UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Una Nueva Familia de Topologías Híbridas para la Interconexión de Redes de Gran Escala

Tesis de Máster en Ingeniería de Computadores
Curso 2011/2012

Departamento de Informática de Sistemas y Computadores

Roberto Peñaranda Cebrián

Directores:
María Engracia Gómez Requena
Pedro Juan López Rodríguez
Crispín Gómez Requena

Julio de 2012

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Topologías para Redes de Interconexión

El tamaño de los super-computadores está creciendo, año tras año. Las máquinas que se encuentran en lo más alto de la lista del top 500 [top] de super-computadores más potentes del mundo llegan a tener cientos de miles de nodos de procesamiento. En la última lista, el número uno, supera el millón de nodos. Todos estos nodos trabajan en conjunto para solucionar problemas de gran tamaño en el menor tiempo posible. Este trabajo conjunto se hace posible gracias a una red de interconexión que permite que todos los nodos de procesamiento compartan sus datos. La red de interconexión debe permitir una comunicación eficiente entre los nodos de procesamiento, ya que influye directamente en el rendimiento global del sistema en el que esté implantado. En concreto, el tiempo de transmisión de datos a través de la red que se suma al tiempo de procesamiento, lo que hace que tenga un notable efecto en el tiempo requerido para poder ejecutar cierta aplicación.

Los principales objetivos de una red de interconexión son proporcionar comunicaciones con muy poca latencia para reducir el tiempo de ejecución de las aplicaciones, y una alta productividad para permitir tantas comunicaciones simultáneas como sea posible. Otro requisito muy importante de las redes de computadores es proporcionar una implementación simple y de bajo coste hardware.

Además de las prestaciones y el coste, otro factor importante a tener en cuenta es su capacidad para tolerar fallos. La gran cantidad de hardware que se puede encontrar en una red de interconexión en los super-computadores de altas prestaciones hace que la probabilidad de que se produzca un fallo en el sistema sea mayor. Cada componente hardware puede fallar de forma independiente, y por eso, la probabilidad de tener un fallo en el sistema crece de forma considerable con el número de elementos que componen ese sistema. Por lo tanto, la tolerancia a fallos es esencial en una red de interconexión.

Dos de los aspectos más importantes en el diseño de una red de interconexión son la topología de red y el algoritmo de encaminamiento [DT04, DYL02]. La topología define como se interconectan entre sí los nodos de procesamiento, y el algoritmo de encaminamiento determina cual va a ser el camino que recorrerá un paquete desde el nodo fuente hasta el nodo destino. La topología de una red también repercute en el coste de la red. Normalmente, las topologías utilizan estructuras regulares para

simplificar el algoritmo de encaminamiento, además de su implementación y facilitan la posibilidad de expandir la red. Entre las diferentes formas de clasificar las topologías, la más utilizada las divide en topologías directas e indirectas[DT04, DYL02].

Las topologías directas suelen adoptar una estructura ortogonal, donde los nodos están organizados en un espacio n–dimensional y cada nodo de procesamiento tiene un router asociado. Los nodos están conectados en cada dimensión en forma de anillo. Las topologías directas de 2 ó 3 dimensiones son relativamente fáciles de construir, ya que cada dimensión de la topología se implementa como una dimensión física. La implementación de topologías directas con más de tres dimensiones no solo implica el incremento de la complejidad del cableado, sino también el aumento de la longitud de los cables cuando pasamos al espacio físico de tres dimensiones. Además, el número de puertos de los routers aumenta con el número de dimensiones (se necesitan dos puertos por cada dimensión). Como consecuencia, las topologías directas no son las más recomendables para grandes computadores, ya que la limitación en el número de dimensiones hace que se incremente el número de nodos por dimensión, que a su vez hace que aumente la latencia de comunicación, perjudicando las prestaciones.

La alternativa es usar una topología indirecta. La principal diferencia, comparada con una topología directa, se basa en que no todos los routers están asociados a un nodo de procesamiento. Las topologías indirectas más comunes son las redes indirectas multietapa (Multistage Interconnection Networks o MINs) donde los routers están organizados en n etapas. Las topologías indirectas suelen proporcionar mejores prestaciones para un gran número de nodos de procesamiento que las topologías directas. Sin embargo, estos mejores resultados son debidos al uso de una gran cantidad de routers y enlaces, y su implementación física es compleja, debido al hecho de que la complejidad del cableado aumenta con el tamaño de la red, mientras que en las topologías directas crece con el número de dimensiones de la topología.

1.2. Propuesta de una Topología Híbrida

Para poder aprovechar las ventajas de las topologías directas e indirectas, este trabajo propone una nueva familia de topologías híbridas, donde se combinan las mejores características de ambas topologías. Se propone una topología n–dimensional donde los anillos que conectan los nodos en cada dimensión son reemplazados por pequeñas redes indirectas. De esta forma, conseguimos que la latencia de comunicación dentro de las dimensiones esté limitada gracias al uso de una topología indirecta. Por otro lado, el pequeño tamaño de esta topología indirecta permite tener una complejidad de cableado razonable. Con esta combinación, obtenemos una familia de topologías que proporciona unas altas prestaciones, con productividad y latencia cercanas a las obtenidas con topologías indirectas, con un coste hardware reducido. El diseño de esta topología es mucho más simple que en las topologías indirectas. La tolerancia a fallos en este tipo de topologías es tan alta o más que la que proporciona una topología directa o indirecta.

1.3. Estructura del Trabajo

El resto del trabajo está organizado de la siguiente manera. El Capítulo 2 explica de forma breve las propiedades de las topologías directas e indirectas. También describe algunos trabajos actuales que se han presentado sobre esta temática. El Capítulo 3 describe la nueva familia de topologías híbridas. En primer lugar, se presenta como se implementaría la nueva familia. Después, se especifican los algoritmos de encaminamiento que se utilizarán en esta nueva familia de topologías, tanto deterministas como adaptativos. El Capítulo 4 describe el simulador utilizado para evaluar las distintas topologías. También mostrará los parámetros que se han utilizado en el simulador para dicha evaluación. El Capítulo 5 analiza y compara las prestaciones y el coste hardware entre las diferentes topologías evaluadas, y por último, el Capítulo 6 presenta las conclusiones más relevantes del trabajo, además de enumerar las publicaciones que se han realizado y posibles trabajos futuros.

Capítulo 2

Topologías Regulares

La taxonomía más comúnmente utilizada clasifica las topologías regulares en dos grandes grupos [DYL02, DT04]:

- Topologías directas.
- Topologías indirectas.

En este capítulo se describirán con detalle los dos tipos de topologías y se mostrarán las diferencias de cada tipo. También se describirán otras topologías que han sido diseñadas en trabajos previos a este.

2.1. Topologías Directas

En las topologías directas, cada nodo tiene su propio router que lo conecta a sus nodos vecinos del sistema por medio de enlaces punto a punto. Las topologías directas normalmente adoptan una estructura ortogonal, donde los nodos están organizados en un espacio n–dimensional, de forma que atravesar un enlace equivale a un desplazamiento solo en esa dimensión. Es decir, todos los enlaces de la red están organizados en varias dimensiones de forma regular, y cada nodo tiene al menos un enlace en cada dimensión. La simetría y regularidad de estas redes simplifican en gran medida su implementación y el diseño del algoritmo de encaminamiento, ya que el movimiento de un paquete en una dimensión no modifica el número de saltos restantes en otras dimensiones para alcanzar el destino del paquete. Este tipo de topologías son conocidas como k_d–ary n_d–cubes, donde k_d es el número de nodos en cada dimensión, y n_d el número de dimensiones de la red. El número total de nodos en el sistema viene dado por $N = {k_d}^{n_d}$. Para distinguir los nodos, éstos son etiquetados con un identificador con tantas componentes como dimensiones que tenga la topología $\{p_{n_d-1}, ..., p_0\}$, y la componente asociada a cada dimensión está en el rango de 0 a $k_d - 1$ (por ejemplo, los nodos serán numerados desde $0, 0, ..., 0$ hasta $k_d - 1, k_d - 1, ..., k_d - 1$). Los identificadores de los nodos vecinos en una dimensión dada solo difieren en la componente correspondiente de esa dimensión, mientras que las otras componentes del identificador tendrán el mismo valor. En particular, dos nodos p y p' son vecinos en la dimensión d si y solo si $p_d = p'_d \pm 1$ y $p_i = p'_i$ para todas las demás dimensiones.

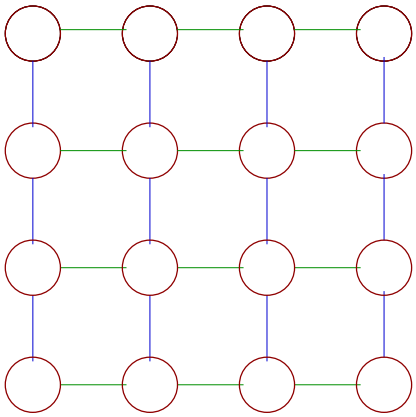

Figura 2.1: Una malla de dos dimensiones con $k_d = 4$.

Las topologías directas más utilizadas son la malla, el toro y el hipercubo. En una malla, todos los nodos de una dimensión componen un vector lineal. La Figura 2.1 presenta un ejemplo de una malla de 2 dimensiones, con 4 nodos de procesamiento por dimensión ($k_d = 4$) y un total de 16 nodos de procesamiento ($N = 4^2 = 16$). En una topología de tipo toro, todos los nodos de cada dimensión forman un anillo. El hipercubo es un caso particular de una malla, donde sólo hay dos nodos de procesamiento en cada dimensión ($k_d = 2$), lo cual fuerza que el número de dimensiones (n_d) tenga que ser mayor para ser capaz de interconectar todos los nodos de procesamiento del sistema, es decir, $n_d = log_2 N$. Las topologías directas están siendo usadas por las super-computadoras más potentes del momento, siendo el toro de 3 dimensiones la más utilizada. Por ejemplo, la actual primera (Sequoia), segunda (k Computer) y la sexta (Jaguar) super-computadora de la lista Top500 [top] usan redes basadas en toros.

Para un número dado de nodos N, las topologías directas proporcionan una mejor conectividad a medida que aumenta el número dimensiones de la red. Como el hipercubo tiene un mayor número de dimensiones, proporciona una mejor conectividad que las mallas y los toros; pero esto causa un elevado coste ya que se usa más enlaces y los routers tienen un alto grado (número de puertos en el router). La latencia está relacionada con la distancia media, medida como el número de enlaces que un paquete tiene que atravesar para llegar a su destino. En relación a esto, el diámetro de una topología mide la máxima distancia entre dos nodos de la topología, usando el camino más corto entre ellos. Es decir, el máximo número de enlaces que un paquete tiene que cruzar en la red. Por ejemplo, en una topología de tipo malla, el diámetro viene dado por $(n_d * (k_d - 1))$, ya que en el peor caso un paquete tiene que cruzar todos los nodos de cada dimensión.

Para un número constante de nodos de procesamiento ($N = k_d^{n_d}$), el diámetro aumenta a medida que disminuimos el número de dimensiones, o lo que es lo mismo, aumentamos el número de nodos de procesamiento por dimensión. El aumento de la distancia recorrida por los paquetes también aumenta la probabilidad de contención de otros paquetes que también están cruzando la red. Desde este punto de vista, puede parecer interesante maximizar el número de dimensiones para un número dado de nodos de procesamiento (N). Sin embargo, hay que considerar otros aspectos. Las topologías

directas de hasta tres dimensiones pueden ser implementadas usando enlaces relativamente cortos en nuestro espacio tridimensional, independientemente al tamaño del sistema. Sin embargo, la implementación de una topología con más de tres dimensiones implica un incremento de la longitud de los enlaces, ya que tenemos que asignar todas las dimensiones al espacio físico tridimensional. Además, requiere el uso de routers con un alto numero de puertos (dos puertos bidireccionales por cada dimensión). Debido a esta limitación, las topologías directas no son las más recomendables para grandes máquinas, ya que requieren usar un gran número de nodos de procesamiento por dimensión (k_d), lo cual incrementa el diámetro de la red y la probabilidad de contención, incrementando así la latencia de las comunicaciones e impactando de forma negativa a las prestaciones de la red.

