Redes de interconexión para sistemas masivamente paralelos

Informe de investigación
EHU-KAT-IK-05-04

José Miguel-Alonso

Departamento de Arquitectura y Tecnología de Computadores
UPV/EHU

Resumen

La mayor parte de los sistemas masivamente paralelos están basados actualmente en componentes de cómputo de consumo: se emplean procesadores, o incluso nodos completos, iguales a los utilizados en los computadores personales. Sin embargo, no se ha llegado aún a este nivel de reutilización de tecnologías de consumo en lo referido a las redes de interconexión: las económicas LAN no ofrecen un rendimiento apropiado, o no escalan lo suficiente, por lo que se necesitan diseños específicos.

En este trabajo hacemos una evaluación de algunas alternativas de diseño para redes de interconexión específicamente concebidas para procesadores masivamente paralelos: aquellos que constan de varios miles de nodos. Esta exploración se hace a partir de un simulador funcional, que no recoge en detalle el coste hardware que las alternativas propuestas tendrían en un diseño real. Sin embargo, la experiencia de nuestro grupo de investigación facilita el descarte de aquellas alternativas con alto coste hardware, siendo por tanto conservadores en nuestras propuestas de diseño.

Trabajamos en todo momento con redes directas. La conmutación es virtual cut-through. El encaminamiento es adaptativo por caminos mínimos (aunque también se emplea encaminamiento estático en orden de dimensión en algunos canales virtuales). Para evitar interbloqueos se opta por la técnica de la burbuja. Estas decisiones iniciales, todas ellas aceptadas como buenas en cuanto a su rendimiento, reducen notablemente el ámbito de nuestro estudio.

Los factores que se exploran son la topología de la red (malla vs. toro, 2D vs. 3D), las políticas de selección y arbitraje de canales[1], el número de canales virtuales (mecanismo usado fundamentalmente para aumentar el rendimiento), el tamaño de burbuja (mecanismo para evitar interbloqueos), la longitud de las colas de tránsito, y el tamaño del paquete.

La propuesta final que se ha obtenido a través de este estudio es la de usar toros 3D con selección SMART_REQ, dos canales virtuales adaptativos y burbuja de tamaño 2. El sistema de arbitraje apenas tiene impacto en el rendimiento: basta elegir uno que sea justo y no plantee problemas de inanición (OLDEST_ARB y ROUNDROBIN_ARB son buenas opciones). El tamaño del paquete tampoco afecta de forma sustancial al rendimiento, por lo que se puede elegir el más adecuado para cada aplicación. En cuanto a la longitud de las colas de tránsito, el retardo es menor usando colas cortas, por lo que no se considera necesario incorporar mucha memoria en los encaminadores.

Los resultados obtenidos en este trabajo, pendientes de refinamientos adicionales, coinciden en gran medida con los escasos datos aparecidos recientemente en la literatura sobre el sistema BlueGene/L de IBM, quizá el mayor de los computadores paralelos actualmente en construcción.

[1] Estos conceptos se describirán con detalle más adelante.

Índice

1 Introducción 4
1.1 Motivación y objetivos 4
1.2 La simulación como herramienta 5
1.3 Estudios a realizar 6
1.4 Organización este trabajo 6
2 El simulador funcional 7
2.1 Estructura del encaminador simulado 7
2.2 Parámetros de entrada 9
2.3 Salida generada por el simulador 10
2.4 Patrones de tráfico 11
3 Metodología de experimentación con el simulador 13
4 Topología y encaminamiento 15
4.1 Topologías para procesadores paralelos de gran tamaño 15
4.2 Encaminamiento 17
4.3 Experimentos sobre topología y encaminamiento 18
5 Políticas de selección y de arbitraje 22
5.1 Política de selección 22
5.2 Arbitraje 23
5.3 Experimentos sobre selección y arbitraje 25
6 Número de canales virtuales 27
6.1 Razones para incorporar canales virtuales 27
6.2 Experimentos sobre el número de canales virtuales 28
7 Tamaño de burbuja 29
7.1 Evitación del interbloqueo 29
7.2 Experimentos sobre el tamaño de la burbuja 31
8 Longitud de las colas 32
9 Tamaño del paquete 33
10 Conclusiones y líneas de trabajo abiertas 35
10.1 Conclusiones 35
10.2 Líneas abiertas 36
11 Referencias 37
12 Agradecimientos 38
Apéndice A: Visión general del BlueGene/L 39
Apéndice B: Visión general del Red Storm 43

1 Introducción

1.1 Motivación y objetivos

Actualmente están en fase de diseño varias máquinas con el objetivo de alcanzar potencias de cálculo del orden del PetaFLOP/s. Salvo excepciones, el planteamiento es construir estas máquinas basándose en componentes hardware de consumo y software libre. Desde nuestro punto de vista, lo más interesante de estos diseños es la red de interconexión, puesto que aquí no se recurre a elementos de consumo: las LAN actuales, aunque tienen unas elevadas prestaciones en cuanto a ancho de banda (1 Gb/s y, en breve, 10 Gb/s) introducen unos retardos muy elevados. Por lo tanto, se recurre a redes diseñadas específicamente para estos sistemas.

El propósito de este trabajo es explorar alternativas de diseño de redes de interconexión para sistemas paralelos de gran tamaño (varios miles de nodos), y contrastar estas propuestas con la mínima información disponible sobre las máquinas en construcción, dedicando especial atención a dos: BlueGene/L [ADI 2002] y Red Storm [CT 2003].

IBM tiene en marcha en el proyecto BlueGene, toda una saga de máquinas que, en unos pocos años, alcanzarán la potencia de cálculo del PetaFLOP/s tomada como objetivo. La primera de estas máquinas, prevista para 2005, es el BlueGene/L (BG/L para abreviar). La información disponible sobre el BG/L es, de momento, escasa: un artículo general [ADI 2002], y otro sobre la administración del sistema [ALM 2003]. El primer artículo indica que se publicará "elsewhere" más información sobre la red de interconexión, objeto principal de interés de nuestro grupo de investigación. Hasta lo que sabemos, no ha sido así. Sin embargo, en algunos workshops en EE.UU., organizados por los grandes laboratorios de investigación (LLL, Sandia, LANL, etc.) se han realizado presentaciones sobre esta futura máquina; entre ellos, [BAA 2002] y [CHS 2003]. También se dispone de información obtenida en conferencias impartidas sobre el tema: [KLE 2003]. No sabemos con detalle, por lo tanto, cómo está diseñada esta máquina, pero sí vamos sabiendo cosas, como que la red de interconexión es un toro 3D, con encaminamiento adaptativo, y que usa la técnica de canales de escape con control de flujo por burbuja [PUE 2001] para evitar interbloqueos.

La información sobre Red Storm (RS para abreviar), sistema que está siendo construido por Cray, es aún menor. La referencia más completa encontrada es [CT 2003]. En ella se indica que se va a optar por una interconexión específicamente diseñada para esta máquina, aunque se considera la posibilidad de recurrir a la tecnología QsNet de Quadrics [PET 2002] (con modificaciones) si es necesario. En cuanto a la topología, optan por una malla 3D.

La experiencia en el diseño de redes de interconexión del grupo del que formo parte, avalada además por ser los inventores de la técnica de la burbuja, nos ha permitido lanzarnos a esta iniciativa: intentar predecir, mediante simulación, el comportamiento de redes de interconexión para sistemas de gran tamaño, como pueden ser las del BG/L y RS, así como realizar aportaciones para mejorar el diseño de estas redes.

1.2 La simulación como herramienta

Este es un trabajo basado en simulación: se trata de explorar alternativas de diseño con el objetivo de obtener buenos rendimientos. Si un estudio mediante simulación (en condiciones ideales) no ofrece mejoras respecto a alternativas conocidas, no merece la pena pensar en su implementación. Una vez que se tiene una propuesta "aparentemente" válida, sería necesario volver a comprobarla mediante simuladores más detallados (y, por lo tanto, mucho más lentos) o mediante la implementación en hardware de prototipos.

Como base de este trabajo podríamos pensar en utilizar SICOSYS [PGB 2002], un potente y flexible simulador de redes de interconexión desarrollado por nuestro grupo en la Universidad de Cantabria. Sin embargo, este simulador es *demasiado* detallado y, por lo tanto, lento. SICOSYS es una herramienta excelente de predicción de rendimientos de redes, con un coste muy inferior al que tendría un simulador hardware, pero inviable para redes del tamaño previsto para el BG/L (65.536 nodos): el consumo de memoria y de tiempo de CPU se dispararía.

Por lo tanto, hemos optado por realizar otro simulador (funcional), esta vez en la UPV/EHU que, sin modelar tan detalladamente el hardware, nos de ideas del comportamiento (y, por lo tanto, del rendimiento) de una red de interconexión, de tal forma que se puedan evaluar con él diferentes alternativas de diseño. La ventaja de esta aproximación es poder simular, en un procesador, una red de miles de nodos—algo imposible con SICOSYS. El inconveniente es que ciertas alternativas de diseño que parezcan óptimas en una simulación pueden tener, en la práctica, un costo inaceptable en cuanto a complejidad o retardo en su implementación hardware. Aún así, la experiencia del grupo nos llevará a proponer ideas que sean implementables en hardware. Una evolución de este trabajo iría por la evaluación del coste hardware de aquellas propuestas más prometedoras[2].

El simulador funcional empleado en los experimentos llevados a cabo es un programa secuencial, en el que el tiempo avanza ciclo a ciclo (dirigido por el tiempo). Se podría pensar en realizar simulación paralela. Al fin y al cabo, tenemos experiencia en esta área [MIG 1996]: dado el tipo de modelo, con un gran número de eventos que suceden en cada instante de tiempo, se podría realizar una simulación conservadora o incluso una simulación sincronizada SPED (Syncronous Parallel Event Driven), ambas fácilmente implementables en un sistema de paso de mensajes [MIG 1998]. El uso de simuladores paralelos optimistas no parece recomendable; de hecho, en [UGS 2002] se comenta que, para el diseño del BG/L, se ha analizado el comportamiento de un simulador paralelo optimista denominado SPEEDES, consiguiendo un speedup de tan sólo 8 con 128 procesadores.

En cualquier caso, hemos optado por el simulador secuencial, empleando las máquinas paralelas simplemente para aumentar la productividad. Esto nos permite lanzar multitud de simulaciones al mismo tiempo, con el inconveniente de tener que esperar unas 10-15 horas para completar una tanda de experimentos. Otro inconveniente añadido es la cantidad de memoria consumida por el simulador: para una red 3D de 20x20x20 nodos se necesitan cerca de 2 GB de RAM, y la mayoría de nuestras máquinas disponen tan sólo de 1 GB. Por esta razón, nuestros experimentos se limitan

[2] En buena medida, [PUE 2001] ya contiene una evaluación del coste hardware de mecanismos como los propuestos en este trabajo.

a redes de 4096 nodos (16x16x16). Para simular redes mayores será necesario ampliar la memoria o, definitivamente, paralelizar el simulador.

En la bibliografía sobre el BG/L [HB 2002], se describe cómo IBM está usando en la etapa de diseño un simulador "preciso casi al ciclo", con el que se han probado aspectos como el número y tamaño de los canales virtuales y diferentes políticas de arbitraje de canales. Se indica que funciona en paralelo en un SMP, con buen speedup, y que simula 0,5 microsegundos del BG/L por segundo[3] (toro grande, tráfico denso), usando patrones de generación de tráfico que vienen dados por seudo-códigos y trazas UTE [UTE 1995]. También se cita un banco de pruebas multi-nodo usando VHDL, para verificar el diseño hardware. En un artículo más reciente, se ofrecen datos sobre un simulador del sistema completo, que corre sobre un cluster Linux [CEZ 2003].

1.3 Estudios a realizar

Son muchos los parámetros de diseño que se pueden modificar a la hora de buscar una buena red de interconexión para un sistema masivamente paralelo. Como compendio de todos ellos, resulta muy recomendable leer [DYN 2003] donde, además, se puede encontrar una excelente lista de referencias.

La experiencia del grupo, así como las tendencias observadas en la industria, nos ha llevado a fijar algunos de esos parámetros, y centrarnos en sólo unos cuantos. Estas son las características de los sistemas simulados:

- Sólo trabajaremos con redes directas.
- Utilizaremos un número reducido de patrones de tráfico, básicamente uniforme y hotspot.
- Conmutación: virtual cut-through [KK 1979].
- Topologías: mallas y toros de baja dimensión (2, 3).
- Evitación de interbloqueo mediante burbuja [PUE 2001].
- Encaminamiento estático en orden de dimensión (primero en X, luego en Y, luego en Z) o adaptativo por caminos mínimos.
- Uso de canales virtuales (CV) [DAL 1992] para incrementar el rendimiento y evitar problemas de inanición [PUE 2001].

Fijados estos parámetros básicos, y considerando diferentes patrones de tráfico, en las siguientes secciones llevaremos a cabo los siguientes estudios:

- Selección de topología. Sección §4.
- Políticas de selección del canal virtual de salida (para encaminamiento adaptativo) y de arbitraje de canales de salida. §5.
- Número de canales virtuales. §6.
- Tamaño de la burbuja. §7.
- Tamaño de las colas de tránsito. §8.
- Tamaño del paquete. §9.

1.4 Organización este trabajo

Tras esta introducción, en §2 se facilitan detalles sobre el simulador empleado. En §3 se describe la metodología de experimentación utilizada para la realización de este trabajo. A partir de ahí, vienen los experimentos descritos en el apartado anterior,

[3] Se necesitan más de 23 días para simular un segundo del funcionamiento del BG/L.

quedando para §10 las conclusiones y líneas de trabajo abiertas. Se incluyen, además, dos apéndices con información sobre los proyectos BlueGene/L y Red Storm.

2 El simulador funcional

El Grupo de Paralelismo de la UPV/EHU, con la colaboración con el Grupo de Arquitectura y Tecnología de Computadores (ATC) de la Universidad de Cantabria, ha desarrollado un simulador de redes de interconexión, que nos permite explorar diferentes alternativas de diseño de encaminadores y redes. Las limitaciones fundamentales de este simulador vienen derivadas de su carácter funcional: no trata de emular el detalle hardware de un encaminador. Por lo tanto, no sirve para darnos idea precisa de la complejidad de la implementación en hardware de las propuestas de diseño que se hagan, ni del impacto que puedan tener esas propuestas en los tiempos de ciclo.

2.1 Estructura del encaminador simulado

La Figura 1 representa la estructura del encaminador simulado. Se ha procurado que las decisiones tomadas sobre el diseño del simulador estén cerca, en todo momento, de soluciones hardware de sencilla implementación, bajo coste y utilidad contrastada. Algunas de ellas vienen marcadas por la experiencia del grupo en estas tareas. Por ejemplo:

- La técnica de conmutación es virtual cut-through [KK 1979].
- No se realizan políticas de selección y arbitraje globales que consideren todas las entradas y todas las salidas, para tomar una decisión óptima (al menos, dentro del nodo). Cada una de las entradas toma sus propias decisiones a la hora de solicitar salidas, y cada una de las salidas toma sus propias decisiones a la hora de arbitrar entradas.
- Las colas de tránsito se han colocado a la entrada, porque esta alternativa tiene una implementación más sencilla que otras: sólo se necesita un puerto de escritura.
- Puesto que algunas combinaciones topología/encaminamiento pueden llevar a situaciones de interbloqueo, se implementa el mecanismo de la burbuja para evitarlas [PUE 2001] (véase §7 para más detalles). Una alternativa, de mayor complejidad, podría estar basada en el uso de canales virtuales [DS 1987].
- Se incorporan canales virtuales [DAL 1992] con encaminamiento adaptativo por caminos mínimos para mejorar el rendimiento y evitar problemas de inanición, frente a redes con un único canal por enlace físico (véase §6 para más detalles).
- Cada paquete se etiqueta en origen con un registro de encaminamiento que se va modificando conforme se acerca a su destino.

