*". Estos aspectos se pueden lograr eligiendo el diseño adecuado de entre los parámetros de diseño estático definidos en el capítulo 2, como son la topología, el control del flujo, el

encaminamiento, la técnica de evitación de "*deadlock*", ... entre otros. Con estas características, la red de interconexión se vería como en la Figura 4-11, donde la conexión entre cada par de nodos no presenta el mismo ancho de banda disponible para todos ellos, por lo que comunicar entre cada par de nodos no cuesta el mismo tiempo, por lo que éstos aparecen como si se encontrasen a diferentes "distancias" entre ellos. Este hecho se produce porque el "*hardware*" no es capaz de ocultar esas diferencias de distancia entre nodos.

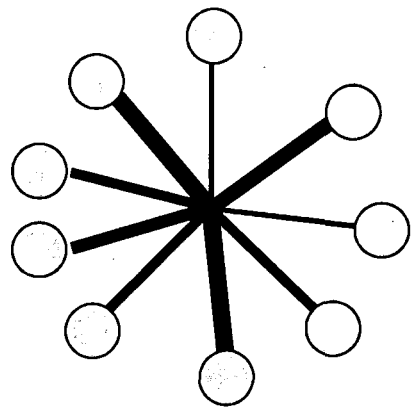


Figura 4-11 Conexión todos con todos con diferente ancho de banda

4.4.3 Características dinámicas

Respecto a las características dinámicas, sería deseable que una comunicación para acceder a un dato remoto costase un tiempo comparable a un acceso a memoria local de acceso aleatorio. Es decir, es importante que la latencia de los mensajes esté acotada y que no haya grandes diferencias de latencia debidas a la distancia física entre los nodos de cómputo o a la contención entre mensajes. Esto significa que la latencia de los mensajes debería tender hacia un valor medio con una varianza pequeña.

Por otro lado, el "*throughput*" de la red, entendido como número de mensajes que la red entrega a su destino por unidad de tiempo, debería ser maximizado [110]. Estos dos objetivos, latencia y "*throughput*" son contradictorios porque un mayor "*throughput*" significa un mayor número de mensajes en la red y, por lo tanto, una mayor latencia. El argumento complementario se puede enunciar diciendo que para que los mensajes tengan poca latencia debe haber pocos de ellos presentes en la red de interconexión, lo que implica un "*throughput*" bajo.

Por lo tanto, debe buscarse un equilibrio entre ambos. Aquí se han establecido de esta forma, latencia pequeña e uniforme y "*throughput*" grande, pensando en cómo debería ser una red de interconexión de propósito general. Un aspecto importante es conseguir una latencia uniforme para los mensajes que circulan por la red, es decir,

acotada por un valor máximo y con pequeña dispersión. Esta latencia uniforme lo debería ser independientemente del camino que recorren los mensajes, la distancia o el nodo destino, o la carga, es decir, la contención de la red.

Con esta definición de latencia uniforme, se conseguiría una conexión de los nodos de cómputo del computador de altas prestaciones a través de la red de interconexión de todos con todos, con la misma capacidad de comunicación independientemente de la carga de la red y el patrón de comunicaciones, con lo que los nodos aparentarían estar a la misma "distancia" lógica entre ellos. Con estas características la red se vería como en la Figura 4-12, una red ideal virtualizada por capas "*hardware*" y "*software*".

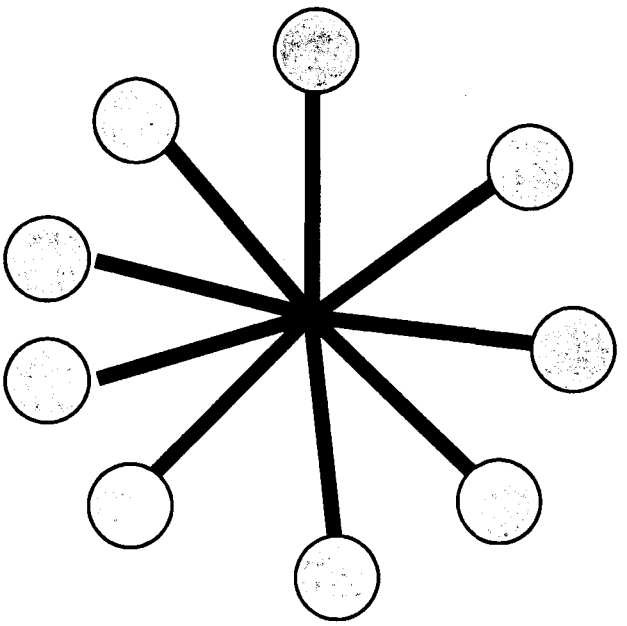


Figura 4-12 Conexión todos con todos al mismo ancho de banda

4.4.4 Objetivos de diseño de una red de interconexión

Idealmente, por lo tanto, pediríamos a una red de interconexión que ofreciese un ancho de banda ilimitado, de manera que el coste en tiempo de la comunicación fuese nulo o al menos constante. En este sentido, el programador desearía que el tiempo de enviar o recibir un mensaje fuese igual o comparable al tiempo de acceder a una variable en memoria local de manera que no hubiese diferencia entre acceder a la propia memoria del nodo de computación local que a la memoria de otros nodos vecinos, más o menos lejanos.

Este objetivo general se podría cumplir si los nodos de cómputo estuviesen conectados cada uno de ellos con todos los demás. Pero ya hemos visto que es

imposible de cumplir por razones de coste económico, de que sea factible físicamente o porque sea posible según la lógica de funcionamiento. Solamente una red de interconexión que ofrezca una conexión directa de cada uno de los nodos de procesadores con todos los demás, tipo "crossbar" podría ofrecer esos requerimientos. Pero el coste de un "crossbar" de tal tipo crece cuadráticamente con el número de nodos de cómputo y, a partir de un cierto número, no es posible realizarlo físicamente. Además, en el caso de accesos simultáneos por parte de varias tareas a la memoria de un mismo nodo de procesamiento, aparecería una contención debida a la propia lógica del programa imposible de eliminar [33]. La otra alternativa, conexión común a través de un único elemento tipo "*bus*" compartido, cuyo coste si que es escalable, tampoco es viable por la contención por el acceso común al "*bus*" cuando el número de nodos de cómputo conectados a él aumenta ligeramente, ya que el ancho de banda es fijo y no escalable.

Es por estas razones que una cierta latencia de comunicaciones se debe asumir en la red de interconexión. Pero, por otro lado, después de estudiar el comportamiento de la latencia en el capítulo anterior, hemos visto también que ciertos efectos, como variaciones bruscas de la latencia, deben ser evitados. Nuestra propuesta para conseguir conjugar ambas cuestiones es asegurar que la red de interconexión presente un comportamiento con una latencia baja y controlada de manera que se eviten los "*hot-spots*" [34].

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

- ✓ La granularidad 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, desde red libre de mensajes hasta niveles próximos al de saturación. De hecho, el punto de saturación se sitúa en un punto de carga más elevado.

