UNIVERSIDAD DE CANTABRIA
Facultad de Ciencias

*Trabajo Fin de Máster*

# EVALUACIÓN DE TOPOLOGÍAS DE SISTEMA BASADAS EN ROUTERS DE ALTO GRADO

## (Evaluation of system topologies based upon high-radix routers)

Para acceder al Titulo de

**MÁSTER DE COMPUTACIÓN**

**Autor: Pablo Fuentes Sáez**

**Septiembre - 2012**

## Agradecimientos

A Enrique y Carmen, los directores de este trabajo. Habéis estado a mi lado ayudándome desde el principio, con paciencia y muy buen humor. ¡Sois geniales! Habéis aguantado estoicamente las tardes trabajando hasta las tantas, las prisas, los retrasos. Sin vosotros este año habría perdido la cabeza. Gracias.

A Mon, por toda la ayuda que me has dado. Siempre has estado ahí para darme una opinión, un consejo, una oportunidad.

A Emilio, Marina, Cristóbal, Miguel, David... porque sin vosotros venir día a día sería infinitamente más aburrido. Esa pausa del café que siempre viene bien, esa mano que siempre estáis dispuestos a echar, hay tantas razones que no caben en esta hoja. Más que compañeros, sois amigos.

A Esteban, José Luis, Rafa y Fernando, porque durante este año he tenido la oportunidad de trabajar a vuestro lado y aprender de vosotros. Sólo deseo poder seguir haciéndolo.

A Sebio, Alma, Warren y Paula, porque aunque este año no nos hayamos podido ver tanto, siempre estáis ahí. Sé que os debo un sinfín de cafés, pero de momento aprovecho estas líneas para daros las gracias por haber hecho 5 años de carrera bastante más divertidos.

A mis padres, mis abuelos y mi hermana, por haberme apoyado y ayudado tanto. Porque una barbacoa con vosotros siempre despeja la cabeza. Porque como las patatas de Ubiarco no hay ninguna. Porque...

Y a Lorena, por seguir aguantando mi mal humor matutino, mis momentos 'plof'. Por ser la culpable de los buenos momentos, de los paseos, de las risas. Porque sigues intentando ver un toro en mis "dibujos raros". Porque eres especial.

## Resumen

Las distintas aplicaciones en las que se emplean los sistemas de computación de altas prestaciones requieren una alta capacidad de cómputo. En la actualidad sólo se puede alcanzar mediante el empleo de múltiples nodos de cómputo unidos a traves de una red de interconexión. En los últimos años se han propuesto diversas topologías de red para sistemas de HPC con el objetivo de alcanzar 1 Exaflop ($10^{18}$ instrucciones de punto flotante por segundo). En este trabajo vamos a analizar dos de ellas, la red Dragonfly y los toros concentrados.

Las redes Dragonfly se han desarrollado orientadas al uso de routers de alto grado y con el objetivo de mantener un bajo coste de implementación a la par que una alta escalabilidad. Los toros concentrados son una variante de las redes toroidales en las que se incrementa el número de nodos de cómputo unidos al router y el número de enlaces de red.

El objetivo de este trabajo es determinar las prestaciones que cada topología puede ofrecer, y seleccionar el escenario de aplicación en el que su uso es más acertado. Con este fin vamos a realizar un análisis realista de la escalabilidad, coste y rendimiento de ambas topologías para distintos tamaños de red. Para los cálculos más técnicos vamos a tomar como referencia a las tecnologías commodity, de modo que los resultados obtenidos sean alcanzables con tecnologías actuales o disponibles en un futuro próximo.

Este trabajo propone una relación entre los parámetros del toro concentrado que permite garantizar un correcto aprovechamiento del ancho de banda de la bisección. Con este trabajo se aporta una evaluación de la escalabilidad de la red, restringida a casos en los que la red está correctamente dimensionada de modo que el ancho de banda de la bisección no limite el rendimiento. También se desarrolla un estudio del coste de la red, considerando la distribución física de los elementos. Además se proporcionan nuevas herramientas para el análisis del rendimiento y se analizan los resultados obtenidos. En definitiva, el trabajo supone una comparativa entre dos topologías de interés para los sistemas de HPC actuales.

Los resultados obtenidos muestran que los toros concentrados tienen un rendimiento superior y un coste menor a las Dragonflies para tamaños de red intermedios (menos de 10000 nodos de cómputo) empleando routers de alto grado, hasta 64 puertos.

# Índice general

# Índice de figuras

# Capítulo 1

# Introducción y trabajos previos

Desde sus orígenes con la máquina de Babbage [9] la computación ha tenido como uno de los principales objetivos servir como herramienta de análisis y desarrollo en todo tipo de ámbitos de investigación y ciencia. Este objetivo se convirtió por sí mismo en una rama de la computación con la aparición del primer supercomputador, el CDC 6600 (1964) [33]. A esta rama se la denomina HPC: *High Performance Computing* (Computación de Alto Rendimiento).

Los requerimientos en HPC se han incrementado debido a las necesidades de áreas tales como la investigación en Bioingeniería (formación de proteínas), Defensa (criptoanálisis), o la Meteorología (predicción meteorológica). Estas necesidades consisten en mayor capacidad de cómputo, para poder realizar tareas más complejas en menor tiempo. La capacidad actual resulta insuficiente porque no permite afrontar algunas tareas, o el tiempo requerido es inabordable. Por ejemplo, una predicción meteorológica no puede necesitar más tiempo de cálculo que el transcurrido hasta el momento a predecir. La industria ha enfocado dichos requerimientos en la búsqueda de una máquina capaz de ofrecer un rendimiento de 1 Exaflop, que equivale a $10^{18}$ instrucciones de punto flotante por segundo. Dicha búsqueda ha logrado la aparición de máquinas de rendimiento de hasta 20 PetaFlops, en el caso del supercomputador más potente en la actualidad, el IBM Sequoia [4].

Para lograr tan alta capacidad de cómputo, y dado que el rendimiento de un chip de 16 cores como el empleado en Sequoia está en torno a los 200 TeraFlops [4], es necesaria la unión de un gran número de chips para su funcionamiento en paralelo. Las aplicaciones que se ejecutan sobre estos sistemas típicamente dividen la tarea en partes más pequeñas que se puedan ejecutar en paralelo. Esto hace que cada cierto tiempo los diferentes cores que están ejecutando la aplicación tengan que comunicarse entre sí. Para realizar esta comunicación es necesaria una red de interconexión capaz de unir varias parejas de cores de forma simultánea, ofreciendo una alta capacidad de conmutación de datos y con baja latencia. El incremento de las necesidades de cómputo también ha forzado un crecimiento de los requisitos exigidos a la red de interconexión. Con las arquitecturas de los sistemas actuales, las soluciones pasan necesariamente por una mejora tecnológica de los elementos que componen la red de interconexión. Dichas mejoras son costosas en términos de tiempo y dinero, y están constreñidas por diversos límites tecnológicos.

Por este motivo, se están desarrollando nuevas topologías de red que permitan afrontar las necesidades generadas sin incrementar el coste de forma proporcional. En [25], Kim *et al.* proponen una nueva red denominada *Dragonfly* que promete una alta escalabilidad (incremento del número de nodos a comunicar según el número de recursos) con bajo coste.

Otra opción que ya ha comenzado a ser explotada de forma comercial son los denominados *toros concentrados*. Esta topología supone una evolución de otras previas, y

comparte con ellas muchas similitudes. Por este motivo, resulta más fácil de implementar con tecnología ya presente en el mercado. En los últimos años, los sistemas HPC han mostrado una cierta convergencia con otro sector de la computación: los data centers. Un *data center* es un sistema de computación orientado al tratamiento de datos y comprende una rango heterogéneo de aplicaciones, generalmente empresariales: servidores web, servidores de aplicaciones, servidores de almacenamiento de datos.

Dada su motivación, los data centers están fuertemente dominados por la entrada/salida de información. La red de interconexión empleada en los data centers suele emplear soluciones estandarizadas para facilitar la ampliación del sistema, la introducción de elementos redundantes y el remplazo parcial de los elementos que lo componen [14]. De este modo, aumenta el número de sistemas que emplean los mismos componentes, y su precio se reduce por economía de escala. Asimismo se simplifica el diseño del sistema y se facilita la compatibilidad entre distintas soluciones. El empleo de soluciones estándar o de común aceptación por parte de la industria se conoce con el nombre de tecnologías *commodity*.

La convergencia entre data center y sistemas de HPC se basa en el empleo de las mismas redes de interconexión, asegurando que éstas sean capaces de cumplir los requisitos de un gran rango de aplicaciones. En la actualidad existen ciertas soluciones cuyo uso está ampliamente extendido, tales como Myrinet, Infiniband o Gigabit Ethernet. Myrinet es una red desarrollada por la empresa Myricom en 1998 para su empleo en supercomputadores y clusters, capaz de ofrecer baja latencia y alto throughput. Sin embargo, su presencia actual en el mercado es prácticamente simbólica. Infiniband es otra propuesta de red, surgida en 2001 por parte de un consorcio de fabricantes. También tiene como ámbito inicial de aplicación los supercomputadores. Gigabit Ethernet es una evolución del estándar Ethernet empleado en redes de área local. GigE no tiene una orientación tan específica como Infiniband, pero su bajo coste y la compatibilidad con redes Ethernet empleadas comúnmente en la entrada/salida han popularizado su uso. Entre Gigabit Ethernet e Infiniband aglutinan el 87.6 % de las redes empleadas en el Top500. El Top500 [4] es una lista actualizada semestralmente en la que se comparan los sistemas de HPC con mayor capacidad de cómputo del mundo, mediante la ejecución de una serie de programas denominados *benchmarks*.

En este trabajo, se han desarrollado los siguientes objetivos:

- Calcular la relación entre los parámetros del toro concentrado para asegurar un aprovechamiento máximo del ancho de banda de la bisección.
- Determinar la escalabilidad de los toros concentrados y las Dragonflies, calculando el máximo tamaño de red que se puede conseguir para un tamaño de router dado.
- Evaluar los costes de los componentes de la red, tomando como referencia un estudio de costes previo, y desarrollando un esquema con la distribución física de los routers y los nodos de cómputo.
- Realizar una primera aproximación a la tolerancia a fallos de cada topología, a través de la conectividad del router.
- Desarrollar una plataforma de evaluación del rendimiento, tomando como base un simulador de red existente y realizando las modificaciones pertinentes.
- Obtener los resultados del rendimiento alcanzado por cada topología para distintos tamaños de red y evaluar la utilidad de cada una, determinando los escenarios de interés de cada topología.

Para ello esta memoria está dividida en los siguientes capítulos:

- Introducción y trabajos previos. Comienza por una puesta en contexto del estado del arte, para justificar la motivación del trabajo. A continuación se realiza una introducción a los conceptos generales y se explican los aspectos más fundamentales de las redes de interconexión, la arquitectura del router, la topología de red, las técnicas de conmutación, el arbitraje de los recursos, el enrutamiento y control de flujo, el deadlock y los mecanismos para su evitación, y las métricas de rendimiento.

- Estudio de las topologías. En este capítulo se estudian con mayor profundidad las redes Dragonfly y los toros concentrados. Posteriormente se hace un análisis de los parámetros del toro concentrado, para obtener una relación entre éstos que permita obtener un rendimiento óptimo de la red. Seguidamente se estudian la escalabilidad, tolerancia a fallos y coste de cada red.

- Desarrollo de la plataforma de simulación. Aquí se explican las herramientas empleadas para determinar el rendimiento de la red bajo las distintas topologías y tamaños. También se detalla el entorno concreto que se ha simulado, con aspectos como la elección entre distintos enlaces agrupados o las fases de simulación realizadas.

- Resultados. Contiene las gráficas con los resultados de rendimiento fruto del uso de las herramientas explicadas en el capítulo previo. También incluye un análisis aclaratorio de los resultados, relacionando los valores de las gráficas con las causas pertinentes.

- Conclusiones. Supone una recapitulación de los pasos seguidos en el trabajo y condensa los resultados vistos para los distintos objetivos en una justificación de los escenarios de interés para cada topología. Incluye asimismo una explicación de las líneas de trabajo futuro que se han planteado a raíz de los resultados obtenidos.

## 1.1. Definición de red de interconexión

Una red de interconexión es un sistema que permite la comunicación entre diferentes elementos de la arquitectura de un computador. Dicha comunicación se realiza mediante diversos tipos de enlaces que unen los elementos del computador entre sí. El cometido principal de una red de interconexión es asegurar la transmisión de datos entre dos chips o dos elementos de un mismo chip [10]. En el caso de los sistemas HPC, nos centraremos en la comunicación entre distintos nodos, que pueden ser de cómputo (se encargan de ejecutar los programas) o de entrada/salida (permiten la comunicación con el exterior del sistema).

Dos factores clave del rendimiento del sistema son la latencia y el ancho de banda de la memoria. En sistemas actuales gran parte del tiempo de acceso a memoria se debe al retardo de comunicación [10]. Como las redes de interconexión enlazan la memoria con el procesador, el rendimiento de la propia red es relevante en el sistema. La misma situación ocurre en la comunicación entre distintos nodos que se produce en los programas de ejecución paralela, al sincronizar la información disponible entre varios de los chips que ejecutan el programa.

Las redes de interconexión están limitadas porque su escalabilidad es menor que otros componentes del sistema debido a que la capacidad de comunicación demandada crece mucho más rápido que la capacidad de transmisión de los cables de la red [7]. Por tanto, son necesarios mecanismos eficientes que nos permitan aprovechar al máximo los recursos de comunicación disponibles, asegurando al mismo tiempo una entrega de datos con baja latencia.

## 1.2. Tipos de redes de interconexión

Un problema de las redes de interconexión es lidiar con la escasez de recursos. Para realizar comunicaciones de alto rendimiento se podrían establecer uniones punto-a-punto entre cada par de elementos. Sin embargo, como el número de enlaces en la red aumentaría de forma cuadrática con el número de elementos, esta solución no es sostenible ni económica ni físicamente.

La solución más barata que se puede adoptar es comunicar todos los dispositivos mediante un bus. Un bus es un canal físico compartido que transmite las comunicaciones de todos los dispositivos unidos a él. Su mayor inconveniente es que no permite que se comuniquen de forma simultánea más de dos dispositivos, reduciendo la capacidad de transmisión de la red. Se trata de una solución de bajo costo enfocada a sistemas pequeños (comunicaciones dentro del chip como el Front-Side Bus [19], primeras redes de área local [27]) pero que no es apropiada para un sistema de HPC por su falta de escalabilidad, bajas velocidades de transmisión y baja tolerancia a fallos [15].

La falta de escalabilidad se debe a que el ancho de banda de transmisión ha de ser compartido entre todos los dispositivos que se comunican a través de él. Esta compartición implica que el tiempo en que cada dispositivo puede disponer del bus disminuye de forma proporcional al aumento del número de elementos que haya conectados.

Las bajas velocidades de transmisión se deben a que tener muchos elementos conectados al cable aumenta la carga capacitiva de los mismos, y se traduce en un aumento de la latencia. Además, el uso compartido del medio implica un arbitraje para determinar qué elemento ejerce el uso del bus en cada momento. Este arbitraje conlleva una pérdida de tiempo útil de transmisión que también limita la velocidad.

Por último, si un dispositivo muestra un comportamiento anormal impide el resto de las comunicaciones a través del bus. Un fallo en el cable también imposibilita la comunicación entre cualquier conjunto de dispositivos unidos al bus, por lo que la tolerancia a errores es necesariamente muy baja. Hemos visto que el uso de buses en redes de altas prestaciones es bastante ineficiente, por lo que se hace necesario presentar una alternativa. Una solución es el switch o conmutador (en esta memoria se utilizará la palabra original inglesa por ser la más extendida).

Un switch es un circuito digital con múltiples puertos de entrada y salida unidos a enlaces externos, que permite interconectar distintos conjuntos de entradas y salidas de forma simultánea [15] tal y como se puede ver en la figura 1.1.

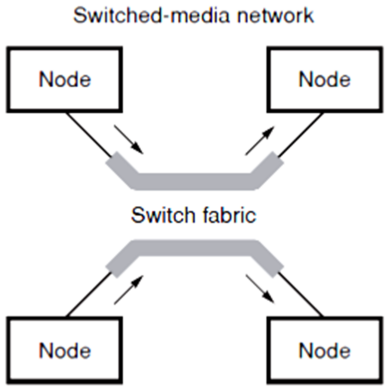



Figura 1.1: Representación del funcionamiento de un switch. Imagen tomada de [8].

El switch sólo es capaz de asignar una salida a una entrada en cada instante de tiempo

y no es posible la comunicación de dos entradas con la misma salida de forma simultánea. Para acumular los datos que no sea posible enviar en un momento dado es necesario disponer de algún tipo de almacenamiento. Este almacenamiento se realiza a través de buffers o colas de almacenamiento, sobre los que hablaremos en la sección 1.6.

El switch permite lograr valores bajos de latencia en la transmisión y es el dispositivo más adecuado para redes de alta velocidad. Sin embargo, el número de dispositivos que un switch permite interconectar está limitado por la tecnología VLSI que utiliza, la cual limita el número de puertos de entrada y salida disponibles [15]. Por causa de estas limitaciones se abandona la idea de un único switch en favor de una red con múltiples switches que se unen entre sí. Dicha red está formada por enlaces punto-a-punto entre distintos nodos, conjuntos de un switch y uno o varios de los dispositivos que se unen.

## 1.3. Arquitectura del router

Al dispositivo de comunicación construido en base a switches se le denominaría router o encaminador. La función del router es determinar un camino hacia el destino de la información, y realizar el envío si se disponen de los recursos necesarios para asegurar la comunicación. Dicho envío se hace efectivo conmutando entre los puertos apropiados mediante el switch.

La arquitectura de un router varía mucho dependiendo de la implementación concreta, con aspectos como la técnica de conmutación, distribución del almacenamiento en buffer o el mecanismo de control de flujo. Aun así, de forma general consta de buffers para almacenar la información recibida a la espera de ser retransmitida; una unidad de enrutamiento y arbitraje que determina la configuración de caminos conmutados que adopta el switch en cada momento; y un switch interno que conmuta entre los puertos de entrada y salida y que denominaremos crossbar. La distribución de estos elementos se muestra en la figura 1.2.

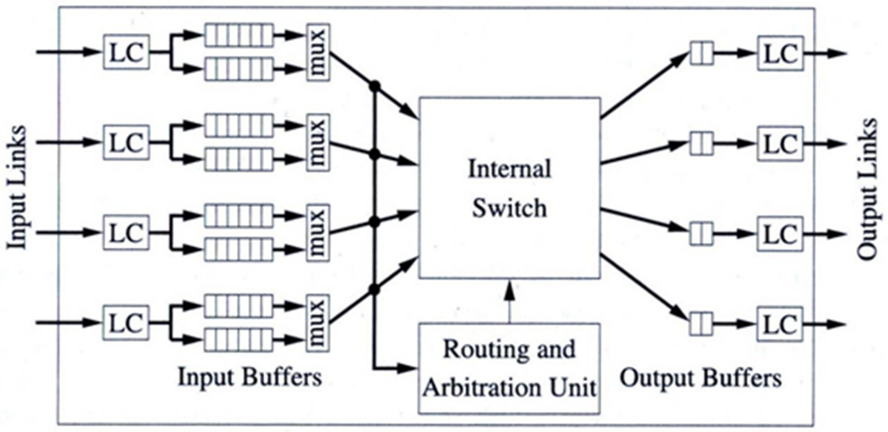



Figura 1.2: Estructura interna típica de un router. Imagen tomada de [15].

En la imagen podemos comprobar la presencia de múltiples buffers para una misma entrada, conectados a un multiplexor. Esta colocación responde al uso de canales virtuales, que permiten realizar una separación del tráfico de un mismo canal físico en varios canales lógicos. Dicha separación permite evitar deadlocks si uno de los canales virtuales se utiliza como escape, asegurando que su encaminamiento garantice la ausencia de deadlock. El uso de canales virtuales no se restringe a la evitación de deadlocks, sino que podría utilizarse

también para diferenciar distintos tipos de tráfico [11] y priorizar algunos tipos sobre otros, y para mitigar el HOL Blocking [32].

## 1.4. Topología de la red de interconexión

Consideramos una red de altas prestaciones aquella en la que se requiere alcanzar un throughput alto de forma sostenida manteniendo al mismo tiempo una latencia baja en la entrega de los mensajes. Tanto el throughput como la latencia son métricas de rendimiento que estudiaremos a lo largo de la sección 1.10. Estas redes tienen como elementos básicos los nodos de cómputo, los encaminadores o routers, y los enlaces que unen a los anteriores.

La forma en que se organizan los enlaces entre los routers constituye la topología de la red. Las topologías de red se clasifican fundamentalmente en redes directas e indirectas. Dicha clasificación se realiza tomando como referente la unión de los nodos de cómputo a los routers. En las redes indirectas, algunos de los routers no están unidos a ningún nodo de cómputo y se dedican únicamente a reenviar la información recibida hacia su destino. Este tipo de redes ha sido tradicionalmente muy popular en HPC, dando lugar a redes jerárquicas en las que los routers se separan en etapas según los routers a los que estén unidos, de forma similar a las estructuras de los árboles.