El número de dimensiones lo puede elegir el usuario: la figura corresponde a un encaminador para una red 3D, pero puede optarse por trabajar con redes 1D y 2D. Se puede seleccionar si el encaminamiento es estático o adaptativo: si se opta por el estático, por cada canal físico sólo se emplea el canal marcado con "E" (Escape). El número de CV adaptativos es variable, aunque en la figura aparecen dos ("A1" y "A2").

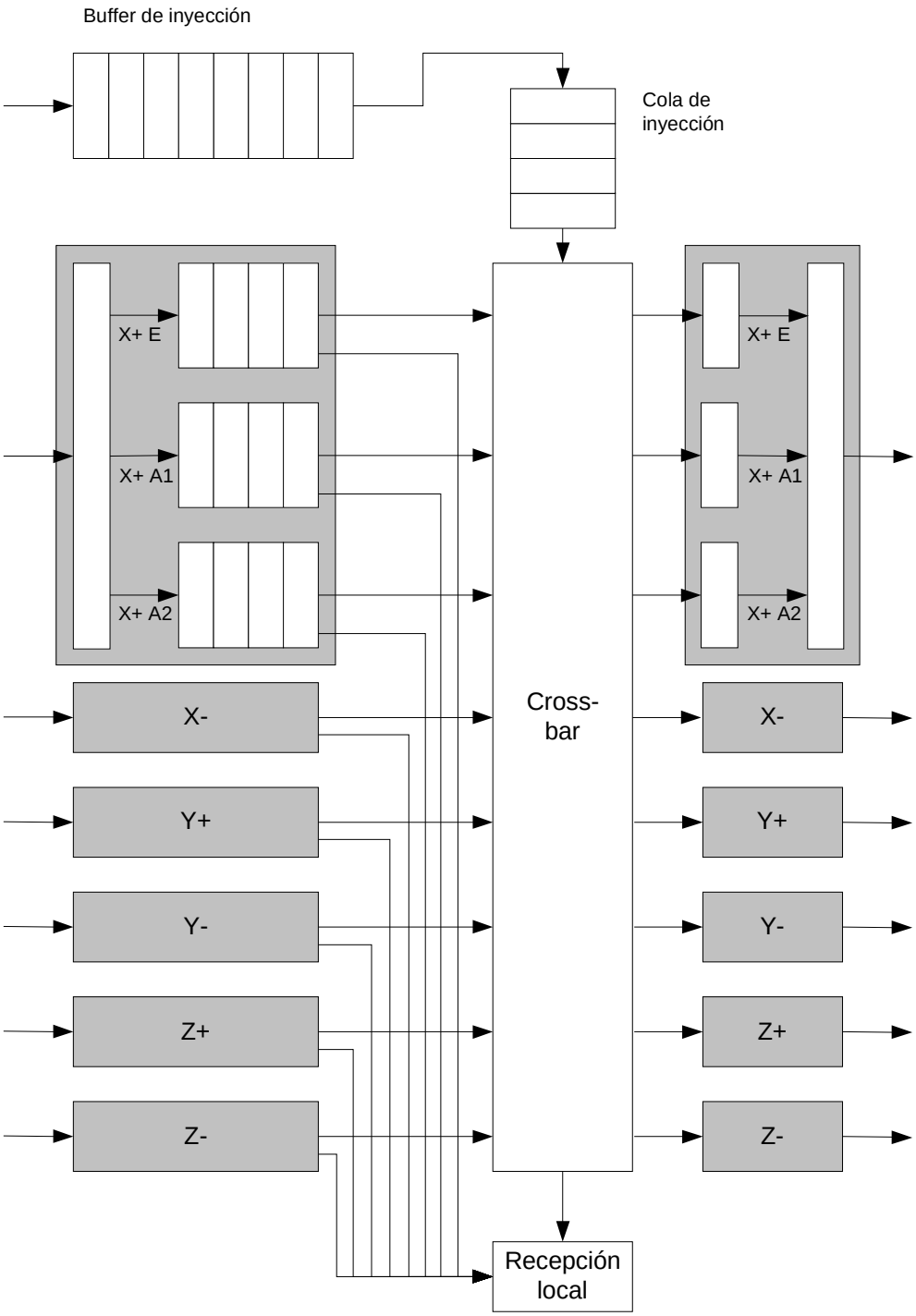

Figura 1. Estructura lógica del encaminador simulado, válida para una red (toro o malla) 3D, con encaminamiento adaptativo en 2 canales virtuales, y un canal virtual adicional de escape con encaminamiento estático.

La anchura de los buses es de un *phit*. Los paquetes pueden ser de 1 ó más phits (aunque el número es fijo para cada simulación). En la práctica, el phit será 8, 16, ... bits. Nótese que todos los CV de una dimensión-sentido, aunque comparten los mismos canales físicos, tienen sus propias colas. La cola de inyección es idéntica a las de tránsito (el puerto de inyección es uno más), pero dispone de un búfer adicional para que el nodo de cómputo (no simulado) deposite en él los paquetes a enviar.

En cuanto al puerto de recepción, puede estar configurado en dos modos: consumo sencillo (sólo se consume un phit por ciclo) o múltiple (se puede consumir, en el mismo ciclo, un phit de cada cola), aunque en este trabajo sólo se considera el consumo múltiple.

La unidad de tiempo manejada por el simulador es el ciclo. No se relaciona este ciclo con los de reloj de un dispositivo real, ni con los retardos que sufre un phit al atravesar un dispositivo real. Se supone que en un ciclo da tiempo a realizar todas estas operaciones en todos y cada uno de los nodos simulados[4]:

- Aceptar (mover) un phit por cada canal físico de entrada y por el canal de inyección local
- Realizar todas las solicitudes de salidas desde todos los canales virtuales de entrada
- Arbitrar a qué canal virtual de entrada se concede cada uno de los canales virtuales de salida
- Consumir los phits correspondientes a paquetes que han alcanzado su destino

Es decir, que un ciclo es el tiempo necesario para que un phit "salte" desde la cola de un puerto en un nodo hasta la cola de otro puerto en otro nodo. Además, en cada ciclo se realizan tareas de recopilación de información para la obtención de datos estadísticos.

2.2 Parámetros de entrada

El simulador es muy flexible en cuanto a su parametrización, permitiéndonos explorar un amplio rango de alternativas de diseño. Esta es una lista de los parámetros que se pueden dar en línea de comandos:

1. Semilla para la generación de números aleatorios, que se utilizan principalmente para la generación del tráfico.
2. Número máximo de ciclos de simulación.
3. Número máximo de paquetes a inyectar en el sistema. La simulación termina cuando se reciban todos los paquetes inyectados, o cuando se alcance el número máximo de ciclos especificado con el parámetro anterior.
4. Longitud (en phits) de los paquetes.
5. Tamaño (en paquetes) de la burbuja. Necesario cuando se utilizan combinaciones de topología y encaminamiento que puedan llevar a interbloqueo. Un valor 0 indica que no se activa el mecanismo de la burbuja.
6. Topología: toro (1, 2, 3 dimensiones), malla (idem.) o midimew[5] (sólo 2D).
7. Nodos por dimensión. El tamaño de la red simulada se calcula elevando este valor al número de dimensiones.
8. Carga aplicada, medida en phits por ciclo y por nodo. Se describe con más detalle en §2.4.
9. Estrategia de encaminamiento: estático, adaptativo.
10. Patrón de inyección de tráfico: uniforme, hotspot, matriz transpuesta y distribución. Más adelante (§2.4) veremos la definición de estos términos.
11. Modo de consumo: sencillo (en cada ciclo se puede consumir localmente un phit) o múltiple (en cada ciclo se pueden consumir, simultáneamente, tantos phits como canales tiene el nodo).

[4] El simulador no utiliza técnicas de segmentación, habituales en los diseños hardware reales.
[5] En este trabajo no exploramos esta topología.

12. Modos de selección de un canal virtual de salida: RANDOM_REQ, SHORTEST_REQ, SMART_REQ. Estos términos los definimos con detalle en §5.1.
13. Modos de arbitraje: ROUNROBIN_REQ, OLDEST_REQ, LONGEST_REQ, RANDOM_REQ. Estos términos se describen en §5.2.
14. Tamaño de las colas de tránsito, medido en paquetes.
15. Tamaño del búfer de inyección, medido en paquetes.

Otros parámetros no pueden ser seleccionados como argumentos al ejecutar el simulador, sino que deben ser facilitados en tiempo de compilación. Son estos:

1. Número de sentidos (enlaces uni- o bi-direccionales).
2. Número de dimensiones: 1, 2 ó 3 (sólo 2 para midimews).
3. Número de canales virtuales por enlace físico. Si se indica V, uno es el de escape y los V-1 restantes son adaptativos.
4. Tamaños máximos (en phits) de las colas de tránsito y de inyección.
5. Nivel de depuración: cuánta información de salida se genera. Los valores habituales son 0 (cuando termina la simulación se obtiene sólo un informe-resumen) y 1 (se obtiene, además, información sobre el uso de los canales). Los niveles 2 y 3 son útiles en las fases de adición o modificación de código.

2.3 Salida generada por el simulador

El resultado de la simulación, una vez detenida por haber alcanzado el número fijado de ciclos o por haber recibido el número de paquetes prefijado, es un informe en formato texto, que resume los datos de entrada más relevantes y recoge una serie de estadísticas con los resultados. Durante la fase de depuración del simulador se puede optar por generar una salida mucho más amplia, pero que no es relevante para realizar estudios de rendimiento. Este es un ejemplo de un informe normal (nivel 1):

```
************************************************************
Random seed:                        13
Topology, uni(1)/bidir(2):          torus 2
Dimensions, nodes/dim:              2, 3 = 9 total nodes
Traffic pattern:                    uniform, packets of 1 phits
Traf./Inj. queue len (pack/ph):     8/8, 1000/1000
Routing:                            adaptive, 2 channels, bubble = 2 phits
Consumption mode:                   multiple
Request mode:                       shortest (only for adaptive routing)
Arbitration mode:                   random
Started at:                         Wed Jul 09 14:15:33 2003
Ended at:                           Wed Jul 09 14:15:34 2003
************************************************************
Simulation clock:                        5000
AvDistance:                           1.50318
Packets  Inj.   Rcv.   Drop.:          37245      28173       3121
Load     Prov.  Inj.   Acc.:          0.90000    0.82767    0.62607
Delay    Avg.   StDev. Max.:        758.46846  435.99838       1539
InjDelay Avg.   StDev. Max.:        757.21162  436.02265       1534
************************************************************
Histogram for Node 1, Port 0: [4999 0 0 0 0 0 0 0 0 ]
Histogram for Node 1, Port 1: [3417 1493 78 11 0 0 0 0 0 ]
Histogram for Node 1, Port 2: [4999 0 0 0 0 0 0 0 0 ]
Histogram for Node 1, Port 3: [3369 1511 110 9 0 0 0 0 0 ]
Histogram for Node 1, Port 4: [4999 0 0 0 0 0 0 0 0 ]
Histogram for Node 1, Port 5: [3678 1257 56 6 2 0 0 0 0 ]
Histogram for Node 1, Port 6: [4999 0 0 0 0 0 0 0 0 ]
Histogram for Node 1, Port 7: [3697 1258 39 5 0 0 0 0 0 ]
Histogram for Node 1, Port 8: [10 1 10 13 4 8 6 3080 1867 ]
Table of destinations for node 1:
```

```
489 0 567 537 478 539 510 512 489
Table of sources for node 1:
357 0 372 376 380 388 382 347 414
Table of source ports for node 1:
0 544 0 574 0 954 0 944
Table of destination ports for node 1:
0 1027 0 949 0 573 0 566
```

La primera sección es la de entrada, e indica los parámetros bajo los cuales ha operado el simulador.

La segunda sección es la relevante cuando se hacen estudios de rendimiento. Indica los ciclos simulados, el número de paquetes procesados, la carga total (phits por ciclo por nodo) inyectada/aceptada por el simulador, el retardo de tránsito experimentado por los paquetes, y el retardo de inyección.

La tercera (opcional: sólo si el nivel de depuración es igual o superior a 1) da una serie de datos adicionales que nos sirven para observar el uso que se ha hecho de las colas asociadas a los puertos de entrada. Para el caso del ejemplo, esta es la relación entre los números y los puertos:

Núm. →	0	1	2	3	4	5	6	7	8
Puerto →	X+E	X+A	X-E	X-A	Y+E	Y+A	Y-E	Y-A	Inyec.

Se trata de una red bidimensional (por eso no hay dimensión Z), un toro de 3x3 nodos, con dos canales virtuales por canal físico: uno de escape y otro adaptativo. El histograma indica el uso que se ha hecho de los canales en el nodo 1[6]. Fijémonos en el puerto 1 (X+ adaptativo). De los 5000 ciclos simulados, ha estado 3417 con su cola completamente vacía, 1493 con un paquete en la cola, 78 con dos paquetes en la cola, 11 con tres paquetes en la cola y ningún ciclo con 4 ó más paquetes en la cola.

Durante la simulación, el nodo 1 ha enviado 489 paquetes al nodo 0, ningún paquete a sí mismo, 567 paquetes al nodo 2, 537 paquetes al nodo 3, etc. En cuanto a la recepción, ha recibido 357 paquetes del nodo 0, ningún paquete de sí mismo, 372 paquetes del nodo 2, 376 del nodo 3, etc.

De todos los paquetes que el nodo 1 ha enviado, 0 han salido por el puerto X+E, 544 por el X+A, 0 por el X-E, 574 por el X-A, 0 por el Y+E, 954 por el Y+A, 0 por el Y-E, 944 por el Y-A. De todos los paquetes que el nodo 1 ha recibido, ninguno ha venido por canales de escape, 1027 han llegado por X+A, 949 por X-A, 573 por Y+A y 566 por Y-A. Esto nos indica que los canales adaptativos han sido capaces de absorber todo el tráfico, y nunca ha sido necesario derivar parte del mismo a los canales de escape.

2.4 Patrones de tráfico

El simulador trabaja con patrones de tráfico sintéticos (queda, como línea de trabajo abierta, añadir la posibilidad de trabajar con trazas, o conectado a un simulador de sistema para trabajar con tráfico generado por aplicaciones reales). Describimos a continuación los patrones implementados.

[6] En el caso de toros y midimews, y tráfico uniforme, todos los nodos exhiben un comportamiento similar, por lo que da lo mismo el nodo elegido para su representación. En cualquier caso, el usuario—previa recompilación—puede decidir qué nodo o nodos desea observar en detalle.

Partimos de que tenemos una red de N nodos en total, identificados desde 0 a N-1. A cada nodo le corresponden, además, unas coordenadas (x), (x,y) o (x,y,z), en función del número de dimensiones de la red. A partir de aquí, definimos estos cuatro patrones de tráfico:

- **Uniforme**: cada nodo envía, cuando le corresponde, un paquete a un nodo elegido al azar (mediante una distribución uniforme) entre todos los demás nodos.

- **Hotspot**[7]: a la hora de elegir destinos, el 25% del tráfico va al primer 12,5% de los nodos. El 75% del tráfico se dirige al resto de los nodos. Los primeros nodos son aquellos que tiene como identificadores los valores 0 a X-1, donde X = Nx0,125.

- **Matriz transpuesta**: el nodo de coordenadas (x,y) envía siempre paquetes al nodo (y,x). En el caso de redes 3D existen dos posibles transpuestas [KMS 1997]: (x,y,z)→(y,z,x) y (x,y,z)→(z,x,y); se ha implementado la primera.