- ✓ Una consecuencia muy importante sería 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.

Actualmente, se afirma que una red asegura un buen rendimiento si se utiliza por debajo de una cierta fracción del ancho de banda disponible en la red. Este límite superior se considera satisfactorio cerca del 20 al 30 por ciento del ancho de banda total

de la bisección de la red ([2] [3] [32]). Para valores superiores, la degradación del comportamiento se produce drásticamente. Esto es debido a que en cualquier momento puede aparecer una zona de congestión o "*hot-spot*" y se necesita tener una ancho de banda de reserva para amortiguar sus efectos. Como no se sabe dónde ni cuándo se formará el "*hot-spot*", ni su duración, ni su influencia sobre el resto de la red de interconexión, ésta debe mantenerse bajo un nivel de utilización muy pequeño. Esto es debido a que cada comunicación debe reservar sobre su propio camino un ancho de banda extra para que no se produzca el "*hot-spot*" cuando aumenta la demanda ya que se utilizan caminos estáticos que no pueden utilizar el ancho de banda potencialmente libre de otros caminos de la red de interconexión.

Actualmente, con las técnicas existentes de encaminamiento para el diseño y construcción de redes de interconexión, este valor es demasiado bajo y además, no se puede predecir la latencia que sufrirá un mensaje en la red. Por lo tanto, debería buscarse la manera de conseguir ofrecer al programa la posibilidad de que pueda hacer un uso mayor de la red de interconexión (de hasta un 60 o 70 por ciento de su capacidad, por ejemplo) junto con una latencia uniforme, aunque ello implique añadir un mecanismo que suponga un "overhead" que consuma un cierto ancho de banda. La cuestión es que, actualmente, debe gastarse en redes más rápidas lo que podría conseguirse incrementando la utilización de la red existente.

Nuestro objetivo aquí expuesto de diseño de la red de interconexión de un computador contempla un modelo en el que se tiene en cuenta el programa de aplicación que se ejecuta sobre ella. La situación sería similar al diseño de procesadores tipo RISC ("*Reduced Instruction Set Computer*", Computador de Repertorio de Instrucciones Reducido), en los que se tiene en cuenta las características del modelo del programa a ejecutar al diseñar la arquitectura.

La aplicación o programa a ejecutar, independientemente de la distribución de tareas entre nodos de cómputo, presentará unas necesidades de comunicación expresadas como un ancho de banda intrínseco, es decir, un volumen de comunicaciones entre las tareas que la componen a lo largo del tiempo. La aplicación, considerada ahora junto con el computador de altas prestaciones, presentará unas necesidades de ancho de banda mayores que serán el ancho de banda intrínseco multiplicado por el número de enlaces de la red de interconexión que recorren los mensajes desde el origen hasta el destino. La situación ideal sería, pues, que este ancho de banda requerido de la aplicación junto con el computador no superase en ningún momento el ancho de banda ofrecido por propio del computador, entendido como el número de enlaces por el ancho de banda de cada enlace.

Este análisis puede hacerse tanto globalmente en una primera aproximación como luego refinarlo considerando cada enlace y/o instante de tiempo.

En una red de interconexión, el ancho de banda total disponible se reparte entre el ancho de banda consumido por los mensajes en tránsito como el ancho de banda de los mensajes que se inyectan o se consumen. Esta división no tiene porque ser estática, sino que puede ir variando a lo largo del tiempo según interese dar más prioridad a los mensajes en tránsito (lo que implicaría bajar la latencia pero también el "*throughput*") o a la inyección de mensajes (lo que incrementaría el "*throughput*" pero también la latencia) No parece que uno de estos anchos de banda dependa del otro, en el sentido que el aumentar el ancho de banda de inyección haga disminuir el de tránsito, o que aumentar el de tránsito haga disminuir el de inyección. Éstas son dos magnitudes independientes y, por lo tanto, no podría haber una autorregulación y llegarse a una situación de equilibrio. El aspecto global a tener en cuenta desde el punto de vista del programador de la aplicación paralela es cuál es el ancho de banda que la red de interconexión entrega a la aplicación para que lo utilice.

4.5 Balanceo de las comunicaciones

En el punto anterior se ha establecido cuál es nuestra visión de una red de interconexión para un computador de altas prestaciones. En este punto expondremos las ideas para conseguir el comportamiento de la red de interconexión como el descrito en el punto de objetivos. La idea está basada en conseguir un balanceo global de las comunicaciones para conseguir un uso uniforme de la red. Por lo tanto, debemos responder a la pregunta de cómo realizar el balanceo de las comunicaciones propuesto.

El método para conseguir los objetivos de diseño de una red de interconexión para un computador de altas prestaciones es realizar un *balanceo distribuido de las comunicaciones* de manera que la carga de comunicaciones esté uniformemente distribuida por los enlaces de la red y se eliminen las zonas de saturación o "*hot-spots*". Esta aproximación de balanceo sería similar y paralela a la que se hace tradicionalmente sobre la carga de cómputo, como se ha explicado aquí. En el balanceo de carga de cómputo se distribuye ésta entre los nodos de cómputo [100]. En el balanceo de la carga de comunicaciones se distribuye ésta entre los enlaces de comunicación que existen en la red de interconexión. Esta aproximación de balanceo de las comunicaciones presenta un nuevo enfoque al tratar el problema de la latencia de comunicaciones de una forma distinta a la de los esquemas tradicionales de encaminamiento dinámico o adaptivo.

El balanceo de las comunicaciones sobre la red de interconexión debería conseguir una serie de objetivos como son una latencia uniforme, latencia predecible y

eliminación de "*hot-spots*". Estos tres objetivos representan caras de un mismo poliedro, de manera que hablar de uno de ellos implica hablar de los otros implícitamente. La latencia uniforme implica que no tenga grandes variaciones al variar la carga de comunicaciones, tal y como se ha definido ésta última anteriormente. Si la latencia es uniforme será fácilmente predecible, es decir, si la carga de comunicaciones se puede conocer de manera global y nosotros la distribuimos uniformemente por la red de interconexión, podemos conocer la latencia que se producirá, a partir de la capacidad de los encaminadores y enlaces de la red de interconexión, porque conoceremos la carga en cada enlace de comunicación.

Por último, el hecho de distribuir uniformemente la carga de comunicaciones por los enlaces de la red, hace que no se produzcan "*hot-spots*", por propia definición, sino que los enlaces de la red se usen uniformemente. De esta manera, los enlaces se pueden aprovechar al máximo mientras sea posible porque la red no esté globalmente saturada.

Nuestra propuesta para conseguir estos objetivos es, pues, balancear la carga de comunicaciones uniformemente por la red de interconexión. El mecanismo que hemos diseñado para conseguir ese balanceo es la búsqueda de caminos alternativos cuando los caminos originales que se están usando empiezan a saturarse. Por lo tanto, el mecanismo se basa en la expansión controlada por la latencia de los caminos que usan los canales para enviar los mensajes. Para ello 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. Los próximos puntos presentan, desarrollan y explican nuestra propuesta con profundidad.