De los tipos de redes indirectas cabe destacar a las redes de Clos y Beneŝ, las fat-tree y las butterfly. Las redes de Clos disponen de 3 etapas de routers: una de entrada, otra de salida, y una intermedia que no está unida a ningún nodo de cómputo. Este tipo de redes se caracteriza por el número de puertos de entrada y salida ($n$), por el número de switches en la etapa intermedia ($m$) y por el número de switches en las etapas de entrada/salida ($r$). Cada switch de la etapa intermedia tiene un puerto de entrada conectado a cada switch de la etapa de entrada y un puerto de salida conectado a cada switch de la etapa de salida, tal y como se muestra en la figura 1.3. Necesariamente los switches emplean crossbars de $n$ x $m$ puertos en la etapa de entrada, de $m$ x $n$ puertos en la etapa de salida, y de $r$ x $r$ puertos en la etapa intermedia.

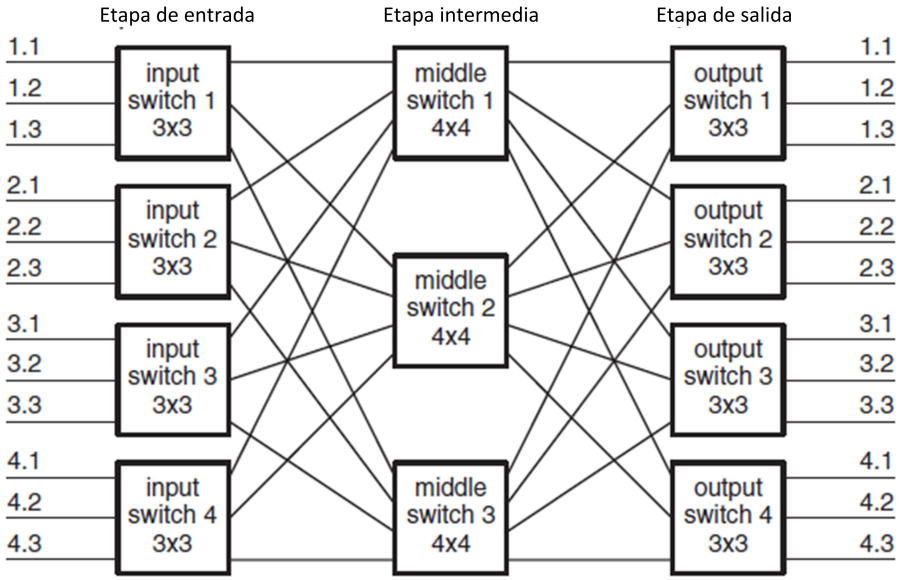



Figura 1.3: Ejemplo de una red de Clos con $m$=3, $n$=3 y $r$=4. Imagen tomada de [10].

Esta configuración permite garantizar una conectividad todos-con-todos entre cada pareja de etapas contiguas, como se aprecia en la figura 1.4. De este modo, se asegura que

cada pareja de nodos de cómputo está unida por una ruta que atraviesa sólo 3 switches. También se garantiza la existencia de $m$ caminos entre cada par de nodos de cómputo, lo que se conoce como "diversidad de caminos" (path diversity).

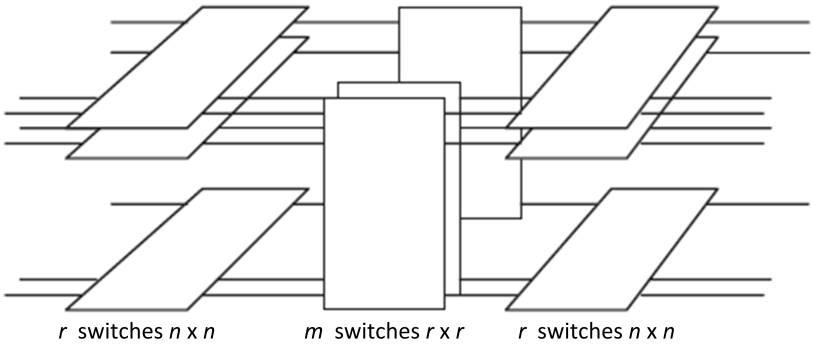



Figura 1.4: Representación de los planos de interconexión entre los puertos de entrada y salida para una red de Clos. Imagen tomada de [10].

Las redes de Beneŝ suponen una variante de las redes de Clos, aplicando el empleo de suficientes etapas intermedias para emplear únicamente switches de 2 x 2 puertos. Ambas redes pueden realizarse mediante switches con puertos bidireccionales, de modo que se produce un plegado o "foldeado" de la red por la mitad, tal y como se ve en la figura 1.5.

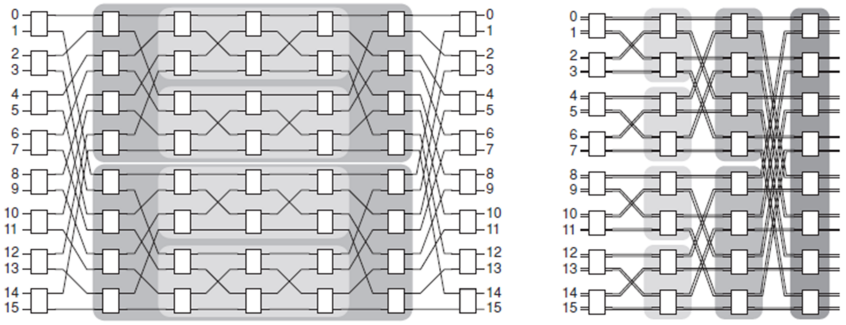


Figura 1.5: Ejemplo de una red de Beneŝ de 16 entradas/salidas y su versión foldeada. Imagen tomada de [20].

De las redes de Clos también se derivan las redes "fat-tree", en las que se emplean más routers por etapa según la cercanía con los nodos de cómputo. Empleando switches de $k$ puertos bidireccionales ($k$ entradas y $k$ salidas), cada etapa reduce el número de switches por un factor $k$. Para que la capacidad de conmutación en cada etapa sea idéntica, se emplean diversas técnicas para aumentar la capacidad de transimisión de cada enlace en la misma medida que se reduce su número. La figura 1.6 muestra con mayor detalle esta red.

Las redes butterfly muestran semejanzas con las anteriores por el empleo de múltiples etapas de switches. Sin embargo, sólo emplean un camino para unir a cada par de nodos de cómputo, por lo que no existe tolerancia a fallos (un único fallo dejaría disconexa una pareja de nodos). Una red $k$-ary $n$-fly requiere $n \cdot k^{(n-1)}$ switches de $k$ entradas y $k$ salidas distribuidos en $n$ etapas para unir $k^n$ nodos de cómputo.

Por el contrario, en las redes directas todos los routers están unidos como mínimo a un nodo de cómputo. Las redes directas más comunes son el anillo, la malla y el toro. En

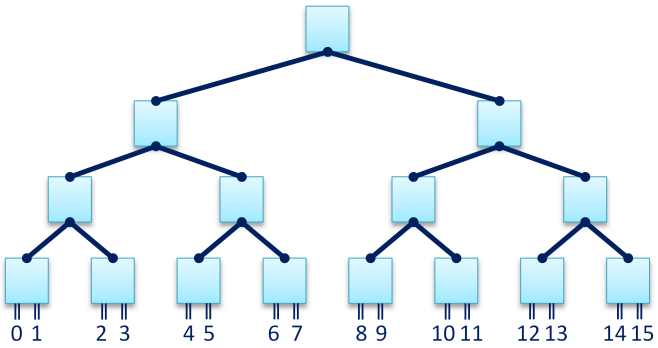



Figura 1.6: Ejemplo de una red fat-tree de 16 entradas/salidas.

un anillo, como su propio nombre indica, los nodos quedan interconectados unos a otros conformando una cadena sin principio ni fin. Es una topología muy sencilla y de bajo coste, pero su rendimiento es bastante limitado.

En una malla, cada nodo tiene múltiples conexiones uno-a-uno con otros nodos vecinos. El número de vecinos se expresa como el doble del número de dimensiones de la malla, a excepción de aquellos nodos situados en la frontera. La topología mallada forma una estructura rectangular o cuadrática dependiendo de su simetría.

El toro es una evolución de la malla en la que los nodos frontera de la red en cada dimensión están conectados entre sí, a través de unos enlaces periféricos. También puede verse como el producto cartesiano de varios anillos, tantos como dimensiones tenga el toro.

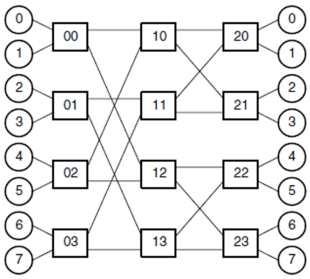



Figura 1.7: Ejemplo de una red butterfly 2-ary 3-fly. Imagen tomada de [10].

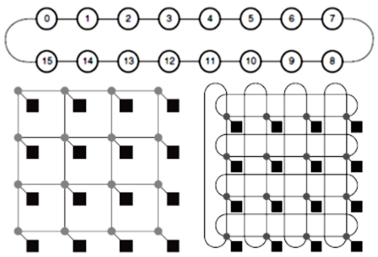


Figura 1.8: Topología en anillo, malla y toro 2D con 16 nodos, respectivamente. Imágenes tomadas de [10] y [20].

## 1.5. Técnicas de conmutación

Un apartado fundamental de las redes de interconexión basadas en switches es la técnica de conmutación utilizada, que define la manera en que se establecen las conexiones entre los recursos de red. Las técnicas más habituales se fundamentan en la conmutación de paquetes en detrimento de la conmutación de circuitos; en la conmutación de circuitos se asigna una conexión hasta que finaliza la comunicación y en la conmutación de paquetes no. Los paquetes son pequeños conjuntos de información en los que se dividen los mensajes para poder compartir el ancho de banda de la red.

La popularidad de la conmutación de paquetes se debe a que permite el uso de TDM (Time Division Multiplex, Multiplexación por División de Tiempo) estadístico logrando comunicaciones más eficientes cuando los paquetes se transmiten de forma intermitente.

TDM es una técnica que permite ocupar un canal de transmisión a partir de distintas fuentes.

La transmisión intermitente de información es una situación habitual en las comunicaciones en el ámbito de computación; por ejemplo, un procesador pide información de la memoria cada cierto tiempo, pero no de forma continua. Estos paquetes pueden ser de longitud fija o variable; más adelante observaremos las condiciones concretas en que se han realizado las pruebas de evaluación.

Existen dos técnicas fundamentales de conmutación de paquetes: Store & Forward y Cut-Through. Store & Forward es más simple, consistiendo en utilizar el enlace de unión entre dos nodos para el envío de un paquete. Cuando el paquete ha sido completamente recibido, el nodo receptor comprueba cuál es su destinatario y lo reenvía al nodo determinado por el algoritmo de enrutamiento. En caso de ser él mismo el destinatario del paquete, directamente lo consume. Este consumo representa una abstracción del comportamiento de la arquitectura por encima del router: puede ser que éste pase el paquete al core o al controlador de memoria que está directamente conectado al router.

Esta técnica es muy fácil de entender e implementar, y permite aprovechar la máxima capacidad de enlaces que tengan distinta tasa de transmisión porque están completamente desacoplados. Su limitación es que no es demasiado eficiente, porque aumenta considerablemente el retardo en la entrega del paquete en su destino respecto a otras técnicas. Además requiere buffers relativamente grandes, lo suficiente como para poder alojar el paquete por completo.

Dado que un requerimiento clave en sistemas de alto rendimiento es lograr baja latencia en la red, Store & Forward no resulta una técnica demasiado apropiada. La conmutación Cut-Through combate esas desventajas reenviando el paquete hacia el siguiente nodo en cuanto se han recibido los datos relativos al destino del paquete (como se ve en la figura 1.9). De este modo reducimos la latencia en la entrega del paquete respecto a la técnica Store & Forward.

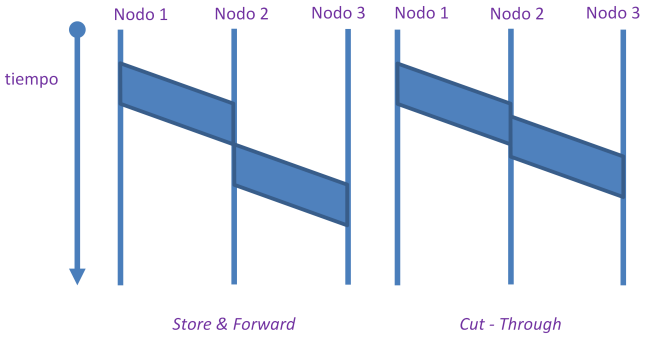



Figura 1.9: Diferencia en la transmisión de un paquete bajo las técnicas de conmutación Store&Forward y Cut-Through.

Dentro de dicha técnica se conocen dos variantes, Virtual Cut-Through y Wormhole. La diferencia entre ambas se basa en el control de flujo que realizan, la asignación que cada una de ellas hace de los recursos de la red.

En wormhole este control de flujo se realiza en fragmentos más pequeños que el paquete denominados flits (flow-control units, unidades de control de flujo), que nos permiten disminuir el tamaño de los buffers necesarios. El principal inconveniente radica en que en caso de congestión el paquete queda detenido en varios nodos a la vez, lo que puede conducir a una saturación prematura de los enlaces. Esta saturación y sus efectos (tales como HOL Blocking) se pueden reducir mediante el uso de canales virtuales (como se

verá en el apartado 1.9) pero éstos utilizan mayor número de buffers. Este aumento se traduce en un incremento de los recursos necesarios en el router.

En la variante Virtual Cut-Through (VCT) el control de flujo se realiza a nivel de paquete [22]. Esta característica hace necesario el uso de buffers más grandes porque es necesario tener hueco para poder recibir paquetes completos, pero evita que un paquete pueda quedar bloqueado en más de un nodo a la vez.

Aquí conviene distinguir dos unidades distintas: el flit y el phit. Un phit (physical transfer unit, unidad de transferencia física) es la cantidad de información que puede transmitir un enlace en un ciclo de reloj. Por tanto, podemos tener paquetes de varios phits, y un flit también puede estar formado por varios phits Un paquete estará compuesta de varios flits sólo si la técnica de conmutación es wormhole; en Virtual Cut-Through es en sí una única unidad de flujo.

Virtual Cut-Through ha obtenido relevancia en productos comerciales porque permite mecanismos de evitación de bloqueos y arbitraje más sencillos, y por tanto routers más simples de implementar manteniendo un alto rendimiento. Sin embargo, wormhole ha tenido gran aceptación en el diseño de algunos tipos de redes de interconexión en las que se buscase minimizar los recursos en área, especialmente en las redes-en-chip [21, 34, 17]. Esta aceptación se debe a que las exigencias de memoria en wormhole son inferiores, aun a pesar de que las tecnologías actuales permiten implementar buffers de longitud relativamente grande dentro del chip.

En este trabajo hemos decidido centrarnos únicamente en el uso de la técnica Virtual Cut-Through, dado que es adecuada para redes de alto rendimiento y reduce el número de enlaces bloqueados por la contención entre paquetes. Esta decisión se ve reforzada por el empleo de la técnica Virtual Cut-Through en la mayoría de sistemas presentes en el Top500, mientras que el uso de wormhole está actualmente relegado al empleo en redes on-chip en las cuales el área ocupado por los buffers supone una restricción de diseño crítica [29].

La técnica de conmutación elegida tiene un fuerte impacto en la organización de los componentes del router, principalmente los buffers y el esquema de arbitraje empleado. La organización de los buffers dentro del router es una decisión de diseño que afecta en múltiples niveles al comportamiento de la red, y la elección del arbitraje influye sobre el coste económico y la latencia lograda.

## 1.6. Organización de los buffers dentro del router

Los buffers pueden estar situados a la entrada del router, a la salida, en ambos lados, o de forma centralizada. Una de las ventajas de que los buffers estén a las salidas es que se elimina completamente el denominado Head-Of-Line Blocking (HOL, Bloqueo por Cabecera de Línea). El HOL Blocking se produce cuando en una cola en la que hay almacenados varios paquetes, el que está en cabeza queda bloqueado porque su salida está ocupada e impide avanzar a otros paquetes que sí podrían salir. Situar los buffers a las salidas evita esta situación porque todos los paquetes de la misma cola tienen el mismo destino inmediato, de modo que la posibilidad de avance es la misma para todos ellos. La aparición de HOL Blocking disminuye el rendimiento de la red, ya que supone paradas en el trayecto hacia el destino del paquete.

No obstante, situar las colas a las salidas impide que un paquete con varias salidas posibles pueda cambiar de buffer en caso de bloqueo, siendo perjudicial para la adaptatividad del router. Asimismo, para evitar una posible pérdida de datos las colas deben estar preparadas para recibir de forma simultánea paquetes de todos los puertos de entrada. Esta necesidad supone unos requerimientos de tamaño de memoria y especialmente

velocidad de escritura que pueden resultar demasiado exigentes.

Situar los buffers en las entradas del router disminuye estas exigencias de memoria, pero implica la posible aparición de HOL Blocking. En el caso de las memorias más utilizadas en este esquema, de tipo FIFO (First In, First Out, el primero en entrar es el primero en salir) sólo puede avanzar un paquete de forma simultánea, de modo que el HOL Blocking se hace más probable. La aparición de HOL Blocking suele mitigarse mediante el uso de canales virtuales (tal y como veíamos en el apartado 1.3).

Utilizar buffers a la entrada y salida del router minimiza el HOL Blocking y la necesidad de escribir en los buffers a mayor velocidad, y además independiza al crossbar del enlace externo. Sin embargo, a pesar de su uso extendido en sistemas comerciales, es una solución bastante costosa (requiere el uso de muchos más buffers) y que ocupa mayor espacio en área.

Por último, el uso de buffers centralizados está fundamentalmente asociado a la técnica de conmutación wormhole y permite asignar memoria dinámicamente según la carga de cada enlace, en lugar de tener buffers de longitud fija.

En nuestro caso, optaremos por un esquema de almacenamiento a la entrada del router, para tener unos requerimientos de memoria menos exigentes. Por este motivo, será necesario tener en cuenta la aparición de HOL Blocking.

## 1.7. Esquema de arbitraje

Del mismo modo que los buffers pueden estar organizados de distinta forma dentro del router, existen diversas formas de realizar el arbitraje. Ante todo, hemos de diferenciar claramente entre árbitro y allocator. El árbitro es un elemento que se encarga de asignar el acceso a un recurso concreto a uno de los elementos que compiten por su uso; por ejemplo la salida por un puerto concreto, atravesando el crossbar. El allocator realiza una asignación entre un grupo de recursos y otro de solicitantes que pueden realizar una o varias peticiones, para lo cual típicamente estará compuesto por un conjunto de árbitros. Es decir, el allocator considera todas las entradas y salidas.

En la sección 1.9 veremos la explicación del uso de *canales virtuales*, pero por el momento comentaremos que supone tener en las entradas múltiples buffers. Éstos compartirán un único acceso al crossbar por entrada, porque la complejidad del crossbar depende del número de entradas y salidas entre las que tiene que conmutar. La función del allocator en este contexto será asignar el acceso al crossbar a uno de los buffers de cada entrada, y una salida a cada entrada.

Existen tres grandes tipos de allocators: centralizados, separables y paralelos. En el caso de los allocators centralizados reciben las peticiones de todos los buffers situados en las entradas a la vez, y generan un esquema de asignación global para todo el crossbar. Al esquema de asignación le llamaremos "máximo" si no es posible realizar ninguna otra conexión entrada-salida. Este sistema permite realizar asignaciones que aprovechen el uso de los recursos de forma eficiente, pero penaliza el tiempo invertido en realizar la asignación. Dally afirma en [10] que dicho tiempo tiene un grado de escalabilidad de $O(p^{2,5})$ para un crossbar de $p$ puertos, por lo que resulta irrealizable para crossbars con un número elevado de entradas y salidas.

Los allocators paralelos consiguen realizar el esquema de asignaciones en mucho menos tiempo. Para ello resuelven de forma paralela las distintas peticiones de asignación de recursos, permitiendo que la asignación pueda requerir diversas iteraciones. Sin embargo, son poco escalables, porque para asegurar que se realiza asignación por ciclo de reloj es necesario utilizar tantos allocators como iteraciones tenga que dar cada uno de ellos. El último tipo que nos queda por conocer es el separable allocator (podría entenderse como

allocator "divisible"), en el que la asignación se divide en dos etapas. En la primera etapa se utiliza un conjunto de árbitros a las entradas para conceder el acceso al crossbar a uno de los buffers de cada entrada. En la segunda etapa se dispone de árbitros a las salidas para conceder el uso del enlace a una de las entradas. Este allocator será el que utilicemos en este trabajo, y cuando detallemos la arquitectura elegida en nuestra simulación (sección 3.2) veremos este aspecto con mayor profundidad.

## 1.8. Enrutamiento

El enrutamiento es la elección del camino empleado a través de la red para transmitir un paquete entre su origen y el destino. Dicha elección puede ser estática o dinámica. Un enrutamiento *estático* es aquel que define el trayecto en el momento en que se inyecta el paquete en la red. Un enrutamiento *dinámico* permite emplear diferentes caminos para que el paquete llegue a su destino, cambiando la ruta de forma dinámica en función del estado de los buffers del destino inmediato en cada avance. El uso de adaptatividad realiza un balance de la carga de tráfico entre los enlaces, puesto que elige nodos con almacenamiento disponible si la primera opción de avance del paquete no está lista para recibirle. Este balance de la carga suele traducirse en un aumento del throughput y una disminución de la latencia, puesto que facilita un mejor aprovechamiento de los recursos e intenta evitar las detenciones de un paquete durante su trayecto. También es capaz de afrontar la aparición de fallos en los enlaces ya que permite el uso de rutas alternativas.