- **Distribución**: a la hora de elegir destinos, el nodo n siempre usa esta secuencia: n+1, n+2, ... N-1, 0, 1, 2, ... n-1, n+1, etc.

El simulador decide, en cada ciclo, si a un nodo le corresponde inyectar o no. Esta decisión depende de la carga que se ha facilitado como parámetro de entrada. Esta carga es un valor entre 0 y 1, e indica cuántos phits por ciclo genera cada nodo. Una carga de 1, la máxima, indica que todos los nodos están constantemente intentando inyectar.

A partir de los valores de entrada L (carga a intentar inyectar, medida en phits por ciclo y por nodo) y M (Longitud de los paquetes, medida en phits) se genera un L' = L/M. L' es la carga a inyectar, en paquetes por ciclo y por nodo. En cada ciclo se visitan todos y cada uno de los nodos, para realizar el proceso de generación de tráfico, de esta manera:

1. Se genera un número aleatorio r entre 0 y 1
2. Si r < L'
    a. Si hay espacio en el búfer de inyección, se inserta en él un paquete (completo) siguiendo el patrón de tráfico que corresponda
    b. En caso contrario se incrementa un contador de paquetes "dropped"

En la información de IBM [HB 2002] se indica que, en las evaluaciones de su red, se han utilizado los patrones uniforme y hotspot tal y como los hemos expuesto aquí. Otros experimentos están hechos a partir de trazas de aplicaciones.

[7] Esta definición del tráfico hotspot está tomada de [HB 2002]. En la literatura nos podemos encontrar otras. Lo más habitual es definir el tráfico hotspot como aquel en el que un cierto porcentaje de todos los paquetes generados en la red va destinado a un único nodo, no a una zona.

3 Metodología de experimentación con el simulador

En las tablas y gráficos de este trabajo centramos nuestra atención en dos indicadores del rendimiento de la red:

- La **carga aceptada**: cuántos phits por ciclo ha consumido, como promedio, cada nodo. La **carga aplicada**, como ya se ha dicho, es un parámetro de entrada que indica cuántos phits por ciclo y por nodo se intentan inyectar en el sistema. Cuando la red no está saturada, la carga aplicada y la aceptada deben ser prácticamente idénticas. Cuando la red ha alcanzado la saturación, la carga aceptada es menor que la aplicada. También se mide, aunque no se recoge en este trabajo, la carga inyectada: cuántos paquetes por ciclo y por nodo han sido aceptados por el sistema, lo que suma a los paquetes ya consumidos aquellos que están en tránsito, ocupando colas y búferes.

- El **retardo** de tránsito medio experimentado por los paquetes, medido en ciclos. Mide el tiempo desde que el paquete entra en el sistema (colocándose en un búfer de inyección) hasta que su último phit es consumido en el nodo de destino. También se mide (aunque no se representa) el retardo de inyección, que indica cuánto tiempo, como promedio, se pasan los paquetes en el búfer y la cola de inyección, antes de empezar a transitar por los nodos que lo llevarán a destino.

En el informe de salida, el contador *dropped* indica el número de mensajes que se han intentado inyectar sin éxito por no haber espacio en el búfer de inyección, lo que indica que se ha llegado a un punto de saturación (que puede ser temporal y localizada), y justifica la diferencia entre carga aplicada y carga aceptada.

En un sistema teórico con búfer infinito, cuando la carga aplicada es mayor que la aceptada el retardo no llegaría a estabilizarse, sino que crecería constantemente. En el simulador la saturación se detecta a través del contador dropped, y el retraso de tránsito se estabiliza al cabo de un tiempo. En un sistema real no es posible que se inyecte más que lo que la red acepta, luego las situaciones de saturación se reflejan en que las aplicaciones tienen que bloquearse (y perder tiempo) a la espera de que la red acepte paquetes; por lo tanto, el retardo real de un paquete sería la suma del experimentado en la red más el tiempo que el nodo de cómputo ha estado bloqueado a la espera de inyectarlo.

En casi todos los experimentos se ha optado por aplicar la máxima carga posible (1), para saturar el sistema y comprobar qué carga es capaz de aceptar, y con qué retardo. Estos parámetros varían con el tiempo simulado: el sistema pasa por un periodo transitorio, hasta alcanzar un régimen estacionario. En este trabajo, nos centramos en el estacionario, aunque los transitorios no dejan de tener su interés.

Con el fin de que los resultados de la simulación sean significativos, se han tomado estas dos medidas:

1. Se ha ejecutado el mismo experimento con varias semillas aleatorias. Se ha comprobado que no hay variación significativa en los resultados obtenidos. Finalmente, se ha optado por presentar los realizados con semilla 13.

2. En cuanto a la duración de los experimentos, puesto que nos interesa el régimen estacionario, se ha hecho un estudio que nos indique cuándo se alcanza este régimen. Ese momento depende de diferentes parámetros de entrada, por lo que no sirve un valor único. Sin embargo, en todos los casos explorados, una ejecución de 150.000 ciclos de simulación (para una red de 16x16x16 nodos) resulta suficiente. Puesto que la exploración no ha sido completa, se ha optado por ejecutar todos los experimentos durante 200.000 ciclos.

A modo de ejemplo de la efectividad de la primera medida, la Tabla 1 recoge los resultados obtenidos para una colección de experimentos, todos ellos con los mismos parámetros[8] pero con diferentes semillas aleatorias.

Semilla	Carga aceptada	Retardo
13	0,47748	2250,77171
17	0,47758	2249,92775
71	0,47751	2250,18423
43	0,47749	2249,37316
11	0,47750	2250,46264

Tabla 1. Experimentos sobre el efecto de la semilla de generación de números aleatorios en la estabilidad de los resultados.

Para ilustrar la segunda medida, se presentan unas gráficas (Figura 2 y Figura 3) con la evolución de los parámetros de rendimiento (carga aceptada y retardo) a lo largo del tiempo de simulación[9]. Lo más significativo de estas gráficas es que la estabilidad se consigue mucho antes con el tráfico uniforme, mientras que para el tráfico hotspot se producen oscilaciones, que van siendo menores según avanza el tiempo (simulado).

En [HB 2002] se realizan experimentos similares a estos con el simulador del BG/L, con resultados también similares, destacando que el régimen transitorio tiene menor duración cuanto más cortas son las colas de tránsito.

[8] Datos de la simulación: 200.000 ciclos, paquetes de 32 phits, burbuja tamaño 2, topología toro 16x16x16, carga aplicada máxima, encaminamiento adaptativo (2 CV adaptativos), tráfico uniforme, consumo múltiple, selección SMART_REQ, arbitraje OLDEST_ARB, colas de 8 paquetes, búfer de inyección de 16 paquetes.

[9] Los datos de la simulación son los mismos del experimento anterior, salvo que se consideran dos patrones de tráfico: uniforme (U) y hotspot (HS).

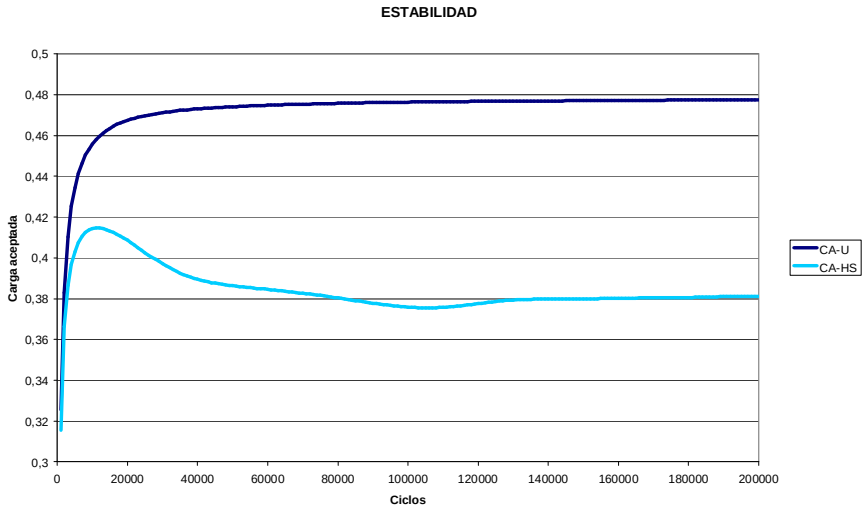

Figura 2. Evolución de la carga aceptada a lo largo del tiempo de simulación. Una ejecución de 200.000 ciclos nos coloca en un régimen estacionario. Este régimen se alcanza mucho antes para el caso de tráfico uniforme.

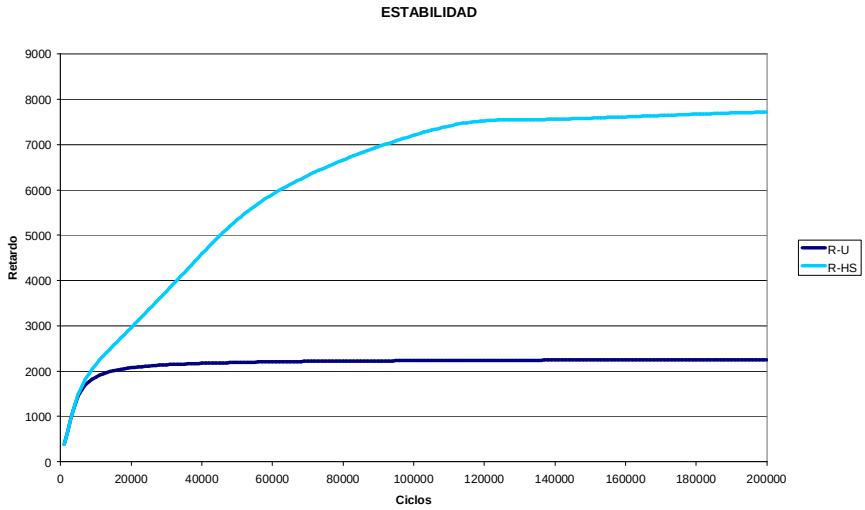

Figura 3. Evolución del retardo medio de los paquetes a lo largo del tiempo de simulación.

4 Topología y encaminamiento

4.1 Topologías para procesadores paralelos de gran tamaño

La topología, aunque no el único, es un importante parámetro de diseño. En la literatura sobre el tema se han propuesto muchas topologías distintas (véase el proyecto investigador), aunque las tendencias más recientes han ido hacia dos grandes grupos:

- Redes multietapa
- Redes directas tipo n-cubo k-ario

Ejemplos de las primeras son el sistema IBM SP Switch (que forma parte del IBM RS6000/SP) [IBM] y las SAN comercializadas por Myricom (Myrinet) [MYR] y Quadrics (QsNet). Ejemplos de las segundas son [DYN 2003] [HP 2003]:

- Cray T3D (toro 3D, hasta 1024 nodos 8x16x8, 300 MB/s por enlace)
- Cray T3E (toro 3D, hasta 2048 nodos, 600 MB/s por enlace)
- Intel Delta (malla 2D, 540 nodos, 40 MB/s por enlace)
- Intel Paragon (malla 2D, hasta 2048 nodos, 175 MB/s por enlace)
- Intel ASCI Red (malla 2D, 2536 nodos, 800 MB/s por enlace)
- MIT M-Machine (malla 3D, 800 MB/s por enlace)

Entre las máquinas que están en fase de diseño, IBM ha optado por un toro 3D para el BG/L, mientras que Cray ha seleccionado una malla 3D para RS (véase el apéndice).

Fijando nuestra atención en las redes directas, diferentes investigadores han realizado trabajos para seleccionar los valores más adecuados de n y de k. El trabajo de Dally [DAL 1990] indicaba que, considerando las limitaciones impuestas por el cableado de una red directa, la topología óptima es el toro 2D, para patrones de tráfico uniforme y hotspot. Un trabajo posterior de Agarwal [AGA 1991] matiza estas conclusiones, indicando que, si se tiene en cuenta el retardo introducido en los nodos (tanto por tiempos de conmutación como por contención), las mejores topologías son las de 3 ó incluso 4 dimensiones.

En cuanto a la decisión de si es adecuado o no disponer de enlaces periféricos (mallas frente a toros), es importante señalar que las mallas presentan ciertos inconvenientes:

- El ancho de banda de la bisección es la mitad, a pesar de tener el mismo número de puertos por encaminador.
- Esta topología no es simétrica, lo que lleva a un uso no homogéneo de los recursos. Los nodos periféricos procesan menos paquetes que los centrales.
- Si se usa encaminamiento adaptativo, con ciertos patrones de tráfico (como el uniforme) existe una tendencia a que los paquetes se acumulen hacia el centro de la red, convirtiéndolo de hecho en una zona caliente.

También presentan ciertas ventajas:

- No son necesarios los cables periféricos, por lo que todos los enlaces son de la misma longitud. Esta aparente ventaja queda mitigada si se configuran los enlaces de forma no convencional: un toro plegado (Figura 4) tiene todos los enlaces del mismo tamaño—aunque estos enlaces son más largos que los que tendría una malla con el mismo número de nodos. Una figura en [COT 2003] sugiere que el BG/L usa esta estrategia de tendido de cables.
- Otra ventaja de las mallas es que su particionado es trivial. Partir un toro 3D exige tener mucho cuidado con el cálculo de los registros de encaminamiento y, posiblemente, perder capacidad de movimiento (por los enlaces periféricos) en alguna de las dimensiones.
- El encaminamiento estático en las mallas está libre de interbloqueos, puesto que con esta topología no aparecen ciclos. En el toro no ocurre esto, así que se complica el encaminador, al requerir un mecanismo para evitar este problema. Sin embargo, la técnica de la burbuja, desarrollada por nuestro grupo, es (además de efectiva) muy sencilla, lo que permite su implementación en toros con muy poco coste hardware añadido.

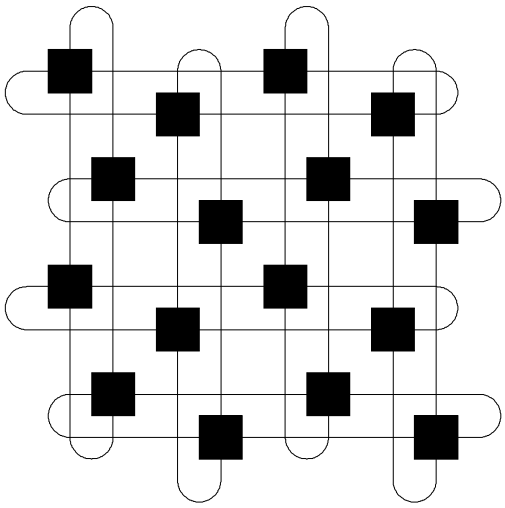

Figura 4. Un toro plegado 2D de 4x4 nodos. Nótese cómo las longitudes de todos los enlaces son muy similares.

Es de suponer que IBM y Cray han tenido en cuenta estas consideraciones a la hora de elegir sus topologías. Tanto el BG/L como el RS están diseñadas para poder ser particionadas. La compartición en estas máquinas, de hecho, está basada en la asignación de trozos de la misma para su uso en exclusiva por un cierto trabajo: no se usa *time-sharing* sino *space-sharing*. El BG/L puede dividirse en trozos con topología anillo (toro 1D), toro 2D, y toro 3D con algunos enlaces periféricos eliminados. El RS puede dividirse fácilmente en submallas 3D.

4.2 Encaminamiento

Se podrían escribir varios tratados sobre métodos de encaminamiento, con las ventajas e inconvenientes relativos de cada uno. En [DYN 2003] hay un capítulo dedicado en exclusiva a este tema, que incluye un amplio número de referencias adicionales para profundizar en él. De todas las propuestas existentes, nos quedamos con dos tipos de encaminamiento: estático DOR y adaptativo por caminos mínimos.