2.2. Topologías Indirectas

La alternativa a las topologías directas son las topologías indirectas. En estas topologías, los nodos de procesamiento no tienen la capacidad de encaminar, ya que los routers están separados del nodo. Los routers pasan a ser dispositivos independientes conocidos como switches. Además, a diferencia de las topologías directas, todos los switches no tienen porque estar conectados a un nodo de procesamiento. De hecho, la mayoría de los switches están conectados a otros switches pero no están conectados a nodos de procesamiento. Estos nodos de procesamiento se conectan a los switches mediante tarjetas de interfaz de red (NICs).

Las redes indirectas más comunes son las redes de interconexión multietapa (MINs). En las MINs, los switches están organizados por etapas. Los nodos de procesamiento están conectados con la primera etapa, y las demás etapas están conectadas entre ellas usando cierto patrón de conexión que proporcione una conectividad total. Existen dos tipos de MINs. Las MINs unidireccionales (UMINs) usan switches y enlaces unidireccionales, por lo tanto los paquetes atraviesan la red solo en una dirección. En este caso, los nodos de procesamiento también están conectados a la última etapa de la red y solo hay un camino entre cada par de nodos fuente–destino (usando el menor número de etapas). Las MINs bidireccionales (BMINs) usan switches y enlaces bidireccionales. En este tipo de redes, los paquetes atraviesan la red en dos fases, una fase de subida y otra de bajada. Las BMINs proporcionan varios caminos entre cada par de nodos fuente–destino.

En las MINs, los patrones de conexión entre las etapas están basadas en permutaciones de los identificadores de los nodos. Dependiendo del patrón de conexión utilizado entre las etapas adyacentes se pueden proponer distintas MINs. La MIN más utilizada en sistemas comerciales es la topología fat–tree [Lei85], que se trata de una BMIN. Muchos de los vendedores de redes de interconexión de altas prestaciones, como Mellanox (fabricante de la tecnología Infiniband) [IBA], Myricom (fabricante de Myrinet) [myr] o Quadrics (fabricante de QsNet) [Qua] recomiendan usar la topología fat–tree y proporcionan switches específicos para construir este tipo de topología. Además, se utilizan en algunas de las super-computadoras más potentes del momento, como por ejemplo en la supercomputadora Tianhe 1A, la número cinco de la lista Top500 [top].

La implementación más extendida de la topología fat–tree es la k_i–ary n_i–tree. k_i es el número de enlaces que conectan un switch a la etapa siguiente o anterior, y n_i es el

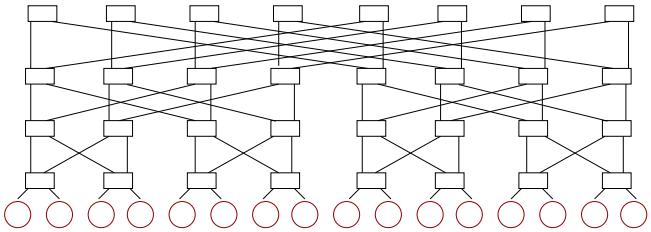

Figura 2.2: Un fat–tree con 4 etapas , con $k_i = 2$

número de etapas de la red. Por lo tanto, cada switch tiene $2k_i$ puertos bidireccionales, de los cuales, los primeros k_i enlaces conectan con la etapa anterior, y los demás con la etapa siguiente. De esta manera, los enlaces de bajada están numerados de 0 a $k_i - 1$, y los de subida de k_i a $2k_i - 1$.

En una MIN, para interconectar N nodos de procesamiento se necesitan como mínimo $log_{k_i}N$ etapas, y cada una de ellas tiene N/k_i switches, lo que hace un total de $(N/k_i)log_{k_i}N$ switches y $N = {k_i}^{n_i}$ nodos de procesamiento. En la Figura 2.2 se muestra un fat–tree con 4 etapas ($n_i = 4$), con $k_i = 2$ y un total de 16 nodos de procesamiento ($N = 16$).

Para una red con n nodos de procesamiento, si usamos switches de un alto grado (un elevado número de puertos), se necesitarán pocos switches, pero cada switch será más complejo. Sin embargo, si se usan switches de bajo grado, se necesitarán más switches, pero mucho más simples. El número de etapas también aumenta cuando utilizamos switches de bajo grado. En los fat–trees, el diámetro de la red depende solo del número de etapas y viene dado por $2n_i$, o lo que es lo mismo por $2log_{k_i}N$, en el peor de los casos donde un paquete debe atravesar todas las etapas en la fase de subida y en la de bajada. En el caso de las UMINs, la distancia que recorre un paquete siempre es n_i. En otras palabras, para un número fijo de $N = {k_i}^{n_i}$ nodos de procesamiento, cuando se eleva el número de etapas, k_i disminuye pero la distancia que deben recorrer los paquetes aumenta. Las redes indirectas proporcionan una mayor escalabilidad que las redes directas, porque proporcionan diámetros mas pequeños y por lo tanto menores latencias en redes de gran tamaño. No obstante, tienen un alto coste, porque requieren una gran cantidad de switches y enlaces, y su implementación física es compleja, ya que la complejidad del cableado de la red crece con su tamaño, no como en las redes directas.

2.3. Otras Topologías

Existen otros trabajos previos que proponen topologías diferentes a las presentadas en la sección anterior, pero éstas nunca o pocas veces han sido usadas en productos comerciales o super-computadoras. Este es el caso de la topología WK–recursive que fue propuesta en [DVS87] para redes de interconexión, y más reciente para redes en chip [RKHSA06]. Esta topología tiene dificultades para garantizar un algoritmo de encaminamiento libre de bloqueos.

Otra topología muy popular en trabajos de investigación es la flattened–butterfly [KDA07], la cual es obtenida de la combinación de los routers de cada fila de una MIN

butterfly convencional, obteniendo así una red directa n–dimensional (sus parámetros serán referidos en este trabajo como k_f y n_f) donde los nodos de cada dimensión no están conectados con un anillo como en el toro, o mediante una pequeña topología indirecta como en la propuesta presentada en este trabajo, en su lugar están totalmente conectados. De esta combinación sale resultante una topología muy similar a un hipercubo generalizado [BA84], pero conectando varios nodos de procesamiento (parámetro p) a un mismo switch. Esta topología, como el hipercubo generalizado, tiene un alto coste, especialmente en máquinas de gran tamaño, máquinas a las que va orientada la familia propuesta en este trabajo. En el capítulo 5, esta topología se compara con la propuesta de este trabajo.

Por otro lado, como se hace en este trabajo, otros proponen la combinación de varias topologías. Muchas han sido propuestas para redes en chip y por lo tanto, tienen diferentes objetivos. Por ejemplo en [MKA07], cada nodo está conectado a dos redes de árbol diferentes en un entorno en chip para superar las prestaciones que proporciona una topología de árbol. Esta propuesta no es recomendable para máquinas de gran tamaño, debido a la complejidad del diseño del cableado y las pocas prestaciones que se obtienen.

También hay propuestas para redes en chip basadas en topologías jerárquicas. En este tipo de topologías hay un conjunto de pequeñas redes locales creando subredes de nodos, las cuales se conectan mediante una red global. Este no es el caso de la propuesta de este trabajo, ya que en este caso los nodos de procesamiento están conectados a ambas topologías. Una topología semejante a la propuesta es la malla de árboles (MoT), la cual se describe en [Lei92] y más tarde usada para redes en chip [BQV09]. Está basada en una topología n–dimensional donde los nodos de una dimensión dada están conectados mediante un árbol. En realidad, este tipo de topología sería un caso particular dentro de la familia de topologías que se propone en este trabajo, ya que usa un árbol, como topología indirecta para conectar los nodos de cada dimensión dada.

En [DEM+09], los autores proponen usar como red local un simple bus y una malla como red global. Como cabe esperar, esta no es una topología apropiada para máquinas de gran tamaño debido a las pocas prestaciones que proporcionan los buses y las mallas. Los autores de [GD06] proponen una herramienta para seleccionar la topología más apropiada para un diseño de red dado. La herramienta explora los diseños de la red y utiliza redes híbridas Clos-toros. En este caso, los diseños explorados son topologías jerárquicas, a diferencia de la propuesta de este trabajo, donde las redes locales son redes Clos y éstas están conectadas mediante una red toro global.

Otra topología jerárquica es la DragonFly [KDSA08], la cual proporciona una topología que está basada en agrupar routers como un router virtual para incrementar el radix efectivo de la red. Esta topología usa dos redes diferentes, una intragrupal y otra intergrupal. Es aconsejable para esta red utilizar un algoritmo de encaminamiento adaptativo, con rutas no mínimas, para equilibrar la carga a través de los canales globales. Estos canales globales, los cuales unen los diferentes grupos, son enlaces largos, con lo que se espera una gran latencia en este tipo de redes. Se espera tener grandes latencias y poca productividad en los diseños jerárquicos, porque se deben atravesar varios tipos de redes para muchos pares de fuente-destino.

Por último, en [YFJ+01] se combinan varias redes toro para crear una nueva topología. La propuesta se centra en grandes super-computadores, pero su utilización

está limitada a 2^{16} nodos y el diseño de su cableado es muy complejo para máquinas de gran tamaño. La propuesta se basa en toros de dos dimensiones y en proporcionar conexiones por medio de enlaces diagonales tantas veces como sea necesario. Esta propuesta no mejora el número de saltos en una sola dimensión, ya que no añade nuevos enlaces para conectar nodos de la misma dimensión. En su lugar, reduce el número de saltos cuando se atraviesan varias dimensiones. Además, esta topología tiene el mismo problema que la topología WK–recursive, que resulta muy complicado diseñar un algoritmo de encaminamiento libre de bloqueo.

Capítulo 3

Una Nueva Familia de Topologías Híbridas

En este capítulo se presenta la nueva familia de topologías híbridas que permite conectar de forma eficiente un gran número de nodos de procesamiento. Cuando se dice de forma eficiente, nos referimos a que proporciona una buena relación coste/prestaciones. Se propone la utilización de topologías híbridas, que se basan en la combinación de redes directas n–dimensionales con pequeñas redes indirectas. La idea principal es combinar las ventajas de las topologías directas e indirectas para obtener una familia de topologías que permita conectar un gran número de nodos de procesamiento, mientras que se proporciona una latencia baja, una productividad alta y un alto nivel de tolerancia a fallos con un coste más bajo que las redes indirectas. En particular, se propone usar dos tipos diferentes de routers en la red. Primero, unos switches de grado bajo para mover los paquetes entre las dimensiones. A estos switches se les llamarán routers, y tendrán por lo menos un nodo de procesamiento conectado a ellos y tantos puertos como dimensiones. Estos routers pueden ser parte de la red de interconexión, pero otra opción es tomar la ventaja de las tarjetas de interfaz de red con capacidad de encaminamiento como las tarjetas ATOLL [LFNB08], de forma que los nodos de procesamiento, además de inyectar y recibir mensajes de la red, también permitan recibir paquetes de los cuales no es destino y encaminar dichos paquetes para que alcancen sus destinos sin extraer los paquetes de la red. En otras palabras, dentro de la NIC de los nodos de procesamiento hay un router, como en las redes directas. Además de estos elementos de encaminamiento, también se usan otros switches para implementar las pequeñas redes indirectas que interconectan los nodos de cada dimensión.

Por otro lado, también se presentarán los algoritmos de encaminamiento que se utilizarán en esta familia de topologías híbridas. Se describirán dos diferentes algoritmos de encaminamiento, determinista y adaptativo.