4.6 Antecedentes: Encaminador Universal

En este trabajo, se han utilizado como base las ideas desarrolladas por L. Valiant y M. May presentadas en [130] y [88], respectivamente, de diseño de encaminadores universales. A continuación se presenta un breve resumen de los objetivos, las ideas y las técnicas de sus trabajos. Al principio de los años 80, L. Valiant desarrolló una teoría para conseguir una latencia uniforme de los mensajes distribuyendo de manera aleatoria los mensajes entre los caminos disponibles en la red de interconexión.

Posteriormente, M. May y el grupo de la Universidad de Kent en el Reino Unido, utilizaron las ideas de L. Valiant para desarrollar lo que ellos llamaron el encaminador universal. En sus trabajos [88] afirman que las colisiones entre mensajes en la red de interconexión pueden afectar de manera drástica al rendimiento de la aplicación. Es más, en ausencia de algún límite superior para la latencia de los mensajes es difícil, sino

imposible, diseñar programas paralelos eficientemente. Por lo tanto, el problema de construir computadores paralelos de propósito general depende de responder a la siguiente cuestión:

¿Es posible diseñar un sistema de encaminamiento universal, es decir, una red de interconexión y algoritmo de encaminamiento, realizables, que puedan implementar todos los patrones de comunicación con una latencia de mensajes acotada?

De hecho, un sistema de encaminamiento universal que permite la construcción de computadores paralelos de propósito general escalables ya fue descubierto por Valiant en 1980 [129]. Este sistema cumple dos importante requerimientos:

- ✓ El "*throughput*" de la red se incrementa proporcionalmente con el número de nodos de la misma.
- ✓ El retardo a través de la red se incrementa ligeramente con el número de nodos, digamos, proporcionalmente al logaritmo del número de nodos.

El objetivo era maximizar la capacidad y minimizar el retardo bajo carga altas de tráfico porque una red de interconexión *paralela* es un componente vital del computador paralelo. Esto no es lo mismo que, por ejemplo, minimizar el retardo en una red cuasi vacía.

Su propuesta para conseguir estos objetivos, pasaba por utilizar encaminamiento aleatorio para enviar los mensajes en dos fases de la siguiente forma: Para cada mensaje a enviar entre dos puntos, digamos A y B, se elige un tercer nodo C aleatoriamente entre todos los de la red de interconexión, y, entonces, se envía el mensaje primeramente desde A hasta C y a continuación desde C hasta B. Todos los nodos de la red de interconexión eran candidatos a ser elegidos como nodos intermedios con la misma probabilidad, independientemente de las posiciones de A y B o de la distancia que los separase o del tráfico presente en la red.

Con este mecanismo se conseguiría aleatorizar el patrón de comunicaciones de la red de interconexión de manera que el patrón resultante aplicado sobre la red presentaría siempre una distribución uniforme. Sin aplicar este sistema, la latencia máxima de los mensajes depende del patrón de tráfico presente en cada momento en la red de interconexión. Algunos patrones serán rápidos mientras que otros serán lentos. El encaminamiento universal conseguido con el método de aleatorizar las comunicaciones superaba este problema acotando el tiempo que un conjunto de comunicaciones tomará en la red. Según demuestra en sus trabajos, la probabilidad de superar esta cota puede hacerse arbitrariamente pequeña. La mejora de esta cota superior es de gran beneficio

para el computador paralelo, y puesto que sólo interesaba esa cota superior, cualquier empeoramiento en los patrones rápidos no era relevante.

La idea intuitiva del encaminamiento universal es directa. Con el hecho de introducir el comportamiento aleatorio en la red, se perturba el patrón de tráfico sistemático, que son los que causan los retardos excepcionales, y se dispersa la carga a lo largo de la red de interconexión de manera que una mayor cantidad de enlaces se puede usar concurrentemente.

A partir de este punto, argumentan que sería posible construir un computador paralelo de propósito general caracterizado por tres parámetros únicamente: el número de procesadores p, el retardo en la red de interconexión l y la relación entre el "*throughput*" de comunicaciones y el "*throughput*" de cómputo g.

Este computador tendría dos importantes características: la primera es que sería totalmente escalable y la segunda que sería eficiente y, por lo tanto, fácil de usar por parte del programador.

Con este computador paralelo, el procesamiento paralelo se caracterizaría por tres propiedades:

- ✓ Simplicidad, la cual es necesaria para desarrollar fácilmente aplicaciones paralelas sobre computadores paralelos.
- ✓ Eficiencia, necesaria para tener un aprovechamiento sostenido de la capacidad del computador paralelo por encima de un valor digno.
- ✓ Portabilidad, para que las aplicaciones puedan ser migradas de una plataforma "*hardware*" a otra, contemporánea o futura, independientemente de sus características específicas.

Los argumentos de P.H. Welch en [132] abogan por una reconciliación del "*software*" y del "*hardware*" paralelos. Según este autor, el divorcio entre las aplicaciones, el "*software*" de desarrollo de programas paralelos y los computadores paralelos hace que actualmente se vea el paralelismo como una técnica difícil de utilizar, no determinista, difícil de depurar y difícil de portar a otras plataformas.

La cuestión planteada es que la naturaleza de las aplicaciones sería paralela, como lo es el mundo real, compuesto de entes que actúan independientemente y que se relacionan entre si. Hasta la fecha actual, con el desarrollo de los computadores serie, se ha desarrollado un "*software*" de desarrollo de aplicaciones serie, que encaja muy bien

con el "*hardware*", pero que no permite expresar la naturaleza intrínsecamente paralela de las aplicaciones. El puente que hace de encaje entre "*hardware*" y "*software*" es el conocido modelo de Von Neuman. Pero hace falta un "puente" entre "*software*" y aplicaciones. Esta situación se muestra en la Figura 4-13.

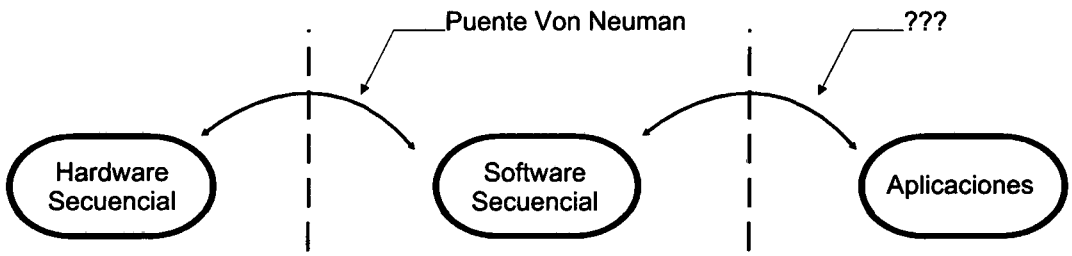


Figura 4-13 Situación del mundo secuencial

Con el procesamiento paralelo, se precisan resolver los dos puentes entre los componentes. Para permitir expresar el paralelismo inherente de las aplicaciones y aislar al programador de los detalles coyunturales del computador paralelo, hace falta un lenguaje de programación paralelo que incluya las ideas de concurrencia, sincronismo y comunicación en todos los niveles de descripción de la aplicación, desde el algoritmo hasta el programa paralelo. Este primer puente se construiría con un lenguaje que incluyese esas características al nivel de su sintaxis y semántica. Con ello se conseguiría verificar mejor los programas paralelos. El segundo puente es el reclamado por Valiant. El paralelismo del programa paralelo, ahora ya reconciliado con la aplicación, puede ser diferente del paralelismo del computador paralelo, por ejemplo en la granularidad, la comunicación o la topología de interconexión. Hace falta, pues, un puente que permita asignar eficientemente el uno sobre el otro, independientemente de los estilos subyacentes de cada uno de ellos. Valiant propone su concepto de "*Bulk Synchronous Parallelism* (BSP)" para construir este puente [131]. Con ello la computación paralela se vería como en la Figura 4-14.