En el momento en el que se inyecta un paquete, el nodo de origen calcula (y le inserta en la cabecera) su *registro de encaminamiento*: un vector <hx, hy, hz>[10] que indica cuántos saltos "hacia delante" (por el lado positivo de un eje) o "hacia atrás" (por el lado negativo) son necesarios para llevar el paquete a destino. El cálculo del registro de encaminamiento depende de las coordenadas de los nodos de origen y de destino, así como de la topología. Conforme el paquete va dando saltos de nodo a nodo, su registro de encaminamiento va siendo modificado, actualizándose el número de saltos restante. Cuando el registro vale <0, 0, 0> es que el paquete ha completado su camino.

Partiendo de este registro de encaminamiento configurado en origen, se pueden definir estos dos algoritmos de encaminamiento:

1. **Estático** ("oblivious") en orden de dimensión, también conocido como DOR (Dimension Order Routing). Este algoritmo exige que primero se den todos los

[10] Estoy suponiendo una topología 3D. Las componentes hx, hy y hz del registro de encaminamiento son números enteros. Sea el registro <2, 0, -1>. Esto indica que el mensaje tiene que dar dos saltos hacia adelante por el eje X, ninguno por el Y, y uno hacia atrás por el eje Z.

saltos en el eje X, hasta que hx = 0. Después se dan los saltos en el eje Y. Por último, los saltos en el eje Z. Este algoritmo es muy sencillo de implementar, y distribuye bien el tráfico por toda la red. Además, como ya se ha comentado, en el caso de las mallas está libre de interbloqueos.

2. **Adaptativo** por caminos mínimos. Este algoritmo permite avanzar en cualquiera de los ejes posibles, siempre que su correspondiente campo en el registro de encaminamiento sea distinto de cero, y siempre acercando el paquete a su destino[11]. El mayor inconveniente del encaminamiento adaptativo es que introduce el riesgo de aparición de interbloqueos, por lo que no se puede llevar a la práctica si no se dota a los encaminadores de mecanismos de tratamiento de esta anomalía. Otro inconveniente es que dos paquetes consecutivos que salgan de un mismo nodo origen hacia un mismo destino pueden llegar desordenados (al circular por diferentes rutas), lo que obliga a realizar procesos de etiquetado en origen y reordenación en destino.

Para todos los experimentos descritos en este trabajo, cuando se indique la estrategia de encaminamiento empleada, debe entenderse lo siguiente:

1. Si se indica que el encaminamiento es estático, sólo existe un canal por enlace físico: el de escape.
2. Si se indica que el encaminamiento es adaptativo, cada acceso a un canal físico se organiza en varios canales virtuales—se indicará el número. El primero de ellos es el CV de escape, y el resto son adaptativos. Siempre se usa encaminamiento estático en los CV de escape, por lo que el encaminamiento adaptativo se limita a los CV adaptativos.

4.3 Experimentos sobre topología y encaminamiento

Hemos realizado varias tandas de experimentos para evaluar el impacto de la topología y el encaminamiento en el rendimiento de una red de interconexión, con estos parámetros fijos: 200.000 ciclos de simulación; paquetes de 32 phits; red de 16x16x16=4096 nodos; selección SMART_REQ, arbitraje OLDEST_ARB[12]; tamaño de las colas: 8 paquetes; tamaño del búfer de inyección 16 paquetes.

Parámetros variables:

- Toplogía: toro, malla
- Burbuja: 2 en el caso del toro (para evitar interbloqueos). La malla no necesita burbuja
- Carga aplicada: entre 0,1 y 0,5, y máxima.
- Encaminamiento: estático, adaptativo (2 CV adaptativos)
- Patrón de tráfico: uniforme, hotspot

En la Tabla 2 se recogen los resultados de estos experimentos. Se han hecho representaciones gráficas de los mismos, organizadas por patrón de tráfico, en la Figura 5 (uniforme) y en la Figura 6 (hotspot).

[11] Otros algoritmos adaptativos permiten "dar rodeos" y transportar los mensajes por rutas más largas que la distancia entre origen y destino (*misrouting*).
[12] Experimentos posteriores mostrarán la idoneidad de estas políticas de selección y arbitraje.

Carga aplicada	Toro				Malla			
	E. Estático		E. Adaptativo		E. Estático		E. Adaptativo	
	T-E-U	T-E-H	T-A-U	T-A-H	M-E-U	M-E-H	M-A-U	M-A-H
0,1	0,09988	0,09988	0,09988	0,09989	0,09987	0,09987	0,09987	0,09988
0,2	0,19984	0,19984	0,19986	0,19990	0,19151	0,18164	0,19968	0,19949
0,3	0,29933	0,28696	0,29975	0,29971	0,19394	0,18430	0,22819	0,21800
0,4	0,32771	0,29369	0,39932	0,36566	0,19441	0,18428	0,22957	0,21901
0,5	0,32765	0,29361	0,47526	0,37412	0,19452	0,18398	0,22960	0,22009
1,0	0,32686	0,29428	0,47748	0,38120	0,19395	0,18402	0,22926	0,22052

Tabla 2. Carga aceptada para diferentes combinaciones topología-encaminamiento. Claves: T=Toro, M=Malla. A=encaminamiento Adaptativo, E=encaminamiento Estático, U=tráfico Uniforme, H=tráfico Hotspot.

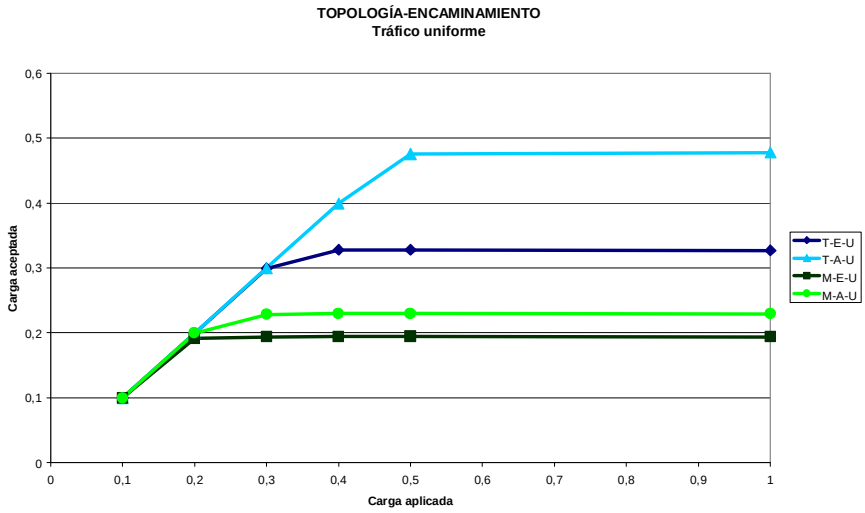

Figura 5. Carga aceptada para diferentes combinaciones topología-encaminamiento, para tráfico uniforme.

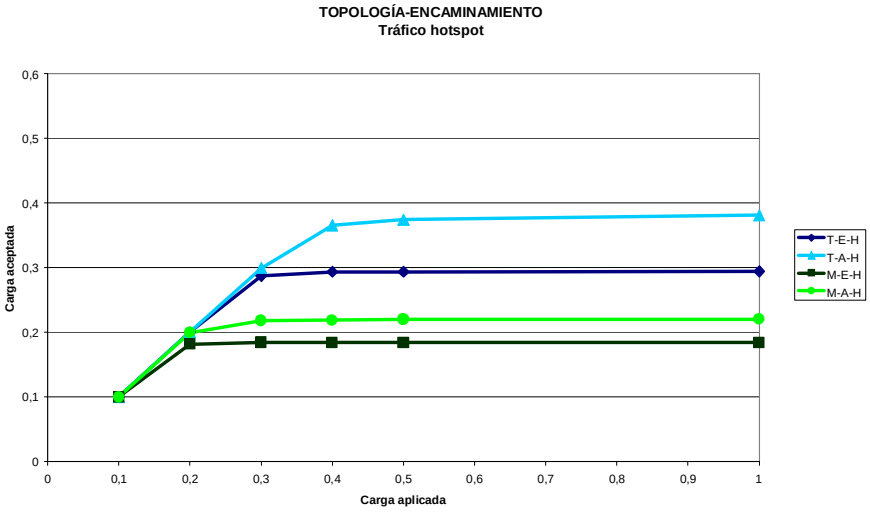

Figura 6. Carga aceptada para diferentes combinaciones topología-encaminamiento, para tráfico hotspot.

Los datos anteriores fijan la atención en uno sólo de los parámetros del rendimiento: la carga aceptada. En la Figura 7 y la Figura 8 representamos (gráficamente) el otro indicador de interés: el retardo experimentado por los paquetes.

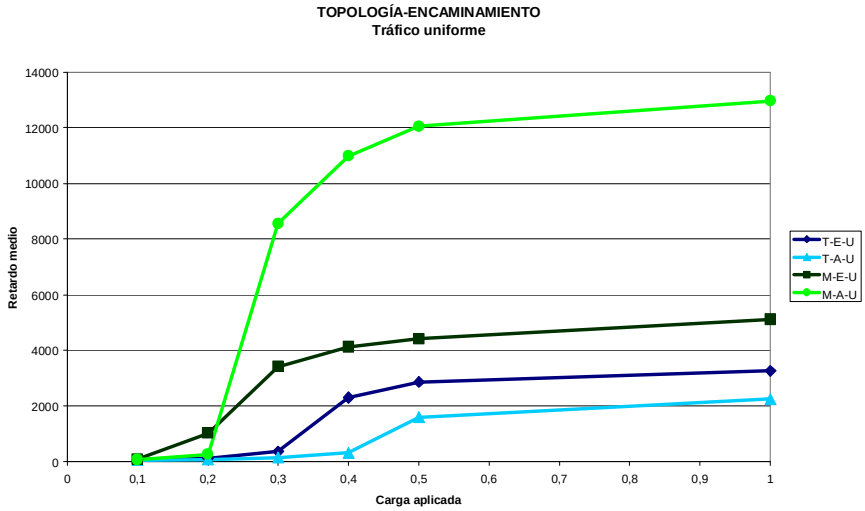

Figura 7. Retardo experimentado por los paquetes para diferentes combinaciones de topología-encaminamiento. Tráfico uniforme.

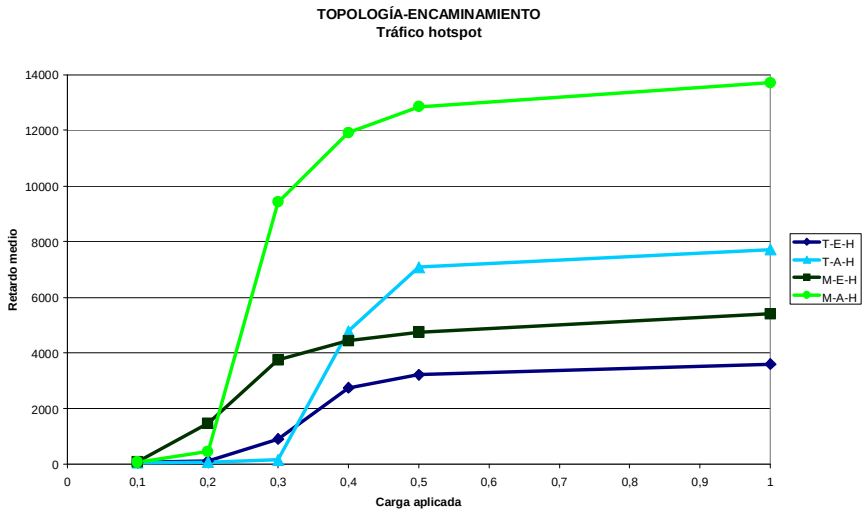

Figura 8. Retardo experimentado por los paquetes para diferentes combinaciones de topología-encaminamiento. Tráfico hotspot.

La primera conclusión que se puede extraer de estos experimentos es que los indicadores de rendimiento son mucho mejores para el tráfico uniforme que para el hotspot. Es lo que cabe esperar, ya que el tráfico uniforme distribuye de manera más homogénea los paquetes por la red. Esto es más notorio en el caso del toro que en la malla, porque en el primer caso la homogeneidad es total (el toro es una topología simétrica) pero en el segundo los nodos de los bordes procesan menos mensajes que los de la parte central.

Centrándonos en la topología, se comprueba que el toro siempre ofrece un rendimiento mejor que la malla: para la misma configuración encaminamiento-tráfico, la carga máxima aceptada es aproximadamente el doble, y el retardo experimentado por los paquetes es bastante menor. Sin embargo, debe resaltarse que bajo nuestro esquema experimental la malla está infravalorada con respecto al toro. El hecho de tener ecualizados los cables, y de no necesitar mecanismos específicos de evitación de interbloqueos, son características muy apropiadas para diseñar encaminadores que operen a mayores frecuencias.

En lo referente al encaminamiento, el adaptativo ofrece mejores niveles de rendimiento que el estático, siendo la mejora aportada mucho más marcada en el toro que en la malla. También es notable que con este encaminamiento, cuando el sistema llega a la saturación, los retardos son mucho mayores que con encaminamiento estático. Este efecto no se debe a que el comportamiento del encaminamiento adaptativo sea peor, sino a que el encaminador adaptativo tiene mucho más espacio en sus colas (el triple, puesto que usa tres canales virtuales, dos de ellos adaptativos) que el estático. En §6 se estudia este fenómeno, observándose que, a igualdad de memoria local, los retardos en saturación son prácticamente idénticos.

Por otra parte, se aprecia la estrecha relación que existe entre el patrón de tráfico y la mejora de rendimiento obtenida al usar encaminamiento adaptativo. Hemos hecho estudios adicionales para el toro 3D, con los patrones de tráfico distribución y matriz transpuesta antes descritos, y los hemos recogido en la Tabla 3 junto con los de los patrones uniforme y hotspot. Las ganancias en carga aceptada son: 46% para tráfico uniforme, 30% para tráfico hotspot, 15% para tráfico distribución, y 116% para tráfico matriz transpuesta. Los excelentes resultados el último patrón son fáciles de explicar. En él, cada nodo A (x,y,z) envía mensajes siempre al mismo destino B (y,z,x). El encaminador estático siempre usa la misma ruta entre A y B, ruta que puede coincidir parcialmente con las de otras parejas de nodos. El adaptativo puede elegir rutas alternativas, reduciendo la contención en los segmentos compartidos por varias rutas.

	E. Estático		E. Adaptativo	
Tráfico	Carga aceptada	Retardo	Carga aceptada	Retardo
uniforme	0,32686	3273,04496	0,47748	2250,77171
hotspot	0,29428	3596,82503	0,38120	7719,64221
distribución	0,21584	4968,35457	0,24865	11962,77242
transpuesta	0,06758	6123,94708	0,14585	17761,20799

Tabla 3. Rendimiento de los toros 3D, para diferentes patrones de tráfico, con encaminamiento estático y adaptativo.

Hasta ahora sólo hemos considerado redes 3D. ¿Cómo sería el rendimiento con redes 2D y encaminamiento adaptativo? Dejando a un lado las posibles ventajas derivadas de la reducción del tiempo de ciclo en el encaminador, la Tabla 4 es un resumen de los resultados que podemos esperar con experimentos similares a los anteriores, para toros y mallas 2D con el mismo número de nodos (64x64 = 4096 nodos). La carga aplicada es la máxima. Como se puede comprobar, las cargas aceptadas son mucho menores para 2D que para 3D. Del mismo modo, los retardos son mucho más elevados.

Topología	Tráfico	Carga aceptada	Retardo
Toro 3D	Uniforme	0,47748	2250,77171
	Hotspot	0,38120	7719,64221
Malla 3D	Uniforme	0,22926	12964,64002
	Hotspot	0,22052	13711,91949
Toro 2D	Uniforme	0,10181	28064,78968
	Hotspot	0,09013	31511,21032
Malla 2D	Uniforme	0,05189	26309,56402
	Hotspot	0,04995	27783,19217

Tabla 4. Comparativa entre topologías 2D y 3D, para redes con 4096 nodos.