3.1. Descripción de la Familia

La nueva topología propuesta organiza los nodos de procesamiento en n dimensiones, como una topología directa, pero cada nodo de procesamiento de una dimensión dada no está exclusivamente conectado a sus nodos adyacentes en cada dimensión, como en el caso de las topologías malla y toro. En este caso, todos los nodos de una

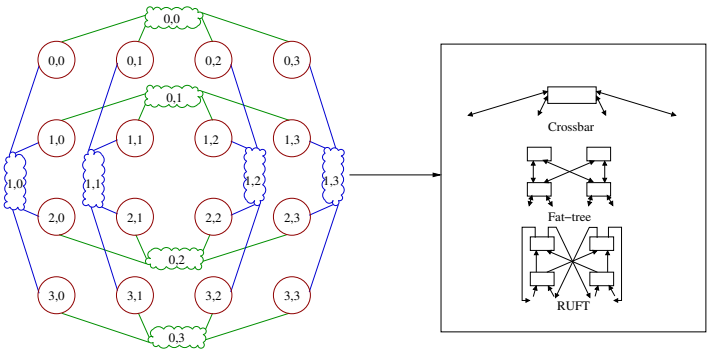

Figura 3.1: Un ejemplo de la topología propuesta con $n_n = 2$, $k_n = 4$ y $d_n = 4$.

dimensión dada están directamente conectadas por medio de una red indirecta. Esta red indirecta puede ser un simple crossbar si el número de nodos por dimensión es pequeño. Si el número de nodos por dimensión es mayor al número de puertos que dispone el crossbar, se tendrá que utilizar una red indirecta, es decir, una MIN. Por lo tanto, la familia propuesta se define con tres parámetros. Dos de ellos son heredados de las redes directas, y son el número de dimensiones n_n y el número de nodos por dimensión k_n. El número de nodos de procesamiento que pueden ser interconectados en estas topologías viene dado por $N = k_n^{n_n}$. Además de estos dos parámetros, tenemos uno más, el número de etapas de la subred indirecta, referenciado como m_n. El número de etapas necesarias para interconectar los k_n nodos de procesamiento de una dimensión dada depende del número de puertos de los switches que implementa la subred indirecta, que será referenciado como d_n. La relación entre k_n y d_n define la forma de interconectar los nodos en cada dimensión. Si $k_n \leq d_n$, un simple crossbar puede ser capaz de interconectar todos los nodos de la dimensión, y m_n tendrá un valor de 1. Por otro lado, si $k_n > d_n$, se necesitará una MIN para interconectar los nodos de una dimensión dada, y el número de etapas vendrá dado por $log_{\frac{d_n}{2}} k_n$. En este trabajo se referirá a la nueva familia de topologías como k_n–ary n_n–direct m_n–indirect, donde k_n será el número de nodos por dimensión, n_n el número de dimensiones, y m_n el número de etapas de las subredes indirectas.

En este trabajo se han considerado dos MINs diferentes para conectar los nodos de una dimensión dada. La primera es la topología fat–tree, una MIN bidireccional usada en productos comerciales. La segunda MIN considerada es RUFT [GGG+08], una MIN unidireccional derivada de un fat–tree usando un algoritmo de encaminamiento determinista con equilibrado de carga [GGG+07], la cual requiere menos recursos hardware reduciendo un poco las prestaciones.

La Figura 3.1 muestra un ejemplo de la nueva topología con 2 dimensiones ($n_n = 2$) y 4 nodos de procesamiento por dimensión ($k_n = 4$), con un total de 16 nodos de procesamiento. Como se puede observar, los nodos se conectan a cada una de las dimensiones con un enlace diferente, a diferencia de las mallas y toros, que utilizan dos enlaces por dimensión. En este ejemplo, como tenemos 4 nodos por dimensión y el grado de los switches es 4 ($d_n = 4$), con un único crossbar es suficiente para interconectar los 4 nodos de cada dimensión. Por lo tanto, la red implementa una 4–ary 2–direct 1–

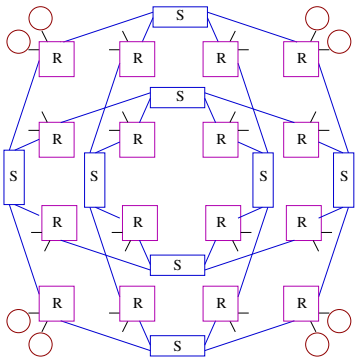

Figura 3.2: Un ejemplo de la topología propuesta con $n_n = 2$, $k_n = 4$, $d_n = 4$ y $p_n = 2$.

indirect. Sin embargo, si tenemos un mayor número de nodos por dimensión, como por ejemplo 8, y switches del mismo tamaño, tendríamos que utilizar una MIN, por ejemplo un fat–tree o un RUFT con 3 etapas y 12 switches con 4 puertos (bidireccionales para fat–tree y unidireccionales para RUFT), implementando en ese caso una 8–ary 2–direct 3–indirect.

Es posible conectar varios nodos de procesamiento en el mismo router (lo que se conoce en la bibliografía como concentración). Esto se ve en la Figura 3.2, donde los nodos y los routers se muestran separados. En este caso, dos nodos de procesamiento están conectados a cada router. Solo se muestran los nodos de las esquinas por motivos de claridad. Esta extensión introduce un nuevo parámetro en la topología, p_n, que representa el número de nodos de procesamiento conectados a cada router. En este caso el número de nodos de procesamiento totales se calcula como $N = p_n k_n^{n_n}$. En la Figura 3.2, se pueden distinguir dos elementos de encaminamiento de diferente tamaño. Los switches (etiquetados con una "S") solo se conectan con otros elementos de encaminamiento y tienen un grado de k_n, mientras que los routers (etiquetados con "R") se utilizan para conectar los nodos de procesamiento a la red. Estos routers están conectados por un lado con los nodos de procesamiento y por otro lado con los switches y su grado es $n_n + p_n$.

Como se puede ver en la Figura 3.2, si se usa un crossbar como subred indirecta, los switches permiten a los paquetes cambiar su posición en una dimensión dada atravesando solo dos enlaces, mientras que los routers permiten cambiar de dimensión. Por lo tanto, el diámetro de la nueva topología es $2n_n$. Si se usa una MIN como subred indirecta, el diámetro de la red será el diámetro de la MIN usada multiplicado por n_n.

La tabla 3.1 resume los parámetros que definen la familia híbrida de topologías propuesta en este trabajo. Cabe notar que solo el parámetro d_n o m_n es necesario, ya que uno puede ser derivado a partir del otro y k_n. En este trabajo se utilizará m_n. Además, también se muestran los parámetros que definen las tradicionales redes directas e indirectas.

Las topologías híbridas propuestas pueden sacar ventaja del alto número de puertos de los switches actuales. Estos pueden tener hasta 64 puertos [SAKD06], aunque en recientes trabajos se han propuesto switches de hasta 144 puertos [BDJ+11]. Con este tipo de switches, una pequeña MIN con solo 2 ó 3 etapas o con solo un simple switch es

Topología	Parámetros	
Malla o Toro	n_d	número de dimensiones
	k_d	número de nodos por dimensión
Fat–tree	n_i	número de etapas
	k_i	aridad del switch
k_n–ary n_n–direct m_n–indirect	n_n	número de dimensiones
	k_n	número de nodos por dimensión
	m_n	número de etapas de la subred indirecta
	d_n	grado del switch
	p_n	número nodos por router

Tabla 3.1: Parámetros de los diferentes tipos de topologías analizadas en este trabajo.

posible conectar todos los nodos de cada dimensión en la mayoría de los casos. Como la MIN que se utilice no será de gran tamaño, no introducirá una latencia muy alta y será de fácil implementación.

Las topologías híbridas presentan varias ventajas principales. La primera, que permite reducir el diámetro comparado con las topologías directas. Si se utilizan crossbars, el diámetro de la nueva topología es $2n_n$, mientras que en una malla es $n_d(k_d - 1)$. Con esto se espera que aumenten las prestaciones de la red, ya que se incrementa la productividad y disminuye la latencia. Además, el número de switches y enlaces se reduce si lo comparamos con una topología indirecta, como se mostrará en el capítulo 5. Con todo esto, se espera que la nueva familia de topologías pueda reducir los costes de la red. Finalmente, como el número de caminos alternativos en la topología propuesta es muy alto proporciona un alto nivel de tolerancia a fallos, mayor que en las topologías directas y parecido al de las indirectas (ver Capítulo 5).

3.2. Algoritmo de Encaminamiento para la Nueva Familia de Topologías

En esta sección se describen los algoritmos de encaminamiento propuestos para la nueva familia de topologías. Primero se describirán los algoritmos propuestos para las topologías k_n–ary n_n–direct 1–indirect, es decir, las que utilizan un crossbar como subred indirecta. Después, se describirán los algoritmos para el caso general, es decir, para las topologías k_n–ary n_n–direct m_n–indirect, usando fat–tree o RUFT como subredes indirectas. En ambos casos se proporcionarán algoritmos de encaminamiento adaptativos y deterministas.

En primer lugar se explicará como se han etiquetado los routers y switches en la nueva familia de topologías híbridas con el fin de poder identificar mejor dichos recursos. Cada router será etiquetado como en una malla o en un toro, con el conjunto de componentes o coordenadas (tantas como dimensiones tenga la red) $\langle r_{n_n-1}, r_{n_n-2}, ..., r_1, r_0 \rangle$. Cada coordenada representa la posición de cada router en cada una de las dimensiones. Por otro lado, los switches están etiquetados mediante 2 tuplas *[d,p]*, donde *d* corresponde a la dimensión en la cual está localizado el switch, y *p* a la posición que tiene el

switch en la dimensión en la que se encuentra (ver Figura 3.1).

Cabe destacar que los routers no pertenecen a ninguna dimensión, ya que están conectados a todas las dimensiones, y los paquetes cambian de dimensión a través de ellos. Por el contrario, los switches no permiten cambiar de dimensión, sino moverse por una dimensión en la que se encuentran.

3.2.1. Encaminamiento en las Topologías k_n–ary n_n–direct 1–indirect

Encaminamiento Determinista

El algoritmo de encaminamiento determinista para las topologías k_n–ary n_n–direct 1–indirect, que será referenciado como Hybrid–DOR, es una variante del algoritmo de encaminamiento determinista DOR para mallas adaptado a la topología k_n–ary n_n–direct 1–indirect. En DOR, los paquetes son encaminados a través de las diferentes dimensiones siguiendo un orden establecido hasta que se alcanza el nodo destino. En cada dimensión, los paquetes van atravesando varios routers hasta que se ha recorrido toda la distancia en esa dimensión para alcanzar el destino. Por otro lado, como cada router tiene dos enlaces por dimensión, los paquetes tienen que ser enviados por el sentido correcto para garantizar el camino mínimo.

En Hybrid–DOR, las dimensiones de la red también son cruzadas en un orden establecido para garantizar un algoritmo libre de bloqueos. Sin embargo, como hay un switch que permite alcanzar directamente el siguiente router de la siguiente dimensión, los paquetes no tienen que hacer varios saltos en cada dimensión. Por lo tanto, en Hybrid–DOR, los routers envían los paquetes directamente por el único enlace de la dimensión que se tiene que atravesar, y este enlace está conectado al correspondiente crossbar. Cabe destacar que en las topologías k_n–ary n_n–direct 1–indirect, no se requiere elegir el sentido en cada dimensión, ya que solo se dispone de un enlace que conecta con el crossbar. Respecto a cómo encaminan los paquetes los crossbars, éstos envían los paquetes a través del enlace indicado por la componente del destino en la correspondiente dimensión, necesitando solo un salto para alcanzar el siguiente router.

A continuación, se mostrará el pseudo-código del algoritmo Hybrid–DOR para los routers y los crossbars. Se asume que el número de dimensiones de la topología es n_n, el destino y el router actual vienen dadas por $\langle x_{n_n-1}, ..., x_{d+1}, x_d, x_{d-1}, ..., x_1, x_0\rangle$, y $\langle r_{n_n-1}, ..., r_{d+1}, r_d, r_{d-1}, ..., r_1, r_0\rangle$, respectivamente. En el caso de los crossbars, las coordenadas del actual crossbar son *[d,p]* (el *p-ésimo* switch de la dimensión d). El enlace elegido para enviar el paquete sera devuelto en *link*.

Routers:

```
i = 0;
Done = False
while (i < n_n) ∧ (!Done) do
  if x_i ! = r_i then
    Done = True
    link = i
  end if
  i = i + 1
end while
```

Crossbars:

```
link = x_d
```

Como se puede apreciar, los routers seleccionan la siguiente dimensión (i) para encaminar el paquete, el cual es también el enlace del router que se debe usar, ya que hay un enlace por dimensión. Y los crossbars seleccionan el enlace por la coordenada del destino de la correspondiente dimensión.