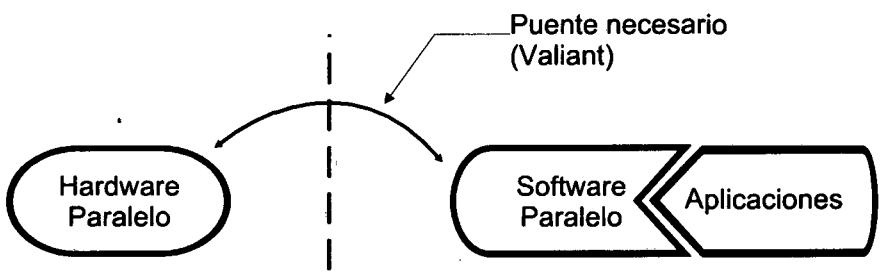


Figura 4-14 Situación ideal del mundo paralelo

La situación actual real de una parte del computo paralelo es intentar aprovechar las aplicaciones existentes y desarrollar nuevas aplicaciones con un enfoque de

"*software*" secuencial sobre un "*hardware*" paralelo. Las razones para ello son aprovechar las inversiones hechas tanto en aplicaciones existentes como en formación de los programadores que realizan aplicaciones serie. La verdad es que este enfoque parece muy arriesgado por los costes ocultos entre las aplicaciones y el "*software*" serie y el coste desconocido hoy en día entre el "*software*" serie y el "*hardware*" paralelo. Esta parece la peor situación de las tres, tal y como se ve en la Figura 4-15.

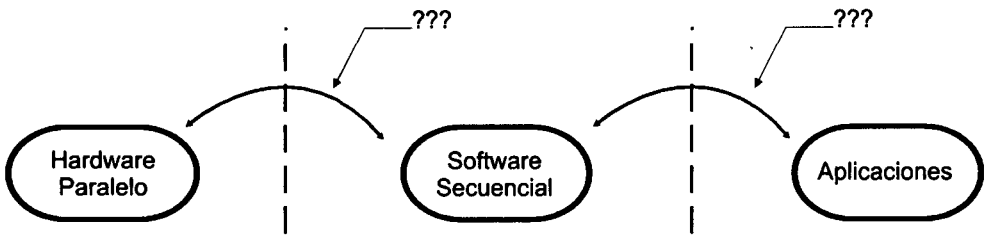


Figura 4-15 Situación actual del mundo paralelo

4.7 Balanceo distribuido del encaminamiento

Tal y como se comentó en la introducción, el método propuesto en este trabajo llamado balanceo distribuido del encaminamiento (DRB por sus siglas en inglés) es un método para crear caminos alternativos entre fuente y destino en la red de interconexión. DRB distribuye la carga de mensajes de cada par fuente-destino entre un camino "multicarril" constituido por varios caminos simples. Esta distribución esta controlada por el nivel de carga del camino multicarril. Estos nuevos caminos harán uso de los enlaces disponibles de los encaminadores.

Los nuevos caminos utilizados pueden ser de distancia mínima o no, dependiendo de la situación. En el desarrollo del método, veremos que se puede configurar para usar solamente caminos de distancia mínima o permitir un cierto alargamiento de los caminos, siempre de manera controlada.

Como ya se comentó, el objetivo de DRB es una distribución uniforme de la carga de tráfico sobre la red de interconexión entera para mantener una latencia baja y uniforme y eliminar la generación de "*hot-spots*". En este sentido, la distribución realizada por DRB mantendrá una latencia baja y uniforme sobre toda la red de interconexión siempre que la demanda total de ancho de banda de comunicaciones no supere la capacidad total de los enlaces y encaminadores de la red de interconexión. Dependiendo de la carga de tráfico y de su distribución sobre la red de interconexión, el

método DRB distribuye la carga de los caminos mas cargados hacia los menos cargados.

La idea principal de la validez del método se basa en el comportamiento de la latencia respecto al nivel de carga de tráfico de mensajes en la red de interconexión. Este comportamiento es el que se ha mostrado en la sección 2 a través de una curva con dos tipos de comportamiento: una parte plana y otra vertical. En la parte vertical, recordemos, la latencia toma grandes valores y es inestable, pues pequeños cambios de la carga de tráfico pueden provocar grandes cambios en la latencia. Por lo tanto, la zona vertical es una zona de trabajo no deseable y se debe evitar.

Veamos con un ejemplo el principio de funcionamiento del método. Supongamos ahora que tenemos una red de interconexión con una carga de mensajes en los que unos canales están en zona de saturación y otros no están en esa zona, sino que se encuentran trabajando en la zona plana y, por lo tanto, no están aprovechando el máximo de capacidad del camino que utilizan.

Los caminos saturados se encuentran en esa situación porque la carga es muy alta debido al número de canales que los utilizan, la frecuencia de inyección de mensajes o su longitud, tal y como se vio en el capítulo anterior.

Entonces, de acuerdo con el comportamiento de la latencia descrito en ese capítulo, el método DRB distribuye los canales desde la zona congestionada a zonas no saturadas con objeto de mover el punto de trabajo de la curva de respuesta de latencia desde el punto de saturación a otro punto de baja latencia en la parte plana de la curva.

Este efecto se consigue, pues, modificando la distribución de canales para reducir el tráfico en los caminos más cargados de la red de interconexión. Evidentemente, los caminos que estaban poco cargados en la zona plana y reciben una parte de la carga de los caminos muy cargados, verán su carga aumentada. Estos caminos, se moverán por la parte plana de la curva hacia un punto de carga mayor, pero la latencia no se verá aumentada en gran cantidad precisamente porque están trabajando en la zona plana de la curva en ausencia de saturación.