5 Políticas de selección y de arbitraje

5.1 Política de selección

Sea un paquete en la cabeza de una cola asociada a un cierto puerto de entrada E. Si ese paquete ha alcanzado su destino final, podrá ser consumido. Si no es así, tendrá que avanzar a otro nodo. Con el fin de acercar el paquete a su destino, tendrá que seleccionarse, dentro de los caminos posibles, un cierto puerto de salida S (Figura 9).

En el caso de que encaminamiento sea estático, la selección de S es trivial: sólo se considera la opción que determina el algoritmo DOR. Sin embargo, si el encaminamiento es adaptativo, la decisión es más difícil. Se tienen estas opciones (siempre sujetas a que el registro de encaminamiento las permita):

- Elegir un puerto de salida S correspondiente a un CV adaptativo, en la misma dimensión y sentido en la que está circulando el paquete. Puede haber varios candidatos.
- Elegir un S correspondiente a un CV adaptativo, pero cambiando de dimensión. De nuevo, puede haber varios candidatos.
- Elegir un S correspondiente al CV de escape indicado por el algoritmo DOR. Aquí sólo cabe un candidato.

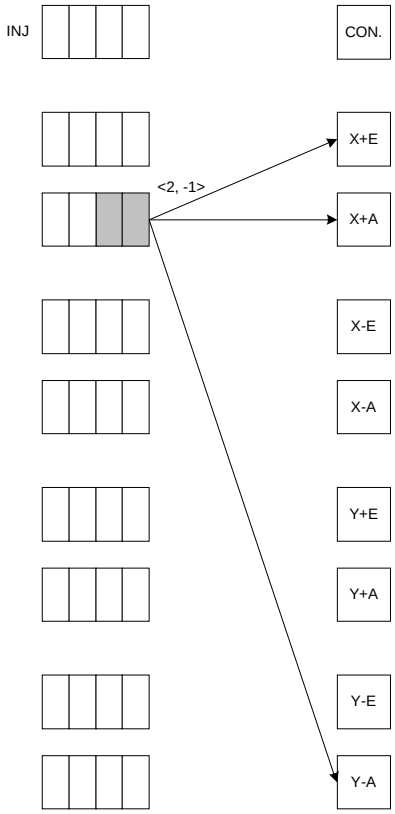

Figura 9. Proceso de selección, para una red 2D con un canal virtual adaptativo. El puerto de entrada X+A tiene un paquete en tránsito, con destino a un nodo que dista dos saltos en el eje X+ y uno en el eje Y-. Las flechas indican los posibles puertos de salida. En función de la política de selección y del estado de las colas en los nodos vecinos se optará por solicitar la salida por uno (y sólo uno) de los puertos candidatos.

Se han implementado, en el simulador, tres diferentes políticas de selección del puerto de salida. Son estas:

1. **RANDOM_REQ**. Se elige como puerto de salida S, al azar, cualquiera de los puertos adaptativos posibles (de entre aquellos que acerquen el paquete a su destino por un camino mínimo). En el caso de que no se pueda por ninguno, se elige el canal de escape indicado por el algoritmo DOR.

2. **SHORTEST_REQ**. Esta política de selección es similar a la anterior, excepto que el canal adaptativo S que se elige es aquel con más espacio en la cola de entrada del nodo de destino. De nuevo, se elige el puerto de escape indicado por el algoritmo DOR cuando no hay opción adaptativa.

3. **SMART_REQ** [PUE 2001]. Esta política puede necesitar varios ciclos para completarse. Al elegir S se intenta primero seguir por el canal adaptativo por el que estaba circulando el paquete; si E es el puerto de inyección, se intenta acceder a un CV adaptativo del primer eje con componente no nula en el registro de encaminamiento. Si no se concede este canal, *en el siguiente ciclo* se pide otro canal adaptativo, en la siguiente dimensión[13] (Y si la inicial es X, Z si la inicial es Y). Se repiten los intentos, si es preciso, hasta agotar las dimensiones. Si no se ha concedido ninguna de estas solicitudes, se solicita el canal de escape correspondiente. Si tampoco se consigue este, se vuelve a empezar.

Común a todas estas formas de solicitud es que siempre están supeditadas a que haya espacio en la cola de recepción seleccionada, en el nodo de destino.

El objetivo de la primera política es un uso uniforme de los recursos: se eligen los CV al azar, luego se emplearán, de promedio, todos por igual. La segunda trata de minimizar los retardos: se elige el CV más vacío. La tercera intenta simplificar la implementación en hardware, siendo al mismo tiempo eficaz en el uso de los recursos: lo más sencillo es intentar seguir por donde se va; si no se puede, *en el siguiente ciclo* se opta por cambiar de dimensión; y si tampoco se puede, *en el siguiente ciclo* se pasa al canal de escape.

5.2 Política de arbitraje

Es posible que varios puertos de entrada E, incluyendo al de inyección, soliciten salir por el mismo puerto de salida S. Se necesita arbitrar entre todas las solicitudes, para asignar S sólo a uno de los solicitantes (Figura 10). Para poder asignar el puerto es necesario que exista espacio suficiente para almacenar el paquete en la cola receptora, condición que ya se comprobó en el proceso de selección. Además, en el caso de que S sea un puerto de escape, se debe cumplir una restricción adicional: la de la burbuja, explicada con detalle en §7.1. Esta restricción obliga a que siempre quede al menos un hueco libre a lo largo del anillo topológico al que pertenece S.

[13] Cuando se inyecta o se cambia de dimensión, de entre todos los CV adaptativos posibles, se selecciona uno al azar. Por supuesto, nunca se intenta cambiar a una dimensión con componente nula en el registro de encaminamiento.

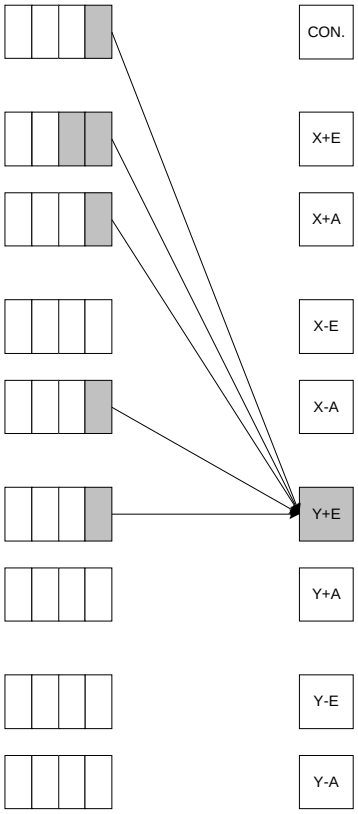

Figura 10. Proceso de arbitraje, para una red 2D con un canal virtual adaptativo. Los puertos de entrada INJ (inyección), X+E, X+A, X-A, Y+E han solicitado (todos ellos) salir por el puerto Y+E. Sólo se va a conceder ese puerto de salida a uno de los solicitantes, en función de la política de arbitraje y de las restricciones que pueda imponer la burbuja.

Se han implementado en el simulador cuatro políticas de arbitraje diferentes:

1. **RANDOM_ARB**. Se asigna el puerto S, al azar, entre aquellos solicitantes que cumplen todas las restricciones.

2. **ROUNDROBIN_ARB**. Todos los puertos están organizados en una lista ordenada. Un índice asociado al puerto S nos indica el número de orden del último puerto de entrada al que se le asignó esta salida[14]. A la hora de buscar un solicitante válido (uno que cumpla todas las restricciones), se empieza a recorrer la lista de solicitantes a partir de la posición siguiente a la que indica el citado índice. Si es necesario, se recorre toda la lista hasta encontrar a uno que cumpla las restricciones, y a él se le asigna—y se actualiza el índice.

3. **OLDEST_ARB**. Asociado a cada cabeza de cola de un puerto de tránsito, tenemos un contador que nos indica cuánto tiempo lleva un paquete (su cabecera) esperando a salir de la cola. Se asigna el puerto S a aquel solicitante que, cumpliendo todas las restricciones, tiene el paquete más antiguo. Los casos de empate se resuelven con un sistema de turnos roundrobin.

4. **LONGEST_ARB**. Cada cola de tránsito tiene una cierta longitud, que es un indicador del uso que está teniendo el canal correspondiente. Se asigna S a

[14] En un ciclo de arbitraje anterior.

aquel solicitante que tiene más ocupada su cola de tránsito. De nuevo, los casos de empate se resuelven con un sistema de turnos roundrobin.

La política LONGEST_ARB tiene un problema: puede dar lugar a inanición. Veamos un caso muy sencillo, con una red en anillo (Figura 11). Sean tres nodos n1, n2 y n3 de ese anillo. El nodo n1 está constantemente enviando paquetes a n3, con n2 siempre como intermediario. El nodo n2 únicamente quiere enviar un paquete a n3. En el nodo n2, las colas de los canales que sirven para comunicar n1 con n3 estarán constantemente llenas (máxima longitud), mientras que la de inyección tendrá longitud 1. Esta política de arbitraje nunca aceptaría la salida del paquete generado en n2.

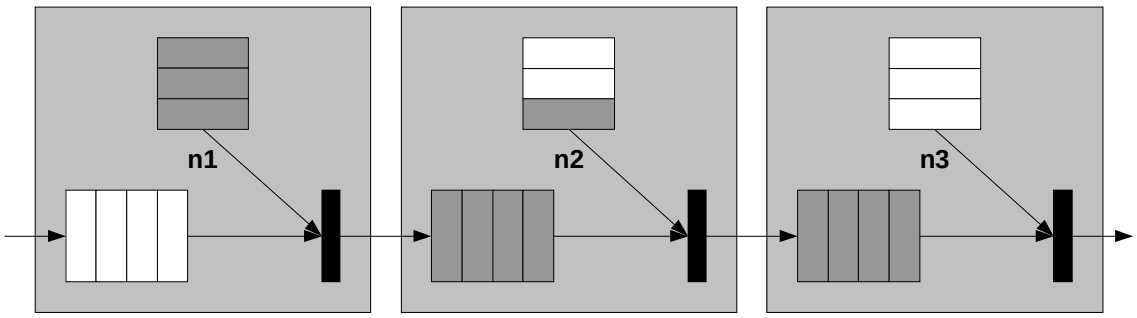

Figura 11. Escenario de inanición cuando se usa la política de arbitraje LONGEST_ARB. El nodo n1 está constantemente generando tráfico para n3, por lo que la cola de tránsito de n2 está siempre llena. Al arbitrar el puerto de salida, la cola de tránsito de n2 tendrá siempre más prioridad que la de inyección—luego n2 nunca puede inyectar.

5.3 Experimentos sobre selección y arbitraje

Con el fin de evaluar el impacto de las políticas de selección y arbitraje de puertos en el rendimiento de la red, se ha realizado una tanda de experimentos con los siguientes parámetros en común: 200.000 ciclos; paquetes de 32 phits; burbuja de 2 paquetes; topología toro 16x16x16; carga aplicada máxima; colas de 8 paquetes; búfer de inyección de 16 paquetes.

Parámetros variables:

- Patrones de tráfico: uniforme y hotspot
- Encaminamiento: estático, adaptativo (2 CV adaptativos)
- Política de selección:
  - smart = SMART_REQ
  - random = RANDOM_REQ
  - shortest = SHORTEST_REQ
- Política de arbitraje:
  - rrobin = ROUNDROBIN_ARB
  - longest = LONGEST_ARB
  - oldest = OLDEST_ARB
  - random = RANDOM_ARB.

Tenemos por tanto una colección de experimentos selección_arbitraje (smart_rrobin, smart_longest, smart_oldest, smart_random, random_rrobin, etc.) por cada patrón de tráfico. Para comparar, también se han recopilado datos para un encaminador estático—donde no es necesaria la política de selección, pero sigue siendo necesario el arbitraje. Los resultados quedan recopilados en la Tabla 5.

Combinación selección_arbitraje	Tráfico uniforme		Tráfico hotspot	
	Carga aceptada	Retardo	Carga aceptada	Retardo
Estático_rrobin	0,32701	3271,32395	0,29333	3603,12936
Estático_longest	0,33329	3340,86357	0,29812	3663,35437
Estático_oldest	0,32686	3273,04496	0,29428	3596,82503
Estático_random	0,31635	3322,56665	0,29343	3561,41173
Smart_rrobin	0,47744	2252,42115	0,38520	7482,20880
Smart_longest	0,47738	2262,12865	0,38447	8239,25797
Smart_oldest	0,47748	2250,77171	0,38120	7719,64221
Smart_random	0,47702	2263,77565	0,37709	7947,66035
Random_rrobin	0,48373	4589,59016	0,38877	7931,12129
Random_longest	0,48078	5906,40123	0,38258	9048,10550
Random_oldest	0,48442	4445,90105	0,38614	8095,66191
Random_random	0,48332	4552,13849	0,37980	8182,91344
Shortest_rrobin	0,41818	2324,73094	0,37322	4351,40420
Shortest_longest	0,29158	11882,78582	0,31838	11625,83059
Shortest_oldest	0,42570	2262,38508	0,37867	4050,90660
Shortest_random	0,42070	2296,14576	0,37580	4159,85405

Tabla 5. Resultados de los experimentos sobre políticas de selección y arbitraje.

Esta tanda de experimentos nos da estas pistas:

1. Para los dos tipos de tráfico, la política de selección tiene más impacto en el rendimiento que la política de arbitraje, aunque destaca por su mal rendimiento el arbitraje LONGEST_ARB—sobre todo cuando se combina con una estrategia de selección SHORTEST_REQ.
2. Para el tráfico uniforme, y fijándonos en la carga aceptada, las mejores políticas de selección son RANDOM_REQ y SMART_REQ. El menor retardo asociado a SMART_REQ nos hace inclinarnos por esta política.
3. Las cosas son distintas para el tráfico hotspot: las tres políticas de selección ofrecen cargas máximas similares, pero SHORTEST_REQ se destaca por tener un retardo bastante menor. Se han hecho más experimentos, con patrones de tráfico distribución y transpuesta, en los que no se observa este fenómeno, y se confirma la superioridad de RANDOM_REQ y SMART_REQ. Quedan pendientes experimentos adicionales que deberían confirmar las especificidades de cada patrón de tráfico.

De forma global, podemos seleccionar (de manera no rotunda) la política de selección SMART_REQ como la que mejor compromiso carga-retardo nos ofrece. En combinación con esta selección, podemos utilizar cualquier política de arbitraje distinta de LONGEST_ARB.

En [HB 2002] se indica que para el BG/L la política de selección a emplear será SHORTEST_REQ, mientras que la de arbitraje será LONGEST_ARB. A la vista de nuestros experimentos, no parece la elección más acertada. Realizaremos más estudios para comprobar hasta qué punto el cambio en otros parámetros que puede condicionar la elección de estas políticas.

6 Número de canales virtuales

6.1 Razones para incorporar canales virtuales

En la Figura 1 veíamos cómo todos los canales virtuales de una misma dimensión y sentido comparten el mismo canal físico; por ejemplo, tanto el canal de escape X+E como los adaptativos X+A1 y X+A2 comparten el canal físico X+. A todos los efectos, el encaminador considera distintos a todos los canales virtuales, salvo a la hora de mover un phit de un nodo a otro: cuando un phit de un circuito virtual dado va a atravesar el enlace físico, es posible que lo haga inmediatamente (si tiene suerte) o que tenga que esperar su turno. En el encaminador simulado, el canal físico se distribuye entre sus canales virtuales por turnos rigurosos. Eso sí, cuando el canal físico le corresponde a uno virtual sin envío pendiente, no se pierde ciclo, sino que se da la oportunidad al que tenía el turno siguiente.