Encaminamiento Adaptativo

Tal y como se hace en las mallas, el algoritmo de encaminamiento adaptativo para las topologías k_n–ary n_n–direct 1–indirect se consigue permitiendo cruzar las dimensiones de la red en cualquier orden. Por lo tanto, los routers, a la hora de seleccionar la siguiente dimensión para enviar el paquete, comprueban por que dimensiones pueden acercar el paquete a su destino y después, la función de selección selecciona una de ellas siguiendo algún criterio. El algoritmo de encaminamiento para los crossbars es el mismo que el usado en el encaminamiento determinista

El correspondiente pseudo–código se muestra a continuación. Las dimensiones que aproximan el paquete a su destino son almacenadas en el conjunto *Links*, y *Selection* lleva a cabo la función de selección:

Routers:
while $(i < n_n)$ **do**
  if $x_i! = r_i$ **then**
    $Links = LinksUi$
  end if
  $i = i + 1$
end while
$link = Selection(Links)$

Como en las mallas, cuando se usa el encaminamiento adaptativo en las topologías k_n–ary n_n–direct 1–indirect, se necesitan al menos dos canales virtuales en los routers para evitar bloqueos [DYL02]. Uno de los canales virtuales se usará para proporcionar un canal de escape para evitar los bloqueos, y los demás canales virtuales se utilizaran de forma adaptativa. Los canales virtuales adaptativos se seleccionaran siguiendo el algoritmo propuesto en esta sección, y, en los canales de escape, las dimensiones se atravesarán siguiendo un orden establecido, como en Hybrid–DOR, para evitar bloqueos. Además, en el crossbar también se necesitarían al menos dos canales virtuales para mantener las restricciones cuando un paquete sea enviado a un router después de atravesar el crossbar.

3.2.2. Encaminamiento en las Topologías k_n–ary n_n–direct m_n–indirect

En las topologías k_n–ary n_n–direct m_n–indirect, los crossbars son reemplazados por pequeñas MINs.Como ya se ha indicado anteriormente, se han considerado fat-trees o RUFTs en este trabajo.

En estas topologías, todos los switches dentro de un fat–tree o RUFT están siempre en la misma dimensión y en la misma posición relativa a los routers. Para poder identificar los switches dentro de un fat-tree o RUFT, se han extendido las coordenadas

clásicas de las MINs, incluyendo las coordenadas correspondientes de la MIN en la topología global. De esta manera, las coordenadas del switch en las topologías k_n–ary n_n–direct m_n–indirect vienen dadas por 4 tuplas *[d,p,e,o]*, donde d es la dimensión en la que se encuentra la MIN, p es la posición de la MIN en esta dimensión, e es la etapa del switch dentro de la MIN y o es la posición del switch en esa etapa. Hay que recordar que d y p son las coordenadas del crossbar equivalente en las topologías k_n–ary n_n–direct 1–indirect.

Dado que los routers son los mismos independientemente de la topología indirecta que se use, su algoritmo de encaminamiento es el mismo que para las topologías k_n–ary n_n–direct 1–indirect, tanto para encaminamiento determinista como adaptativo. Sin embargo, el algoritmo de encaminamiento de los switches depende de la red indirecta que se esté usando.

Primero nos centraremos en la topología k_n–ary n_n–direct m_n–indirect que usa fat–trees en las subredes indirectas. A pesar de que un fat–tree tiene varios caminos para cada par fuente–destino, lo que permite encaminamiento adaptativo, se propone utilizar el algoritmo de encaminamiento determinista presentado en [GGG+07], ya que es simple y permite superar al encaminamiento adaptativo para muchos patrones de tráfico. El encaminamiento está basado en dos fases. Primero, la fase de subida, donde el paquete es enviado hacia la raíz del fat–tree. Una vez el paquete ha terminado esta fase, empieza la fase de bajada hasta su destino. En particular, el enlace que se debe usar en ambas fases es dado por las coordenadas del destino correspondiente a la etapa en donde se encuentra el switch. Por ejemplo, si un switch localizado en la etapa e_1 encamina un paquete cuyo destino (en el fat–tree) es $\langle t_{n_i-1}, ..., t_{e_1+1}, t_{e_1}, t_{e_1-1}, ..., t_1, t_0 \rangle$, entonces el paquete será enviado por el enlace $k_i + t_{e_1}$ en la fase de subida y por el enlace t_{e_1} en la fase de bajada. Cabe recordar que k_i es la aridad de los switches de la topología fat–tree.

En las subredes fat–trees de las topologías k_n–ary n_n–direct m_n–indirect, el algoritmo de encaminamiento es el mismo, pero dado el identificador del destino $\langle x_{n_n-1}, ..., x_{d+1}, x_d, x_{d-1}, ..., x_1, x_0 \rangle$ (ver página 23) solo se usa la parte del identificador del mismo correspondiente a la dimensión en la que se encuentra el fat–tree (x_d). De esta manera, el paquete es entregado al mismo router que sería alcanzado en el caso de usar un crossbar, en una topología k_n–ary n_n–direct 1–indirect.

Este algoritmo de encaminamiento se muestra a continuación. $GetFTIdentifier$ obtiene de x_d las coordenadas para encaminar de forma local dentro del fat–tree. $UpwardsPhase$ devuelve *true* si el paquete está en la fase de subida o *false* si esta en la de bajada. Y la etapa del switch que esta encaminando el paquete es e:

Switches:

$ID = GetFTIdentifier(x_d)$
if $UpwardsPhase()$ **then**
  $link = k_i + ID[e]$
else
  $link = ID[e]$
end if

En RUFT, solo hay un camino entre cada par fuente–destino, y los paquetes viajan subiendo todas las diferentes etapas, alcanzando la última etapa, que está directamente conectada a los nodos de procesamiento. El enlace que tiene que ser utilizado en un

switch en particular es dado por la componente del destino correspondiente a la etapa donde el switch esta localizado. Véase [GGG+08] para más detalles. En este caso, el pseudo–código para el encaminamiento en los switches es el siguiente:

Switches:

$ID = GetFTIdentifier(x_d)$

$link = ID[e]$

Capítulo 4

Entorno de Simulación

Este capítulo presenta el simulador utilizado para evaluar las prestaciones de la topología propuesta. Además, para realizar una comparación, se evaluará el toro, la malla, el fat–tree, la flattened–butterfly y la nueva familia de topologías, usando como subredes indirectas el crossbar, fat–tree y RUFT.

4.1. Simulador

Para la evaluación de estas topologías se ha implementado un simulador detallado basado en eventos. El simulador modela varias topologías, incluyendo toros, mallas, fat–tree, flattened–butterfly y la nueva familia de topologías presentadas en este trabajo, k_n–ary n_n–direct m_n–indirect. Este simulador utiliza virtual cut–through como control de flujo. Cada switch tiene un crossbar con colas de dos paquetes tanto en los puertos de salida como en los de entrada. Para implementar el mecanismo de control de flujo se ha utilizado la técnica de créditos.

Se han utilizado distintos patrones de tráfico para la evaluación de las distintas topologías: uniforme, *hot–spot* y *complement*. En el patrón de tráfico uniforme, el destino del mensaje se elige de forma aleatoria de entre todos los posibles destinos. En el patrón de tráfico *hot–spot*, un porcentaje de tráfico se envía a un solo nodo, mientras que el tráfico restante se envía de forma uniforme. En el patrón de tráfico *complement*, el nodo destino se calcula complementando los bits del identificador del nodo fuente. O lo que es lo mismo, en nodo i enviará paquetes al nodo $N - i - 1$.

Para cada simulación, se ha inyectado tráfico en un rango completo (desde carga baja hasta carga de saturación).

4.2. Parámetros para el Entorno de Simulación

La Tabla 4.1 resume los parámetros que se han elegido para el entorno de simulación. El tamaño del paquete se expresa en flits, que es el tamaño mínimo de información cuya transferencia puede ser aceptada o rechazada. El tiempo de encaminamiento es el tiempo en el que se calcula por donde se va a encaminar un paquete. El tiempo de switch es el tiempo de transferencia de un flit por el switch. El tiempo de enlace es el tiempo en el que se transmite un flit en el canal físico. Por último, el tiempo de vuelo

Entorno de simulación	
tiempo de encaminamiento	20 ciclos de reloj
tiempo de switch	1 ciclo de reloj
tiempo de enlace	1 ciclo de reloj
tiempo de vuelo	8 ciclos de reloj
Tamaño del paquete	128 flits
	256 flits
	512 flits

Tabla 4.1: Parámetros de entorno del simulador.

se refiere al tiempo que tarda un flit en cruzar un enlace físico.

Estos valores fueron usados para modelar la red Myrinet en [FMLD00]. Además de estos parámetros, para la nueva topología que utiliza RUFT como subred indirecta, el tiempo de vuelo para sus enlaces largos que conectan la MIN unidireccional a los routers, se ha asumido que serán 8 ciclos de reloj por cada etapa que tenga la MIN.

Capítulo 5

Resultados Experimentales

En este capítulo, se compara la familia propuesta k_n–ary n_n–direct m_n–indirect utilizando diferentes subredes indirectas (crossbar, fat–tree y RUFT) con otras topologías bien conocidas y comúnmente utilizadas, como el toro, la malla y los fat–trees. Además. también se compara con la topología recientemente propuesta, la flattened–butterfly. Se ha considerado tanto el encaminamiento determinista como adaptativo en la evaluación de las distintas topologías. En el caso de la familia de topologías propuesta, se usarán los algoritmos descritos anteriormente en este trabajo. Para las mallas, toros y la topología flattened–butterfly se usará el algoritmo de encaminamiento DOR para encaminamiento determinista [DYL02], y para encaminamiento adaptativo se utilizará la técnica de la vía de escape de Duato [DYL02]. En los fat–trees, se ha usado un algoritmo de encaminamiento con equilibrado de carga propuesto en [GGG+07] para el encaminamiento determinista, y la función de selección SADP también presentada en [GGG+07] para el encaminamiento adaptativo.

Se han evaluado un alto rango de tamaños para estas topologías, desde 16 nodos hasta 4096 nodos. No se han podido simular topologías mayores por problemas de memoria en las máquinas disponibles. Para las topologías directas, como la malla y el toro, para la topología flattened–butterfly y para la nueva familia de topologías se han probado diferentes valores para el número de dimensiones y el número de nodos por dimensión. En concreto, se han evaluado redes de dos dimensiones con 4, 8, 16, 32 y 64 nodos por dimensión; tres dimensiones, con 4, 8, 16 nodos por dimensión; cuatro dimensiones, con 4 y 8 nodos por dimensión; y seis dimensiones, con 4 nodos por dimensión. A menos que se indique lo contrario, solo habrá un nodo de procesamiento conectado a cada router. Si varios nodos de procesamiento están conectados, se utilizará la notación p–c, siendo p el número de nodos de procesamiento conectado a cada router. Estas redes son comparadas con fat–trees con el mismo número de nodos de procesamiento, donde en varios casos se pueden dar varias configuraciones distintas.

5.1. Evaluación

En esta sección evaluaremos las prestaciones de las distintas topologías con las distintas configuraciones. Utilizaremos tanto algoritmos de encaminamiento deterministas como adaptativos.

5.1.1. Encaminamiento Determinista

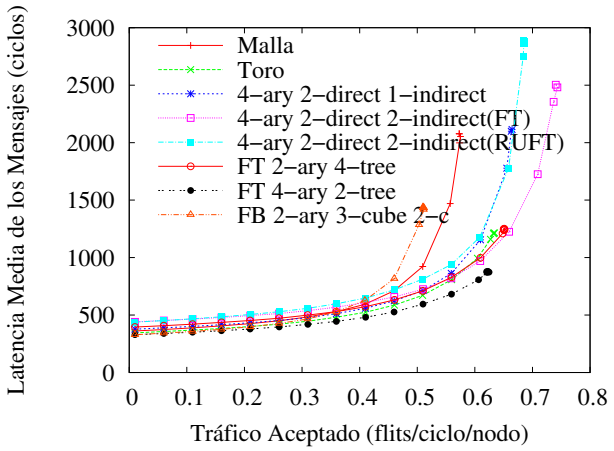

Figura 5.1: Tráfico uniforme para diferentes topologías con encaminamiento determinista y 2 dimensiones con 4 nodos por dimensión (16 nodos) para las topologías directas.

Patrón de tráfico uniforme: La Figura 5.1 muestra los resultados para una red muy pequeña (16 nodos) con tráfico uniforme. Como se puede ver, la topología flattened–butterfly es de las que menos productividad alcanza. Se ha seleccionado una 2–ary 3–cube flattened–butterfly para comparar con la propuesta de este trabajo con una complejidad hardware similar, como veremos en las siguientes secciones. La malla es la segunda peor topología y, el toro, las diferentes configuraciones de la familia propuesta y los fat–trees obtienen una productividad muy similar.