Al margen de su topología, los routers de una red conmutada han de realizar siempre 3 tareas claves para asegurar la entrega del paquete: enrutamiento, arbitraje y switching (conmutación) [20]. El enrutamiento es la fase en la que se decide el camino (o caminos) que puede seguir el paquete hacia su destino. Si se trata de un enrutamiento mínimo, la longitud del camino será la mínima de entre todas las posibles rutas entre el nodo origen y el nodo destino.

Con el objeto de aliviar la congestión y mejorar el rendimiento de la red, existen hoy en día numerosos algoritmos de enrutamiento que permiten la elección de caminos no mínimos, lo que se conoce con el nombre de "misrouting". Uno de sus inconvenientes es que pueden producirse livelocks, situaciones en las que los paquetes nunca llegan a su destino aunque siguen avanzando por la red [10]. En este trabajo contemplaremos únicamente algoritmos mínimos por ser más sencilla su implementación y permitir ejecuciones más rápidas a igualdad de recursos de la red.

De entre los algoritmos de enrutamiento mínimos empleados en mallas y toros, uno de los más comunes es el conocido como Enrutamiento por Orden de Dimensión (Dimension-Order Routing, DOR) [13]. En el algoritmo DOR, se ordenan las dimensiones de la topología, de modo que un paquete intenta siempre agotar el camino en la dimensión más prioritaria antes de avanzar por otra. Se trata de un algoritmo estático (en contraposición a los adaptativos descritos previamente) puesto que sólo permite un único camino entre cada par de nodos origen y destino.

## 1.9. Control de flujo y evitación de deadlocks

Un problema al que se enfrentan los algoritmos de enrutamiento es evitar la presencia de "deadlocks". Un deadlock es un bloqueo permanente de los paquetes en la red debido a la espera de recursos que nunca se liberan y que impide que los paquetes lleguen a su destino [20]. Dichos bloqueos se forman debido a la generación de un ciclo de dependencias entre varios nodos de la red que sean comunes a varios caminos. La existencia de deadlocks

se debe a la presencia de un número finito de recursos en la red, por lo que la probabilidad de aparición aumenta con el tráfico que intenta atravesarla.

Para simplificar la explicación, contemplaremos un caso típico de deadlock en mallas (figura 1.10) que puede ser fácilmente extrapolado a un toro. Los paquetes viajan del nodo origen $s_x$ al nodo destino $d_x$. Como podemos comprobar, los paquetes con destino $d_1$, $d_2$, $d_3$ y $d_4$ se bloquean entre sí, impidiendo que ninguno de ellos avance y bloqueando a otros paquetes como por ejemplo los de $s_5$.

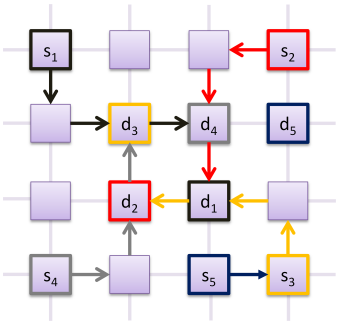



Figura 1.10: Formación de un deadlock en una malla 2D debido a una dependencia cíclica. Basado en imagen tomada de [20].

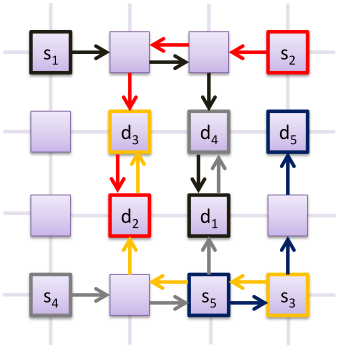



Figura 1.11: Representación del avance de los paquetes con DOR. Basado en imagen tomada de [20].

Para combatir los efectos adversos del deadlock existen dos estrategias: subsanarlo cuando se produce o evitarlo. En la primera estrategia, se implementan mecanismos de detección de la existencia de deadlock y se eliminan paquetes de los recursos afectados para disolverle cuando aparece. En la segunda estrategia (evitación de deadlock) se evita la elección de rutas que puedan provocar la formación de deadlock.

La estrategia de recuperación de deadlock encuentra su utilidad en redes en las que la pérdida de un paquete no sea crítica (como por ejemplo Ethernet). En dichas redes es necesario el uso de mecanismos de control de flujo extremo a extremo y recuperación de errores, para que la pérdida de paquetes no implique pérdida de datos (por ejemplo, en Ethernet se utiliza el protocolo TCP para asegurar una entrega libre de errores).

Sin embargo, en redes para sistemas de HPC pueden existir paquetes relativos a peticiones de datos que en caso de pérdida provocarían un bloqueo infinito de la operación. Este bloqueo es extremadamente grave y podría incluso causar la caída del sistema. Por este motivo, los algoritmos de evitación de deadlock son los más extendidos en este tipo de redes. Estos algoritmos se integran en el uso de mecanismos de control de flujo, que permiten el control del estado de los recursos para realizar una asignación de los mismos. El control de flujo posibilita así un envío de paquetes en el que no se produzca saturación ni paradas innecesarias.

Existen múltiples implementaciones de mecanismos de control de flujo, pero prácticamente todas se fundamentan en dos variantes: Stop&Go y uso de créditos. La diferencia entre ambos radica en la forma en que controlan el estado de los buffers de los nodos adyacentes. Stop&Go realiza este control mediante mensajes o señales exclusivas, mientras que el uso de créditos consiste en mantener un registro del número de huecos o posiciones libres en los buffers de los nodos adyacentes. Así, cuando enviamos un paquete disminuimos dicho número de créditos, y cuando el paquete es extraído del buffer en el nodo vecino, éste informa al anterior de que debe aumentarlo. Este estado de los recursos puede actualizarse mediante piggybacking, envío de información de estado dentro de la cabecera de paquetes de información. A lo largo de este trabajo nos centraremos en el segundo sistema, que

proporciona mayor rendimiento.

Una ventaja de las topologías malladas frente a otras como los toros o los anillos es que no tienen una estructura cíclica en cada una de sus dimensiones individuales, de modo que los ciclos pueden evitarse más fácilmente. La solución libre de deadlock más habitual en mallas es emplear el algoritmo DOR antes descrito. Por ejemplo, en el caso de deadlock ejemplificado anteriormente, se comportaría como la representación de la figura 1.11. En este caso, se evita el deadlock, y todos los paquetes consiguen llegar a su destino.

En los toros esta solución no garantiza la ausencia de deadlock, puesto que la propia topología tiene una estructura cíclica. Para eliminar este inconveniente, las soluciones empleadas impiden que los paquetes transiten formando dicha estructura.

Una solución es separar el tráfico según su distancia recorrida en distintas clases que viajarán por canales virtuales diferentes [18]. Los paquetes cambian en cada router de canal virtual avanzando hacia una clase inmediatamente superior. Tal y como muestra la figura 1.12 de este modo nunca se produce una interdependencia entre elementos situados en el mismo anillo.

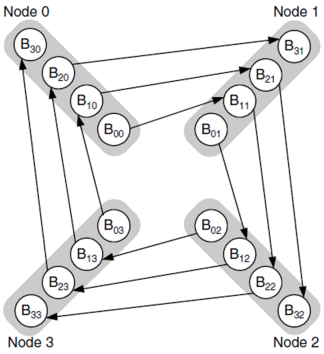



Figura 1.12: Empleo de la prevención de deadlock mediante división de los recursos en clases sobre un anillo de 4 nodos. Imagen tomada de [10].

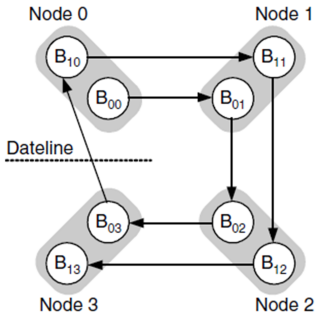



Figura 1.13: Empleo de la prevención de deadlock mediante dateline sobre un anillo de 4 nodos. Imagen tomada de [10].

Pese a impedir la aparición de deadlocks, este sistema no resulta conveniente debido a que requiere un número muy alto de canales virtuales, tantos como el diámetro de la red. Para reducir el número de recursos necesarios existe una solución comúnmente extendida que aprovecha la topología del toro: el dateline.

El dateline consiste en dividir el tráfico que atraviesa un anillo de la red en dos clases diferentes, tal y como se ilustra en la figura 1.13. Se toma uno de los enlaces del anillo como referencia, y al atravesar dicho enlace se realiza un cambio de clase. A cada clase se le asigna un canal virtual diferente. De este modo se produce una ruptura de las dependencias cíclicas de la red.

En el toro cada dimensión está formada por un anillo. Si en cada anillo/dimensión empleamos la división en clases según el dateline, el toro se convierte en una malla a efectos de uso de recursos. Empleando enrutamiento DOR podemos asegurar que no se produce dependencia cíclica entre paquetes que atraviesen distintas dimensiones, y el resultado es una red libre de deadlock.

Existe otra posibilidad de evitar el deadlock empleando canales virtuales pero permitiendo realizar un enrutamiento adaptativo: el protocolo de Duato. En [12] Duato demuestra que si existe un camino de escape libre de deadlock, es posible encaminar los paquetes de forma completamente adaptativa. En el momento en que un paquete no puede avanzar de forma adaptativa, se envía a través del camino de escape.

La elección de dicho camino de escape depende de la implementación del algoritmo. Uno de ellos es el basado en el control de flujo Burbuja [31], presente en algunos sistemas comerciales [5]. Este mecanismo combina un enrutamiento DOR y un control de flujo con evitación de deadlock mediante un algoritmo denominado *burbuja*.

El algoritmo Burbuja impide la formación de deadlocks restringiendo la inyección y el cambio de dimensión de los paquetes: los paquetes siguen un avance Cut-Through cuando avanzan en la misma dirección, y solicitan hueco para un paquete extra si se quiere realizar un cambio de dimensión.

Este algoritmo se basa en dos conceptos clave: usando enrutamiento DOR prevenimos la aparición de dependencias cíclicas entre diferentes dimensiones, pero puede producirse deadlock entre paquetes que avanzan en la misma dimensión, dentro de un anillo. Si garantizamos el flujo constante de paquetes a lo largo del anillo los paquetes terminarán llegando a su destino. Garantizando la entrega de paquetes dentro del anillo, éstos serán retirados de la red y se evitará el deadlock porque no se llenarán todas las colas.

Para garantizar ese flujo de paquetes basta requerir que todos los paquetes que no van a seguir avanzando en esa dimensión necesiten hueco para dos paquetes en el siguiente salto. De este modo, estamos restringiendo la inyección de paquetes procedentes de otra dimensión. Así garantizamos que los paquetes que sigan avanzando a través del mismo anillo tengan prioridad, ya que liberan un hueco en esa dimensión y con ello favorecen el tránsito. El uso de este mecanismo burbuja previene la formación del deadlock dentro de un anillo, por lo que permite su aplicación sobre anillos o toros.

## 1.10. Métricas de rendimiento

Una herramienta indispensable para poder trabajar en cualquier campo es el uso de métricas de rendimiento y comparativas de propiedades. Ellas nos indicarán las ventajas de cada red y nos permitirán determinar su campo de uso más adecuado. Una de las comparativas más comunes está relacionada con la propia topología de la red, y es la cota de tráfico establecida por el *ancho de banda de la bisección*. Éste se define como el menor ancho de banda o cantidad de datos que pueden enviarse, del total de enlaces situados en la unión de todos los posibles cortes que dividen a la red en dos partes iguales, y se expresa usualmente en bits/segundo. A lo largo de la memoria denotaremos el ancho de banda de la bisección por $BBW$.

La topología de red también determina otras características, como la distancia media, el diámetro o la conectividad. La distancia media es el número de nodos que en media atraviesa un paquete hasta llegar a su destino. El diámetro de la red es la máxima distancia en ciclos que existen entre dos nodos de la red, y tiene especial importancia en redes en las que se requieran varios ciclos para atravesar algunos de los enlaces. La conectividad es el número mínimo de enlaces que han de ser desconectados para impedir que un nodo de cómputo envíe y reciba mensajes del resto de la red. Esta propiedad refleja el máximo número de fallos que se puede tolerar sin aislar ningún nodo.

Cuanto más alto sea el $BBW$, mayor capacidad tendrá de soportar cargas de tráfico altas y teóricamente mejor será su rendimiento. En la realidad esto no siempre se cumple, puesto que existen más limitaciones que pueden invalidar la comparativa. En el caso del toro concentrado todos los enlaces tienen la misma capacidad y la topología es toroidal. Por estos motivos, el corte que menor número de enlaces atraviesa está situado en el centro de la red, a lo largo de todo el eje X ó Y (la distribución de estos ejes se puede observar en la figura 1.14). Así, el ancho de banda de la bisección se reduce al producto de la capacidad de cada enlace por el cociente entre el número total de nodos y el número de nodos para

una misma dimensión. A este número de nodos en cada dimensión se le conoce como raíz de la red. A lo largo de esta memoria vamos a considerar topologías simétricas, con el mismo valor para el sentido de entrada que para el de salida. Esta situación se produce en muchas redes, por lo que constituye un caso genérico.

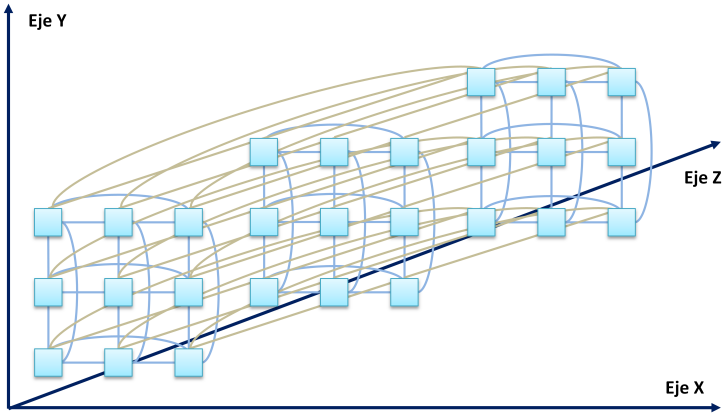



Figura 1.14: Representación de los ejes X, Y y Z sobre un toro 3D.

En cuanto a métricas de rendimiento, principalmente son dos: el throughput y la latencia. El throughput es la tasa de datos en bits por segundo (o en phits por nodo y ciclo) que admite la red. Por su parte, la latencia mide el tiempo transcurrido desde que un paquete es inyectado en la red hasta que llega a su destino. Esta última métrica puede ser más interesante en redes que necesiten entregar los paquetes rápidamente, mientras que el throughput hace referencia a la cantidad de datos que pueden enviarse de forma sostenida.

# Capítulo 2

# Estudio de las topologías

En el capítulo 1 de introducción hemos visto varios tipos de topologías de red, tanto directas como indirectas. Las primeras se ven favorecidas por la aparición de routers comerciales de alto grado y gran capacidad de conmutación que permiten unir al router con un gran número de routers vecinos y con varios nodos de cómputo de forma simultánea. Dado que el coste de dichos routers no es excesivamente elevado y que se requiere un menor número de routers y de enlaces frente a las redes indirectas, en el ámbito de los sistemas HPC las más extendidas son las topologías de red directas.

No obstante, las redes directas vistas en la sección 1.4 sufren una serie de limitaciones. Estas limitaciones se deben a una escalabilidad limitada y a un coste que crece a un ritmo superior al número de nodos de cómputo con los requisitos exigidos. El incremento en la capacidad de cómputo requerida implica aumentar el número de nodos de cómputo de la red. Asimismo crece el número de routers necesarios para comunicarlos entre sí.

Para aumentar el tamaño de la red existen dos opciones básicas: aumentar el número de nodos por dimensión, y aumentar el número de dimensiones. La primera solución aumenta la latencia en la red y reduce el ancho de banda disponible para cada nodo de cómputo. La segunda opción aumenta el grado del router (número de puertos de entrada/salida) necesario. Como la implementación física se debe realizar necesariamente en 3D, algunos nodos vecinos van a estar físicamente alejados, aumentando la longitud promedio de los cables.

Ambas soluciones requieren una mejora de la tecnología disponible. Al aumentar el tamaño de las dimensiones, es necesario incrementar el ancho de banda de los cables y puertos del router para que el rendimiento de las comunicaciones de cada nodo de cómputo permanezca invariable. Por otro lado, aumentar el grado del router implica desarrollar crossbars con capacidad de conmutar un mayor número de puertos. También afecta al número de árbitros y de buffers que se necesitan.

En este capítulo vamos a hacer un estudio y comparativa en términos de escalabilidad, conectividad y coste de las redes Dragonfly frente a los toros concentrados. Para ello, en la sección 2.1 introduciremos las redes Dragonfly y los parámetros que las caracterizan mientras que en la sección 2.2 se considerarán en estos mismos términos los toros concentrados, calculando sus paránetros en la sección 2.3. Seguidamente, en las siguientes secciones 2.4, 2.5 y 2.6 se realiza la comparativa de las topologías introducidas previamente.

## 2.1. Redes Dragonfly

Para combatir las limitaciones de la tecnología (incremento del ancho de banda de los cables, crossbar con mayor número de puertos, aumento de la complejidad del router) y alcanzar tamaños de red mayores se han propuesto algunas topologías novedosas. De entre

estas, cabe destacar las redes Dragonfly, por estar enfocadas a obtener un alto rendimiento y escalabilidad manteniendo un bajo coste [25]. Se trata de una red directa jerárquica con dos niveles, orientada al uso de routers de alto grado.

Los routers de la red se encuentran divididos en varios grupos. El primer nivel de la jerarquía está constituido por los routers dentro de cada grupo. A cada router se unen uno o varios nodos de cómputo. El segundo nivel de la jerarquía corresponde con los distintos grupos, que a su vez están directamente interconectados mediante enlaces *globales*. Estos enlaces globales están distribuidos entre los routers del grupo: cada router del mismo grupo tiene uno o varios enlaces globales a routers de otros grupos.

Las redes Dragonfly pueden emplear diferentes topologías dentro del grupo y entre distintos grupos. En este trabajo nos vamos a restringir al uso de grafos completos en ambos niveles. Esta decisión implica que los routers de un mismo grupo están interconectados todos entre sí de forma directa mediante los denominados enlaces *locales* o cortos, y que los grupos también están unidos todos con todos mediante los enlaces globales.

La red se caracteriza mediante los siguientes parámetros:

- $k$: grado del router (número de puertos de entrada/salida).
- $p$: número de nodos de cómputo unidos a cada router.
- $a$: número de routers por grupo.
- $h$: número de enlaces globales por cada router.
- $g$: número de grupos.
- $n$: número de routers en la red.
- $N$: número de nodos de cómputo en la red.

A partir de los parámetros $p$, $a$ y $h$ podemos determinar el valor de los restantes según las siguientes relaciones:

$$g = a \cdot h + 1 \tag{2.1}$$

$$n = a \cdot g = a \cdot (a \cdot h + 1) \tag{2.2}$$

$$N = p \cdot n = p \cdot a \cdot (a \cdot h + 1) \tag{2.3}$$

$$k = p + h + (a - 1) \tag{2.4}$$

Si bien la red tiene 3 grados de libertad ($a$, $p$ y $h$) para que el uso de los enlaces locales y globales esté correctamente balanceado es necesario establecer una dependencia entre ellos. Empleando un enrutamiento mínimo un paquete va a tener un diámetro 3, dado que el mayor camino de entre todas las rutas mínimas requiere 3 saltos. Dicho camino comprende un salto local entre el router origen y otro router del mismo grupo conectado al grupo destino, un salto global hacia el grupo destino, y un salto local hacia el router destino. Para que el balance del tráfico entre enlaces locales y globales sea equitativo se debe cumplir que el número de enlaces locales de cada router es el doble que el de globales ($(a - 1) = 2h \Rightarrow a \approx 2h$). En caso contrario, alguno de los dos tipos de enlaces es más restrictivo que el restante y limita el rendimiento de la red.

Además, el número de enlaces globales por cada router debe ser igual que el número de nodos de cómputo porque cada paquete emplea un salto global para alcanzar su destino. Esta restricción fuerza la relación $(a - 1) = 2p \Rightarrow a \approx 2p$ para que no haya oversubscription. La *oversubscription* se produce cuando se genera más tráfico en los nodos de cómputo

del que los enlaces de red pueden transmitir. Si la capacidad de almacenamiento es ilimitada, la oversubscription implica que la latencia crece hasta el infinito. Si planteamos un modelo más realista con tamaño finito de los buffers la oversubscription impedirá que la tasa de tráfico generado sea la esperada, porque se saturan los buffers de inyección. Una red oversubscribed tiene unos costes menores, pero se convierte en el cuello de botella del rendimiento del sistema.

La figura 2.1 ilustra un caso de red Dragonfly balanceada, en la que se cumple $a = 2p = 2h$. La consideración de red balanceada es muy útil porque permite que haya un enlace global por cada nodo de cómputo ($p = h$) y el ancho de banda de la bisección sea siempre proporcional al número total de nodos.