Esta explicación del efecto conseguido con DRB sobre el ejemplo se muestra en la Figura 4-16 . En ella se muestra el resultado en latencia del ejemplo 1 de la sección 2. Supongamos que algunos canales de la aplicación se encuentran trabajando a baja carga (Canales L) y otros en la zona de saturación (Canales S), los cuales están trabajando por encima de la latencia umbral, tal y como se definió en el último punto de la sección 2. Con la distribución de tráfico conseguida mediante DRB, cierta parte de la

carga de los canales saturados se moverá hacia los no saturados, disminuyendo en aquellos y aumentando en estos en la misma medida. El resultado en latencia serán grandes reducciones en los canales que al principio estaban saturados, saliendo de esta condición y pasando a la parte plana de la curva (Canales S'), pasando del punto A al A', por ejemplo; y pequeños aumentos en la latencia de los canales que se encontraban no saturados ya que éstos continúan aún trabajando en la parte plana de la curva (Canales L'), pasando del punto B al B'. La consecuencia es que el efecto global en la latencia de la red de interconexión es positivo.

Debemos señalar que el comportamiento de DRB:

- ✓ Es un comportamiento para todos los pares fuente-destino
- ✓ Es dinámico, es decir, en función de las altas latencias se dispara el mecanismo de crear nuevos caminos para los canales saturados lo que influencia a los no saturados.

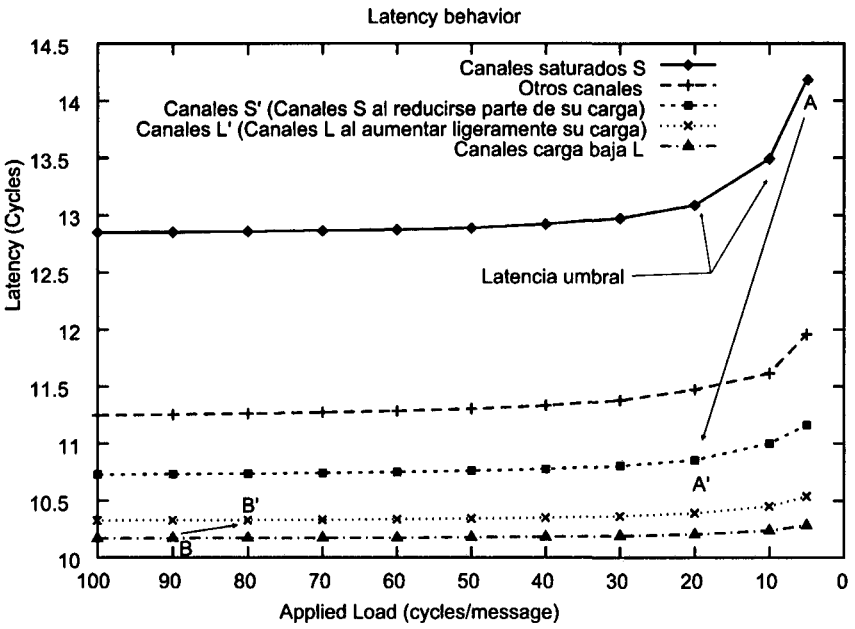


Figura 4-16 Ejemplo de balanceo de la carga de comunicaciones

Después de esta introducción genérica al método DRB, podemos ahora concretar sus objetivos:

- ✓ La reducción de la latencia de los mensajes bajo un cierto valor umbral por medio de la adaptación dinámica del número de caminos que utilizan los canales, mientras se mantiene una latencia uniforme para todos los mensajes. Este objetivo se

consigue optimizando el uso de los recursos de la red de interconexión con objeto de minimizar los retardos de comunicación

- ✓ La minimización del alargamiento del camino, de manera que se ajuste lo más posible al camino de distancia mínima. Este punto es importante para las técnicas de control del flujo tipo "*Wormhole*" y "*Cut-Through*" porque alargar el camino supone incrementar el uso de ancho de banda y el número de puntos de colisión, que como ya hemos visto en el estudio sobre el comportamiento de las redes de interconexión visto en el capítulo anterior, es un factor muy importante. Para redes controladas bajo "*Store-and-forward*" también es muy importante pues el retardo de transmisión depende directamente de la longitud del camino que recorre el mensaje.

- ✓ La optimización del uso de los recursos de la red, de todos los recursos disponibles en todo momento, en general, y de los enlaces de los encaminadores que inyectan y reciben mensajes, en particular, mediante la distribución equitativa de los mensajes entre todos ellos.

Hasta ahora hemos descrito las bases de DRB. En los siguientes dos apartados vamos a mostrar cómo DRB crea los caminos alternativos y se introducirá la política de encaminamiento diseñada para seleccionar en cada envío un determinado camino de entre los posibles caminos alternativos disponibles.

4.7.1 Creación de caminos alternativos mediante DRB

La forma de crear caminos alternativos para un canal lógico del programa de aplicación, tal y como se definió en el capítulo 3, entre los nodos fuente y destino se basa en enviar los mensajes a nodos intermedios de la red antes de enviarlos al nodo destino final. De esta forma, el camino del mensaje se recorre en diversas fases o etapas. En la definición actual, consideramos dos destinos intermedios, las razones de lo cual se explicarán mas adelante, de manera que el camino queda dividido en tres segmentos y se realiza en tres etapas: desde el nodo origen al primer destino intermedio, de ahí al segundo nodo intermedio, y, finalmente, desde este punto al destino final. Cada uno de los pasos individuales se realiza utilizando encaminamiento estático mínimo.

Esta forma de operar pretende hacer que los mensajes utilicen un camino diferente que el camino original determinado por el encaminamiento estático definido para la red de interconexión. Esta técnica pretende desacoplar el patrón de tráfico de la aplicación y la topología física de la red de interconexión. Este sistema de pasos intermedios pretende distribuir uniformemente por toda la red de interconexión todo el tráfico de mensajes aunque la aplicación tenga un patrón en el que aparezcan nodos saturados y

otros con poca carga de comunicaciones. La utilización de la red se va haciendo en función de la carga de mensajes, al ir distribuyéndolos para mantener una ocupación de los caminos físicos que permita la existencia de valores bajos de latencia.

El primero de los nodos intermedios se escoge de entre un subconjunto de nodos "cercano a", o "centrado en", el nodo origen, mientras que el segundo de los nodos intermedios se elige de un subconjunto "cercano a", o "centrado en", el nodo destino. En este punto se define cómo se crean los conjuntos de nodos cercanos a uno dado de manera que nos sea útil a nuestros propósitos. Estos subconjuntos de nodos, que siempre incluyen al nodo fuente o destino, se llaman *supernodos*. La Figura 4-17 muestra el concepto de expansión de caminos mediante el uso de saltos intermedios a zonas formadas por nodos cercanos a los nodos fuente y destino. De esta manera la comunicación entre un par nodo fuente y destino se realiza mediante múltiples caminos formados entre los nodos intermedios llamados "carriles", formando un camino expandido.