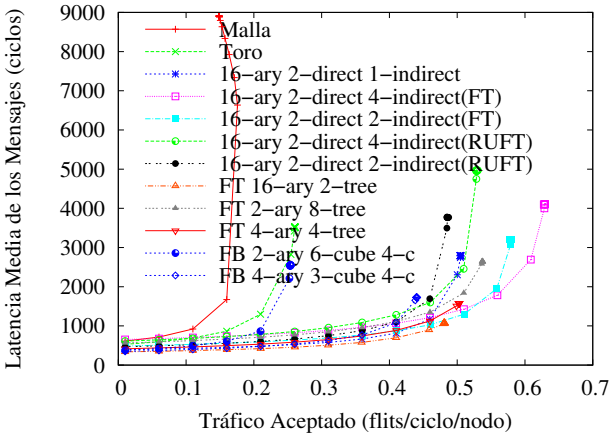

Figura 5.2: Tráfico uniforme para diferentes topologías con encaminamiento determinista y 2 dimensiones con 16 nodos por dimensión (256 nodos).

Al incrementar el número de nodos por dimensión dejando constante el número de dimensiones en las topologías n–dimensionales, como se puede ver en las Figuras 5.2 y 5.3, la productividad disminuye en todas las topologías. En el caso de las topologías

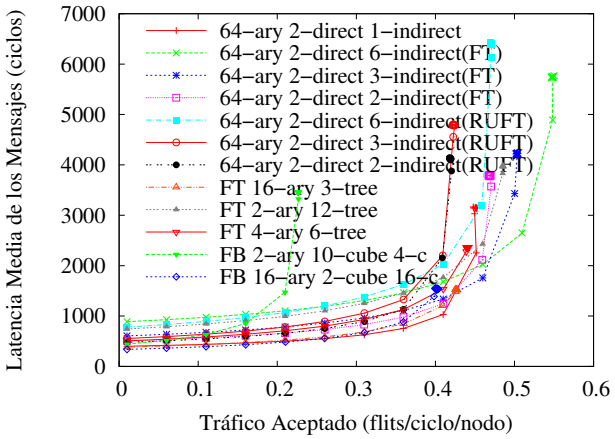

Figura 5.3: Tráfico uniforme para diferentes topologías con encaminamiento determinista y 2 dimensiones con 64 nodos por dimensión (4096 nodos).

malla y toro, la productividad decae en mayor medida, ya que la distancia media entre dos nodos crece mucho más que en las demás topologías. De hecho, en la Figura 5.3 no se muestran las topologías malla y toro porque su productividad es muy baja (un 89 % y un 84 % mas baja que la k_n–ary n_n–direct 1–indirect, respectivamente). Como se puede ver, la topología k_n–ary n_n–direct m_n–indirect es mucho más escalable que la malla y el toro. En todos los casos, la topología con mejores prestaciones es una de la nueva familia. En concreto, la que usa fat–trees como subred indirecta. Cabe destacar que cada topología tiene diferente coste hardware, que será evaluado en siguientes secciones.

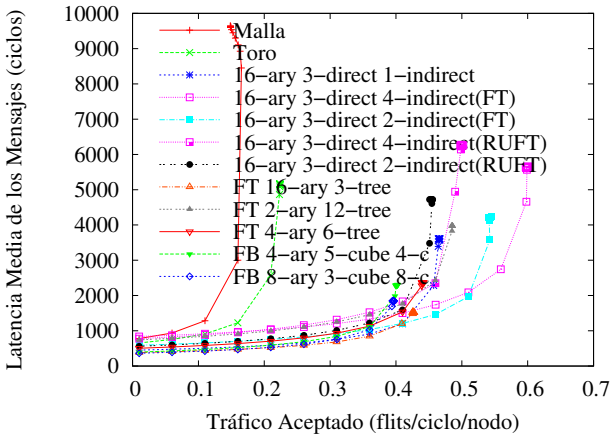

Figura 5.4: Tráfico uniforme para diferentes topologías con encaminamiento determinista y 3 dimensiones con 16 nodos por dimensión (4096 nodos).

En las Figuras 5.2 y 5.3 también se puede ver la diferencia de usar más etapas en las MINs de las topologías k_n–ary n_n–direct m_n–indirect. Para un mismo número de nodos por dimensión, si disminuimos el número de etapas, la aridad de los switches

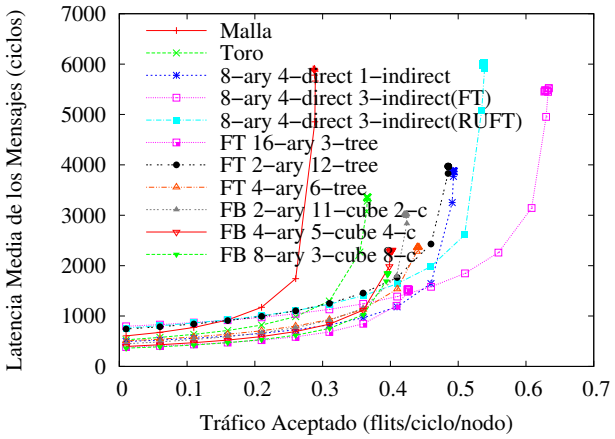

Figura 5.5: Tráfico uniforme para diferentes topologías con encaminamiento determinista y 4 dimensiones con 8 nodos por dimensión (4096 nodos).

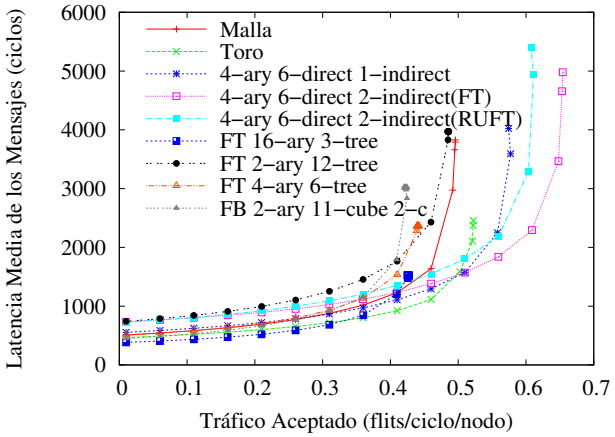

Figura 5.6: Tráfico uniforme para diferentes topologías con encaminamiento determinista y 6 dimensiones con 4 nodos por dimensión (4096 nodos).

aumenta, y se obtendrá una menor latencia. Es decir, cuanto más etapas tenga la subred indirecta, más latencia se obtendrá, ya que se tienen que cruzar más switches. Pero el hecho de tener más etapas también hace que se tenga más productividad. Este incremento de productividad se explica por el efecto del HoL (Head of Line) blocking. Con pocas etapas, cada topología indirecta tiene menos switches para servir a la misma cantidad de routers. Por eso, cada switch tiene que manejar más tráfico, lo que ayuda a tener más efecto HoL blocking y, por lo tanto, menos productividad.

Ahora pasaremos a analizar la latencia base. En una k_n–ary n_n–direct 1–indirect, la latencia base es estable, pues el número de saltos entre nodos no depende del número de nodos por dimensión. Sin embargo, en la k_n–ary n_n–direct m_n–indirect, y los fat–trees, la latencia base aumenta porque aumenta el número de etapas, y por tanto, el número de switches que se tienen que cruzar para mandar un paquete de un nodo a otro. En el caso de los toros y mallas, la latencia base está fuertemente ligada con el número de

nodos por dimensión, ya que la distancia media entre nodos aumenta.

Las Figuras 5.4, 5.5 y 5.6 analizan el impacto del número de dimensiones en las diferentes topologías. Se han analizado redes de 4096 nodos implementadas con diferentes números de dimensiones. Se puede distinguir tres comportamientos distintos. Las topologías malla y toro tienen un comportamiento similar. Cuanto más dimensiones y menor número de nodos por dimensión, mayor productividad se alcanza. También disminuye la latencia, porque la distancia media se reduce. Sin embargo, el comportamiento de la k_n–ary n_n–direct 1–indirect es diferente. La productividad aumenta con el número de dimensiones porque el tamaño de los switch de las subredes indirectas (un crossbar en este caso) es reducido, y por lo tanto, el pernicioso efecto del HoL blocking se reduce. Sin embargo, la latencia base no mejora con el incremento de dimensiones. Esto se debe al hecho de que el número de saltos que el paquete tiene que hacer crece con el número de dimensiones. En las topologías k_n–ary n_n–direct m_n–indirect tenemos un comportamiento similar, pero con una diferencia. La latencia base, en este caso, disminuye cuando incrementamos las dimensiones. Esto es porque hay menos nodos por dimensión, por lo tanto, las subredes indirectas tienen menos etapas, y los paquetes tienen que cruzar menos etapas. La configuración seleccionada para la topología flattened–butterfly (la única con un coste hardware similar al de la nueva familia propuesta) proporciona una productividad intermedia. Finalmente, cabe destacar que según vamos incrementando las dimensiones de las topologías directas, más competitivas son, ya que se reduce el número nodos por dimensión, y con ello, el número de saltos que debe hacer un paquete para llegar a su destino, disminuyendo la latencia y aumentando la productividad. De cualquier forma, cabe destacar que la nueva familia de topologías propuesta, usando crossbar, RUFTs o fat–trees para conectar los nodos de una dimensión, siempre obtiene la mejor productividad independientemente del número de dimensiones.

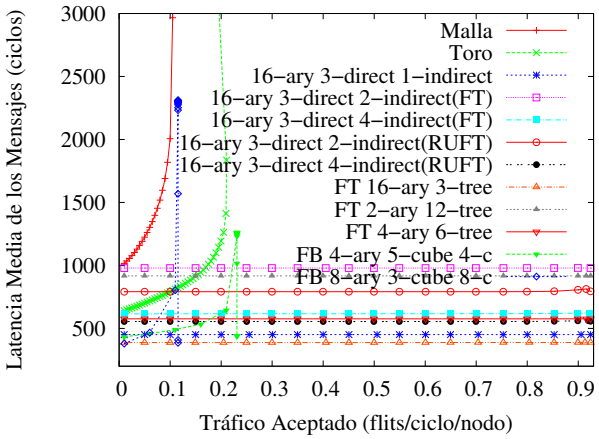

Figura 5.7: Tráfico *complement* para diferentes topologías con encaminamiento determinista y 3 dimensiones con 16 nodos por dimensión (4096 nodos).

Patrón de tráfico *complement*: Las Figuras 5.7, 5.8 y 5.9 muestra los resultados obtenidos para este patrón de tráfico para diferentes redes de 4096 nodos. Se pueden distinguir dos diferentes comportamientos entre las topologías analizadas. En

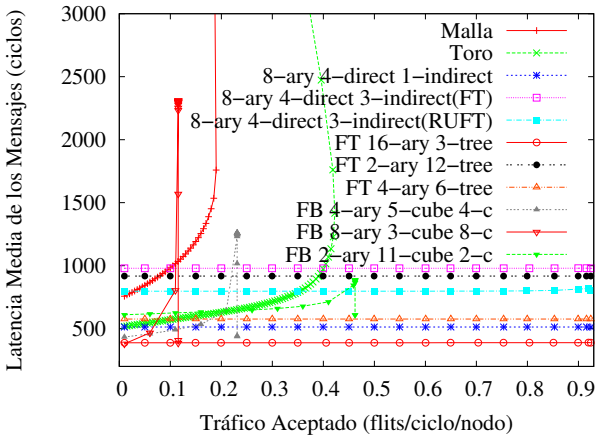

Figura 5.8: Tráfico *complement* para diferentes topologías con encaminamiento determinista y 4 dimensiones con 8 nodos por dimensión (4096 nodos).