Existen dos razones fundamentales por las cuales un canal físico se organiza en forma de varios canales virtuales. En [DYN 2003] podemos encontrar una explicación detallada de estas razones. Resumimos aquí lo más relevante:

1. Incrementar el rendimiento, al reducir el efecto HOLB (head-of-line blocking). Si un paquete no puede avanzar porque no tiene espacio en el siguiente nodo, se bloquea y bloquea a todos los que le siguen. Si tenemos V canales virtuales es menos probable que esto ocurra, ya que todos los paquetes que van por un determinado camino se distribuyen entre esas V colas, y el bloqueo en una de ellas no supone el bloqueo en las V-1 restantes.

2. Evitar interbloqueos. Esta fue la primera razón para usarlos [DS 1987]. Determinadas combinaciones de topologías y algoritmos de encaminamiento pueden provocar la aparición de dependencias en ciclo, que acaban haciendo que grupos de paquetes se bloqueen unos a otros y, al final, ninguno avance. Se puede organizar un canal físico con ciclos en una colección de canales virtuales sin ciclos, en los que no hay interbloqueo.

Los encaminadores modelados por nuestro simulador usan canales virtuales, pero no por el segundo motivo, evitando el interbloqueo mediante la técnica de la burbuja. En [PUE 2001] se explica esta técnica, que introduce como efecto secundario la posibilidad de inanición (incapacidad de inyectar mientras otros paquetes atraviesan la red); además, se explica cómo es posible añadir a una red con encaminamiento estático DOR y burbuja una colección de CV totalmente adaptativos. Esta combinación de un canal "de escape" con encaminamiento estático y varios canales adaptativos consigue eliminar el riesgo de inanición, además de aumentar el rendimiento total del sistema y mantener la ausencia de interbloqueos.

Desde el punto de vista de la implementación hardware, se trata de una buena solución: la burbuja es barata de implementar (sólo se trata de comprobar el estado de una cola local). Incorporar CV adaptativos sí que tiene un coste: se complica el crossbar, y se necesita incorporar mecanismos de selección/arbitraje más complejos. De todas formas, este coste sería necesario en un toro sin burbuja en el que la evitación del interbloqueo se hiciese con canales virtuales.

Lo que nos planteamos ahora es ¿cuántos canales virtuales adaptativos conviene incorporar para optimizar el rendimiento? Lo examinamos a continuación.

6.2 Experimentos sobre el número de canales virtuales

Hemos realizado una tanda de experimentos para evaluar el impacto sobre el rendimiento del número de canales virtuales utilizados. Los parámetros fijos son: 200.000 ciclos, paquetes tamaño 32 phits, burbuja tamaño 2, toro de 16x16x16, carga aplicada máxima, consumo múltiple, selección SMART_REQ, arbitraje OLDEST_ARB, colas de tránsito de 8 paquetes, búfer de inyección de 16 paquetes.

Parámetros que cambian:

- Encaminamiento: estático, adaptativo
- Canales virtuales adaptativos: 0 (encaminamiento estático), 1, 2, 3, 4
- Tráfico: uniforme y hotstpot

Encaminamiento – CV adap.	Tráfico uniforme		Tráfico hotspot	
	Carga aceptada	Retardo	Carga aceptada	Retardo
Estático – 0	0,32686	3273,04496	0,29428	3596,82503
Adap. – 1	0,46445	5204,27713	0,38279	6413,97539
Adap. – 2	0,47748	2250,77171	0,38120	7719,64221
Adap. – 3	0,46880	2130,37234	0,37174	9153,52990
Adap. – 4	0,46134	2120,44341	0,37543	10000,45176

Tabla 6. Resultados de los experimentos sobre el número de CV adaptativos. La memoria asignada a cada CV es la misma en todos los casos: 8 paquetes— luego la memoria total por encaminador aumenta con el número de CV.

Los resultados quedan recogidos en la Tabla 6. Se puede comprobar que el uso de canales adaptativos, independientemente de su número, mejora sustancialmente la carga aceptada, comparando con el encaminamiento estático (es decir, con usar sólo los canales de escape). En lo que se refiere al retardo, en el caso de tráfico uniforme observamos un aumento cuando incorporamos un CV adaptativo, pero una notable reducción cuando el número de estos CV es dos o más. Con el tráfico hotspot, el retardo aumenta, de forma notable, con el número de CV.

Antes de sacar conclusiones en este apartado, es importante que nos fijemos en que, en los experimentos, la memoria total por nodo es proporcional al número de canales virtuales. Por lo tanto, el encaminador con cuatro CV, aparte de ser más complejo, usa más memoria. En un experimento posterior (§8) veremos que aumentar la memoria en los nodos por encima de un cierto valor tiene un impacto casi nulo en la carga aceptada, pero aumenta el retardo de los paquetes; esto no es más que otra prueba del mismo efecto.

Podemos aislar el efecto de la memoria manteniéndola fija: aprox. 40 paquetes, ó 1280 phits de almacenamiento por canal físico (lo utilizado en el experimento con 4 CV adaptativos). Con esta nueva configuración, se han obtenido los resultados recogidos en la Tabla 7.

Encaminamiento – CV adap. – Long. colas	Tráfico uniforme		Tráfico hotspot	
	Carga aceptada	Retardo	Carga aceptada	Retardo
Estático – 0 – 40	0,38010	9983,39145	0,33444	10825,61849
Adap. – 1 – 20	0,46473	10350,44486	0,38455	14168,42552
Adap. – 2 – 13	0,47756	2578,56580	0,37917	11507,92699
Adap. – 3 – 10	0,46875	2264,68870	0,37395	10921,09343
Adap. – 4 – 8	0,46134	2120,44341	0,37543	10000,45176

Tabla 7. Resultados de los experimentos sobre el número de CV adaptativos. El tamaño de las colas de los CV varía para que la memoria total por encaminador sea constante. Con respecto a la Tabla 6, en todos los experimentos (menos el último) se ha usado más memoria por nodo: hasta 5 veces más en el caso de encaminamiento estático.

Se observa que el aumento de la memoria (en esta tanda de experimentos, salvo para el último caso, las colas son de mayor longitud que en los anteriores) aumenta el rendimiento de forma notable sólo para el caso del encaminador estático. Se pueden obtener mejoras de rendimiento usando CV adaptativos, sin coste en cuanto al uso de memoria. La introducción un único CV adaptativo ya mejora la carga aceptada (de forma sustancial en el caso del tráfico uniforme) con cierta penalización en el retardo. Si se aumenta el número de CV adaptativos se sigue manteniendo la mejora en cuanto a la carga, y se reduce (tráfico uniforme) o mantiene (tráfico hotspot) el retardo. El número óptimo de CV adaptativos sería dos: es la solución de mínimo coste, y de mejor rendimiento global.

En [HG 2001] se indica que en el BG/L se usan canales virtuales (Figura 16) de una forma simular a como lo hace nuestro simulador (Figura 1). En concreto, se usa un canal de escape, dos adaptativos[15], y otros dos más, denominados "de alta prioridad" y "bypass" cuyo uso no está claramente descrito—salvo que el de alta prioridad se emplea para señales del sistema operativo.

7 Tamaño de burbuja

7.1 Evitación del interbloqueo

En la sección anterior ya se ha presentado la técnica de la burbuja como mecanismo sencillo de evitación de interbloqueos [PUE 2001]. La técnica se aplica en los canales de escape, donde pueden aparecer interbloqueos debidos a la topología empleada (toro). Los canales adaptativos no necesitan usar esta técnica, ya que si un paquete se bloquea en uno de ellos acaba pasando al canal de escape, libre de interbloqueos y, por tanto, la solución final sigue estando libre de intebloqueos [PUE 2001, DUA 1993].

La técnica de la burbuja se aplica cuando se inyecta en un canal de escape, bien directamente desde la cola de inyección, bien cuando se pasa de otro canal a uno de escape. Sea E el canal virtual de entrada (origen), y S el de salida (destino), donde S es un canal de escape. Para que un paquete m encolado en E pueda ser introducido en la cola de tránsito correspondiente al canal S en el nodo vecino, es necesario:

1. Que la cola asociada a S en el nodo de destino tenga espacio para acoger el paquete (condición general de la conmutación virtual cut-through).

[15] Lo cual puede interpretarse como una confirmación de nuestras conclusiones.

2. Que la cola asociada a S en el nodo local tenga espacio para tantos paquetes como indique la burbuja.

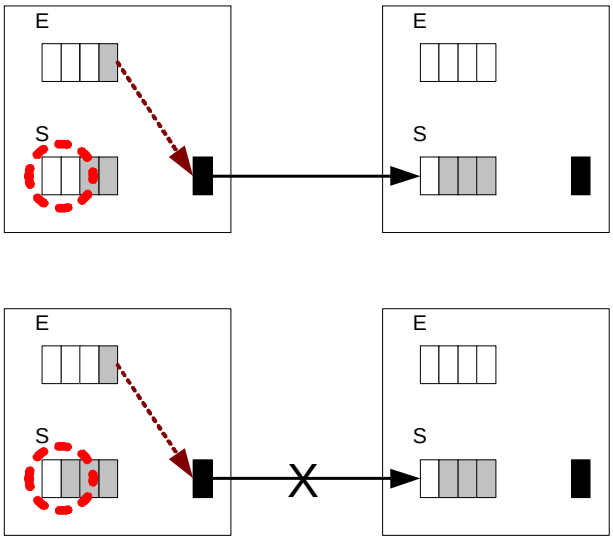

Figura 12. Ilustración de la técnica de la burbuja. E es el canal de origen, S el de destino (uno de escape), y la burbuja es de tamaño 2. Arriba, una situación en la que el paso de un paquete de un canal a otro (por ejemplo, de la cola de inyección al canal X+E) es posible. Abajo, una situación en la que no se permite ese paso: no se garantizaría la existencia de una burbuja 2 en canal S.

De esta manera nos aseguramos de que en el ciclo formado por todas las colas de un mismo canal virtual *siempre* queda al menos un hueco (lo podemos ver gráficamente en la Figura 12). Y esto es suficiente para evitar el interbloqueo. El tamaño mínimo de burbuja para conseguir esta propiedad es dos. La Figura 13 muestra una situación de interbloqueo que se podría producir usando una burbuja menor.

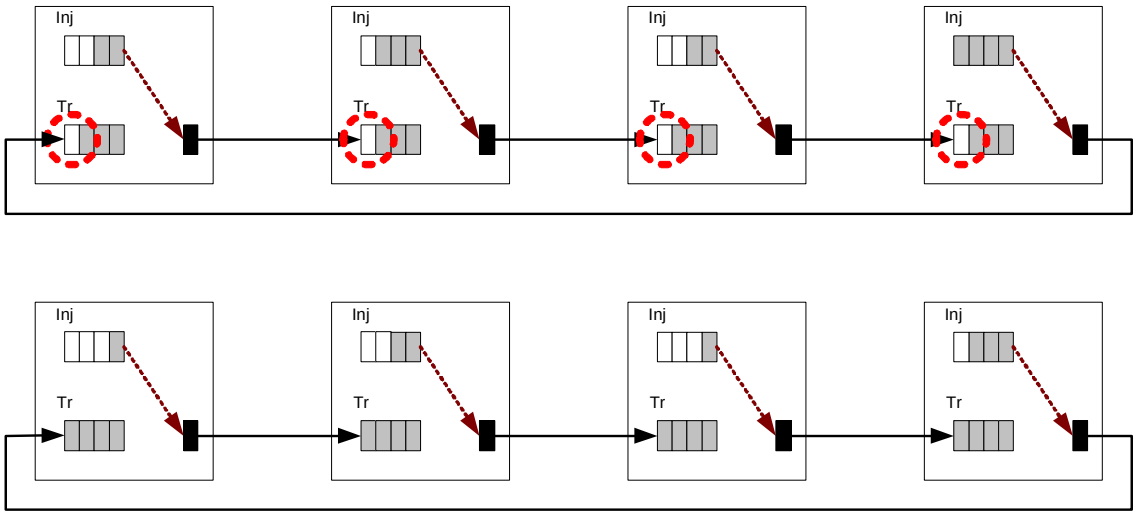

Figura 13. Ilustración de por qué no es suficiente una burbuja de tamaño 1 para garantizar la ausencia de interbloqueos. Caso patológico, en un anillo unidireccional. Con una burbuja de tamaño 1, puede darse la situación de la parte superior: todos los nodos quieren inyectar, y en todos los casos se cumplen las condiciones (hueco en la cola de tránsito vecina, burbuja en la cola de tránsito local). Si se completan todas las inyecciones, se llenan todas las colas de tránsito simultáneamente, tal como se muestra en la parte inferior.

A continuación exploramos hasta qué punto el tamaño de burbuja afecta al rendimiento en redes con topología toro 3D.

7.2 Experimentos sobre el tamaño de la burbuja

En este apartado revisamos una colección de experimentos realizados para evaluar el impacto en el rendimiento del tamaño de la burbuja. Usar una burbuja grande implica limitar la inyección y el cambio de dimensión, lo que favorece el tránsito de los paquetes que ya están en circulación. Una burbuja de tamaño 2 es lo mínimo necesario para garantizar interbloqueos. Nos planteamos si mejoran las cosas con una burbuja mayor, y realizamos para ello una serie de experimentos con estos parámetros fijos: 200.000 ciclos, paquetes de 32 phits, topología: toro 16x16x16, carga aplicada máxima, encaminamiento adaptativo con 2 CV adaptativos y uno de escape, consumo múltiple, selección SMART_REQ, arbitraje OLDEST_ARB, cola de tránsito de 8 paquetes, búfer de inyección de 16 paquetes.

Parámetros que cambian:

- Tamaño de burbuja: 0 (no burbuja), 1, 2, 3, 4
- Tráfico: uniforme y hotstpot

Los resultados obtenidos se resumen en la Tabla 8. Nótese cómo, en el caso del tráfico uniforme, no ha aparecido una situación de interbloqueo durante el intervalo de simulación, ni con burbuja 1 ni sin burbuja. En el caso del tráfico hotspot sí se ha llegado a interbloqueo cuando no se ha empleado burbuja, y el rendimiento obtenido con burbuja 1 ha sido bastante pobre.

Tamaño burbuja	Tráfico uniforme		Tráfico hotspot	
	Carga aceptada	Retardo	Carga aceptada	Retardo
0 (no burbuja)	0,47748	2250,77171	Interbloq.	--
1	0,47748	2250,77171	0,29750	10086,78716
2	0,47748	2250,77171	0,38120	7719,64221
3	0,47754	2251,03148	0,38642	7551,16016
4	0,47747	2249,91895	0,38637	7365,96604
5	0,47753	2250,06530	0,38691	7287,87580
6	0,47752	2250,54405	0,38726	7228,87938
7	0,47754	2252,72311	0,38745	7210,19955
8	0,47822	2366,51529	0,38733	7311,28525

Tabla 8. Resultados de los experimentos para evaluar el efecto del tamaño de la burbuja en el rendimiento de la red. Tráficos uniforme y hotspot.

De los datos recogidos en la tabla, puede resultar llamativo que, para tráfico uniforme, los resultados con burbuja 0, 1 y 2 sean idénticos. Un examen del uso de los búferes en uno de los nodos (el 1) nos da la explicación: a lo largo de la simulación, nunca se han ocupado más de 4 de los 8 espacios disponibles en las colas de los canales de escape. Por lo tanto, la burbuja no ha supuesto ninguna restricción efectiva. Este no ha sido el caso para el tráfico hotspot, donde algunos canales de escape llegan a estar repletos de forma permanente. Para este patrón, aumentar las restricciones (el tamaño de la burbuja) parece mejorar las cosas.