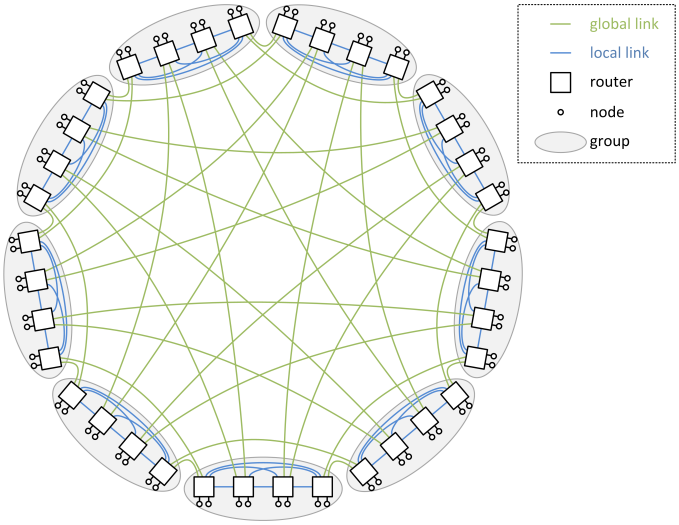



Figura 2.1: Red Dragonfly balanceada con $h = 2$. Imagen tomada de [16].

Los cálculos (2.1) están realizados para una red máxima a partir de un router dado. La topología Dragonfly contempla la posibilidad de tener un número de grupos inferior al máximo posible a partir de un router dado. En este caso, un router puede tener varios enlaces globales al mismo grupo. A lo largo del trabajo supondremos siempre que la red es máxima salvo que se afirme expresamente lo contrario. Cuando la red Dragonfly tenga el máximo número de grupos posible la denominaremos *densa*.

## 2.2. Toros concentrados

Otra topología que se ha propuesto para su aplicación en supercomputadores es la denominada *toros concentrados*. Esta topología supone una evolución de los toros vistos en el apartado 1.4. Dicha evolución se basa en dos cambios fundamentales: unir cada router con varios nodos de cómputo, y realizar una agrupación de los enlaces de red. Dicha agrupación (también conocida como *port link aggregation* o *trunking*) consiste en replicar cada enlace de la topología en varios cables físicos que unen el mismo par de routers. El número de cables que corresponden al mismo enlace es el mismo para todos los enlaces y se define como *factor de trunking*. La figura 2.2 muestra un ejemplo de toro concentrado.

La notación seguida para describir a esta red es la siguiente:

- $k$: grado del router (número de puertos de entrada/salida).
- $p$: número de nodos de cómputo unidos a cada router.

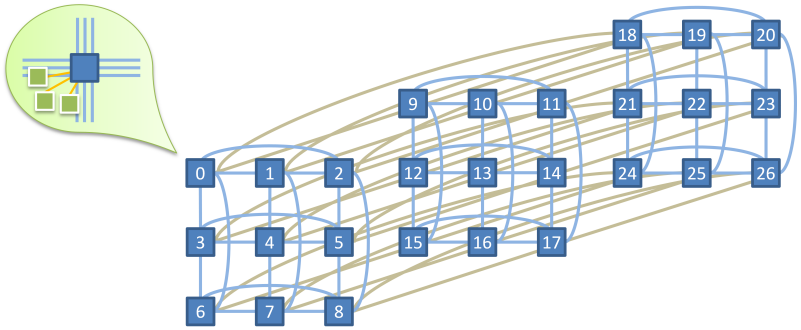



Figura 2.2: Toro concentrado de 3 dimensiones y longitud 3 routers por cada dimensión. El detalle muestra un factor de trunking $t = 3$ y $p = 3$ nodos de cómputo por router.

- $r$: número de routers por dimensión (tamaño de la dimensión del toro). Se considera que los toros son cuadrados, con igual número de routers por cada dimensión.
- $D$: número de dimensiones del toro;
- $t$: número de cables físicos por enlace topológico de red (factor de trunking).
- $n$: número de routers en la red.
- $N$: número de nodos de cómputo en la red.

Para analizar las ventajas e inconvenientes de estas redes, consideraremos su escalabilidad, coste y rendimiento. La *escalabilidad* indica la capacidad de crecimiento del número de nodos de cómputo en la red a medida que añadimos recursos (routers) manteniendo el rendimiento por nodo. Esta métrica constituye un buen indicador para determinar el ámbito de aplicación más apropiado. En general, podemos decir que una red grande necesita una topología que *escale* bien, para realizar un buen aprovechamiento de los recursos disponibles. El coste de una red depende de múltiples variables, y contempla aspectos tanto del diseño e instalación como del funcionamiento. En este último punto destaca las necesidades de un sistema de refrigeración, cuyo coste de funcionamiento en ocasiones supera el de diseño [7]. No obstante, para evaluar el coste de funcionamiento necesitamos realizar un análisis de la potencia consumida. Dicho análisis está supeditado a la existencia de un *layout*, un esquema de la distribución de los componentes del router. El desarrollo de un *layout* del router se aparta del enfoque de este trabajo y queda como línea de trabajo futuro.

Para la evaluación del coste nos centraremos en una serie de consideraciones de diseño tales como el número de routers y de enlaces, y la longitud de los cables empleados. Como veremos posteriormente, el coste de los cables representa una parte significativa del montaje de la red. Para estimar los costes de cada componente, tomaremos como referencia el estudio realizado por Kim et al. en [24].

Por último, el análisis del rendimiento se basará en las métricas de throughput y latencia ya descritas en la sección 1.10. Los resultados se obtendrán a partir de la simulación de la red del sistema mediante las herramientas que se describirán en el capitulo 3.

## 2.3. Cálculo de los parámetros del toro concentrado

Para realizar la comparativa de escalabilidad de la red es necesario disponer de una relación entre el tamaño de red y el grado del router. Esta relación se va a determinar bajo

un patrón de tráfico aleatorio uniforme, en el que los nodos de cómputo envían paquetes a los demás nodos con igual probabilidad. En las redes Dragonfly la relación se consigue a partir del parámetro $h$:

$$N = p \cdot a\,(a \cdot h + 1) = 2h^2\left(2h^2 + 1\right) \tag{2.5}$$
$$k = p + h + (a - 1) = 4h - 1 \tag{2.6}$$

En el caso de los toros concentrados, para disponer de una igualdad similar debemos aplicar una serie de requisitos que limiten los grados de libertad de los parámetros. En primer lugar forzamos que el $BBW$ sea igual al tráfico que atraviesa la bisección de la red, haciendo que se cumpla la siguiente relación:

$$q = \frac{BBW}{n/4} \tag{2.7}$$

donde $q$ es la cantidad de tráfico ofrecida por cada router hacia el resto de la red (en flits/ciclo) y $n/4$ representa el número de routers que se comunican a través de la bisección. Este parámetro depende del número de nodos de cómputo unidos a cada router y del ancho de banda de los puertos del router, $q = p \cdot BW_{port}$. Aunque la comunicación en los enlaces es bidireccional, para simplificar los cálculos se ha considerado uno solo de los dos sentidos en el tráfico y los enlaces.

La segunda restricción es considerar sólo toros cuadrados, con igual número de routers por cada dimensión. Como el $BBW$ es el mínimo número de cables que hay en cada posible corte de la red en dos partes iguales, para que sea lo mayor posible el número de cables ha de ser el mismo en cualquier corte. Esta condición fuerza a que el factor de trunking $t$ sea el mismo en cada dimensión. La relación (2.7) queda así:

$$BBW = 2r^{D-1} \cdot t \cdot BW_{port} \tag{2.8}$$
$$p \cdot BW_{port} = \frac{2r^{D-1} \cdot t \cdot BW_{port}}{n/4} \tag{2.9}$$
$$p = \frac{8r^{D-1} \cdot t}{n} = \frac{8r^{D-1} \cdot t}{r^D} = \frac{8t}{r} \tag{2.10}$$
$$N = p \cdot r^D = \frac{(p \cdot r)^D}{p^{D-1}} = \frac{(8t)^D}{p^{D-1}} = 8t\left(\frac{8t}{p}\right)^{D-1} \tag{2.11}$$

La máxima configuración de red que podemos alcanzar se cumple con $p$ mínimo (sin concentración) y $t$ máximo (máximo factor de trunking). Esta conclusión es lógica si consideramos que el empleo de enlaces más anchos aumenta el $BBW$. No obstante, una restricción razonable es asumir $p \geq t$ para evitar un sobredimensionamiento de los enlaces respecto a los nodos de cómputo. En este caso el máximo tamaño de red se obtiene cuando se cumple la siguiente ecuación:

$$N = 8t(8)^{D-1} = 8^D t \tag{2.12}$$

Por último, relacionamos el tamaño de red con el grado del router:

$$k \geq p + 2D \cdot t = t\,(2D + 1) \tag{2.13}$$
$$t = p = \left\lfloor \frac{k}{2D+1} \right\rfloor \tag{2.14}$$
$$N = 8^D \cdot t = 8^D \left\lfloor \frac{k}{2D+1} \right\rfloor \tag{2.15}$$

Hemos fijado el cálculo de los parámetros del toro concentrado para un aprovechamiento máximo del $BBW$. Esta decisión implica que la red no tiene oversubscription.

## 2.4. Escalabilidad de la red

De acuerdo a los cálculos previos, en esta sección analizamos el tamaño para toros concentrados de 1 a 4 dimensiones variando el tamaño del router ($k$). Nos restringimos a un tamaño $D < 5$ porque un número mayor de dimensiones hace excesivamente complejo el layout (distribución física de la red). Este aspecto se analizará con más detalle en la sección 2.6.

El tamaño de router se ha limitado a un máximo de 64 puertos por considerarlo alcanzable con la tecnología de fabricación de commodities actual o en un futuro cercano. Comparamos el resultado obtenido frente a la escalabilidad de una red Dragonfly máxima para un tamaño de router dado, limitado también a un máximo de 64 puertos. Dicha red responde a la representación de las ecuaciones (2.5) y (2.6) para distintos valores de $h$. El límite corresponde a $h = 16$, que requiere routers de grado $k = 63$.

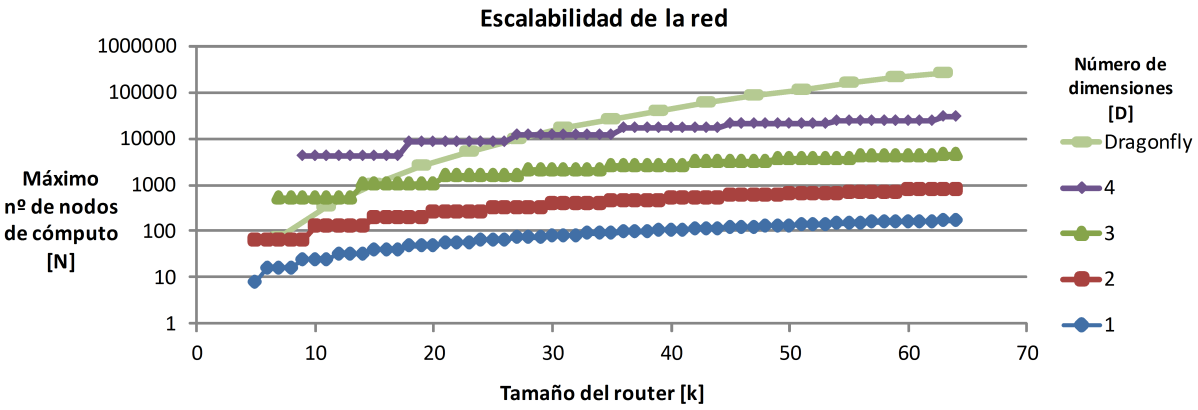



Figura 2.3: Comparativa de la escalabilidad máxima para un tamaño de router dado.

Si observamos los valores obtenidos en la gráfica de la figura 2.3 podemos apreciar que es posible alcanzar mayor número de nodos de cómputo con toros concentrados frente a redes Dragonfly, cuando el grado del routers $k$ es inferior a 27 puertos.

Esta diferencia se produce fundamentalmente para dimensión $D = 4$, y nos permite alcanzar hasta un máximo de 12288 nodos de cómputo. Por tanto, un posible escenario de interés sería la comparativa de los toros concentrados 4D frente a las redes Dragonfly para routers con un grado cercano a los 27 puertos. Dicho grado supone un punto intermedio dentro del rango considerado inicialmente.

Para entender la importancia de este tamaño de red, podemos observar el número de nodos de cómputo que podemos encontrar en un supercomputador actual. Si bien los 3 primeros sistemas del Top500 emplean de 50000 nodos en adelante, dentro de los 10 primeros podemos encontrar máquinas con menos de 10000 nodos de cómputo. Este tamaño equivale a un sistema de altas prestaciones en la actualidad, y de rendimiento medio en un futuro cercano.

## 2.5. Conectividad del router

La tolerancia a fallos de un sistema es uno de los aspectos más importantes a tener en cuenta. Con el crecimiento del tamaño de los sistemas, la posibilidad de fallos aumenta. Típicamente, los estudios de tolerancia contemplan tanto el fallo de enlaces como de routers. En términos de teoría de grafos, esto significa considerar la eje-conectividad o vértice-conectividad del grafo asociado a la topología que define la red de interconexión.

En cualquiera de los dos casos, la conectividad determina cuántos de estos elementos (enlaces ó routers) tienen que caer para que la red quede disconexa. Este tipo de estudios se ha considerado ampliamente, por ejemplo en [26]. En nuestro caso, vamos a considerar la conectividad del router como una primera aproximación para evaluar las características de tolerancia a fallos de la red. Para ello, lo que vamos a calcular es el número de enlaces que tienen que fallar para que un router quede aislado de la red, promediándolo al número de enlaces que posee el router y podremos así comparar los toros concentrados y las Dragonflies en términos de su capacidad a tolerar fallos.

En el caso de las redes Dragonfly, la conectividad del router es de $h + (a - 1)$. Este valor refleja que desconectando $h + (a - 1)$ de los enlaces el router queda disconexo del resto de la red. Esta conectividad supone un ratio de $\frac{h+(a-1)}{p+h+(a-1)} = \frac{3h-1}{4h-1} \approx \frac{3}{4}$ del total de enlaces del router. La figura 2.4 muestra con mayor detalle este resultado, reflejando en color rojo los enlaces que han de desconectarse para aislar el router.

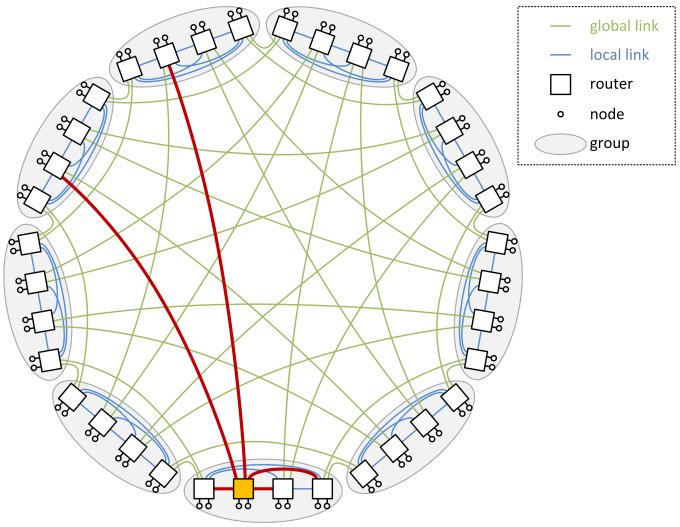



Figura 2.4: Detalle de la conectividad de un router para una red Dragonfly con $h = 2$ (enlaces marcados en rojo).

Asimismo, en el caso del toro concentrado $D$-dimensional, cualquier router está conectado por $2tD$ enlaces a sus $2D$ routers vecinos. Este número de enlaces representa la conectividad del router y corresponde con una proporción de $\frac{2D}{2D+1}$ del total de enlaces. Para los casos particulares de los toros $3D$ y $4D$ este valor es de $\frac{6}{7}$ y $\frac{8}{9}$, respectivamente. Como se puede ver, este resultado es superior en hasta un 18.5 % al obtenido para la Dragonfly, lo que implica en principio que el toro concentrado ofrece una mayor tolerancia a fallos.

## 2.6. Coste de la red

La comparativa de la escalabilidad de la red ha reflejado que existen tamaños de red intermedios en los que la topología de toro concentrado permite un mayor número de nodos de cómputo a igual de grado del router.

Aunque existe una necesidad clara de disponer de redes grandes para poder aumentar el tamaño de los supercomputadores y alcanzar mayores capacidades de cómputo, el coste resulta un fuerte factor limitante. Por este motivo se construyen nuevos supercomputadores que, manteniendo una lucha por alcanzar mayor rendimiento toman en consideración el coste como factor altamente limitante. Un ejemplo puede ser el supercomputador Gordon, en San Diego, el cual emplea routers InfiniBand para abaratar costes. Esta máquina, que ocupa el puesto 62 dentro del Top500, tiene tan solo 64 switches y 1024 nodos de cómputo.

En el caso de la red dicho coste depende fundamentalmente de los routers y de los enlaces más largos, realizados mediante fibra óptica. No obstante, dicha dependencia puede variar fuertemente según el número de cables necesarios. Los toros concentrados pueden suponer una posibilidad interesante para aumentar el rendimiento de la red y posibilitar un número alto de nodos de cómputo, dado que a priori prescinde de cables ópticos para el interconexionado.

En este trabajo nos vamos a restringir al análisis de los costes de los componentes de la red. Dichos componentes van a ser:

- Número de routers: será igual al número de nodos de red, es decir, $n$.
- Número de cables: la suma de las conexiones empleadas para unir los nodos de cómputo al router y los routers entre sí. Los enlaces desde el nodo de cómputo se realizarán mediante cables eléctricos de 1 metro de largo. Para el cálculo de la longitud de los cables empleados en los enlaces de red realizaremos un análisis más detallado a continuación.

$$N^{o}cables_{nodos\ de\ cómputo} = N \tag{2.16}$$

$$N^{o}cables_{red} = D \cdot n \cdot t \tag{2.17}$$

Por un lado es interesante realizar una comparativa de costes frente a las redes Dragonfly, puesto que están enfocadas a alcanzar tamaños grandes para que la capacidad de cómputo sea lo más alta posible. Por otro, conviene contrastar los costes que se alcanzarían frente a redes más similares (regulares, no jerárquicas) tales como los toros, ampliamente extendidos en los supercomputadores actuales.

En el caso de los toros concentrados la unión de los nodos de cómputo con los routers será idéntica, y el número de cables de red se podrá calcular de igual forma considerando $t = 1$. Las redes Dragonfly tienen un cálculo ligeramente distinto, puesto que emplean cables ópticos.

En el análisis de escalabilidad del toro concentrado nos hemos restringido a un máximo de 4 dimensiones. Esta limitación se debe a las consideraciones para realizar el *layout* o distribución física de los componentes de la red. Típicamente los elementos de un supercomputador se almacenan en armarios denominados *racks* o *cabinets*. Cada rack dispone de varios estantes cuya altura se mide en unidades de rack (U), siendo 42U la altura de rack más común. Estos racks se situan en hileras paralelas, intentando minimizar la longitud de las conexiones entre distintos racks.

Como el trabajo está orientado al empleo de tecnologías commodity se ha analizado el espacio ocupado por los nodos de cómputo y los routers que se fabrican bajo dicha tecnología. Para los routers tomaremos como base la tecnología Infiniband, por ser una de

las más extendidas en HPC (en la figura 2.5 se aprecia el número de sistemas basados en ella en los 10 supercomputadores de mayor rendimiento). En el caso de los nodos de cómputo la referencia será la solución TwinBlade de Supermicro [3], que emplea chips Intel Xeon. Un sistema TwinBlade tiene un tamaño de 7U y una capacidad para 20 chips Intel Xeon y un router. Considerando que el espacio ocupado por el router equivale aproximadamente a 1U, esto deja un bloque de 20 nodos de cómputo en 6U. Típicamente un router Infiniband ocupa 1U, pero existen productos como el Mellanox IS5022 [1] que permite situar dos router en una unidad de rack (1U). Es razonable considerar que los router de mayores prestaciones puedan reducir su tamaño hasta llegar a esas dimensiones.

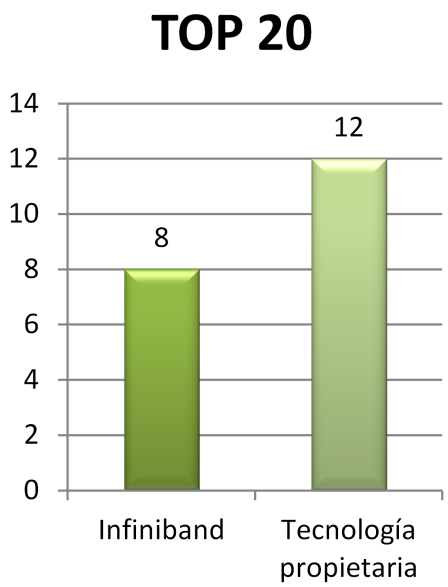



Figura 2.5: Gráfica con las tecnologías empleadas en los 20 sistemas de HPC con mayor rendimiento. Los resultados están extraídos del Top500 [4].

Considerando estos tamaños para nodos de cómputo y routers, no es físicamente razonable emplazar toros cuadrados de 5 ó más dimensiones en racks de 42 U.

En el análisis de escalabilidad de la sección 2.4 se consideró el tamaño máximo de la red (toro concentrado o Dragonfly) para un tamaño de router dado. En el presente análisis de costes tomaremos como referencia un tamaño de router concreto y analizaremos distintos tamaños de red. Mediante esta estrategia podemos igualar los costes del router para cada topología y analizar la relación coste/tamaño de red.