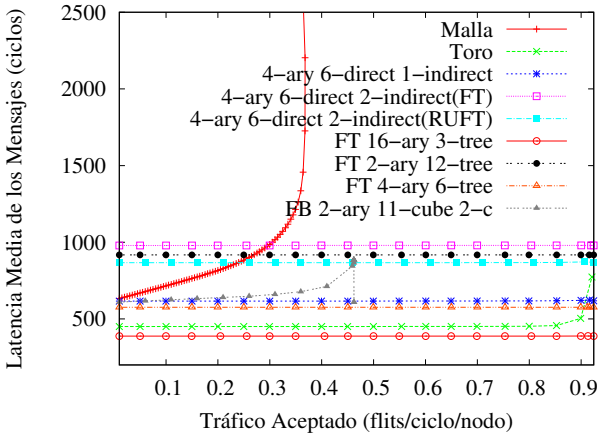

Figura 5.9: Tráfico *complement* para diferentes topologías con encaminamiento determinista y 6 dimensiones con 4 nodos por dimensión (4096 nodos).

las topologías toro, malla y flattened–butterfly, la red se satura rápidamente. Por otro lado, el resto de topologías (todas las topologías k_n–ary n_n–direct m_n–indirect y los fat–trees), la red es capaz de aceptar todo el tráfico inyectado. Y por eso, la latencia de la red es constante con cualquier tipo de inyección de tráfico. La razón es porque para este patrón de tráfico, como se utiliza un algoritmo de encaminamiento con equilibrado de carga en las MINs, los recursos de la red no se comparten entre diferentes pares fuente–destino. Es decir, el camino usado por un par fuente–destino dado, no se utiliza por otra pareja. Por lo tanto, las topologías híbridas y los fat–trees son los ganadores en este patrón de tráfico, y las topologías directas no son una buena opción.

Patrón de tráfico *hot–spot*: Como era de esperar, la concentración de paquetes en un mismo nodo hace que la red se sature con una productividad más baja que con otros patrones de tráfico. Como se puede ver en las Figuras 5.10, 5.11 y 5.12, la red se satura antes, obteniendo para todas las topologías una productividad similar, aunque

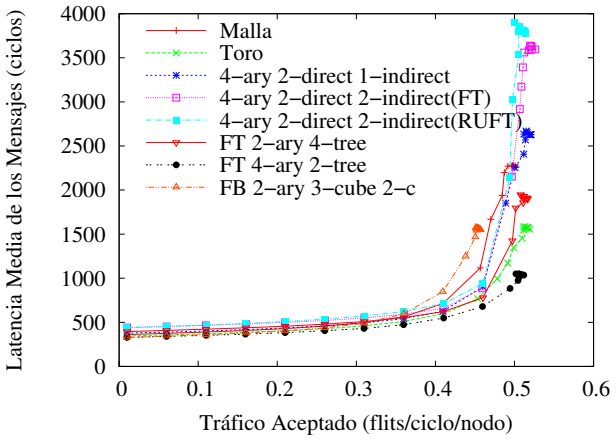

Figura 5.10: Tráfico *hot–spot* al 5 % para diferentes topologías con encaminamiento determinista y 2 dimensiones con 4 nodos por dimensión (16 nodos).

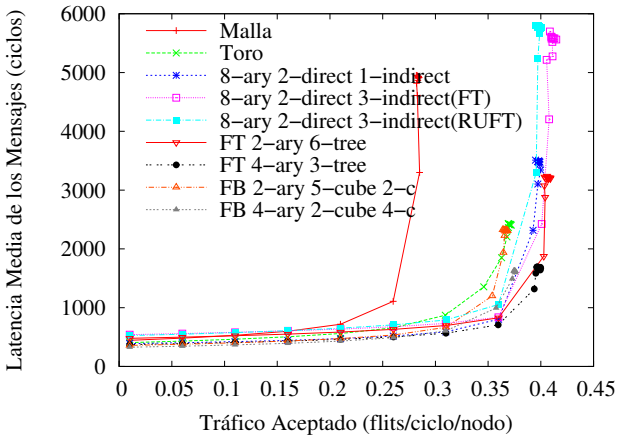

Figura 5.11: Tráfico *hot–spot* al 5 % para diferentes topologías con encaminamiento determinista y 2 dimensiones con 8 nodos por dimensión (64 nodos).

la malla empieza a saturar ligeramente antes que las demás.

Impacto del tamaño del paquete: También se ha evaluado como afecta el tamaño del paquete en los resultados que se obtienen con las diferentes topologías. Los comportamientos son relativamente parecidos independientemente del tamaño del paquete. La única diferencia, como era de esperar, es que a mayor tamaño de paquete mayor latencia base. También, como se inyecta el mismo nivel absoluto de tráfico, hay menos paquetes en la red, y por lo tanto menos contención, con lo que se consigue mayor productividad. Esto se puede ver en la Figura 5.13

Por lo tanto, se puede decir que la nueva familia de topologías puede obtener igual o mejores prestaciones que las topologías directas e indirectas para los diferentes patrones de tráfico.

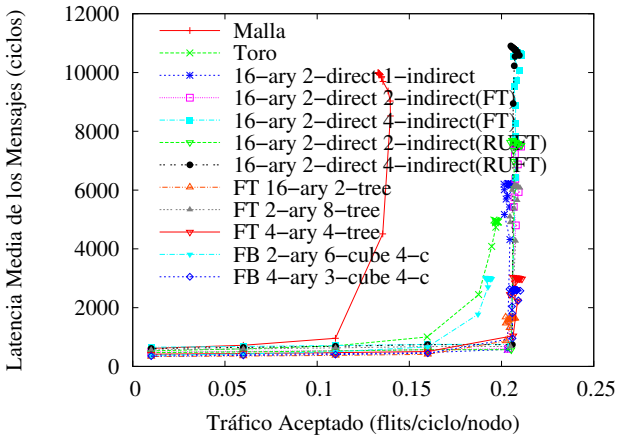

Figura 5.12: Tráfico *hot–spot* al 5 % para diferentes topologías con encaminamiento determinista y 2 dimensiones con 16 nodos por dimensión (256 nodos).

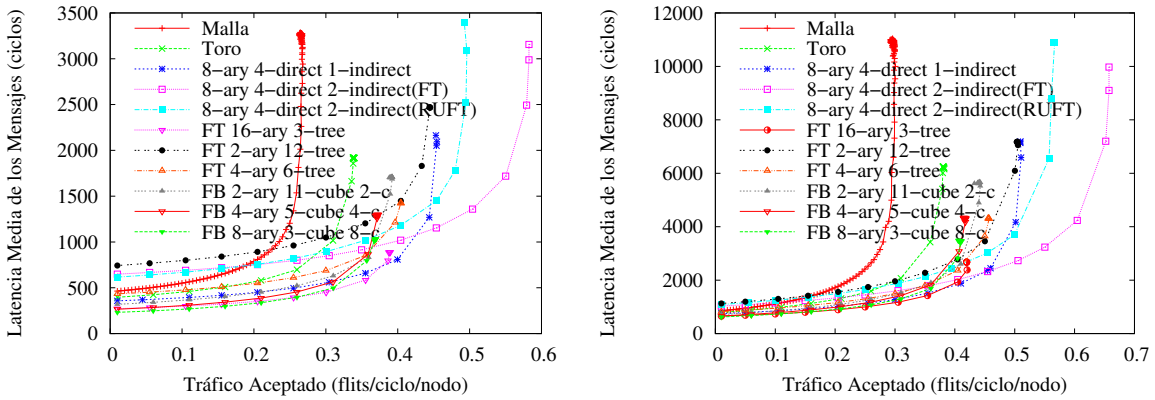

Figura 5.13: Tráfico uniforme para diferentes topologías con encaminamiento determinista y 4 dimensiones con 8 nodos por dimensión (4096 nodos). Tamaño del paquete: (a) 128 flits. (b) 512 flits.

5.1.2. Impacto de usar Encaminamiento Adaptativo

En esta sección, se compara el comportamiento de las diferentes topologías cuando se usa el encaminamiento adaptativo. En las topologías malla, toro y flattened–butterfly y la familia de topologías propuesta se necesitan al menos 2 canales virtuales para evitar bloqueos. Uno de los canales virtuales se usa como vía de escape mientras que los demás se utilizan de forma adaptativa, siguiendo la teoría de Duato [DYL02]. Cabe destacar que, en los resultados presentados, los fat–trees también utilizan 2 canales virtuales para una comparación más justa, aunque no se requieren para su algoritmo de encaminamiento adaptativo.

El comportamiento de las diferentes topologías con encaminamiento adaptativo es muy similar al que se consigue con el encaminamiento determinista, como se puede ver

en la Figura 5.14.(a) comparado con la Figura 5.2.

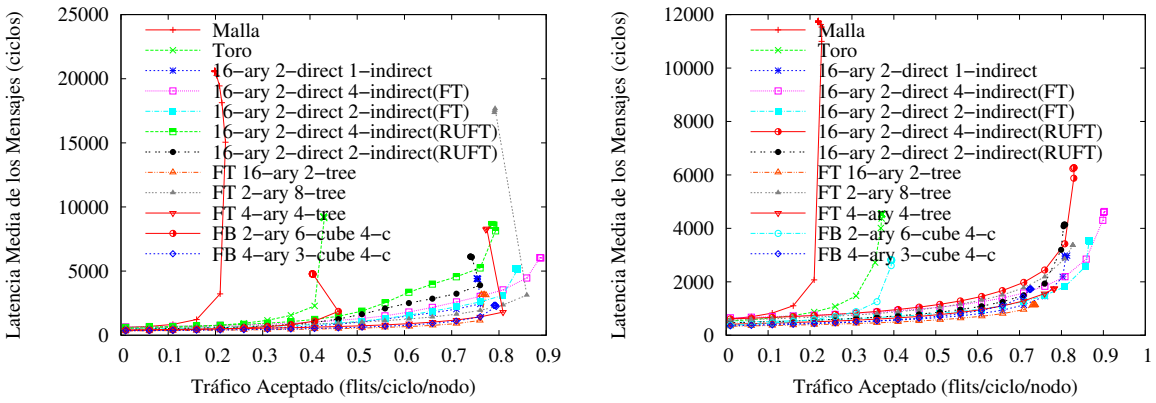

Figura 5.14: Tráfico uniforme en una red de 2 dimensiones y 16 nodos por dimensión (256 nodos) con encaminamiento (a) adaptativo y (b) determinista con dos CV.

Como era de esperar, el uso del encaminamiento adaptativo permite mejorar las prestaciones. Todas las instancias analizadas de la propuesta híbrida han mejorado. Como en el encaminamiento determinista, la que mejores resultados obtiene es 16–ary 2–direct 4–indirect (fat–tree), la cual mejora su productividad hasta un 41 %, si se compara con los resultados de la Figura 5.2.

Pero parte de esta mejora también es debida a que, al introducir encaminamiento adaptativo se ha usado un canal virtual más. Por lo tanto, para hacer una comparación justa, se debe analizar las prestaciones del encaminamiento determinista con el mismo número de canales virtuales. En este caso, el algoritmo no solo devolverá el canal físico, sino que devolverá los dos canales virtuales correspondientes al físico, de los cuales se seleccionará uno de forma aleatoria. La Figura 5.14.(b) muestra los resultados para este caso. Como se puede ver, la mejora del encaminamiento adaptativo respecto al determinista ya no es tan visible y, de hecho, hay ciertos casos donde no hay mejora. Esto es debido también a que, al tratarse de un patrón de tráfico uniforme, el tráfico ya está bien repartido por la red, entonces un encaminamiento adaptativo ofrece poca ayuda en este caso.

Para terminar, se quiere remarcar que las topologías fat–tree y flattened–butterfly, incluso con canales virtuales, no superan a la nueva familia de topologías propuesta, ya sea encaminamiento adaptativo o determinista.

5.2. Análisis Coste–Prestaciones

Esta sección estima y compara el coste hardware de cada topología evaluada en este trabajo. En particular, se analiza, para cada topología, el número de enlaces, el número de switches y el número de elementos de conmutación. En concreto, se estima el coste hardware mediante el número total de elementos de conmutación, ya que considera tanto el total de switches necesarios como su grado, lo que esta relacionado con el

	Malla	Toro	Fat–tree	Flattened–Butterfly
Switches	N	N	Nn_i/k_i	$k_f^{n_f}$
elementos de conmutación	$N(2n_d+1)^2$	$N(2n_d+1)^2$	$4k_in_iN$	$k_f^{n_f}(n_f(k_f-1)+N/k_f^{n_f})^2$
enlaces	$2(k_d-1)k_d^{n_d-1}n_d$	$2k_d^{n_d}n_d$	$2N(n_i-1)$	$k_f^{n_f}(k_f-1)n_f$

k_n–ary n_n–direct m_n–indirect	subred Fat–tree	subredes Crossbar and RUFT
Switches	$N/p_n((m_nn_n/k_i)+1)$	$N/p_n((m_nn_n/k_i)+1)$
elementos de conmutación	$N/p_n((4m_nn_nk_i+(n_n+p_n)^2)$	$N/p_n(m_nn_nk_i+(n_n+p_n)^2)$
enlaces	$2(k_n^{n_n}n_n+(m_n-1)Nn_n/p_n)$	$2k_n^{n_n}n_n+(m_n-1)Nn_n/p_n$

Tabla 5.1: Comparación analítica de la Malla, Toro, Fat–Tree, flattened–butterfly (primera tabla), y las topologías k_n–ary n_n–direct m_n–indirect (segunda tabla). k_i en las topologías k_n–ary n_n–direct m_n–indirect se refiere a la aridad de los switches indirectos.

número de enlaces. En este trabajo, consideramos el elemento de conmutación como componente básico de un multiplexor, de tal forma que para un multiplexor de n entradas requiere n elementos de conmutación. Además, en esta sección se introducirán dos características más con el fin de evaluar mejor de forma combinada las prestaciones con el coste hardware de una topología. La primera es la productividad por elemento de conmutación. Cuanto mayor sea este parámetro, mejor relación prestaciones/coste tendrá la topología. El otro parámetro es el producto de la latencia base y el número de elementos de conmutación. Cuanto mayor sea, peor será la topología. La productividad nos dice el tráfico aceptado por la red, y está expresada en flits por ciclo por nodo. La latencia base está expresada en ciclos.