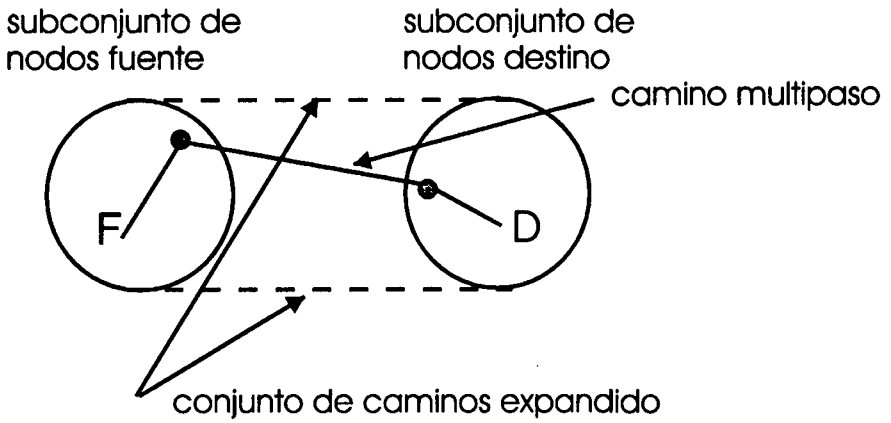


Figura 4-17 Concepto de expansión de caminos en DRB

La técnica de enviar el mensaje a nodos intermedios es similar a la utilizada por el encaminamiento aleatorio de Valiant [129]. La gran diferencia con nuestro método es la determinación de qué nodos pueden ser destinos intermedios. En el encaminamiento aleatorio se considera cualquier nodo con idéntica probabilidad entre todos ellos, con el consiguiente aumento de latencia al escalar la red. En DRB se eligen los nodos intermedios en función del alargamiento del camino que supongan, la saturación de la red y otros factores que explicaremos más adelante.

DRB es un método con dos componentes. Un componente es estático, depende sólo de la topología de la red de interconexión, y es la definición de los posibles caminos alternativos a partir de la construcción de conjuntos de nodos intermedios a los que enviar los mensajes. El otro componente es dinámico y consiste en la selección, en

tiempo de ejecución y, dependiendo de la carga de tráfico en cada momento, de los caminos alternativos a utilizar. El primer componente establece métodos uniformes para construir los supernodos para cada topología y el segundo componente establece las políticas para seleccionar las características de los subconjuntos de nodos (tamaño y forma) acordes con la carga de mensajes existente en cada instante en la red de interconexión. Ambas componentes se gestionan dinámicamente mediante el encaminador DRB, descrito en una sección posterior, que implementa el algoritmo DRB.

El método de distribución de mensajes entre conjuntos de caminos tiene un doble efecto sobre los mensajes que se envían por los canales:

- ✓ Los nuevos caminos alternativos estarán menos cargados que el camino original, el cual también disminuye su carga, y la suma del ancho de banda utilizado por el canal, al utilizar múltiples caminos o "carriles", puede igualar el ancho de banda del camino original simple sin contención.

- ✓ Los mensajes pueden ser enviados concurrentemente por los diferentes carriles del camino múltiple y estar usando, consecuentemente, un ancho de banda mayor todavía. Se produce una reducción de la latencia efectiva por efecto del solapamiento.

Aunque DRB mantiene algunos objetivos con el encaminamiento aleatorio, la gran diferencia es que DRB no solo pretende mantener un alto "*throughput*", sino que también intenta mantener una latencia de los mensajes baja.

A continuación, establecemos una serie de definiciones que corresponden a la primera componente de DRB, la definición de los supernodos.

4.7.1.1 Definiciones previas:

- ✓ Una *red de interconexión I* se define como un grafo dirigido *I=(N, E),* donde *N* es un conjunto compuesto por *MaxN* nodos definido como $N = \bigcup_{i=0}^{MaxN} N_i$ y *E* un conjunto de arcos conectando pares de nodos. Generalmente, cada nodo está compuesto por un encaminador y está conectado a otros nodos por medio de enlaces, representados por los arcos del grafo formando una cierta topología. La topología puede ser regular, con una serie de parámetros constantes definidos, o irregular dependiendo de la red en cuestión. La red asimismo tendrá definidos un algoritmo de encaminamiento estático mínimo, una técnica de control del flujo ("*wormhole*", "*cut-through*" o "*store-and-forward*"), etc. Por ejemplo, para "*n-*

cubos k-arios", n es la dimensión y k el tamaño de cada dimensión, el encaminamiento es "*Dimension Order Routing* (DOR)".

- ✓ Si dos nodos N_i y N_j están directamente conectados por un enlace, entonces, se dice que N_i y N_j son *nodos adyacentes.*

- ✓ Se define la *Distancia* (N_i, N_j) como el mínimo número de enlaces que deben ser atravesados para viajar desde N_i hasta N_j siguiendo el grafo *I.*

- ✓ Un *Camino* $P(N_i, N_j)$ entre dos nodos N_i y N_j es el conjunto formado por los nodos atravesados seleccionados al ir de N_i a N_j con arreglo al encaminamiento mínimo estático definido para la red de interconexión. N_i es el *nodo fuente* y N_j es el *nodo destino* del camino

- ✓ Longitud de un camino *P*, *Long(P)* es el número de enlaces entre N_i y N_j siguiendo el camino *P*. En el caso de encaminamiento estático mínimo:

$$Long(P(N_i, N_j)) = Distancia(N_i, N_j)$$

Ecuación 4-1 Longitud de un camino mínimo

4.7.1.2 Definición 1: Supernodo

Un *Supernodo S(tipo, tamaño,* N_0^S $) = \bigcup_{i=0}^{l} N_i^S$, se define como una región de la red de interconexión formada por el conjunto de l nodos adyacentes N_i^S alrededor de un nodo "central" N_0^S, que cumplen las siguientes condiciones:

- ✓ N_i^S, $\forall i$ cumple una propiedad dada en el "*tipo*" especificado

- ✓ Distancia (N_i^S, N_0^S)<=tamaño.

La Figura 4-18 muestra el concepto de Supernodo. El nodo central forma parte del Supernodo. Como casos particulares, cualquier nodo individual (l=0) y la red de interconexión entera (l=*MaxN*) son Supernodos. Un Supernodo que contiene sólo un único nodo se dice que está en forma **canónica menor**; si un Supernodo contiene todos los nodos de la red se dice que está en forma **canónica mayor**.

Los Supernodos así definidos son, como se mencionó anteriormente, los conjuntos de nodos candidatos a ser elegidos como nodos intermedios para enviar los mensajes

mediante DRB. Se debe hacer notar que, en un caso general, un mismo nodo puede pertenecer a más de un Supernodo.