Nos centramos inicialmente en dos grados del router: $k = 36$ y $k = 64$. Estos representan un buen referente del estado del arte presente y alcanzable en un futuro cercano. Fijando un tamaño de grupo y variando el número de grupos en la red Drangofly podemos alcanzar tamaños de red inferiores al máximo. En este caso el número de enlaces globales por router ($h$) se calcula mediante la ecuación (2.18), tal y como se detalla en [23].

$$h = \left\lfloor \frac{ap+1}{g} \right\rfloor \tag{2.18}$$

El tamaño de grupo lo fijamos a partir del valor máximo de $h$ para un tamaño de router. En el caso del router con grado $k = 64$, la red Dragonfly más grande que podemos realizar se cumple para $h = 16$. Fijando el balanceo de carga obtenemos el número de routers por grupo $a = 32$, y el de nodos de cómputo por router $p = 16$. Esto fija un total

de 512 nodos de cómputo por grupo. A partir de este tamaño de grupo, consideramos distintos tamaños de red variando el número de grupos $g$. Para el caso $k = 36$, el número de nodos de cómputo por grupo se establece en 162 ($a = 18$, $p = 9$).

La longitud de los cables se ha establecido a partir de las suposiciones realizadas para una variante de las redes butterfly en [24]. Se considera que en 1 rack caben hasta 128 nodos de cómputo, de modo que para $k = 64$, cada grupo ocupa 4 racks. Tomando como referente las medidas del rack y haciendo una distribución física de los recursos en los racks, determinamos que la longitud promedio de los cables cortos es de $2{,}605m$. Las conexiones entre grupos se consideran siempre mediante tecnología óptica. Las uniones entre nodos de cómputo y router se realizan mediante enlaces a través de un *backplane*. El backplane es una placa de circuito impreso unida de forma transversal a los routers y nodos de cómputo que dispone de conexiones y de pistas de material conductor que unen entre sí dichos conectores.

Para el caso del toro concentrado fijamos un tamaño de router y vamos iterando el número de nodos de cómputo por router. Para cada valor de $p$ podemos determinar un factor de trunking y con éste determinar el máximo tamaño de red que va a seguir cumpliendo la ausencia de oversubscription. Las siguientes ecuaciones modelan este cálculo:

$$t = \left\lfloor \frac{k-p}{2D} \right\rfloor \tag{2.19}$$

$$r = \left\lfloor \frac{8t}{p} \right\rfloor \tag{2.20}$$

La longitud de los cables del toro concentrado se ha calculado en base al número de racks necesarios para albergar los nodos de cómputo y routers de cada tamaño de red. Mediante un pequeño programa que recibe el tamaño y capacidad de cada bloque de nodos de cómputo o routers y los parámetros $p$, $t$, $r$ y $D$ se calcula el número de filas de racks y de racks por fila para minimizar la longitud media de los cables. Una práctica muy común en el desarrollo del layout para redes toroidales es aplicar lo que se conoce como *folding* [10]. Esta técnica reorganiza los enlaces para evitar el empleo de un cable periférico de gran longitud, a costa de duplicar la longitud del resto de uniones. De este modo, todos los enlaces de una misma dimensión tienen igual longitud. La figura 2.6 ilustra con mayor detalle la aplicación de esta técnica.

Como el coste asociado a los cables es elevado y aumenta con la longitud promedio de los cables, hemos aplicado el uso de folding para reducir el coste total de la red. Las conexiones entre elementos de un mismo rack se realizan mediante cables de 1 metro de longitud. Para el cálculo de la longitud de las conexiones entre racks, se ha asumido el siguiente tamaño de rack: 600 mm de anchura, 1991 mm de altura y 1070 mm de profundidad. La figura 2.7 ilustra la organización del sistema con mayor detalle.

Como las longitudes de los cables resultantes son reducidas, se ha considerado el empleo de cables eléctricos para todos los enlaces del toro concentrado. Esta decisión reduce fuertemente el coste total de la red, dado que un cable óptico supone de uno a dos órdenes de magnitud más en el coste de cada enlace.

Para la estimación de los costes de cada componente nos hemos basado en el estudio realizado por Kim en [24]. Los costes asociados a cada elemento de la red son los siguientes:

- Router: 390$. Este precio aglutina la amortización de los costes de desarrollo (aproximadamente 300$) y los costes asociados al material empleado (aproximadamente 90$).

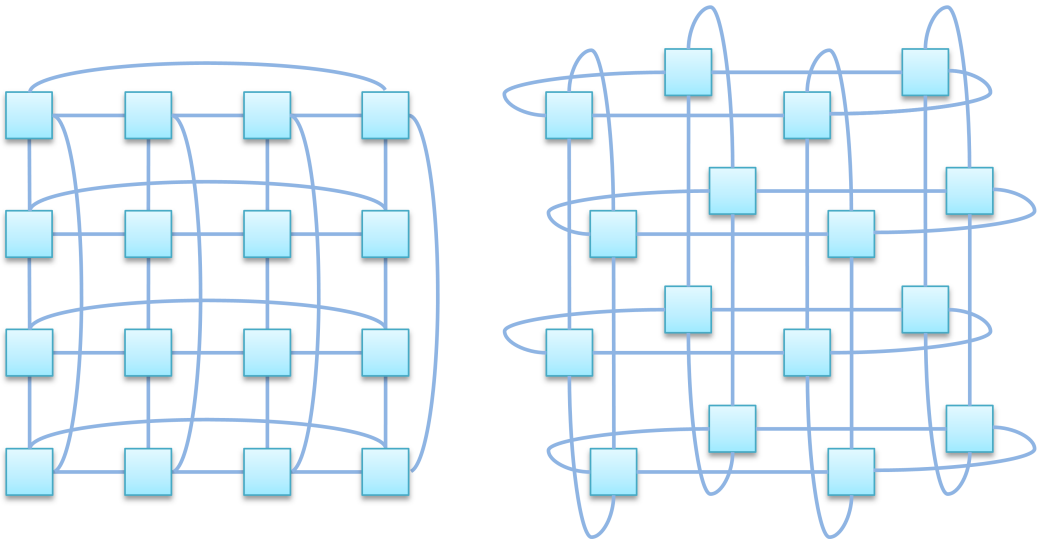


Figura 2.6: Ejemplo de la técnica de folding. La figura de la izquierda es un toro 2D de longitud 4, y la figura de la derecha muestra el mismo toro pero aplicando folding.

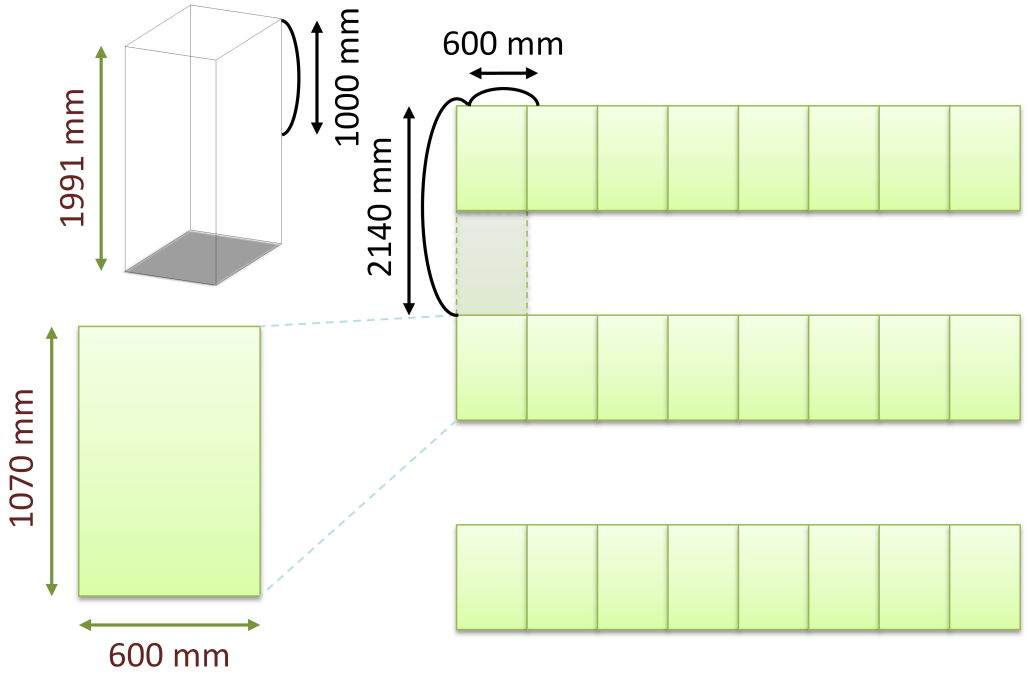



Figura 2.7: Layout del sistema y dimensiones del rack y los cables. Las longitudes del rack se han tomado del modelo APC NetShelter AR3100 [2].

- Unión mediante backplane: 1.95$ por enlace. Estas uniones deben limitarse a longitudes de unión reducidas, como las que se pueden producir entre un nodo de cómputo y un router.
- Cables eléctricos: se considera un coste fijo de 3.72$ debido a los conectores y a la fabricación, y un coste variable de 0.81$ por metro de longitud. Este modelo de coste es válido cuando la longitud del cable es lo suficientemente reducida para no

necesitar amplificadores de señal. En general, podemos situar este límite en torno a los 15 m [6].

- Cables ópticos: 200$ de coste fijo por enlace, y un coste variable de 1.01$ por metro de longitud. El coste fijo corresponde a los transceptores ópticos en ambos extremos.

El resultado de este análisis para un router de tamaño $k = 64$ queda reflejado en la figura 2.8. Como podemos comprobar, el coste es mayor a medida que la red aumenta su tamaño. Esta tendencia se debe a la necesidad de emplear cables de mayor longitud. En los toros concentrados la pendiente de la curva es mayor que en la Dragonfly porque el factor de trunking es mayor cuanto mayor es el tamaño completo de la red. Este crecimiento fomenta un aumento supralineal del número de enlaces necesarios. El mismo comportamiento se observa para el caso del router con grado $k = 36$ (figura 2.9).

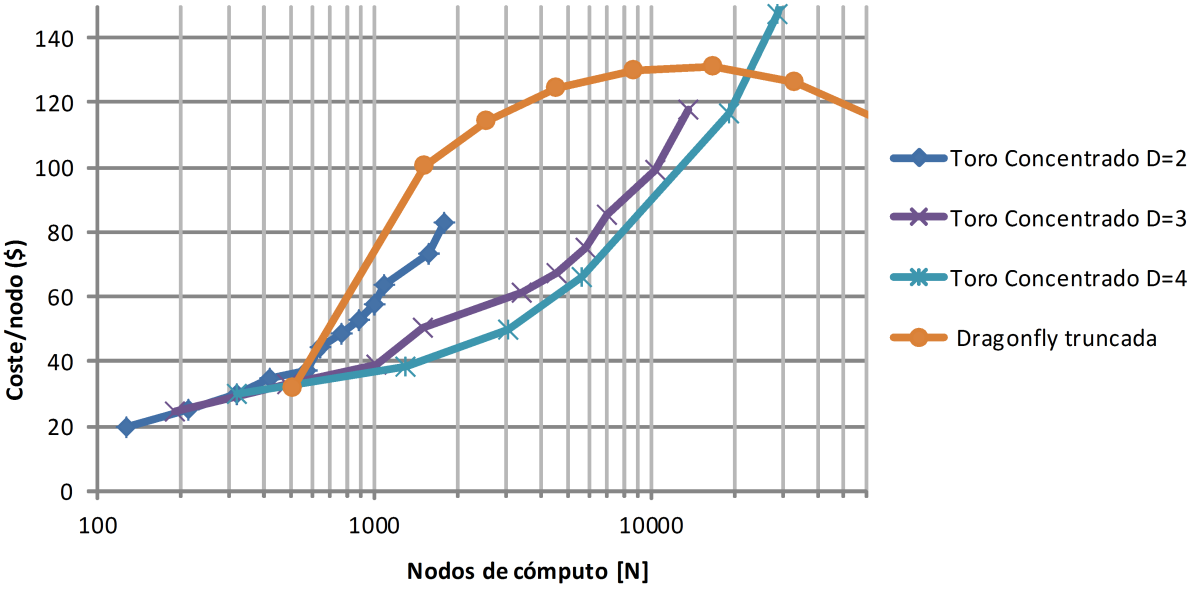



Figura 2.8: Comparativa del coste para un router de grado $k = 64$.

No obstante, el coste de la red Dragonfly para tamaños intermedios ($N < 20000$) es superior. La diferencia se debe al empleo de cables ópticos, cuyo coste es 2 órdenes de magnitud superior al de un cable eléctrico, y que requiere de un número alto de nodos de cómputo para amortizarlo. Esto muestra un área de interés para tamaños intermedios de los toros concentrados. Los toros concentrados con un número de dimensiones inferior a 3 no revisten excesivo interés en tanto que el número máximo de nodos de que podemos disponer es de 1792. Por este motivo nos centraremos en el estudio de los toros concentrados 3D y 4D.

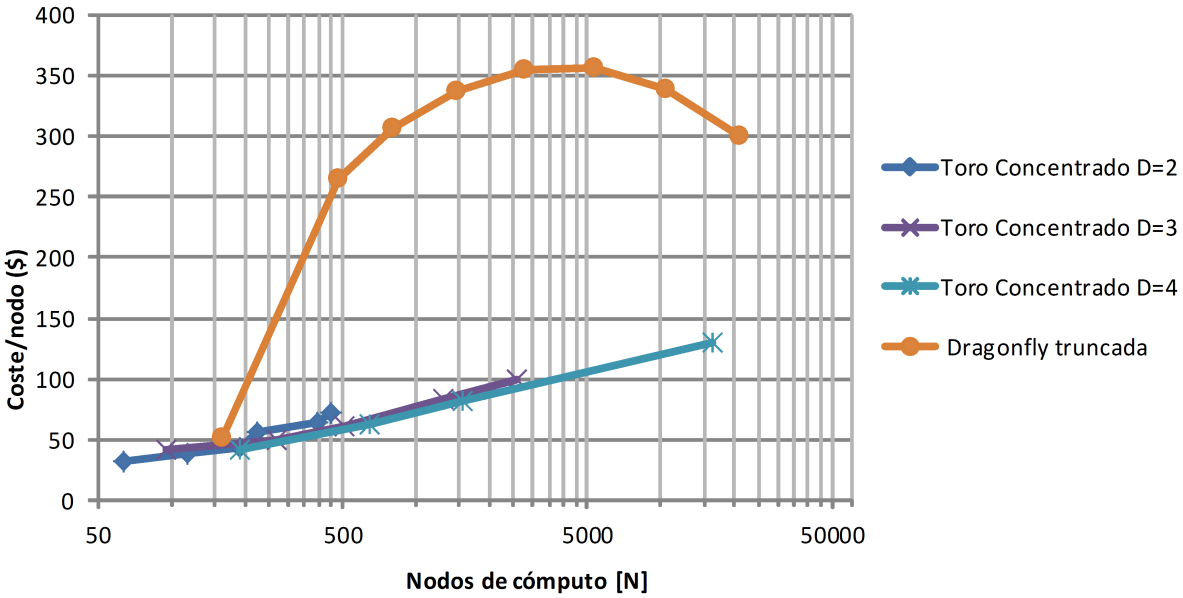



Figura 2.9: Comparativa del coste para un router de grado $k = 36$.

# Capítulo 3

# Desarrollo de la plataforma de simulación

Para el análisis del rendimiento de la red se ha optado por emplear herramientas de simulación. Este sistema de análisis es el más común en computación, puesto que el desarrollo de un prototipo es costoso. Además, resulta ineficiente para hacer una primera comparación entre distintas redes que puedan resultar más interesantes de cara a un desarrollo pormenorizado.

Con esta motivación se ha decidido partir de un simulador desarrollado en el grupo de Arquitectura y Tecnología de Computadores (ATC), del departamento de Electrónica y Computadores. Este simulador está inicialmente diseñado para modelar una topología Dragonfly. Dado que uno de los objetivos de este trabajo es realizar una comparativa del rendimiento de las redes Dragonfly y los toros concentrados, se ha optado por ampliar el desarrollo del simulador para admitir la simulación de la topología del toro concentrado. De este modo se permite a los demás miembros del grupo aprovechar los esfuerzos realizados para este trabajo. Además, el uso de un simulador destinado a la evaluación de las redes Dragonfly permite asegurar resultados contrastados del rendimiento obtenido. Por último, el empleo de este simulador garantiza que la arquitectura del router simulado es la misma para ambos casos y evita que surjan discrepancias ajenas a la diferencia en la topología de red.

En este capítulo se describe el entorno de simulación que se ha considerado. Para ello, en la sección 3.1 se explica el modelo de simulación considerado. En la sección 3.2 se describe la arquitectura del router simulado, detallando el empleo que se hace de los enlaces agrupados y las fases que conforman el funcionamiento del router. Finalmente, en la sección 3.3 se describe el simulador y su entrada y salida, así como los cambios que se han efectuado sobre el código original.

## 3.1. Elección del modelo de simulación

Para obtener datos del consumo y rendimiento de los distintos algoritmos, se ha utilizado como herramienta de trabajo la simulación computerizada del comportamiento de la red. A la hora de realizar una simulación se puede optar principalmente por 3 modelos: tráfico sintético, trazas, y simulación conducida por ejecución.

La simulación conducida por ejecución es el modelo más realista de los tres, puesto que se basa en la emulación de un sistema completo. De este modo, cada mensaje que se origina está determinado por las necesidades del sistema, y se reproducen las condiciones exactas de tráfico en un sistema real. No obstante, el desarrollo de un sistema así o la mera adaptación de uno ya existente a las necesidades de los algoritmos propuestos requiere

mucha complejidad y esfuerzo, que en este caso hemos juzgado innecesario. Como se trata de una primera aproximación al rendimiento que ofrecen, no es necesario tener una visión tan detallada del sistema.

La simulación mediante trazas es otro modelo bastante extendido, aunque algo complejo de desarrollar. A partir de un sistema real, se capturan los distintos mensajes generados y los instantes de generación, de modo que en el simulador podamos forzar la generación de dichos mensajes en sincronía con las trazas capturadas. Aunque este modelo también puede considerarse realista, presenta algunas dificultades importantes. Una de las más importantes es que necesitamos disponer de un sistema muy similar al simulado para poder realizar dichas trazas, lo cual en topologías innovadoras puede resultar difícil, y presenta costes elevados. Aunque el desarrollo de un simulador así no es excesivamente complejo *per se*, se ha considerado que su empleo no es atractivo para una primera toma de contacto con el rendimiento de las dos topologías. No debemos dejar de lado que en este trabajo se pretende realizar una evaluación en distintos aspectos, y se buscan aproximaciones rápidas que nos permitan determinar los escenarios más interesantes.

En este caso emplearemos la tercera opción, el empleo de tráfico sintético. Éste consiste en la inyección de paquetes en la red de acuerdo a distintos patrones de tráfico. Un patrón de tráfico determina la distribución espacial de mensajes dentro de la red de interconexión, estableciendo el destino del paquete a partir de información sobre la red y el nodo inyector. Para obtener unos resultados que sean representativos, se ha optado por emplear un patrón de tráfico aleatorio uniforme. En este patrón, cada nodo origen envía paquetes a cualquier otro nodo con igual probabilidad. Este patrón resulta un buen referente del comportamiento promedio de las redes respecto a distintas aplicaciones. Se trata de un patrón sencillo y por ello muy extendido en este tipo de simulaciones. Cabe resaltar, no obstante, que este patrón balancea la carga en la red y distorsiona ligeramente el comportamiento de algoritmos que realizan un mal balanceo de la carga.

## 3.2. Arquitectura del router simulado

Para poder tener unos resultados realistas, el desarrollo que hagamos en nuestro simulador debe tener en cuenta la limitación de recursos que tienen los elementos de la red. Por ello, la simulación del comportamiento del router y el paso de los paquetes por éste estará inspirada en una arquitectura concreta y conocida. Así garantizamos que se reproduzcan las mismas dificultades que en una red real con dispositivos comerciales.

Para este trabajo hemos tomado como referente el router de la arquitectura de red del procesador Alpha 21364 [28]. Este router emplea canales virtuales y tiene situados los buffers a la entrada, además de emplear una conmutación de tipo Virtual Cut-Through. Por otro lado, es una arquitectura que ha sido analizada y descrita de forma detallada en diversos artículos. En [28] podemos encontrar una explicación minuciosa de la asignación de los recursos mediante árbitros en las entradas y salidas del router.

En nuestra particular implementación del router, aunque nos hemos inspirado en el comportamiento de la arquitectura 21364, consideramos la existencia de hasta 64 puertos de entrada/salida. En el caso del toro concentrado, consideramos que varios de estos puertos van a estar unidos al mismo puerto. Este aspecto se verá con mayor detalle en la sección 3.2.1.

Consideramos también la existencia de múltiples inyectores, determinados por el parámetro $p$. El número de canales virtuales empleados y la forma en que se utilizan variará para los distintos mecanismos de evitación de deadlock empleados. Para este trabajo nos limitaremos a dos alternativas según la topología de red simulada. En el caso de los toros concentrados emplearemos el protocolo de Duato, usando un canal de escape con DOR y

burbuja (1 ó más VCs). En el caso de la red Dragonfly se realiza una separación en clases según el tipo de enlace empleado. Como se emplea un encaminamiento mínimo, cada paquete recorre como máximo un enlace global y dos enlaces locales. Utilizando 2 VCs podemos separar el tráfico en los enlaces locales según si el salto se produce en el grupo origen o en el grupo destino. De este modo garantizamos que los paquetes no se interfieran entre sí. Ambos mecanismos se explicaron con mayor detalle en la sección 1.9.