La Tabla 5.1 muestra las fórmulas para calcular el número de enlaces, switches y elementos de conmutación para cada topología. En el cálculo de los enlaces, solo se consideran los enlaces de la red, es decir, no están incluidos los enlaces entre los nodos de procesamiento y los switches. Aunque el número de enlaces que conectan estos nodos a la red son los mismos para todas las topologías. Otro aspecto a considerar es que en todas las topologías, los enlaces son bidireccionales pero, cuando se usa RUFT como subred indirecta, estas subredes utilizan enlaces unidireccionales. Además, el número de elementos de conmutación en las subredes RUFT es cuatro veces menor que en una subred fat–tree, debido a que se usan switches unidireccionales.

La Tabla 5.2 muestra en la parte de arriba los resultados para las topologías de 16 nodos. Las redes son de dos dimensiones en el caso de las topologías k_n–ary n_n–direct m_n–indirect, malla y toro. En esta comparación se considera tráfico uniforme y encaminamiento determinista. También se ha considerado una red de dos dimensiones para la topología flattened–butterfly y otra configuración de tres dimensiones que posee un coste hardware similar a la familia propuesta, con el mismo número de nodos de procesamiento. Como puede observarse, la topología flattened–butterfly de dos dimensiones es la que mayor productividad alcanza, seguida de la k_n–ary n_n–direct m_n–indirect usando fat–trees como subredes. Pero con un gran número de elementos de conmutación. Las siguientes son la k_n–ary n_n–direct m_n–indirect con RUFT y crossbar como subredes indirectas.

Ahora comparemos los costes de las distintas topologías. La topología flattened–butterfly es la que menos elementos de conmutación utiliza. Después le sigue las topologías k_n–ary n_n–direct 1–indirect y k_n–ary n_n–direct m_n–indirect utilizando RUFT

#Nodos	Topología	Lat. base	Prod.	Enlaces	Switches	E.C.	Prod./E.C.	Lat. base*E.C.
16	Flattened–Butterfly 2-ary 3-cube 2-c	329,00	0,51133	24	8	200	0,0025567	65800
16	4–ary 2–direct 2–indirect(RUFT)	437,01	0,69003	96	48	272	0,0025369	118866
16	4–ary 2–direct 1–indirect	375,48	0,66259	64	24	272	0,0024360	102131
16	Toro	345,30	0,63534	64	16	400	0,0015883	138120
16	Malla	360,38	0,57466	48	16	400	0,0014367	144150
16	Fat-Tree aridad 2 etapas 4	415,57	0,65030	96	32	512	0,0012701	212773
16	Fat-Tree aridad 4 etapas 2	327,46	0,62456	32	8	512	0,0012198	167661
16	4–ary 2–direct 2–indirect(FT)	439,24	0,74223	128	48	656	0,0011314	288139
16	Flattened–Butterfly 4-ary 2-cube 1-c	328.47	0,81492	96	16	784	0,0010394	257524
4096	64–ary 2–direct 6–indirect(RUFT)	770,56	0,47438	57344	28672	135168	0,0000035	104155095
4096	64–ary 2–direct 3–indirect(RUFT)	546,71	0,42546	32768	10240	135168	0,0000031	73898072
4096	64–ary 2–direct 2–indirect(RUFT)	463,11	0,41971	24576	6144	167936	0,0000025	77772700
4096	64–ary 2–direct 6–indirect(FT)	866,40	0,55148	98304	28672	430080	0,0000013	372620624
4096	64–ary 2–direct 3–indirect(FT)	591,82	0,50569	49152	10240	430080	0,0000012	254531077
4096	Fat-Tree aridad 2 etapas 12	865,20	0,48493	90112	24576	393216	0,0000012	340210585
4096	Fat-Tree aridad 4 etapas 6	561,23	0,43872	40960	6144	393216	0,0000011	220683070
4096	Flattened–Butterfly 2-ary 10-cube 4-c	439,94	0,22688	10240	1024	200704	0,0000011	88297728
4096	64–ary 2–direct 2–indirect(FT)	499,96	0,46912	32768	6144	561152	0,0000008	280553722
4096	64–ary 2–direct 1–indirect	389,29	0,44818	16384	4224	561152	0,0000008	218448488
4096	Toro	1292,88	0,07323	16384	4096	102400	0,0000007	132390963
4096	Flattened–Butterfly 16-ary 2-cube 16-c	338,47	0,40332	7680	256	541696	0,0000007	183349068
4096	Fat-Tree arity 16 stages 3	388,45	0,41601	16384	768	786432	0,0000005	305487961
4096	Malla	1606,12	0,05084	16128	4096	102400	0,0000005	164467087
4096	Flattened–Butterfly 64-ary 2-cube 1-c	339,65	0,60032	516096	4096	66064384	0,0000000	22435873084

Tabla 5.2: Resultados para diferentes topologías 2–D con tráfico uniforme y encaminamiento determinista.

como subred indirecta, las cuales tienen el mismo número de elementos de conmutación. Aunque la última necesita más switches para ser implementada, estos switches son mucho más simples y por esa razón tienen el mismo número de elementos de conmutación. Por otro lado, las topologías flattened–butterfly de 2 dimensiones y k_n–ary n_n–direct m_n–indirect usando fat–trees como subredes indirectas, en ese orden, son las topologías más costosas, respecto a número de elementos de conmutación.

Sin embargo, cuando se considera la relación productividad/coste, y en concreto el parámetro productividad/elemento de conmutación, las conclusiones son totalmente diferentes. En este caso, la mejor topología es la flattened–butterfly de 3 dimensiones, seguida muy de cerca por k_n–ary n_n–direct m_n–indirect con RUFT y k_n–ary n_n–direct 1–indirect. Por otro lado, los fat–trees son menos interesantes debido a su mayor complejidad. Conforme va subiendo el número de etapas de las subredes indirectas, se puede ver que empeora la relación latencia–coste. Sin embargo, si pensamos solo en la productividad, una MIN con más etapas es mejor, ya que reduce el efecto HoL blocking.

La Tabla 5.2 muestra también los resultados para redes más grandes. Como se puede observar, cuando el número de nodos por dimensión aumenta, las prestaciones de red de las topologías malla y toro empeoran en gran medida. Respecto a las topologías híbridas que se han presentado en este trabajo, los resultados mostrados confirman lo que se dijo anteriormente. Las mejores prestaciones las obtienen las topologías flattened–butterfly y las configuraciones de k_n–ary n_n–direct m_n–indirect (subredes fat–tree), con una ventaja mayor para las topologías propuestas en este trabajo a medida que vamos aumentando el número de nodos por dimensión. Desde el punto de vista coste–prestaciones, las diferentes nuevas topologías con RUFT como subredes indirectas son la mejor opción. Aunque la topología k_n–ary n_n–direct m_n–indirect (RUFTs) no alcanza la mayor productividad, tiene la ventaja de que requiere un coste hardware

#DIM	Topología	Lat. base	Prod.	Enlaces	Switches	E.C.	Prod./E.C.	Lat. base*E.C.
3	16–ary 3–direct 4–indirect(RUFT)	766,14	0,50088	61440	28672	163840	0,0000031	125524540
3	16–ary 3–direct 2–indirect(RUFT)	544,01	0,45590	36864	10240	163840	0,0000028	89130980
3	16–ary 3–direct 1–indirect	443,32	0,46498	24576	4864	262144	0,0000018	116213728
3	16–ary 3–direct 4–indirect(FT)	815,39	0,60257	98304	28672	458752	0,0000013	374060903
3	Fat-Tree arity 2 stages 12	865,20	0,48493	90112	24576	393216	0,0000012	340210585
3	16–ary 3–direct 2–indirect(FT)	576,95	0,50412	49152	10240	458752	0,0000011	264674815
3	Torus	654,40	0,22376	24576	4096	200704	0,0000011	131340348
3	Fat-Tree arity 4 stages 6	561,23	0,43872	40960	6144	393216	0,0000011	220683070
3	Flattened–Butterfly 4-ary 5-cube 4-c	396,56	0,40179	15360	1024	369664	0,0000011	146595250
3	Flattened–Butterfly 8-ary 3-cube 8-c	361,82	0,39680	10752	512	430592	0,0000009	155798223
3	Mesh	762,46	0,16553	23040	4096	200704	0,0000008	153029131
3	Fat-Tree arity 16 stages 3	388,45	0,41601	16384	768	78432	0,0000005	305487961
3	Flattened–Butterfly 16-ary 3-cube 1-c	365,00	0,63116	184320	4096	8667136	0,0000001	3163526160
6	4–ary 6–direct 2–indirect(RUFT)	714,24	0,61761	73728	28672	299008	0,0000021	213564147
6	4–ary 6–direct 1–indirect	546,78	0,57877	49152	10240	299008	0,0000019	163492186
6	Fat-Tree arity 2 stages 12	865,20	0,48493	90112	24576	393216	0,0000012	340210585
6	Flattened–Butterfly 2-ary 11-cube 2-c	450,03	0,42385	22528	2048	346112	0,0000012	155761542
6	4–ary 6–direct 2–indirect(FT)	715,60	0,65648	98304	28672	593920	0,0000011	425011201
6	Fat-Tree arity 4 stages 6	561,23	0,43872	40960	6144	393216	0,0000011	220683070
6	Torus	463,94	0,52038	49152	4096	692224	0,0000008	321150900
6	Mesh	502,79	0,49203	36864	4096	692224	0,0000007	348041789
6	Fat-Tree arity 16 stages 3	388,45	0,41601	16384	768	78432	0,0000005	305487961
6	Flattened–Butterfly 4-ary 6-cube 1-c	416,37	0,66666	73728	4096	1478656	0,0000005	615671623

Tabla 5.3: Resultados para topologías de 4096 nodos con tráfico uniforme y encaminamiento determinista.

más reducido, ya que RUFT combina una MIN unidireccional con un algoritmo de encaminamiento optimizado, obteniendo así la mejor productividad por elemento de conmutación.

La tabla 5.3 muestra los resultados para las mismas grandes redes (4096 nodos) pero ahora considerando más dimensiones (3 y 6). Como se puede observar, cuando el número de dimensiones utilizadas aumenta se consiguen mejoras, especialmente para las topologías malla y toro. Sin embargo, la topología que siempre obtiene mejor productividad es la flattened–butterfly, seguida de la configuración k_n–ary n_n–direct m_n–indirect (fat–trees), aunque con un mayor coste en el caso de la topología flattened–butterfly. La mejor relación coste–prestaciones la consigue la topología k_n–ary n_n–direct m_n–indirect (RUFT). En concreto, la productividad por elemento de conmutación de esta topología es mucho mejor que las obtenidas por las topologías fat–tree o flattened–butterfly.

Por otro lado, al incrementar el número de dimensiones, en el caso de las topologías k_n–ary n_n–direct m_n–indirect (fat–trees y RUFTs), toro y malla, el número de switches directos (routers) se mantienen, pero estos tienen ahora una mayor aridad, lo que hace que aumente el número de elementos de conmutación. En el caso de la topología k_n–ary n_n–direct 1–indirect, el número de switches se incrementa con el número de dimensiones, los switches indirectos (crossbars) tienen una menor aridad y los switches directos (routers) una mayor aridad, y por eso también se incrementa el número de elementos de conmutación.