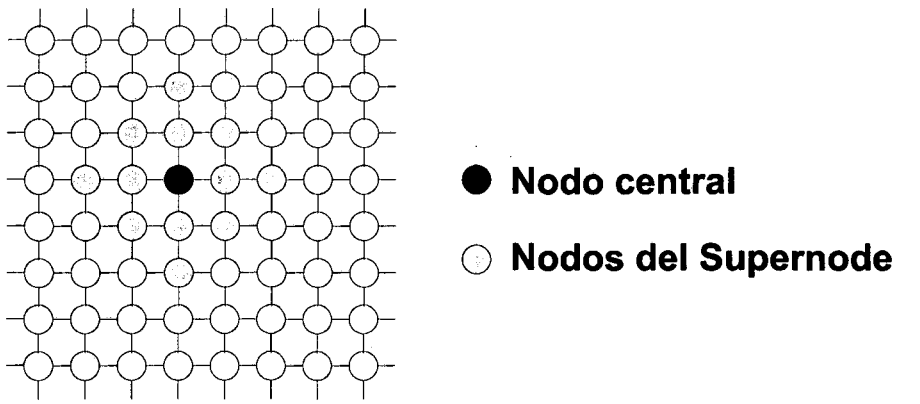


Figura 4-18 Concepto de Supernodo

Dado un cierto Supernodo, hay una serie de características que se pueden definir y que dependen del *tipo* y el *tamaño*, como son:

- ✓ Forma topológica, es decir, cómo están interconectados los nodos del Supernodo.
- ✓ Número de nodos que componen el Supernodo
- ✓ Grado del Supernodo, definido como el número de enlaces de los nodos del Supernodo N_i^S que no están conectados a otros nodos N_i^S que también pertenezcan al Supernodo, es decir, el número de enlaces conectados "al exterior" del Supernodo
- ✓ Grado Central definido como el número de nodos del Supernodo conectados al nodo central N_0^S.

De estas características se deducirán los criterios de uso de los Supernodos para enviar los mensajes.

Los parámetros *tipo* y *tamaño* determinan qué nodos se incluyen en el Supernodo. DRB define dos tipos diferentes de Supernodos adecuados a cualquier topología. El primer tipo se llama *Área de Gravedad* y el segundo *Subtopología.*

A continuación, se presentan con detalle cada uno de los dos tipos de Supernodos *Area de Gravedad* y *Subtopología.*

4.7.1.2.1 Area de Gravedad

Un Supernodo tipo *Área de Gravedad S("Área de Gravedad", tamaño,* N_0^S*)* está formado por el conjunto de nodos que se encuentran a una distancia del nodo central

N_0^S igual o menor que *tamaño*. Este tipo de Supernodo es adecuado tanto para redes regulares como irregulares. La Figura 4-19 muestra un ejemplo de un nodo tipo Area de Gravedad.

El tipo de Supernodo *Área de Gravedad* se expande desde el nodo central como un árbol, siguiendo todas las ramas posibles. Debido a su propia definición, este tipo de Supernodo maximiza el número de caminos a usar mientras minimiza el alargamiento del camino y maximiza el uso del grado del nodo. Esto es así, porque, para un tamaño de Supernodo dado, el tipo de *Área de Gravedad* selecciona *todos* los nodos más cercanos al nodo central. En redes regulares con topología bidimensional de grado 4, el número de nodos del Supernodo con respecto al tamaño d_G del mismo se puede calcular mediante la siguiente fórmula, donde el primer término corresponde al nodo central y cada término del sumatorio corresponde a los nodos que se encuentran a una cierta distancia del nodo central:

$$numero\ de\ nodos = 1 + \sum_{i=1}^{i=d_G} 4 * i$$

Ecuación 4-2 Número de nodos del Supernodo *Área de Gravedad* para redes de grado 4

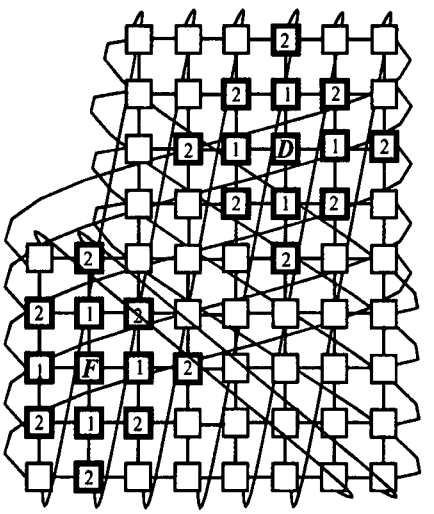


Figura 4-19 Supernodos tipo Área de Gravedad de tamaño 2 para una red "midimew"

4.7.1.2.2 Subtopología

Un Supernodo *S("Subtopología", tamaño, N_0^S)* de tipo Subtopología es un conjunto de nodos el cual tiene la misma forma topológica total o parcial que la red de interconexión en la que se define pero su *dimensión* y/o *tamaño* se reducen respecto los

de la red. Esta reducción puede hacerse, por ejemplo, por una potencia de 2 exacta. El Supernodo puede ser visto, entonces, como una proyección topológica de la red de interconexión.

Este tipo de Supernodo puede aplicarse a redes regulares, con una topología estructurada, una dimensión y un tamaño determinados. En una red que sea un n-cubo k-ario, cualquier m-cubo j-ario con j<k y m<n puede ser un Supernodo de ella. Por ejemplo, para una red con topología 2-cubo 256k-ario, es decir, un toro, o una red con topología "*midimew*", una fila o columna de longitud 256 pueden ser Supernodos. La reducción en tamaño de la longitud de la fila o la columna se puede hacer dividiendo secuencialmente por 2, dando filas o columnas de tamaños 128, 64, 32, 16, 8, 4, y 2, respectivamente. Para un 16-cubo k-ario pueden ser Supernodos 8-cubo k-ario, 4-cubo k-ario, 2-cubo k-ario. La Figura 4-20 muestra ejemplos de Supernodos tipo Subtopología.

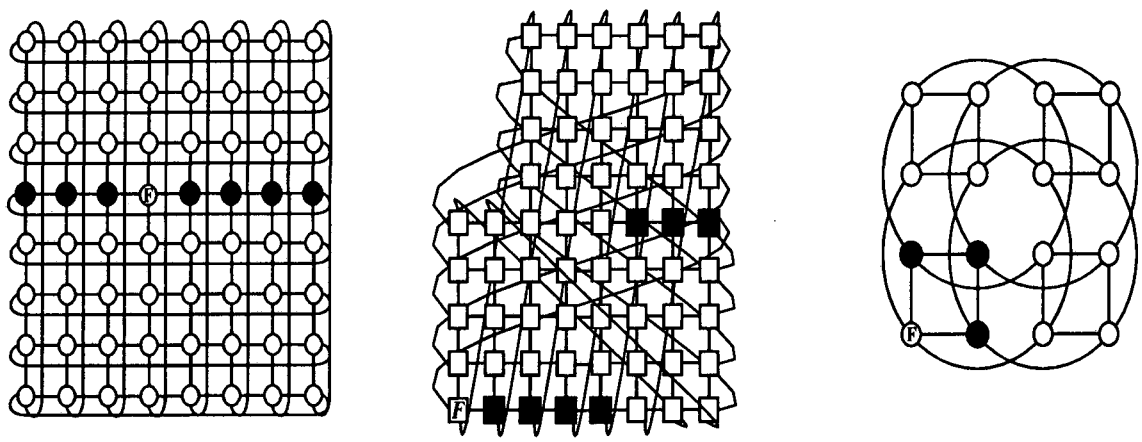


Figura 4-20 Supernodos tipo Subtopología

4.7.1.3 Definición 2: Camino Multipaso ("MultiStepPath": MSP)

Un camino multipaso "*MultiStepPath*" $MSP(SOrigen, N_i^{SOrigen}, N_j^{SDest}, SDest)$ es el camino generado entre dos Supernodos, *SOrigen* y *SDest*, de la siguiente manera (donde el símbolo * significa concatenación y *P1*, *P2* y *P3* son caminos simples):

$$MSP = \prod (N_0^{SOrigen}, N_i^{SOrigen}, N_j^{SDest}, N_0^{SDest}) =$$
$$= P1(N_0^{SOrigen}, N_i^{SOrigen}) * P2(N_i^{SOrigen}, N_j^{SDest}) * P3(N_j^{SDest}, N_0^{SDest})$$