Todas estas diferencias no aumentan excesivamente la complejidad del esquema interno del router. Por ello, hemos imitado en nuestro simulador las fases que describe el router desde que se inyecta el paquete en uno de los buffers a la entrada hasta que se envía por uno de los puertos de salida.

El router dispone de tantos árbitros en cada entrada como canales virtuales empleados, y de un árbitro por cada salida. Cada salida dispone de un multiplexor para poder seleccionar una entrada cualquiera. Todos los árbitros han de elegir a quién asignan el recurso que controlan. Los árbitros de las entradas regulan el acceso de los paquetes al crossbar, y los árbitros a las salidas conceden el uso del enlace a una de las entradas del crossbar. El crossbar dispone de tantas entradas/salidas como puertos tenga el router, de modo que sólo se necesite un puerto de lectura en cada buffer de entrada.

Hemos mencionado que en esta arquitectura se utiliza una técnica de conmutación Virtual Cut-Through. Una peculiaridad de los routers que utilizan control de flujo Virtual Cut-Through es que cambia el esquema de arbitraje con respecto a wormhole. En el caso de usar control de flujo wormhole, el uso de canales virtuales hace necesario realizar dos arbitrios, uno a nivel de canales virtuales y otro a nivel de flit. El arbitrio a nivel de canal virtual asigna un canal virtual de salida (es decir, un VC en el siguiente nodo) para un canal virtual en un una de las entradas, mientras que el arbitrio a nivel de flit asigna el uso del crossbar y del enlace de salida en cada ciclo de trabajo. De este modo podemos entrelazar el envío de varios paquetes por la misma salida destinados a canales virtuales diferentes. Esto ocurre si en un determinado momento no queda espacio en un cierto buffer de destino, o si llega un paquete más prioritario.

Como en el caso de Virtual Cut-Through no se libera el recurso hasta que no se ha terminado de enviar el paquete este arbitraje en dos niveles ya no es necesario, y se integra en una sola asignación de recursos. Esto ocurre porque no existe posibilidad de que un paquete se quede bloqueado mientras sigue teniendo una salida asignada, ya que el siguiente nodo va a tener suficiente capacidad de almacenamiento para recibir el paquete completo.

El empleo de árbitros a las entradas y salidas empleado en esta arquitectura del router corresponde con un sistema de asignación de recursos conocido con el nombre de separable allocation. Es un sistema mucho más rápido que otros métodos más tradicionales que no permiten la paralelización ni segmentación de la asignación y permite asegurar bajas latencias en la entrega de paquetes.

Dentro de los separable allocators, existen 2 grandes variantes, según la primera fase de arbitraje que se realiza sea la de entrada o la de salida. Es preferible realizar primero el arbitraje a la entrada si hay más entradas que salidas, debido a que se propagan más peticiones hacia la fase restante. Sin embargo, en allocators con el mismo número de entradas que de salidas, es indiferente el orden en que se realicen los arbitrios de cara al rendimiento logrado. En la arquitectura 21364 y en nuestra simulación la primera fase que se realiza es el arbitrio en las entradas.

Los árbitros tienen que hacer uso de una política de arbitraje que garantice una repartición equitativa de los recursos, de modo que no se produzcan casos de inanición. Se dice que un elemento sufre de *inanición* cuando espera indefinidamente a que se le asigne el recurso solicitado. Existen múltiples políticas de arbitraje cuya idoneidad depende fuerte-

mente de la aplicación del router. En sistemas de tiempo real, por ejemplo, existen puertos cuyas peticiones han de ser asignadas de forma inmediata, aunque con ello se impida la igualdad en el reparto. En el caso concreto de la arquitectura 21364 se ha optado por un esquema LRS (Least Recently Selected, el menos recientemente seleccionado) en el cual se mantiene un registro de uso del recurso por cada elemento. En caso de competición entre varios elementos, se selecciona al que más tiempo lleve sin usarlo.

Los árbitros de las entradas deben determinar para cada paquete que encabeza un buffer qué salida puede utilizar, teniendo en cuenta el destino del paquete y el algoritmo de enrutamiento. Para ello considera el estado de los posibles nodos de destino inmediato, comprobando si están en uso la propia entrada o la salida deseada, o si el destino inmediato a través de esa salida tiene hueco suficiente. Si existe una salida disponible para él, se realiza una petición de uso del enlace entre la entrada y el crossbar. En cada arbitraje, cada canal virtual cuyo paquete en cabeza pueda avanzar realiza una solicitud, y los árbitros seleccionan a un canal virtual de cada entrada mediante la política LRS para realizar una solicitud al árbitro de la salida correspondiente. En el arbitrio de cada salida se determina a quién se le asigna el uso del crossbar y del enlace de salida, también mediante la política LRS.

### 3.2.1. Empleo de los enlaces agrupados (trunking)

En la sección 2.2 se explicó el empleo de múltiples cables para unir cada pareja de routers (trunking). En este simulador de red se han considerado dos formas de emplear los enlaces pertenecientes al mismo grupo:

- Estática: cada paquete emplea siempre el mismo enlace dentro de cada grupo a lo largo de su camino. Se establece una relación unívoca entre la posición del nodo de cómputo dentro del router y el enlace de salida ($link = p \mod t$). Si el número de nodos de cómputo no es múltiplo del factor de trunking, los nodos de cómputo más altos eligen el enlace del grupo de forma aleatoria. De este modo aseguramos un balance de la carga.

- Dinámica: en cada router se elige de nuevo un enlace dentro del grupo. Cada árbitro de entrada analiza cuál es el enlace del grupo con mayor número de créditos para la salida escogida, y realiza una petición al árbitro de salida de dicho enlace. De este modo se balancea la carga que transita por cada enlace del mismo grupo y se equilibra el uso de los recursos.

### 3.2.2. Fases de la simulación del comportamiento del router

La simulación del router va a comprender varias fases:

- Etapa de enrutamiento: por cada paquete que encabece un buffer, se determinará la salida elegida. Esta elección se realiza en función del algoritmo de enrutamiento que se esté simulando y del estado de los recursos, tales como el acceso al crossbar, el enlace de salida y disponibilidad de huecos en el siguiente nodo. En caso de existir una salida disponible, se realizará una petición al árbitro de entrada correspondiente.

- Etapa de arbitrio a la entrada: por cada entrada, se seleccionará la petición del buffer que más tiempo lleve sin atenderse, y se realizará una petición al árbitro de la salida requerida por el paquete.

- Etapa de arbitrio a la salida: por cada salida se seleccionará la petición de la entrada que más tiempo lleve sin atender dicha salida y se establecerá la conexión en el crossbar.

- Envío del paquete al siguiente nodo elegido. Comprende el paso del paquete a través del crossbar y del enlace hasta llegar al siguiente nodo.

La estructura final del router que hemos simulado comprende varios elementos, y se ilustra con mayor detalle en la figura 3.1:

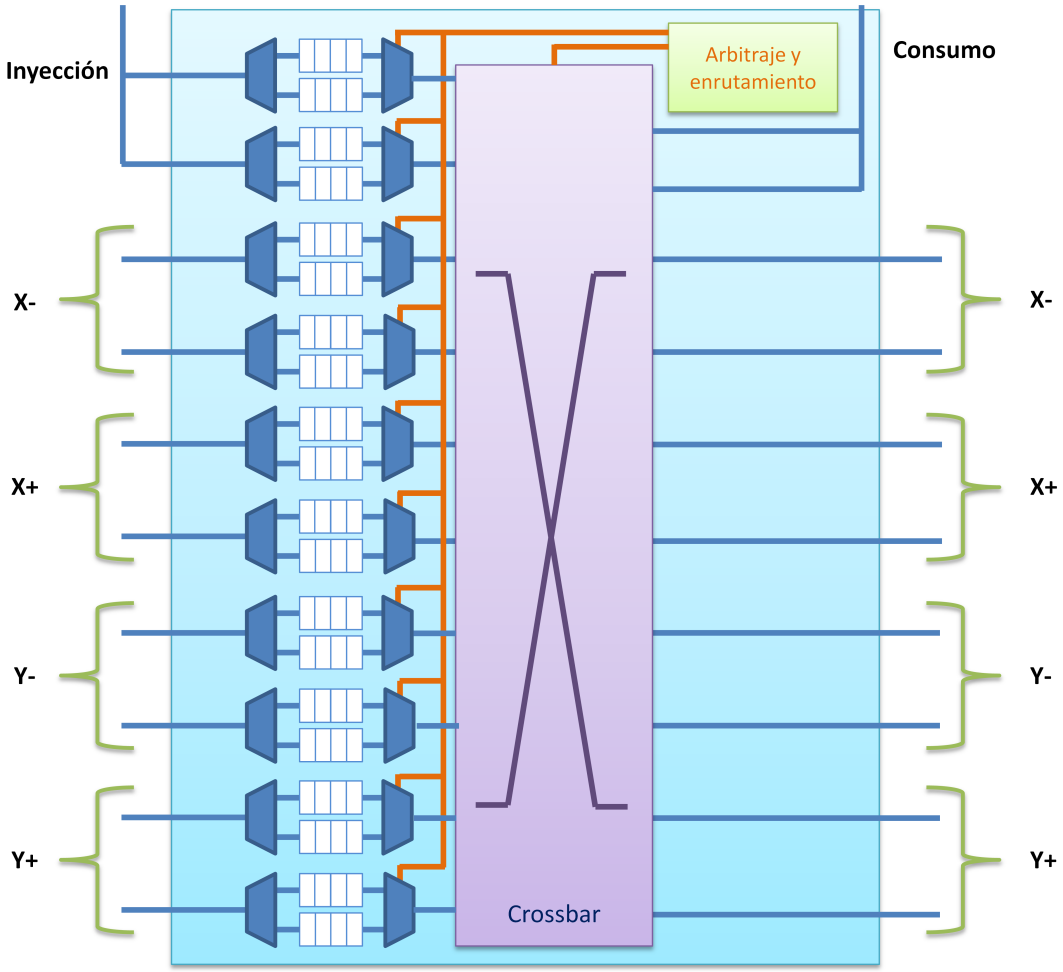



Figura 3.1: Arquitectura interna del router simulado. Ejemplo para $p = 2$, $t = 2$, con dos canales virtuales.

- Buffers. Se trata de memorias de tipo FIFO, con un solo puerto de lectura y otro de escritura. Dado que el control de flujo es de tipo Virtual Cut-Through, mediremos su capacidad de almacenamiento en número de paquetes.
- Allocator. Es del tipo separable, conformado por dos fases de arbitrio: existe un árbitro por cada entrada y otro por cada salida. Al árbitro de entrada, le asignamos además funciones de enrutamiento, para simplificar nuestro diseño. En muchos routers comerciales, la función de encaminamiento se suele realizar en un elemento aparte, que se comunica con el árbitro.
- Crossbar. Es el elemento que realiza la unión entre las distintas entradas y salidas, para permitir la conmutación de los paquetes en la red.

## 3.3. Parámetros, características y salida del simulador

Como se ha comentado al comienzo del capítulo, para determinar el rendimiento de la red se ha partido de un simulador de red desarrollado en el grupo ATC de la Universidad de Cantabria. Se trata de un simulador conducido por tiempo, lo que le permite utilizar menos código y ser más sencillo de programar que un simulador conducido por eventos, siendo penalizado sólo en la duración de la simulación para el caso de redes con baja actividad [30]. El simulador toma una duración inicial de ejecución, y la duplica en dos fases. La primera fase sirve como calentamiento de la red, para evitar efectos anómalos causados por el vacío de los buffers. En la segunda fase se considera que se ha alcanzado un estado fiable para la toma de estadísticas, tanto de throughput como de latencia.

El simulador esta escrito enteramente en lenguaje C++, y tiene un tamaño aproximado de unas 11.000 líneas de código. Como parte de este trabajo se han modificado cerca de 2.500 líneas para soportar la simulación de los toros concentrados. Aunque no forman parte del código del simulador, cabe destacar la existencia de un conjunto de scripts que permiten recoger los resultados de la salida del simulador y condensarlos en varias gráficas. Estos scripts forman parte del trabajo previo y han sido convenientemente modificados para los análisis del ámbito de este trabajo.

El simulador recibe un conjunto bastante amplio de parámetros, que varían ligeramente según la topología simulada. De entre ellos cabe destacar el tamaño de los buffers (separando inyección de tránsito), el número de ciclos de red simulados, el tamaño de los paquetes enviados, el número de canales virtuales y la carga aplicada. Para el caso de los toros concentrados, se detallan los valores de $p$, $t$, $D$ y el número de routers por dimensión $r$, así como el tipo de elección del enlace dentro de un mismo LAG (Link Aggregated Group). En el caso de las Dragonflies, el simulador recibe los valores de $p$, $a$, $h$ y el número de grupos $g$.

La salida del simulador consta de los parámetros de entrada más relevantes para asociar el resultado con la simulación concreta realizada. También incluye los resultados para las métricas de rendimiento consideradas, y distintos parámetros de salida que permiten comprobar la fiabilidad de la simulación. Entre estos últimos se encuentran la ocupación media de los buffers desglosada por dimensión y canal virtual, y el uso promedio de cada puerto de salida.

Como métricas de rendimiento, además de la latencia se muestran el número de paquetes entregados y el throughput. Para que los resultados para distintos tamaños de red sean comparables, se han normalizado los valores de throughput al número de nodos de cómputo de la red, y de latencia al número de paquetes entregados. La latencia se expresa en ciclos por paquete y el throughput en phits/(nodo*ciclo).

# Capítulo 4

# Resultados de rendimiento

Tras haber detallado en el capítulo anterior los aspectos más relevantes de las simulaciones realizadas, en este capítulo vamos a analizar los resultados obtenidos mediante la simulación de la red. En la sección 4.1 determinaremos la diferencia del rendimiento obtenido según el empleo de 1 ó 2 canales virtuales para un toro concentrado. Posteriormente compararemos el rendimiento de las dos topologías analizadas en dos grandes apartados: las simulaciones realizadas para un tamaño de router $k = 36$ en la sección 4.2 y las obtenidas para $k = 64$ en la sección 4.3. En cada caso veremos por separado las gráficas de throughput y latencia para el toro concentrado de 3 y 4 dimensiones y la red Dragonfly. Por último justificaremos los resultados mostrados en el caso de la red Dragonfly mediante las gráficas obtenidas para distintos tamaños de red Dragonfly densa.

Las simulaciones de los toros concentrados están realizadas para tamaños de red vistos en la sección de costes. Los parámetros se determinan fijando un valor para $p$ y calculando $t$ y $r$ a través de las ecuaciones vistas en la sección 2.6. De entre los tamaños contemplados en las figuras 2.8 y 2.9 se han elegido los siguientes casos: 96, 270, 512, 1296, 2560, 192, 648 y 1536 nodos de cómputo.

En el caso de las redes Dragonfly el tamaño de red también está fijado en la sección de costes 2.6, resultando un tamaño de grupo de 162 y 512 nodos de cómputo respectivamente para los dos tamaños del router. El tamaño total de la red depende del número de grupos elegido. En el cálculo de las relaciones entre los parámetros de la Dragonfly de dicha sección se consideró emplear múltiples enlaces entre una pareja de grupos para garantizar que el número total de enlaces globales por router sea igual que para el caso de la Dragonfly densa. Dado que el enfoque del trabajo es realizar un análisis inicial de diversos aspectos de cada topología, se ha optado por modificar la distribución de recursos en la Dragonfly. En este caso se fija un único enlace global entre cada pareja de grupos y se reparte el número de enlaces globales de forma equitativa entre los routers del mismo grupo ($h = \lceil \frac{g}{a} \rceil$). Este cambio incumple la relación inicial entre número de enlaces locales y globales ($a = 2p = 2h$), lo que limitará el rendimiento de la red. En un futuro se plantea dotar al simulador de la capacidad para simular el empleo de varios enlaces globales entre una misma pareja de grupos de la red.

Todos las simulaciones que se han realizado tienen una duración de 40000 ciclos. De estos, los primeros 20000 constituyen un calentamiento de la red y se desechan de cara a las métricas de rendimiento. Para la simulación de los toros concentrados se ha empleado una elección dinámica del enlace dentro del LAG, porque el rendimiento alcanzado es ligeramente mayor que con una elección estática. En todos los casos se ha empleado una longitud de paquete de 16 phits.

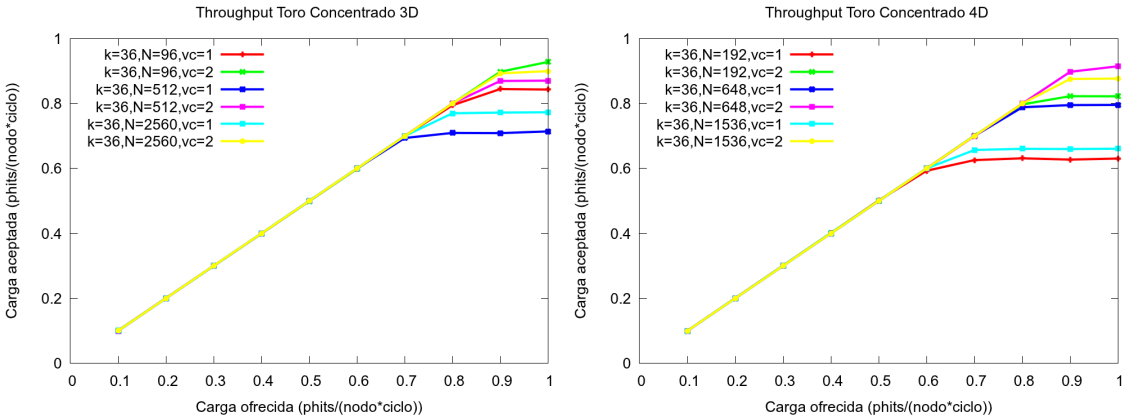
Se ha fijado un tamaño de 100 phits para los buffers de los puertos de red y de 10000 phits en los buffers de inyección. Los enlaces tienen un retardo de 1 ciclo por phit trans-

mitido.

## 4.1. Elección del número de canales virtuales en el toro concentrado

En la sección 3.2 acerca de la arquitectura del router simulado se explicó que como mecanismo de evitación de deadlock de la Dragonfly se emplea la separación de recursos en clases, empleando 2 canales virtuales. También se detallaba que la evitación de deadlock en el toro concentrado se garantiza a través del uso de un canal virtual de escape, de modo que podemos emplear otros canales virtuales adicionales de forma adaptativa. La justificación de su funcionamiento se puede encontrar en el capítulo 1 de introducción y trabajos previos.

En esta sección vamos a analizar la diferencia de rendimiento que se produce en el toro concentrado según el empleo de 1 ó 2 canales virtuales. El canal virtual adicional se va a emplear de forma adaptativa y en caso de congestión los paquetes que circulen por él pueden desviarse al canal de escape. Las figuras 4.1a y 4.1b ilustran la diferencia que se produce sobre el toro concentrado de 3 y 4 dimensiones, respectivamente.




(a) Toro 3D (b) Toro 4D

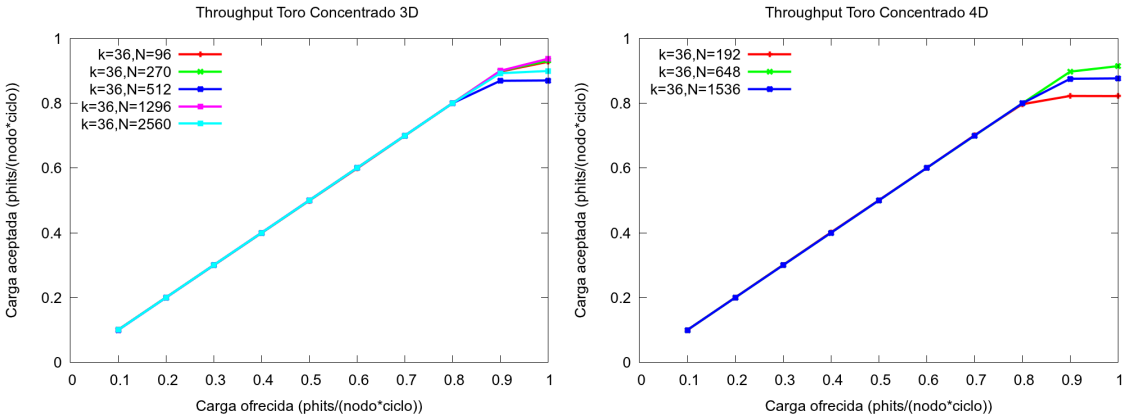
Figura 4.1: Throughput alcanzado para distintos tamaños de toro concentrado con 1 y 2 canales virtuales.

Las gráficas muestran que para un mismo tamaño de red, el empleo de un canal virtual adaptativo adicional proporciona una mejora en el throughput obtenido. El incremento es de hasta un 21 % en el caso del toro 3D, para la red de tamaño $N = 512$ nodos de cómputo. En el caso del toro 4D la diferencia es aún mayor, observando una mejora de un 32 % cuando el tamaño de la red es de $N = 1536$ nodos de cómputo. En ambos casos la diferencia del throughput se observa cuando la red está saturada, siendo el punto de saturación aquel valor de carga ofrecida para el que la carga aceptada se degrada.