Ampliamos el estudio a los otros dos patrones de tráfico, con el objeto de comprobar si el fenómeno observado para el patrón hotspot se repite con otras cargas. La Tabla 9 resume los resultados de los experimentos realizados con tráfico distribución y matriz

transpuesta. De nuevo, si ignoramos la información sobre burbuja 0 ó 1, nos encontramos con que el tamaño de burbuja no es demasiado relevante: en un caso mejora ligeramente el rendimiento, en el otro disminuye.

Tamaño burbuja	Tráfico distribución		Tráfico matriz transpuesta	
	Carga aceptada	Retardo	Carga aceptada	Retardo
0 (no burbuja)	Interbloq.	--	0,12728	20319,98877
1	0,17125	14911,86216	0,14593	18019,10702
2	0,24865	11962,77242	0,14585	17761,20799
3	0,25020	11751,29164	0,14393	17851,19255
4	0,25068	11648,81382	0,14260	17845,89583
5	0,24958	11421,57069	0,14241	17794,57163
6	0,24886	11317,72940	0,14275	17716,56464
7	0,24776	11248,09692	0,14198	17737,45730
8	0,24379	11350,35461	0,14290	17782,73025

Tabla 9. Resultados de los experimentos para evaluar el efecto del tamaño de la burbuja en el rendimiento de la red. Tráficos distribución y matriz transpuesta.

La conclusión de estos experimentos es que restringir las inyecciones y cambios de dimensión en los canales virtuales de escape, además de evitar interbloqueos, tiene cierto impacto en el rendimiento de la red. Una burbuja de tamaño 2 garantiza que no hay interbloqueos; aumentar esta burbuja mejora ligeramente el rendimiento en unos patrones, pero en general no cambia de forma sustancial ni la carga aceptada ni el retardo.

8 Longitud de las colas

Las colas asociadas a los canales virtuales permiten almacenar temporalmente los paquetes pendientes de envío, de recepción o de tránsito hacia otro nodo. A priori, puede parecer conveniente tener colas de gran tamaño, capaces de absorber las ráfagas de tráfico generadas por un nodo de cómputo, o recibidas desde diferentes nodos pero que se dirigen hacia un mismo destino. Sin embargo, el almacenamiento no es gratuito (tiene un coste hardware) y, además, puede tener un impacto negativo en el tiempo de tránsito de los paquetes por la red.

Hemos realizado una serie de experimentos para evaluar el impacto de la longitud de las colas de tránsito en la carga aceptada por la red, y en el retardo medio experimentado por los paquetes. Estos son los parámetros fijos de las simulaciones: 200.000 ciclos, paquetes de 32 phits, burbuja tamaño 2, toro de 16x16x16, carga aplicada máxima, encaminamiento adaptativo (2 CV adaptativos), consumo múltiple, selección SMART_REQ, arbitraje OLDEST_ARB, búfer de inyección de 16 paquetes.

Parámetros que cambian:

- Tráfico: uniforme y hotstpot
- Longitud de las colas de tránsito: 3, 4, 5, 6, 7, 8 paquetes

Tamaño cola	Tráfico uniforme		Tráfico hotspot	
	Carga aceptada	Retardo	Carga aceptada	Retardo
3	0,47712	1863,18751	0,36583	4034,72633
4	0,47742	1963,68009	0,37338	4785,29782
5	0,47745	2044,47217	0,38021	5503,43832
6	0,47751	2115,79274	0,38224	6210,97141
7	0,47752	2183,03376	0,38296	6931,68674
8	0,47748	2250,77171	0,38120	7719,64221

Tabla 10. Resultados de los experimentos para evaluar el efecto de la longitud de las colas de tránsito en el rendimiento de la red.

Los resultados más relevantes están recogidos en la Tabla 10. A la vista de ellos, podemos concluir que el tamaño de las colas no parece que tenga un gran impacto en cuanto a la carga total aceptada por el sistema: la mejora observada conforme se aumenta el tamaño es prácticamente nula en el caso de tráfico uniforme, y marginal en el caso del tráfico hotspot. Por otra parte, el retardo medio aumenta con el tamaño de las colas—de forma considerable para el tráfico hotspot.

Los resultados para el tráfico hotspot pueden explicarse así: los nodos de la "zona caliente" no pueden absorber todo el tráfico que se les envía, puesto que están limitados por sus canales físicos. Los paquetes van llenando, sin remedio, todas las colas que los acercan a la zona caliente, sin que haya posibilidad de que esas colas se vayan vaciando más deprisa. El resultado final es que se inyecta al ritmo que se consume (no es estable una situación en la que se inyecta más), pero cada paquete tiene que atravesar multitud de colas repletas antes de alcanzar su destino—aparte de competir con el resto de paquetes.

La conclusión de este apartado es que, para los patrones empleados, no merece la pena utilizar colas de gran tamaño. Quedaría pendiente de evaluación otro tipo de patrones, con ráfagas de alta generación seguidas de períodos de menos actividad. La hipótesis es que, en estos casos, unos búferes mayores aceptarían con mayor rapidez los datos enviados por los nodos de cómputo, aumentando así la carga aceptada por la red.

9 Tamaño del paquete

El tamaño del paquete es otro importante parámetro de diseño. Los paquetes cortos están ideados para aquellos entornos en los que es necesario un intercambio rápido y frecuente de bloques de información reducidos (una línea de memoria cache, por ejemplo). Los paquetes grandes resultan más adecuados para entornos tipo MPI, donde las aplicaciones intercambian estructuras de datos de tamaño medio-grande. Es importante tener en cuenta que todos los paquetes tienen cierta sobrecarga: como mínimo, la necesaria para llevar la cabecera con el registro de encaminamiento. Por lo tanto, cuanto más corto sea el paquete, mayor es el ancho de banda que no se emplea para transportar carga útil.

Por otra parte, los paquetes pueden ser de tamaño fijo, o de tamaño variable. Tradicionalmente, las redes de computadores han ido hacia paquetes de tamaño variable, con una cabecera indicando ese tamaño. Sin embargo, también hay casos de redes con paquetes de tamaño fijo y pequeño, que pueden exigir procesos de

segmentación y reensamblado en los extremos cuando los datos intercambiados por las aplicaciones sean de tamaño mayor que un paquete.

Nuestro diseño utiliza paquetes de tamaño fijo. La mayor parte de los experimentos se ha realizado con paquetes de 32 phits. Nos planteamos ahora si cambiarían las cosas si la red gestionase paquetes más pequeños, o más grandes. Queda como trabajo futuro la gestión de paquetes de tamaño variable.

Evaluamos el efecto que el tamaño del paquete tiene en el rendimiento de la red con una colección de experimentos cuyos parámetros fijos son: 200.000 ciclos, burbuja tamaño 2, toro de 16x16x16, carga aplicada máxima, encaminamiento adaptativo (2 CV adaptativos), consumo múltiple, selección SMART_REQ, arbitraje OLDEST_ARB.

Parámetros que cambian:

- Tráfico: uniforme y hotstpot
- Tamaño de paquete: 1, 2, 4, 8, 32 y 64 phits
- Tamaño de las colas de tránsito y del búfer de inyección, medido en paquetes, con el fin de que el tamaño en phits se mantenga constante: colas de 256 phits, búfer de 512 phits

Tamaño paquete – colas – búfer	Tráfico uniforme		Tráfico hotspot	
	Carga aceptada	Retardo	Carga aceptada	Retardo
1 – 256 – 512	0,45965	1698,11933	0,37642	7285,50577
2 – 128 – 256	0,46448	1695,83866	0,37257	7665,26652
4 – 64 – 128	0,47086	1712,68776	0,37823	7812,38702
8 – 32 – 64	0,47479	1779,92290	0,37694	7786,89940
16 – 16 – 32	0,47682	1933,34697	0,38175	7664,66678
32 – 8 – 16	0,47748	2250,77171	0,38120	7719,64221
64 – 4 – 8	0,47705	2842,84097	0,38349	7605,03738

Tabla 11. Resultados de los experimentos para evaluar el efecto del tamaño de los paquetes en el rendimiento de la red.

Los resultados se recogen en la Tabla 11. Se aprecia un impacto muy poco significativo del tamaño del paquete en la carga total aceptada por el sistema. En el caso de los paquetes cortos, cada phit es una cabecera, y cada vez que se da un salto hay que tomar una decisión, que puede ser equivocada. En el caso de paquetes de varios phits, las decisiones se toman con la cabecera, y el resto de los phits pasan seguidos por el camino ya abierto.

En cuanto al retardo medio, en el caso del tráfico uniforme se observa que el retardo menor corresponde a paquetes cortos. La explicación es que a éstos les resulta más sencillo encontrar recursos a lo largo del camino: no es lo mismo encontrar un espacio para 64 phits que hacerlo para uno. Hay que tener en cuenta, además, que en el retardo se considera el tiempo que se tarda en consumir todo el paquete, que obviamente es mayor para paquetes largos.

La conclusión de esta sección es que el tamaño del paquete no tiene un impacto demasiado significativo en el rendimiento de la red, lo cual podemos considerarlo como algo positivo, ya que cada aplicación tiene su tamaño de paquete idóneo. No olvidemos, sin embargo, que los paquetes muy pequeños introducen una considerable sobrecarga de tráfico no útil. Por otra parte, como ya se ha dicho, queda pendiente de estudio el trabajo con paquetes de tamaño variable.

En el caso del BG/L, se emplean paquetes de tamaño variable, entre 32 y 256 bytes; la cabecera (4 bytes) indica el número de bloques de 32 bytes que forman el paquete. No tenemos datos de ningún experimento que relacione este tamaño con el rendimiento.

10 Conclusiones y líneas de trabajo abiertas

10.1 Conclusiones

A lo largo de este trabajo se han ido realizando diferentes experimentos, que han ido aportando ideas sobre qué decisiones de diseño son más adecuadas para la construcción de redes de interconexión para sistemas paralelos de gran tamaño. Resumimos y globalizamos aquí estas conclusiones.

- La selección de topologías 3D para sistemas paralelos de gran escala se confirma como una buena opción, puesto que su rendimiento es mucho mejor que el de las alternativas 2D. Aunque la complejidad del encaminador es mayor, no parece que esto sea una limitación real: IBM, para su BG/L, va a integrar un encaminador de este tipo en el mismo chip en el que van, además, dos procesadores e interfaces de acceso a otras redes.

- Dentro de las topologías 3D el toro ofrece un rendimiento superior a la malla. Como inconvenientes principales del toro están la necesidad de enlaces periféricos y la dificultad de partición en toros más pequeños. Sin embargo, siempre es posible dividir el toro en estructuras 3D tipo malla con algunos enlaces periféricos adicionales, lo que aporta flexibilidad adicional.

- La técnica de la burbuja es eficaz para evitar interbloqueos en el toro, y muy sencilla de implementar. Es suficiente usar una burbuja de tamaño 2 para eliminar la posibilidad de interbloqueos. Aumentar el tamaño no aporta mejoras en el rendimiento.

- El encaminamiento adaptativo ofrece una importante mejora de rendimiento con respecto al estático. En el caso de la topología toro 3D, nuestra propuesta es combinar el uso de un canal de escape no adaptativo (con el uso de la técnica de la burbuja) con varios canales virtuales totalmente adaptativos. El resultado final es adaptativo, libre de inanición, libre de interbloqueos y con buen rendimiento. El coste de esta solución es similar a otras que también emplean CV, pero con el objetivo de evitar interbloqueos.

- El número de estos canales virtuales adaptativos no tiene por qué ser muy grande: 2 ó 3 son suficientes. El compromiso adoptado para el BG/L (2 CV adaptativos) es una opción razonable: la mejora de rendimiento es notable, y la complejidad del encaminador no se dispara.

- El tamaño de las colas de tránsito tiene un importante efecto en el rendimiento cuando no se usan canales virtuales y, por lo tanto, sólo hay una cola de tránsito por canal físico y nodo. Sin embargo, cuando se usan varios canales virtuales, el tamaño de las colas de tránsito asociadas a esos canales apenas tiene efecto.

- La política de selección de salida para el caso de encaminamiento adaptativo (en el que un paquete puede seguir varias rutas para alcanzar su destino) tiene, para algunos patrones de tráfico, bastante impacto en el rendimiento del sistema. La política SMART_REQ ofrece un buen compromiso carga aceptada / retardo.

- En cuanto a la política de arbitraje o asignación de un canal de salida a una cola de entrada, el impacto en el rendimiento es pequeño. Resulta suficiente una política que sea justa y no lleve a situaciones de inanición. No nos decantamos por ninguna en particular, aunque los experimentos nos llevan a desechar LONGEST_ARB—precisamente la utilizada en el BG/L.

- El tamaño de paquete no parece un factor muy relevante en el rendimiento del sistema. Los resultados en cuanto a tráfico aceptado son algo mejores para paquetes largos, a costa de unos retardos mayores. Esto lo podemos considerar como algo positivo: una misma red de interconexión puede dar soporte a aplicaciones muy diversas.

10.2 Líneas abiertas

El estudio realizado para este trabajo, aunque amplio, no ha sido todo lo exhaustivo que podría hacerse. A lo largo del mismo se han ido realizando comentarios sobre las líneas de trabajo que quedan abiertas, muchas de las cuales requieren de una modificación del simulador, ampliando su funcionalidad. Resumimos en una lista de puntos esas, y otras, líneas de trabajo futuro.

- Realizar un barrido más exhaustivo de los parámetros. Se han hecho experimentos variando muy pocos parámetros cada vez. Es necesario ver posibles relaciones cruzadas entre parámetros, y su impacto en el rendimiento.

- Incrementar las alternativas de inyección de tráfico, mediante la implementación de nuevos patrones, o mediante el uso de trazas obtenidas tras la ejecución de aplicaciones reales. Relacionado con esto está la capacidad de trabajar con paquetes de tamaño variable. El simulador ya acepta este tipo de paquetes; sin embargo, no está definido ningún método para inyectar esta clase de tráfico. En otras palabras: sería interesante implementar mecanismos de inyección de tráfico que generen paquetes de tamaño variable.

- Sin llegar a emular el hardware de un simulador, sí se puede intentar mejorar el modelado de los tiempos. Relacionado con esto, la mayor parte de los encaminadores funcionan de forma segmentada: varios phits pueden estar siendo gestionados simultáneamente en el chip, cada uno en una etapa distinta. Es interesante llegar a modelar este comportamiento, que afecta al tiempo de paso de un paquete por un encaminador.

- Aunque en un principio hemos optado por trabajar con un simulador secuencial, la falta de memoria RAM en las máquinas empleadas ha terminado siendo un problema, limitando el tamaño de los experimentos. Al margen de ampliar esa RAM, una línea abierta es paralelizar el simulador, para su ejecución eficiente en sistemas tipo SMP de pequeño tamaño y clusters.

- Soporte para operaciones colectivas. Este tipo de operaciones son muy habituales en aplicaciones paralelas desarrolladas con MPI y APIs similares. Es

muy importante que la red de interconexión de un soporte adecuado a operaciones como difusión (broadcast), reducción, barreras, reunir (*gather*), esparcir (*scatter*) etc. Como primer paso, se implementarían mecanismos eficientes de difusión, para después pasar a otro tipo de operaciones.

- Tolerancia a fallos. Este es un tema muy amplio, aunque el grupo ya está trabajando en él [PUE 2003]. La tolerancia a fallos es fundamental en equipos con miles de nodos, donde el tiempo medio entre fallos es muy pequeño.

- Por último, es nuestra intención seguir recopilando la información que vaya apareciendo sobre el BG/L y RS, con el objeto de validar y comparar propuestas de redes de interconexión para sistemas masivamente paralelos.

11 Referencias

[ADI 2002] NR Adiga et al. "An overview of the BlueGene/L Supercomputer". Supercomputing 2002 Technical Papers, November 2002.
Disponible en http://sc-2002.org/paperpdfs/pap.pap207.pdf

[AGA 1991] Anant Agarwal, "Limits on Interconnection Network Performance" IEEE Trans. on Computers, Vol 2, No.4, pp. 398-412, October 1991.