Ecuación 4-3 Camino multipaso

Este camino multipaso está compuesto de los siguientes pasos:

- ✓ Paso 1: Desde el nodo central $N_0^{SOrigen}$ del Supernodo *SOrigen* al nodo $N_i^{SOrigen}$ perteneciente al Supernodo *Sorigen.*
- ✓ Paso 2: Desde el nodo $N_i^{SOrigen}$ al nodo N_j^{SDest} perteneciente al Supernodo *SDest.*
- ✓ Paso 3: Desde el nodo N_j^{SDest} al nodo central N_0^{SDest} del Supernodo *SDest.*

La Figura 4-21 muestra un ejemplo de un camino multipaso entre dos nodos origen y fuente y con dos nodos intermedios.

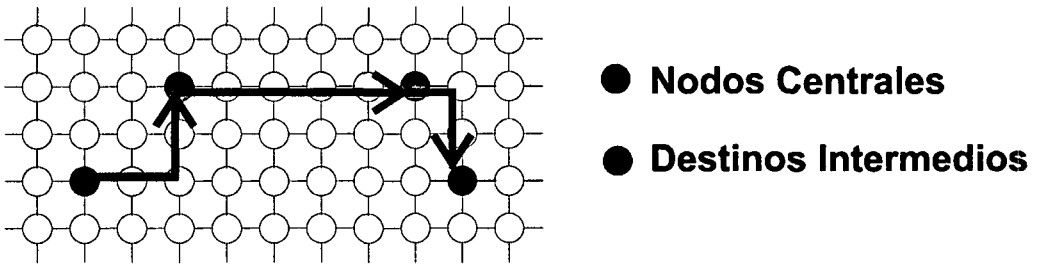


Figura 4-21 Camino multipaso

Los pasos 1 y/o 3 pueden ser nulos si *SOrigen* y/o *SDest* están en forma canónica menor, *Sorigen* = $\{N_0^{SOrigen}\}$, y/o *Sdest* = $\{N_0^{SDest}\}$ En el caso de que ambos Supernodos, *SOrigen* y *SDest*, estén en forma canónica menor, entonces, se dice que el camino multipaso está en **forma canónica**, y es igual al camino entre $N_0^{SOrigen}$ y N_0^{SDest} siguiendo encaminamiento estático mínimo.

Se define la **longitud de un camino multipaso** *long(MSP)* como la suma de las longitudes determinadas siguiendo encaminamiento estático de cada uno de los pasos individuales que lo componen.

$$Long(MSP)=Long(P1(N_0^{SOrigen},N_i^{SOrigen}))+Long(P2(N_i^{SOrigen},N_j^{SDest}))+ Long(P3(N_j^{SDest},N_0^{SDest}))$$

Ecuación 4-4 Longitud de un camino multipaso

A partir de esta definición, se puede ver que algunos de los caminos multipaso entre $N_0^{SOrigen}$ y N_0^{SDest} pueden ser de distancia no mínima. Esta longitud nos dará una medida del tiempo mínimo de transmisión de los mensajes a través de camino multipaso.

Definimos **latencia del camino multipaso** *latencia(MSP)* como la suma del tiempo de transmisión, que es la *long(MSP)*Te* (*Te* es el tiempo de transmitir por un enlace), más el tiempo de espera gastado por el mensaje para viajar desde $N_0^{SOrigen}$ hasta

N_0^{SDest} en las colas de los encaminadores debido a la contención entre mensajes. Es, por tanto, el tiempo que se tarda en enviar un mensaje por el camino multipaso.

$$Latencia(MSP) = Tiempo\ de\ Transmisión + \sum_{\forall nodos \in MSP} Retardo de Encolamiento\ (Nodo)$$

Ecuación 4-5 Latencia de un camino multipaso

Definimos **Ancho de Banda del camino multipaso** *AnchodeBanda(MSP)* como el inverso de la latencia. Esta magnitud informa de la cantidad de mensajes que pueden pasar por el camino multipaso por unidad de tiempo.

$$AnchodeBanda(MSP) = Latencia(MSP)^{-1}$$

Ecuación 4-6 Ancho de Banda del camino multipaso

4.7.1.4 Definición 3: Metacamino

Un metacamino *P*(Supernodo_Origen, Supernodo_Destino)* es el conjunto de todos los caminos multipaso que se pueden generar entre los Supernodos *Supernode_Origen* y *Supernodo_Destino* mediante el producto cartesiano entre los conjuntos formados por los nodos que componen los supernodos:

$$P^* = \bigcup_{\forall i,j} MSP\ (N_0^{SOrigin}, N_i^{SOrigin}, N_j^{SDest}, N_0^{SDest}).$$

Ecuación 4-7 Metacamino

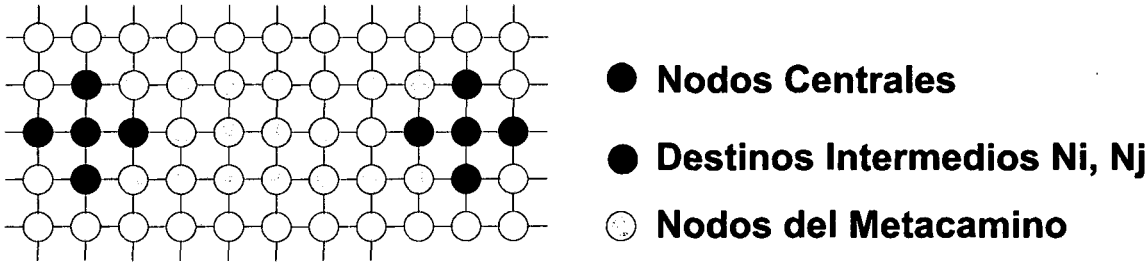


Figura 4-22 Metacamino

Supongamos que *l* es el número de nodos de *Supernodo_Origen* y *k* el número de nodos del *Supernodo_Destino*. Entonces, definimos **Anchura del Metacamino s** como el número de caminos multipaso que componen un metacamino. Estos caminos pueden compartir una cierta parte.