A tenor de los resultados vistos, podemos decir que el empleo de un canal virtual adicional permite aumentar el rendimiento de la red y que la diferencia es apreciable. Como consecuencia negativa de utilizar dicho canal virtual está la necesidad de emplear buffers adicionales para separar el tráfico de cada canal virtual, así como multiplexores a las entradas y salidas de los buffers de una misma entrada. Este aumento de los recursos empleados supone un incremento de la complejidad del router y afecta negativamente al coste de la red. Sin embargo, en la Dragonfly hemos empleado 2 canales virtuales para los

enlaces locales como mecanismo de evitación de deadlock. Por tanto, la complejidad y coste del router para el toro concentrado empleando 2 VCs va a ser comparable al empleado en la Dragonfly. Como se trata de una comparativa más equitativa y el rendimiento se ve incrementado, en la comparativa del rendimiento entre las dos topologías consideraremos sólo el uso de 2 canales virtuales en el toro concentrado.

## 4.2. Resultados para un router de grado $k = 36$




(a) Toro 3D (b) Toro 4D

Figura 4.2: Throughput alcanzado para distintos tamaños de Dragonfly y toro concentrado. Router de grado $k = 36$.

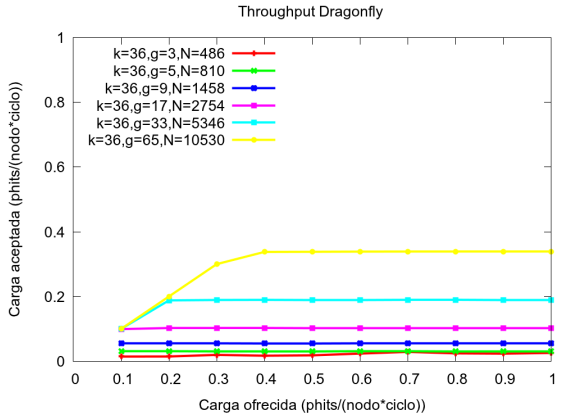



Figura 4.3: Throughput alcanzado para distintos tamaños de Dragonfly con router $k = 36$.

En la figura 4.2a podemos ver el throughput obtenido para los distintos tamaños de toro concentrado con $D = 3$ dimensiones, hasta un máximo de 2560 nodos de cómputo. El rendimiento de la red es bastante elevado, logrando aceptar hasta 0.9 phits/ciclo por cada nodo de cómputo. Además, al alcanzar el punto de saturación el rendimiento se mantiene estable. Analizando la elección de los parámetros en la sección 2.3 observamos que el rendimiento esperado sería alcanzar la carga máxima, 1 phit/(nodo*ciclo), puesto que el número de enlaces está dimensionado acorde con las carga de tráfico que va a atravesarlos. La diferencia entre el rendimiento teórico y el obtenido mediante la simulación se debe

a la aparición de contención en los buffers del router. La *contención* se produce cuando varios mensajes compiten por el mismo puerto de salida, de modo que todos menos uno de ellos han de esperar a que el recurso quede libre. En este caso no se puede transmitir un mensaje por cada puerto de entrada de forma simultánea, y el empleo de los recursos no es óptimo. Al reducir el número de mensajes reenviados la carga aceptada decrece y la latencia se incrementa debido a la espera en el buffer del router.

En la gráfica para el toro concentrado de 4 dimensiones (figura 4.2b los resultados son similares, pero se aprecia un punto de saturación más bajo, en torno a los 0.8 phits/ciclo. Esta diferencia se debe a que al aumentar el número de puertos del router la contención también se hace mayor, ya que existe mayor probabilidad de que dos mensajes de entradas diferentes compitan por el mismo recurso de salida.

En el caso de la red Dragonfly (figura 4.3) el rendimiento es sensiblemente inferior, alcanzando un máximo de 0.34 flits por nodo y ciclo cuando la red tiene $g = 65$ grupos, con un total de 10530 nodos de cómputo. No obstante, al comienzo del presente capítulo se ha explicado que las simulaciones emplean un solo enlace entre cada par de grupos y provocan oversubscription, ya que el número de enlaces globales no está dimensionado para el tráfico que va a atravesarlos. En la sección 2.1 observábamos que para garantizar la ausencia de oversubscription tenía que cumplirse que $a = 2p = 2h$. La comparativa idónea sería evaluar el rendimiento cuando se emplean enlaces agrupados entre cada par de grupos, de modo que el número total de enlaces globales esté adecuadamente dimensionado. No obstante, por razones de tiempo el desarrollo del simulador necesario para caracterizar dicho caso se aparta del ámbito de este trabajo.

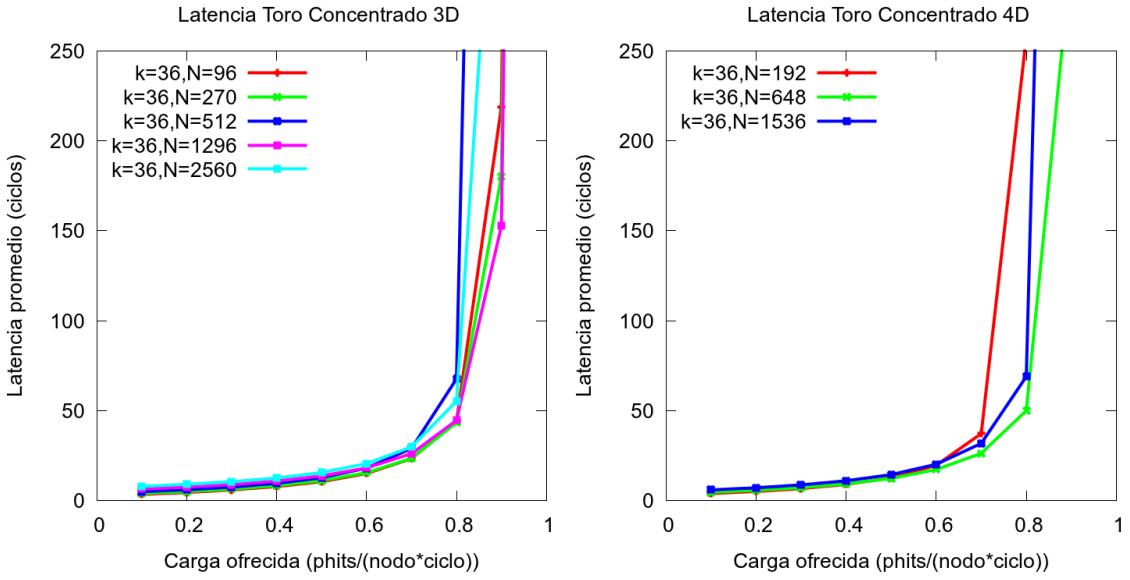
Los resultados de latencia para las mismas simulaciones son coherentes con las gráficas anteriores. Podemos ver en las figuras 4.4a y 4.4b que la latencia tiene un crecimiento exponencial, con 3 fases que se pueden diferenciar fácilmente. Para valores bajos de carga ofrecida la red no está saturada y se observa un crecimiento lineal de la latencia. En el punto de saturación se aprecia que la curva describe un gran salto, porque la red no es capaz de entregar los paquetes al mismo ritmo que se generan, y éstos se acumulan en las colas de inyección durante más tiempo. Cuando la red está saturada el crecimiento es idealmente no acotado, debido a que una carga de tráfico más elevada genera congestión en los buffers de los routers y provoca que los paquetes permanezcan más tiempo a la espera de conseguir el recurso. En la realidad el tamaño de los buffers de inyección es finito y existe un límite superior para la latencia.

En las gráficas no se ha incluido la latencia de la Dragonfly porque al tener un punto de saturación tan bajo la latencia inicial es muy alta y no permite observar la zona lineal de la curva.

Se ha incluido la figura 4.5 para mostrar el crecimiento de la latencia desglosado en sus componentes, de modo que pueda observarse que la saturación motiva un crecimiento de la latencia en las colas de inyección del router. La gráfica corresponde con el caso concreto de un toro concentrado de 3 dimensiones y 2560 nodos de cómputo.

## 4.3. Resultados para un router de grado $k = 64$

Para el caso del empleo de un router con grado $k = 64$ los resultados son similares a los anteriores, como se puede apreciar en las gráficas 4.6a, 4.6b y 4.7. En la sección anterior justificábamos que la elección de los parámetros de la red está realizada intentando dimensionar el número de enlaces de la bisección de acuerdo al tráfico que va a atravesarlos. Sin embargo, no siempre es posible hacer un ajuste óptimo, por lo que se producen variaciones en el rendimiento máximo alcanzado. En general, el punto de saturación se produce para cargas de tráfico ofrecido algo menores que en el caso de los routers de 36 puertos, en torno




(a) Toro 3D (b) Toro 4D

Figura 4.4: Latencia alcanzada para distintos tamaños de toro concentrado. Router de grado $k = 36$.

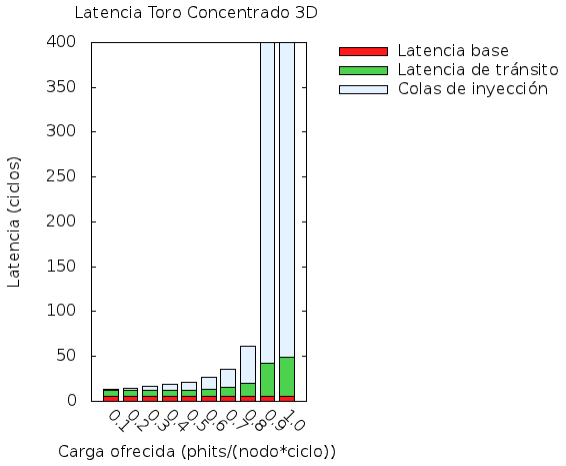
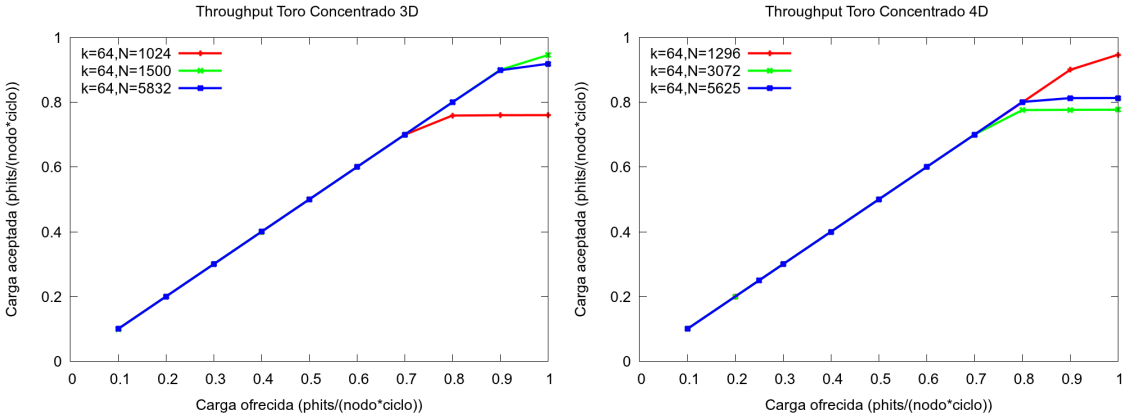



Figura 4.5: Componentes de la latencia en un toro concentrado $D = 3$ con $N = 2560$ nodos de cómputo, cuando la carga ofrecida es de 1 flit por nodo y ciclo.

a 0.8 phits por nodo y ciclo. En la Dragonfly se sigue cumpliendo la tendencia mostrada previamente, el rendimiento crece con el número de grupos porque aumenta el número de enlaces globales y se aproxima más a la relación $a = 2h$.

Los resultados de latencia (figuras 4.8a y 4.8b también son acordes con lo esperado, creciendo de forma exponencial cuando la carga ofrecida supera el punto de saturación de la red. Nuevamente se ha omitido la gráfica con la latencia de la Dragonfly porque no los valores iniciales de latencia son muy altos.




(a) Toro 3D (b) Toro 4D

Figura 4.6: Throughput alcanzado para distintos tamaños de Dragonfly y toro concentrado. Router de grado $k = 64$.

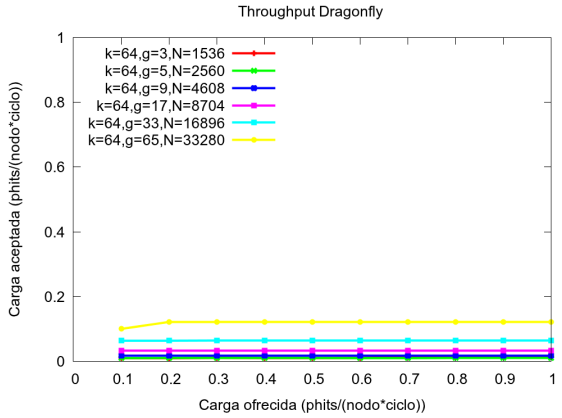
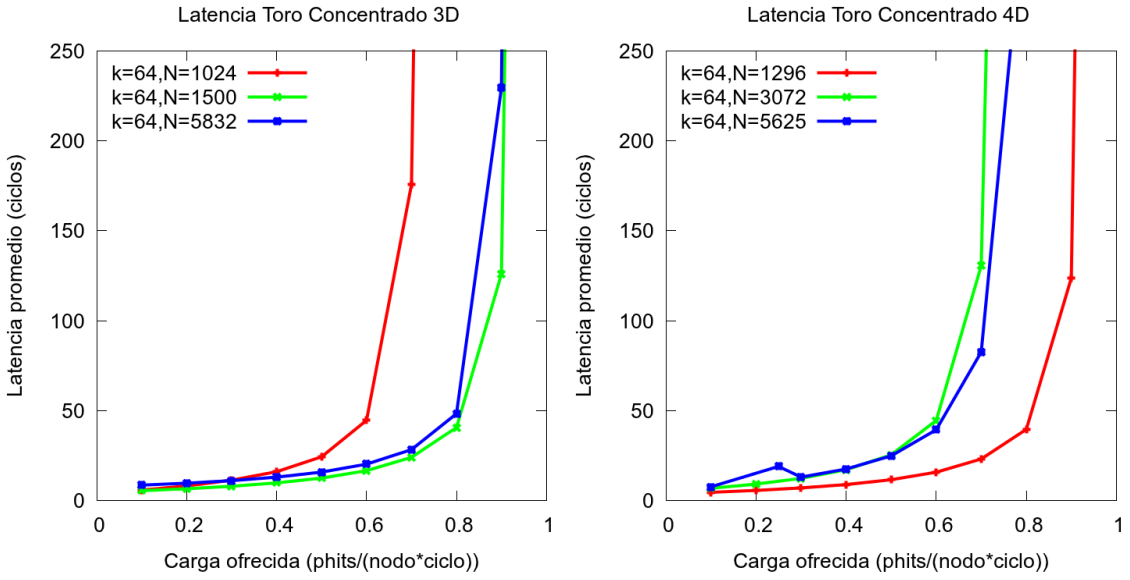



Figura 4.7: Throughput alcanzado para distintos tamaños de Dragonfly con router $k = 64$.

## 4.4. Análisis del rendimiento de una red Dragonfly densa

En las secciones anteriores de este capítulo hemos contrastado el rendimiento entre los toros concentrados y la Dragonflies para dos tamaños de router distintos. No obstante, la comparativa se ha visto perjudicada porque el dimensionamiento de las Dragonflies simuladas no está adecuado a las necesidades de la red y se produce oversubscription. Para poder realizar un análisis más justo en esta sección se analiza el rendimiento de varios tamaños de Dragonfly densa, en la que el número de grupos es el máximo posible. De este modo el dimensionamiento del número de enlaces globales se adecúa a su uso, y se mantiene un único enlace global entre cada pareja de grupos.

Se han contemplado diversos tamaños de red, desde $h = 2$ (72 nodos de cómputo) hasta $h = 6$ (5256 nodos de cómputo). Las figuras 4.9 y 4.10 muestran los resultados de throughput y latencia alcanzados en este caso. Como el factor más limitante para el rendimiento de la red es la relación entre el número de enlaces locales y globales, en la Dragonfly densa todos los tamaños de red muestran un resultado similar, con un punto de saturación en torno a los 0,7 phits/(nodo*ciclo). Aunque son resultados similares, la carga máxima aceptada disminuye ligeramente a medida que aumenta el tamaño de red.




(a) Toro 3D (b) Toro 4D

Figura 4.8: Latencia alcanzada para distintos tamaños de toro concentrado. Router de grado $k = 64$.

Esta tendencia se debe a que la contención crece a medida que aumenta el tamaño de la red, porque se emplean routers de grado más alto.

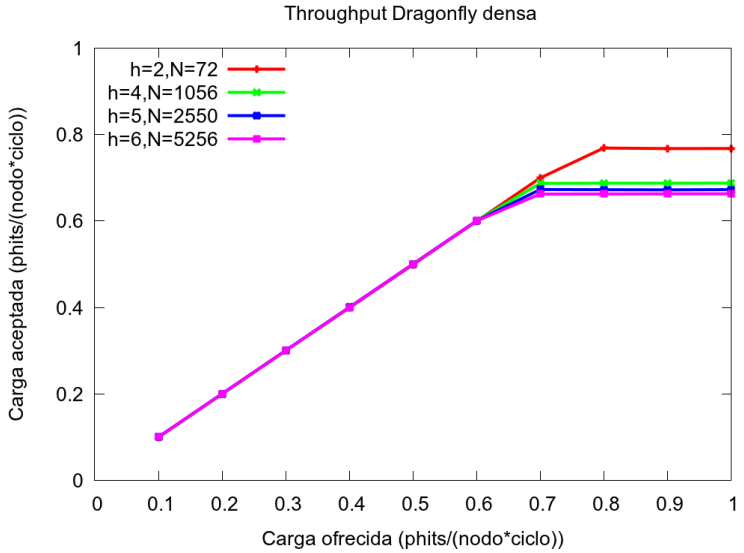



Figura 4.9: Throughput alcanzado para distintos tamaños de red Dragonfly densa.

Como las variaciones de throughput y latencia son relativamente pequeñas aunque se trata de tamaños de red distintos, es de esperar que el rendimiento de la Dragonfly no densa empleando varios enlaces entre cada pareja de grupos permita alcanzar rendimientos similares. En todo caso, como la contención aumenta con el tamaño del router empleado y en la red más grande que se ha simulado ($h = 6$) se emplean routers de grado $k = 23$, estos resultados constituyen una cota superior del rendimiento que se espera alcanzar

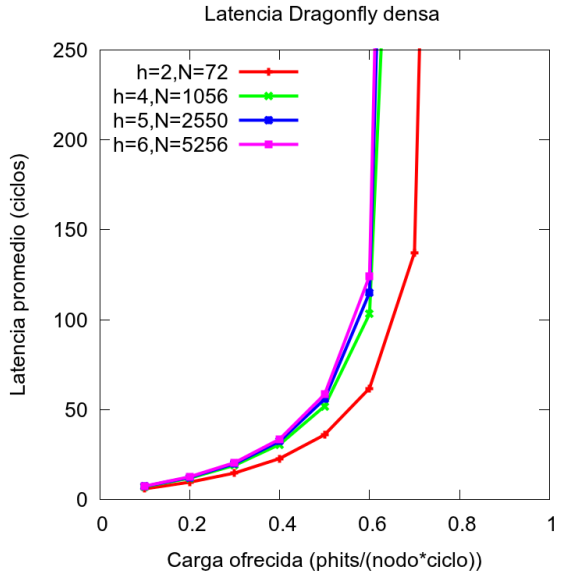



Figura 4.10: Latencia alcanzada para distintos tamaños de red Dragonfly densa.

en ese caso. Este rendimiento es de 0,7 phits/(nodo*ciclo) para una red de 5256 nodos de cómputo, y es hasta un 30 % inferior al obtenido para los distintos tamaños de toro concentrado que se han simulado.

Por tanto se concluye que, si bien no se ha podido realizar la comparativa más adecuada, con los resultados mostrados por la Dragonfly densa podemos afirmar que los toros concentrados representan una alternativa más interesante para tamaños de red intermedios (menos de 10000 nodos de cómputo) tanto en rendimiento como en coste y escalabilidad.

# Capítulo 5

# Conclusiones

## 5.1. Análisis técnico

En este trabajo se han evaluado dos topologías de red para sistemas de computación de alto rendimiento basadas en routers de alto grado. La evaluación se ha realizado en términos de escalabilidad, coste y rendimiento. A lo largo de esta memoria se ha justificado la necesidad de nuevas topologías de red que permitan un rendimiento elevado para un alto número de nodos de cómputo. Se han analizado dos topologías de red directa, las Dragonflies y los toros concentrados. La red Dragonfly es una novedosa propuesta enfocada a tamaños de red elevados (hasta 262656 nodos de cómputo empleando routers de grado $k = 64$) con un coste reducido, y que permite aumentar el tamaño de la red de forma exponencial incrementando el coste de forma logarítmica. Los toros concentrados son una evolución de la topología toroidal, de amplio uso en redes de sistemas de HPC. Se trata de una propuesta que ya se ha empleado de forma comercial en sistemas de alto rendimiento (Gordon, en el Centro de Supercomputación de de San Diego) con tamaños de red intermedios, en torno a 1000 nodos de cómputo.

Para la realización de este trabajo se ha hecho una importante labor de documentación previa. Dicha documentación incluye aspectos técnicos, como manuales y fichas técnicas de productos comerciales, o estándares de tecnologías de comunicación. Asimismo se ha determinado el estado del arte en redes para sistemas de HPC y las tendencias mostradas en los últimos años.