La tabla también muestra el coste en enlaces para cada topología. Como se puede ver, algunas de las configuraciones de la topologías flattened–butterfly, fat–tree y k_n–ary n_n–direct m_n–indirect (fat–trees) tienen el mayor número de enlaces. Sin embargo, en las topologías fat–tree y k_n–ary n_n–direct m_n–indirect (fat–trees), que tienen menor número de enlaces, tienen menos etapas, pero tienen una mayor aridad. Estas

configuraciones tienen también los mejores resultados latencia–complejidad. Respecto al número de dimensiones y el número de nodos por dimensión, si aumentamos el número de dimensiones para el mismo número total de nodos, el número de enlaces aumenta, porque estamos añadiendo más conectividad. La topología k_n–ary n_n–direct m_n–indirect (fat–trees) es una excepción porque, aunque se tienen más fat–trees por el incremento del número de dimensiones, estos fat–trees son más pequeños porque no tienen que interconectar tantos nodos y necesitan menos enlaces para interconectar los switches. Esto hace que la topología k_n–ary n_n–direct m_n–indirect (fat–trees) tenga menos impacto a la hora de incrementar el número de dimensiones. También hay una diferencia significativa entre k_n–ary n_n–direct m_n–indirect usando fat–trees y RUFTs como subredes indirectas, en lo que se refiere a número de enlaces. Esto es porque en RUFT los enlaces son unidireccionales, mientras que en los fat–trees son bidireccionales.

5.3. Tolerancia a fallos

Si tenemos en cuenta la tolerancia a fallos, las topologías propuestas proporcionan muchos caminos alternativos para cada par fuente–destino, lo cual es muy importante desde el punto de vista de la tolerancia a fallos para evitar fallos.

En una topología malla, el peor caso ocurre cuando un enlace conectado a un nodo esquina falla. como cada nodo esquina tiene un número de enlaces igual al número de dimensiones de la red, para mantener la red conectada, el máximo número de fallos que puede tolerar es igual al número de dimensiones menos 1. La topología toro puede tolerar más fallos que la malla debido a que los paquetes pueden ser movidos por ambas direcciones de una dimensión, por lo tanto pueden tolerar un número de fallos igual a 2 veces el número de dimensiones menos 1. La topología fat–tree tolera tantos fallos como puertos de subida o bajada tengan los switches menos 1. La topología flattened–butterfly tolera $k_f(n_f - 1)$ fallos, es decir, el grado del switch. En el caso de las distintas topologías k_n–ary n_n–direct m_n–indirect, el número de fallos tolerados en la parte directa de la red es el número de dimensiones menos 1. Si se usa un crossbar como subred indirecta, esos son los fallos totales que se pueden tolerar. Para las que usan fat–trees, los fallos totales que se pueden tolerar es el mínimo entre los de la parte directa y el número de enlaces de bajada o subida en los fat–trees menos 1. Como se ha comentado en el trabajo, es de esperar que se usen switches con un alto grado en los fat–trees en un futuro próximo, por lo tanto, para las configuraciones más utilizadas, el limite inferior viene dado por la parte directa de la red. Finalmente, si se utilizan RUFTs en las subredes indirectas, no se puede tolerar ningún fallo, porque solo existe un único camino por cada par fuente–destino en las RUFTs. Por lo tanto, las topologías k_n–ary n_n–direct m_n–indirect con RUFT en las subredes indirectas tienen muy buena relación coste/prestaciones, pero no es la mejor opción desde el punto de vista de la tolerancia a fallos. Cabe destacar, que el encaminamiento debe cambiar para dar soporte a la tolerancia a fallos en todas las topologías, pero esto no es el objetivo de este trabajo.

Para realizar una comparación fácil entre todas las topologías, se asume que disponemos de switches de grado d. En una red malla, el número de dimensiones pueden ser $d/2$, por lo tanto, el número de fallos que se pueden tolerar son $d/2 - 1$. En el caso de

una red toro, también tendremos una red con $d/2$ dimensiones, con una tolerancia a fallos igual a $d-1$. En un fat–tree, los switches pueden tener $d/2$ puertos de subida y $d/2$ puertos de bajada, por lo tanto se toleran $d/2-1$ fallos. En la topología flattened–butterfly, el número de fallos tolerados es igual al grado del switch menos 1, por lo tanto $d-1$. Para la nueva topología que utiliza crossbars como subredes indirectas, con switches de grado d, se puede construir una red de d dimensiones, por lo tanto, la red puede tolerar hasta $d-1$ fallos, la máxima tolerancia alcanzada en las topologías evaluadas. Si se usan fat–trees como subredes indirectas, entonces se tolerarán $d/2-1$ fallos.

No se consideran los fallos en los enlaces de inyección, pero las topologías fat–trees y flattened–butterfly tienen normalmente un único enlace que conecta el nodo de procesamiento a la red. Si este enlace falla, el nodo quedará aislado. Por lo tanto, considerando también el enlace de inyección, estas redes no toleran ningún fallo. Por el contrario, en la nueva familia de topologías, si el router está implementado dentro de cada nodo, está conectado a la red mediante tantos enlaces como dimensiones tenga la red, y el nivel de tolerancia a fallos de los enlaces de inyección es de $d-1$ fallos.

Capítulo 6

Conclusiones, Publicaciones y Trabajo Futuro

6.1. Conclusiones

En este trabajo se ha presentado una nueva familia de topologías híbridas para interconectar redes de gran escala con el objetivo de combinar las ventajas de las topologías directas e indirectas.

La familia combina una topología n–dimensional donde los nodos de cada dimensión están conectados mediante una pequeña topología indirecta. Esta topología indirecta puede ser un crossbar, un fat–tree o un RUFT. Esta combinación proporciona un conjunto de topologías que proporciona altas prestaciones, con latencias y productividades cercanas a las redes indirectas, pero con un menos coste hardware. En concreto las nuevas topologías que usan fat-trees como subredes indirectas son las topologías que alcanzan la mayor productividad. Sin embargo, desde el punto de vista coste/prestaciones, las nuevas topologías con RUFT como subredes indirectas obtienen mejores resultados, siendo capaces de obtener el doble de la productividad por elemento de conmutación comparado con las topologías indirectas, y esta ventaja es aún mayor para las topologías directas. Además, el diseño de la topología es mucho más simple que en las topologías indirectas.

Respecto a la tolerancia a fallos, las nuevas topologías que utilizan crossbars o fat–trees como subredes indirectas son capaces de obtener el mismo nivel de tolerancia a fallos que se obtiene en las topologías directas o indirectas, o incluso mejorarlo. Por otro lado, la nueva familia que utiliza RUFT como subredes indirectas no proporciona tolerancia a fallos, por lo tanto, es una opción adecuada para un diseño efectivo donde lo importante sea la relación coste/prestaciones y no tenga mucha relevancia la tolerancia a fallos. Pero si la tolerancia a fallos es relevante, entonces las otras subredes indirectas (crossbar y fat–trees) son más apropiadas.

6.2. Publicaciones

Este trabajo de investigación ha dado lugar a las siguientes publicaciones:

- R. Peñaranda, C. Gómez, M.E. Gómez, P. López, and J. Duato. A New Family of

Hybrid Topologies for Large–Scale Interconnection Networks, *The 11th IEEE International Symposium on Network Computing and Applications (IEEE NCA12)*, Agosto 2012, Cambridge, USA. (Core A). Aceptado, pendiente de publicación.

- R. Peñaranda, C. Gómez, M.E. Gómez, P. López, and J. Duato. IODET: A HoL-blocking-aware Deterministic Routing Algorithm for Direct Topologies, *18th IEEE International Conference on Parallel and Distributed Systems (IEEE ICPADS 2012)*, 2012, Singapore. (Core B). Enviado, pendiente de notificación.

6.3. Trabajo Futuro

Como continuación de este trabajo, hay varios puntos de interés que podrían ser desarrollados:

- Analizar en profundidad la relación entre encaminamiento determinista y adaptativo para distintos patrones de tráfico.
- Por otro lado, hemos visto en este trabajo, un somero análisis sobre la tolerancia a fallos. Se pretende diseñar algoritmos encaminamiento tolerantes a fallos. Una vez diseñados, se implementarán, mediante un simulador, y evaluarán para comparar los resultados y las prestaciones obtenidas con otras topologías utilizando también encaminamiento tolerante a fallos.
- Otro aspecto en el que tenemos previsto centrarnos es en analizar la implementación física real de la topología.
- Por último se pretende añadir nuevas técnicas de control de la congestión especialmente diseñadas para la nueva topología.

Bibliografía

[BA84] L.N. Bhuyan and D.P. Agrawal. Generalized hypercube and hyperbus structures for a computer network. *Computers, IEEE Transactions on*, C-33(4):323 –333, april 1984.

[BDJ+11] N. Binkert, Al Davis, N.P. Jouppi, M. McLaren, N. Muralimanohar, R. Schreiber, and Jung Ho Ahn. The role of optics in future high radix switch design. *SIGARCH Comput. Archit. News*, 39(3):437–448, June 2011.

[BQV09] A.O. Balkan, Gang Qu, and U. Vishkin. Mesh-of-trees and alternative interconnection networks for single-chip parallelism. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(10):1419 –1432, oct. 2009.

[DEM+09] R. Das, S. Eachempati, A.K. Mishra, V. Narayanan, and C.R. Das. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, pages 175 –186, feb. 2009.

[DT04] W.J. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.

[DVS87] G. Della Vecchia and C. Sanges. Recursively scalable networks for message passing architectures. *Parallel Processing and Applications*, 1987.

[DYL02] J. Duato, S. Yalamanchili, and N. Lionel. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[FMLD00] J. Flich, M.P. Malumbres, P. López, and J. Duato. Improving routing performance in myrinet networks. *Parallel and Distributed Processing Symposium, International*, 0:27, 2000.

[GD06] A.K. Gupta and W.J. Dally. Topology optimization of interconnection networks. *Computer Architecture Letters*, 5(1):10 –13, jan.-june 2006.

[GGG+07] C. Gomez, F. Gilabert, M.E. Gomez, P. Lopez, and J. Duato. Deterministic versus adaptive routing in fat-trees. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1 –8, march 2007.

[GGG+08] C. Gomez, F. Gilabert, M.E. Gomez, P. Lopez, and J. Duato. Ruft: Simplifying the fat-tree topology. In *Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on*, pages 153 –160, dec. 2008.

[IBA] Mellanox technology. `http://www.mellanox.com`.

[KDA07] J. Kim, W.J. Dally, and D. Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 126–137, New York, NY, USA, 2007. ACM.

[KDSA08] J. Kim, W.J. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable dragonfly topology. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, pages 77–88, Washington, DC, USA, 2008. IEEE Computer Society.

[Lei85] Charles E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.*, 34(10):892–901, October 1985.

[Lei92] Book review: Introduction to parallel algorithms and architectures : Arrays, trees, hypercubes by f. t. leighton (morgan kauffman pub, 1992). *SIGACT News*, 23(3):31–32, June 1992. Reviewer-Das, Sajal K .

[LFNB08] H. Litz, H. Froning, M. Nuessle, and U. Bruning. Hypertransport nic for ultra-low latency message transfer. feb 2008.

[MKA07] H. Matsutani, M. Koibuchi, and H. Amano. Performance, cost, and energy evaluation of fat h-tree: A cost-efficient tree-based on-chip network. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1 –10, march 2007.

[myr] Myricom. `http://www.myri.com`.

[Qua] Quadrics homepage. `http://www.quadrics.com`.

[RKHSA06] D. Rahmati, A.E. Kiasari, S. Hessabi, and H. Sarbazi-Azad. A performance and power analysis of wk-recursive and mesh networks for network-on-chips. In *Computer Design, 2006. ICCD 2006. International Conference on*, pages 142 –147, oct. 2006.

[SAKD06] S. Scott, D. Abts, J. Kim, and W.J. Dally. The blackwidow high-radix clos network. *SIGARCH Comput. Archit. News*, 34(2):16–28, May 2006.

[top] TOP500 supercomputer site. `http://www.top500.org`.

[YFJ+01] Yulu Yang, A. Funahashi, A. Jouraku, H. Nishi, H. Amano, and T. Sueyoshi. Recursive diagonal torus: an interconnection network for massively parallel computers. *Parallel and Distributed Systems, IEEE Transactions on*, 12(7):701 –715, jul 2001.