Como parte del trabajo se ha analizado la relación entre los parámetros del toro concentrado que garantiza un dimensionamiento del número de enlaces acorde con la carga de tráfico aplicada por los nodos de cómputo. A partir de esta relación, se ha determinado la escalabilidad de la red para el toro concentrado y se ha comparado contra la red Dragonfly. Con este fin, se ha mostrado el tamaño máximo que podemos obtener con ambas redes para un grado fijo del router, manteniendo un aprovechamiento óptimo del $BBW$. Se ha considerado un grado máximo de $k = 64$ por considerarlo realista a corto y medio plazo, considerando las soluciones comercializadas en la actualidad. El resultado nos ha demostrado que un toro concentrado puede permitir tamaños de red superiores a los alcanzados con la Dragonfly para routers con un grado inferior a 27 puertos, obteniendo tamaños cercanos a los 10000 nodos de cómputo. Este tamaño supone un sistema comparable a algunos de los presentes entre los 10 primeros puestos del Top500.

También se ha analizado el coste asociado a la implementación de la red, empleando un modelo de costes propuesto en trabajos previos. Para el cálculo de dicho coste se ha realizado un análisis de la distribución física de los routers y nodos de cómputo en racks a partir de tamaños de soluciones comerciales. El resultado nos permite determinar un escenario para tamaños de red inferiores a los 10000 nodos de cómputo en el que el toro

concentrado permite alcanzar un coste de red inferior empleando routers del mismo grado que la Dragonfly.

Como primera aproximación a la tolerancia a fallos de la red se ha realizado un estudio teórico de la conectividad del router. Dicha conectividad indica el número mínimo de enlaces que han de eliminarse para que un router quede desconectado de la red. Para un toro concentrado de 3 ó 4 dimensiones, se verifica que la conectividad es mayor que en la Dragonfly, aunque la diferencia se reduce a medida que aumenta el tamaño de la red.

Posteriormente se ha realizado un análisis del rendimiento de la red en términos de throughput y latencia para un conjunto de tamaños de red considerados en los escenarios previos. Dicho análisis se ha realizado mediante el uso de un simulador de red ya existente enfocado a las redes Dragonfly, añadiendo las herramientas necesarias para simular el comportamiento de los toros concentrados y de Dragonflies de tamaño inferior al máximo posible para un router dado. Los resultados obtenidos muestran que en todos los casos el rendimiento del toro concentrado es igual o superior al de la red Dragonfly, para tamaños de red similares y con un coste por nodo de cómputo igual o inferior. Por tanto, hemos justificado la existencia de un escenario de aplicación de los toros concentrados en el que resultan más rentables en rendimiento, coste y escalabilidad que las Dragonflies, y con un tamaño razonable para un sistema de HPC.

Además de la presente memoria, este trabajo ha dado lugar a la publicación de un póster (revisión por pares) en el 41 congreso ICPP (International Conference on Parallel Processing) que se celebrará del 10 al 15 de Septiembre de 2012, y del que se adjunta el abstract al final de esta memoria.

Por último, este trabajo ha requerido un gran número de horas de simulación para poder alcanzar los resultados mostrados. Cada punto de la curva de un tamaño y topología concretos ha requerido varias horas, por lo que cada gráfica ha representado la ejecución del simulador de 2 a 3 días. Por este motivo, no ha sido posible simular otros muchos casos interesantes que constituyen las bases del trabajo futuro, junto con posteriores líneas de desarrollo que surjan de los avances en otros ámbitos. Al estar en curso un trabajo más ambicioso centrado en la mejora del rendimiento de las redes Dragonfly, es de esperar que muchos de los avances obtenidos se contrasten con mejoras en los toros concentrados.

## 5.2. Trabajo futuro

Las principales líneas de trabajo futuro se centran en mejoras y modificaciones del simulador de red, para ampliar el rango de evaluaciones. Una primera directiva pasa por el empleo de diversos patrones de tráfico sintético, para que los resultados obtenidos sean más representativos. Se intentará emplear patrones que resulten adversos para el comportamiento de la red, de modo que podamos establecer una comparativa de peor caso. Asimismo, se evaluará el empleo de otros mecanismos de evitación de deadlock en los toros concentrados, como por ejemplo el uso de datelines.

También se plantea realizar mejoras sobre el simulador de cara a su uso en otros trabajos dentro del grupo. La simulación de redes Dragonfly no densas con múltiples enlaces entre un mismo par de grupos constituyen un punto relevante tanto en la comparativa con los toros concentrados como en el estudio de tamaños no maximales para dicha topología.

Otro punto de investigación pasa por un análisis más profundo de los parámetros tecnológicos y su repercusión sobre el rendimiento de la red. En la actualidad, el consumo energético de los sistemas de HPC está alcanzando una fuerte relevancia. El coste de instalación del sistema puede ser ampliamente superado por los costes de mantenimiento y uso, tales como el consumo del sistema y de los medios de disipación del calor. La profundización en el estudio del diseño tecnológico del router impacta en dichos costes.

## 5.3. Evaluación personal

Este trabajo me ha permitido profundizar en el análisis sobre el comportamiento de las redes de interconexión, tanto de forma individual como en sus consecuencias sobre el sistema. Durante la realización de mi PFC pude estudiar dichas redes aplicadas sobre arquitecturas CMP (Chip Multi-Processor). En este trabajo el análisis ha fluido hacia el ámbito de los sistemas de HPC, que me ha resultado muy interesante. Se trata de un área en el que me gustaría seguir trabajando, pues su importancia es indudable. Además, los retos que plantea suponen un desafío y un estímulo para cuantos participan en él.

También me ha brindado la oportunidad de participar en la redacción de un póster y su correspondiente abstract, con todas las dificultades que acarrea pero también la satisfacción de aportar un estudio interesante a la comunidad de investigadores.

En último lugar, he de destacar que este trabajo se fundamenta en un importante esfuerzo previo realizado por diversos integrantes del grupo ATC. Se han aprovechado los desarrollos existentes y se han intentado aportar diversos cambios que ayuden en los diversos proyectos que se sigan realizando. El trabajo en equipo ha sido fundamental y supone un aspecto indispensable en el ejercicio de todo buen ingeniero.

Por estas razones, junto con el amplio rango de líneas futuras que se suscitan y que se han comentado en estas conclusiones, espero poder seguir en esta línea de investigación y ampliar los esfuerzos vertidos en este trabajo.

# Bibliografía


[1] "Mellanox IS5022," accessed 23-August-2012. [Online]. Available: http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=89&menu_section=49/

[2] "NetShelter AR3100," accessed 23-August-2012. [Online]. Available: http://www.42u.com/apc-42u-rack.htm#APC_NetShelter_AR3100

[3] "Supermicro TwinBlade," accessed 23-August-2012. [Online]. Available: http://www.supermicro.com/products/Superblade/TwinBlade/

[4] "Top500 June 2012 report," accessed 23-August-2012. [Online]. Available: http://top500.org/list/2012/06/100

[5] N. Adiga, M. Blumrich, D. Chen, P. Coteus, A. Gara, M. Giampapa, P. Heidelberger, S. Singh, B. Steinmacher-Burow, T. Takken *et al.*, "Blue Gene/L torus interconnection network," *IBM Journal of Research and Development*, vol. 49, no. 2.3, pp. 265–276, 2005.

[6] A. Benner, "Optical interconnect opportunities in supercomputers and high end computing," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference*, march 2012, pp. 1 –60.

[7] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," 2008.

[8] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip," *Circuits and Systems Magazine, IEEE*, vol. 4, no. 2, pp. 18–31, 2004.

[9] M. Campbell-Kelly, "Origin of computing," *Scientific American Magazine*, vol. 301, no. 3, pp. 62–69, 2009.

[10] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[11] W. Dally, "Virtual-channel flow control," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 3, no. 2, pp. 194–205, 1992.

[12] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1320–1331, 1993.

[13] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection networks: An engineering approach*. Morgan Kaufmann, 2003.

[14] N. Farrington, E. Rubow, and A. Vahdat, "Data center switch architecture in the age of merchant silicon," in *Proceedings of the 2009 17th IEEE Symposium on High Performance Interconnects*. Washington,DC,USA: IEEE Computer Society, 2009, p. 93–102.

[15] J. Flich and D. Bertozzi, *Designing Network On-Chip Architectures in the Nanoscale Era*. Chapman & Hall/CRC, 2010.

[16] M. García, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, G. Rodrıguez, J. Labarta, and C. Minkenberg, "On-the-Fly adaptive routing in high-radix hierarchical networks," *ICPP*, 2012.

[17] P. Gratz, C. Kim, K. Sankaralingam, H. Hanson, P. Shivakumar, S. Keckler, and D. Burger, "On-chip interconnection networks of the TRIPS chip," *IEEE Micro*, pp. 41–50, 2007.

[18] K. Gunther, "Prevention of deadlocks in packet-switched data transport systems," *Communications, IEEE Transactions on*, vol. 29, no. 4, pp. 512–524, 1981.

[19] S. Haghighi and N. Songer, "Multiprocessor system including a docking system," Aug. 20 2002, uS Patent 6,438,622.

[20] J. L. Hennessy and D. A. Patterson, *Computer Architecture, Fourth Edition: A Quantitative Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006.

[21] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a teraflops processor," *Micro, IEEE*, vol. 27, no. 5, pp. 51–61, 2007.

[22] P. Kermani and L. Kleinrock, "Virtual Cut-Through: a new computer communication switching technique," *Computer Networks*, vol. 3, pp. 267–286, 1979.

[23] J. Kim, W. Dally, S. Scott, and D. Abts, "Cost-efficient dragonfly topology for large-scale systems," *Micro, IEEE*, vol. 29, no. 1, pp. 33–40, 2009.

[24] J. Kim, W. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," *COMPUTER ARCHITECTURE NEWS*, vol. 35, no. 2, p. 126, 2007.

[25] J. Kim, W. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," *ACM SIGARCH Computer Architecture News*, vol. 36, no. 3, pp. 77–88, 2008.

[26] L. E. LaForge, K. F. Korver, and M. S. Fadali, "What designers of bus and network architectures should know about hypercubes," *IEEE Trans. Computers*, vol. 52, no. 4, pp. 525–544, 2003.

[27] R. M. Metcalfe and D. R. Boggs, "Ethernet: distributed packet switching for local computer networks," *Commun. ACM*, vol. 19, no. 7, pp. 395–404, Jul. 1976.

[28] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb, "The Alpha 21364 network architecture," in *Proceedings of the The Ninth Symposium on High Performance Interconnects*. Washington,DC,USA: IEEE Computer Society, 2001, pp. 113–.

[29] L.-S. Peh and N. E. Jerger, *On-Chip Networks*, 1st ed. Morgan and Claypool Publishers, 2009.

[30] M. J. Pertel, "A critique of adaptive routing," Pasadena,CA,USA, Tech. Rep., 1992.

[31] V. Puente, C. Izu, R. Beivide, J. Gregorio, F. Vallejo, and J. Prellezo, "The adaptive bubble router," *Journal of Parallel and Distributed Computing*, vol. 61, no. 9, pp. 1180–1208, 2001.

[32] K. Shim, M. Cho, M. Kinsy, T. Wen, M. Lis, G. Suh, and S. Devadas, "Static virtual channel allocation in oblivious routing," in *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*. IEEE Computer Society, 2009, pp. 38–43.

[33] J. Thornton, "Considerations in computer design— leading up to the control data 6600," *Control Data Corp*, 1963.

[34] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. Miao, J. Brown, and A. Agarwal, "On-chip interconnection architecture of the tile processor," *Micro, IEEE*, vol. 27, no. 5, pp. 15–31, 2007.

# Comparison study of scalable and cost-effective interconnection networks for HPC

Pablo Fuentes, Enrique Vallejo, Carmen Martínez, Marina García, Ramón Beivide
Department of Electronics and Computing
University of Cantabria
pablo.fuentes@alumnos.unican.es, {enrique.vallejo, carmen.martinez, marina.garcia, ramon.beivide}@unican.es

***Abstract*—This work attempts to compare size and cost of two network topologies proposed for large-radix routers: concentrated torus and dragonflies. We study and compare the scalability, cost and fault tolerance of each network. On average, we found that a concentrated torus can be a cost-efficient option for middle-range networks.**

***Index Terms*—concentrated torus; network topology; dragonfly;**

## I. Introduction

In recent years multiple research fields have increased their computing requirements. Bioengineering (protein folding), Defense (cryptanalysis), or Meteorology (weather forecasting) are examples of this trend. There exists a general effort for pursuing the Exaflop ($10^{18}$ floating point operations per second) [1], [2], [3]. The importance of the interconnection network increases with the system size. Since it can significantly condition performance and power consumption, new topologies are under study. The *Dragonfly* is an example of this search for a scalable and high performance network [4].

Despite this need for highly scalable networks, cost-efficiency is a restrictive factor. In this context, the design of many supercomputers is strongly constrained by both cost and power consumption. The use of *commodity* technology lowers the deployment costs of the systems. The Gordon Supercomputer [5], from San Diego Supercomputer Center, currently reaches $48^{th}$ position in Top500 list using only commodities, both in processor and network and a concentrated Torus topology. Such topology has been studied before in [6].

Network cost strongly depends on routers and longest wires, which must be implemented on optic fiber. However, these dependencies can vary relying on the amount of wires needed. Concentrated tori may happen to be a possibility to increase network performance and size while keeping costs below a reasonable limit. This is due to the lack of optic cables for the interconnection.

In a concentrated torus with $D$ dimensions every router is attached to $p$ compute nodes and to other $2D$ routers. Network connectivity is improved by assigning $t$ physical wires to each topological link between routers (where $t$ is called *"trunking factor"*), thus increasing the amount of traffic the network can cope with.

*Bisection BandWidth* (BBW) is a metric typically employed with regards to network dimensioning. BBW is defined as the aggregate bandwidth of a minimum cut which divides the network in two equal halves. Assuming a uniform traffic pattern, BBW should be enough to carry the traffic generated by the nodes in one side with destination to the other partition. If BBW is lower than such aggregated traffic, the network is *oversubscribed*. Oversubscription lowers network costs, but makes the network a performance bottleneck unless locality is heavily exploited. In this work we will consider networks *without* oversubscription. In such case, the concentrated torus parameters ($D$, $t$ and the number of routers per dimension) restrict the maximum number of compute nodes attached to routers and thus the system scalability. In the present paper we compare network scalability, cost and connectivity for both concentrated torus and dragonflies.

## II. Network scalability

We focus our work on symmetric torus, where the number of routers in any network dimension is the same, denoted by $r$. The routers on each dimension are connected as a ring, and in that case the following relation holds to prevent oversubscription

$$\frac{p}{2}\frac{r}{2} \leq 2t \Rightarrow pr \leq 8t.$$

Therefore, the maximum number of nodes $N$ will be upper bounded by

$$N = pr^D = \frac{(pr)^D}{p^{D-1}} \leq 8t\left(\frac{8t}{p}\right)^{D-1}.$$

The largest configuration is obtained with the minimum $p$ (no concentration) and maximum $t$ (maximum trunking), which is obvious since wider links increase BBW. However, a sensible restriction is to assume that $p \geq t$ to avoid links overdimension with respect to computing nodes. In that case, the maximum size is obtained when the following expressions fulfill, where $k$ denotes the router degree.

$$N = 8^D t \text{ and } t = p = \left\lfloor \frac{k}{2D+1} \right\rfloor$$

In comparison, a balanced dragonfly (as described in [7]) reaches about $\frac{k^3}{16}$ nodes. Figure 1 presents the maximum achievable network size with different router degree, using both balanced Dragonflies and concentrated tori with 1 to 4 dimensions $D$. In this chart we restrict to $D < 5$ since larger number of dimensions make layout very complex; in fact, machines using $D \geq 5$ (such as BlueGene/Q or the K-Computer) use asymmetric designs to be able to wrap multiple

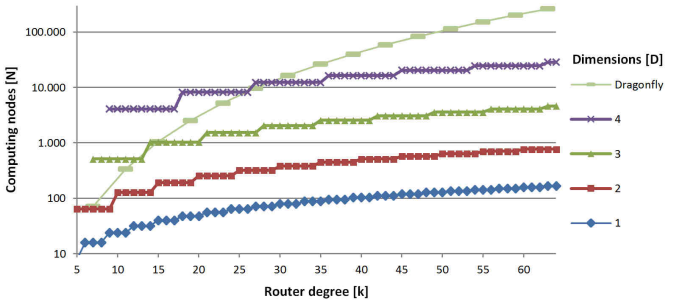



Fig. 1. Maximum scalability for a router size

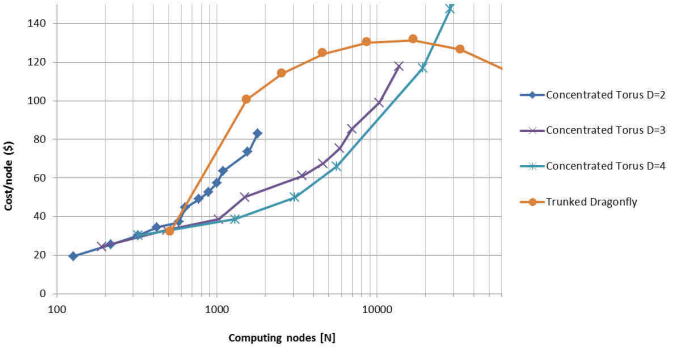



Fig. 2. Cost comparison using router size 64

dimensions within a single rack. Router degree is ranged below 64 ports, which is achievable with current or near future network fabrics for commodities. Figure 1 shows that 4D concentrated tori can reach a higher number of computing nodes without oversubscription when the router degree is below 27 ports.

## III. Cost comparison

Network cost distinguishes between two components. Using the component prices from [8] we estimate that each router has an estimated cost of 390$. The links connecting the nodes to the routers will be 1m length (4,53$). All links in a concentrated torus can use electric wires. The length of links between routers depends on network size considering a folded torus layout and the cabinet distribution. Cost of Dragonfly is calculated similarly to [7], using trunking between groups of 512 nodes. Note that in large sizes the cost diminishes because there are unused global ports due to the rounding in the trunking factor calculation. Dragonfly networks differ to tori because they have some very long optical links (with a price of approximately 220$ per cable). Figure 2 displays the network cost per node built from 64-port routers. Note that for a size bellow 20.000 computation nodes it is cheaper to build a 3D or 4D concentrated torus using large-radix switches than a dragonfly.

## IV. Router Connectivity

The probability of failures grows with the size of the computing system which makes fault tolerance an aspect of great importance in network design. As a first approach for evaluating a topology fault tolerance properties it is usual to take into account the node connectivity. Hence, in this section we consider the router connectivity in both concentrated torus and dragonfly topologies. Therefore, to compare both networks what we look is for the number of links which have to fail in order to completely disconnect a router.

In the case of concentrated $D$-dimensional torus, any router is connected by $2tD$ links to the $2D$ neighbour routers. Therefore, these number of links is exactly the connectivity of any router in the network, which corresponds to a proportion of $\frac{2D}{2D+1}$ to the total number of links. On the other hand, a similar calculation but for the dense dragonfly yields to a ratio of approximately $\frac{3}{4}$ of the links that have to be removed in order to disconnect completely one router. As a consequence, the toroidal topology provides for a greater router connectivity in any case. In particular, the ratio for concentrated 3D torus is $\frac{6}{7}$ while the one for the dragonfly is $\frac{3}{4}$ of the links.

## V. Conclusions and future lines

In this paper we have explored a network topology for high-performance computing which we call concentrated torus. We have compared its size scalability and costs against dragonfly, which is a relatively novel topology expected to be both highly scalable and cost-efficient. Moreover, a first approximation to fault tolerance capacity of the toroidal topology has been made by calculating the router connectivity. As we have seen, the figures of merit that we have considered suggest that concentrated tori could be a good rival in medium-sized networks. Our future goals are to obtain performance charts via network simulation, using synthetic traffic loads and traces.

## Acknowledgment

This work has been supported by the Spanish Ministry of Science under contracts TIN2010-21291-C02-02 and CONSOLIDER Project CSD2007-00050, and by the European HiPEAC Network of Excellence.

## References

[1] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," 2008.

[2] E. C. Joseph, S. Conway, C. Ingle, G. Cattaneo, C. Meunier, and N. Martinez, "A strategic agenda for european leadership in supercomputing: HPC 2020," 2010.

[3] P. Dongarra, "Beckman et al.the international exascale software roadmap. volume 25, number 1, 2011," *International Journal of High Performance Computer Applications*, pp. 77–83.

[4] J. Kim, W. Dally, S. Scott, and D. Abts, "Cost-efficient dragonfly topology for large-scale systems," *Micro, IEEE*, vol. 29, no. 1, pp. 33–40, 2009.

[5] M. L. Norman and A. Snavely, "Accelerating data-intensive science with Gordon and Dash," in *Proceedings of the 2010 TeraGrid Conference*, ser. TG '10. New York, NY, USA: ACM, 2010, pp. 14:1–14:7. [Online]. Available: http://doi.acm.org/10.1145/1838574.1838588

[6] J. Navaridas and J. Miguel-Alonso, "Indirect cube: A power-efficient topology for compute clusters," *Optical Switching and Networking*, vol. 8, no. 3, pp. 162 – 170, 2011.

[7] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *ISCA '08*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 77–88.

[8] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *ISCA '07*. New York, NY, USA: ACM, 2007, pp. 126–137.