[ALM 2003] G. Almasi et al. "System Management in the BlueGene/L Supercomputer". Proc. 3rd. Workshop on Massively Parallel Processing, IPDPS'03. April 2003.
Disponible en
http://www.haifa.il.ibm.com/projects/systems/bluegene/papers/system_management.pdf

[BAA 2002] Presentaciones realizadas en BlueGene/L Applications, Algorithms, and Architectures Workshop. August 2002.
Disponible en http://www.llnl.gov/asci/platforms/bluegene/agenda.html.

[CEZ 2003] L. Ceze et al. "Full circle: simulating linux clusters on linux clusters". Proc. 4th LCI Int. Conf. on Linux Clusters: The HPC Revolution 2003, San Jose, CA, June 2003.

[CHI 2002] George Chiu "BlueGene/L Overview and Status". En [BAA 2002].

[CHS 2003] Presentaciones realizadas en "The Conference on High-Speed Computing". LANL / LLNL / SNL April 21 – 24, 2003. Disponibles en:
http://www.lanl.gov/orgs/ccn/salishan2003/program.htm.

[COT 2002] Paul Coteus "BlueGene/L System Package". En [BAA 2002].

[CT 2003] B. Camp, J. Tomkins "The Red Storm Computer Architecture and its Implementation". En [CHS 2003].

[DAL 1990] William J. Dally. "Performance Analysis of k-ary n-cube interconnection networks". IEEE Transactions on Computers, Vol. 39, No. 6, June 1990.

[DAL 1992] W. J. Dally. Virtual-Channel Flow Control. IEEE Trans. on Parallel and Distributed Systems, Vol. 3, No. 2, May 1992.

[DS 1987] W.J. Dally and C.L. Seitz. "Deadlock-free message routing in multiprocessor interconnection networks". IEEE Transactions on Computers, vol. C-36, no. 5, May 1987.

[DUA 1993] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks". IEEE Trans. on Parallel and Distributed Systems, vol.4, no.12, pp.1320-1331, December 1993.

[DYN 2003] J. Duato, S. Yalamanchili, L. Ni. "Interconnection networks: an engineering approach". Revised printing. Morgan Kaufmann, 2003.

[HB 2002] Philip Heidelberger and Burkhard Steinmacher-Burow. "Overview of the BG/L Torus Network". En [BAA 2002].

[HP 2003] J.L. Hennesy & D.A. Patterson. "Computer architecture: a quantitative approach". 3rd. Ed. Morgan Kaufmann, 2003.

[IBM] Página web de IBM (http://www.ibm.com)

[KK 1979] P. Kermani and L. Kleinrock, "Virtual Cut-Through: A new computer communication switching technique," Computer Networks, vol. 3, 1979. 34.

[KLE 2003] D. Klepacki. "Towards Petascale Computing". Slides of lecture at the Scientific Computing and Visualization Group, Boston University, April 8, 2003. Available at http://scv.bu.edu/SCV/Archive/IBM/LECTURES.html

[KMS 1997] M. Kaufmann, U. Meyer, J.F. Sibeyn. "Matrix transpose on meshes: theory and practice". 11th Int. Parallel Processing Symposium IPPS'97. Geneva, Switzerland, April 1997.

[MIG 1996] J. Miguel. "An empirical evaluation of techniques for parallel simulation of message passing networks". Tesis doctoral. Servicio Editorial de la UPV/EHU, 1996. ISBN 84-7585-741-8.

[MIG 1998] J. Miguel, A. Arruabarrena, R. Beivide, J.A.B. Fortes. "An evaluation of implementations of the CMB parallel simulation algorithm on distributed memory multicomputers". Journal of Systems Architecture 44 (1998) 519-545.

[MYR] Información sobre los productos Myrinet en la página web de Myricom (http://www.myri.com)

[PET 2002] F. Petrini et al. "the Quadrics network: high-performance clustering technology". IEEE Micro, 22(1):46-57, Feb. 2002.

[PGB 2002] V.Puente, J.A. Gregorio, R.Beivide, "SICOSYS: An Integrated Framework for studying Interconnection Network in Multiprocessor Systems", Proceedings of the IEEE 10th Euromicro Workshop on Parallel and Distributed Processing. Gran Canaria, Spain. January 2002.

[PUE 2001] V. Puente, C. Izu, R. Beivide, J.A. Gregorio, F. Vallejo and J.M. Prellezo. "The Adaptative Bubble Router". Journal of Parallel and Distributed Computing. Vol 61 - n. 9, September 2001 pp. 1180-1208.

[PUE 2003] V. Puente, J.A. Gregorio, R. Beivide and F. Vallejo, A Low Cost Fault Tolerant Packet Routing for Parallel Computers, IEEE/ACM IPDPS,International Parallel and Distributed Processing Symposium, Nice, April 2003.

[QSN] Información sobre QsNet en la página web de Quadrics, Inc. (http://www.quadrics.com). Descripción del sistema en F. Petrini et al. "the Quadrics network: high-performance clustering technology". IEEE Micro, 22(1):46-57, Feb. 2002.

[SEA 2003] M. Seager. "ASCI Purple & BlueGene/L Overview". En [CHS 2003].

[UGS 2002] Ed Upchurch, Tom Gottschalk and Paul Springer. "Modeling the BlueGene/L 64K Node Network - Statistical Simulator". En [BAA 2002].

[UTE 1995] C. Eric Wu and Hubertus Franke. "UTE: A unified trace environment for IBM SP systems". Technical Report RC 20048 (88654), T. J. Watson Research Center, Yorktown Heights, NY, May 1995.

12 Agradecimientos

Este trabajo ha sido realizado gracias a la financiación obtenida del Ministerio de Ciencia y Tecnología, proyecto TIC2001-0591-C02-02.

Además del agradecimiento institucional, no puedo dejar de agradecer a todos los miembros del grupo (Grupo de Paralelismo en la UPV/EHU, Grupo de Arquitectura y Tecnología de Computadores en la U. de Cantabria) su apoyo para la realización de este trabajo, con especial mención a los profesores J.A. Gregorio y R. Beivide.

Apéndice A: Visión general del BlueGene/L

El BlueGene/L, actualmente en fase de diseño por parte de IBM, es en realidad el primero de toda una serie de sistemas cuyo objetivo final es alcanzar potencias de cálculo del orden del PetaFLOP/s hacia el año 2010. Esta primera máquina espera alcanzar los 200 TeraFLOP/s en el 2005.

Se trata de un sistema masivamente paralelo, con miles de nodos biprocesador empaquetados (dos a dos) en tarjetas que se integran en placas que después forman un rack. El sistema final consta de múltiples de estos racks interconectados mediante cables externos. La Figura 14 muestra esta estructura.

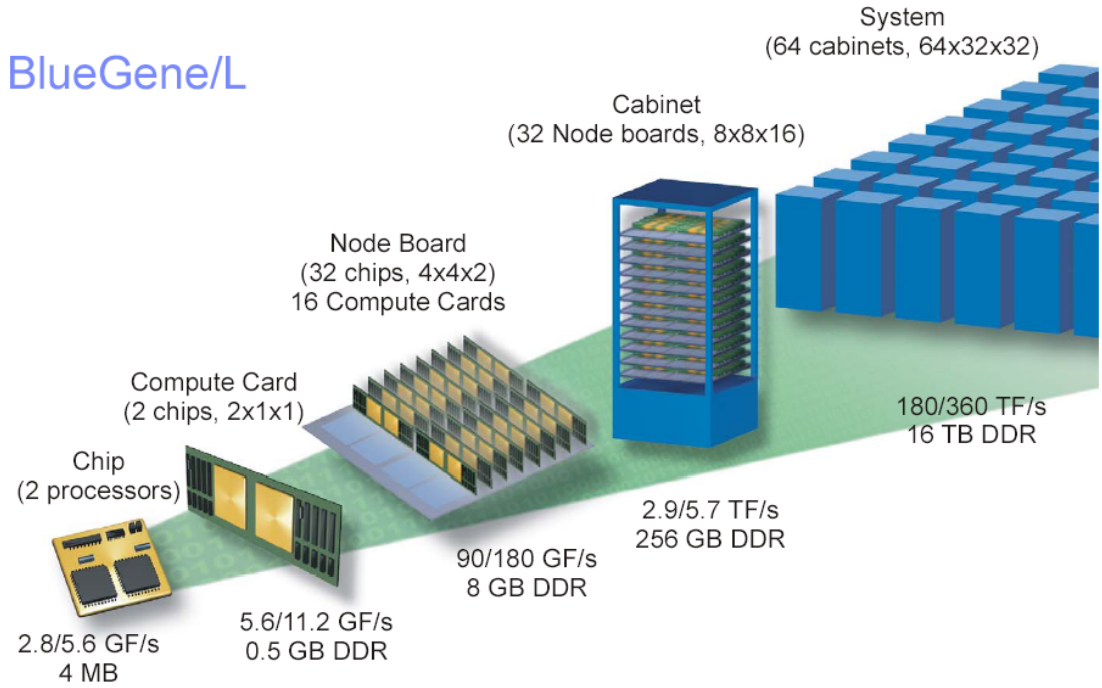

Figura 14. Esquema de la estructura del futuro BG/L [KLE 2003].

Este es un resumen de algunos de los objetivos de diseño del BG/L:

- 65.536 nodos biprocesador
- Bajo consumo
- Alto rendimiento en coma flotante
- Tecnología SIC (System-on-a-chip)
- Interconexión toro 3D (64x32x32)
- Redes auxiliares para E/S y operaciones globales
- Programación: paso de mensajes, sólo un proceso por nodo
- Los nodos de cómputo se usan sólo para aplicaciones, no para operaciones de gestión
    - Se evitan eventos asíncronos (demonios, interrupciones)
    - Se evitan operaciones complejas

- Una colección de nodos de E/S descargan a los de cómputo de esas tareas
    - Un nodo de E/S por cada N nodos de cómputo. N es ajustable, pero se prevé que sea 64
    - Realizan operaciones de administración y de E/S por cuenta de los nodos de cómputo
- La comunicación entre nodos de E/S y de cómputo se hace a través de una red específica, para no interferir con las comunicaciones que circulan por el toro
- También es necesario descargar las operaciones de monitorización y control. Se incorporan nodos de servicio encargados de
    - Arranque, monitorización de estado
    - Monitorización de rendimiento
- Los nodos de servicio se comunican con los de cómputo a través de una red adicional, que no interfiere con las otras dos (toro, E/S)
- La máquina puede ser compartida por muchos usuarios/aplicaciones, pero se usa el concepto opuesto al "Time Sharing": todo es "Space Sharing". Cada trabajo requiere la asignación de una partición (conjunto de nodos asignados en exclusiva)

La Figura 15 representa el detalle de un nodo-biprocesador. Cada CPU es un procesador PowerPC 440 con doble unidad de coma flotante. Todos los elementos aquí representados se integran en un mismo chip.

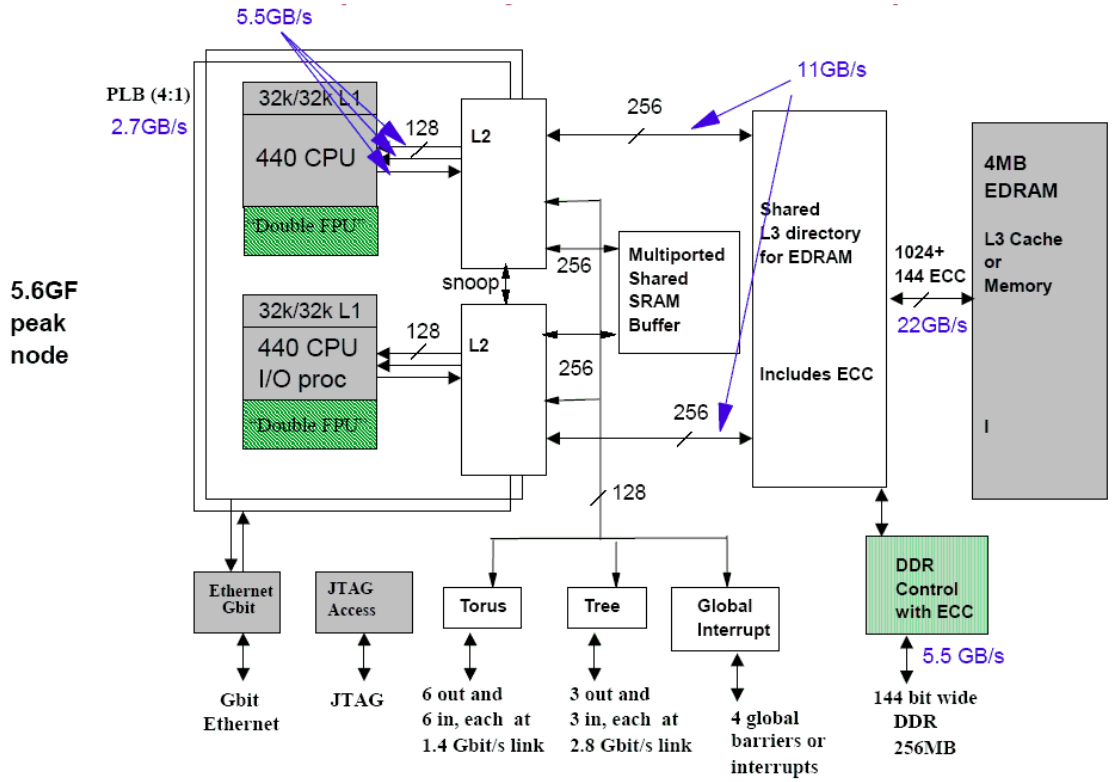

Figura 15. Detalle de un nodo del BL/L [KLE 2003].

En la parte inferior tenemos las interconexiones de cada nodo. Aparte del controlador del sistema de memoria, se incorporan accesos a cinco redes distintas:

- Una red con topología toro-3D para su uso por las aplicaciones, que describiremos a continuación con más detalle

- Una red en árbol, optimizada para operaciones globales (reducciones, difusión), así como para las operaciones de E/S entre los nodos de cómputo y los de E/S.
- Una red para barreras (sincronización global) e interrupciones
- Una red Gigabit Ethernet, a la que están conectados sólo los nodos de E/S (aunque el dispositivo está implementado en todos los nodos)
- Gigabit Ethernet-JTAG para el control del sistema desde un "nodo de servicio"

Desde nuestro punto de vista, lo más relevante de esta arquitectura es la red toroidal que da soporte al paso de mensajes entre aplicaciones[16]. A partir de la información extraída de [ADI 2002, HB 2002], facilitamos un resumen de los datos – objetivo de diseño para esta red:

- Topología: toro 3D
- Conmutación virtual cut-through
- Encaminamiento adaptativo
- Soporte para comunicación punto a punto, y para difusión por una fila
- Paquetes de 32-256 bytes, en múltiplos de 32, con una cabecera de 4 bytes
- Se utiliza un sistema de CRCs y retransmisión para aportar fiabilidad. Los ACKs son de 8 bytes
- Ancho de banda objetivo:
    - 1,4 Gb/s por enlace (serie): 1 byte por cada 4 ciclos del procesador
    - Flexibilidad para multiplicar/dividir por 2 el reloj, dependiendo de los errores
- Latencia por salto: captura + lógica del encaminador + envío + tiempo en el cable
    - Lógica del encaminador: 12 ciclos, 5,7 ns/ciclo = 69 ns
    - Captura + envío = 12 ns
    - Cable: entre 1 y 135 ns → 8 ns de promedio
    - TOTAL (de promedio): 89 ns

[16] En buena medida, la arquitectura de esta red ha sido la que ha motivado la realización de este trabajo.

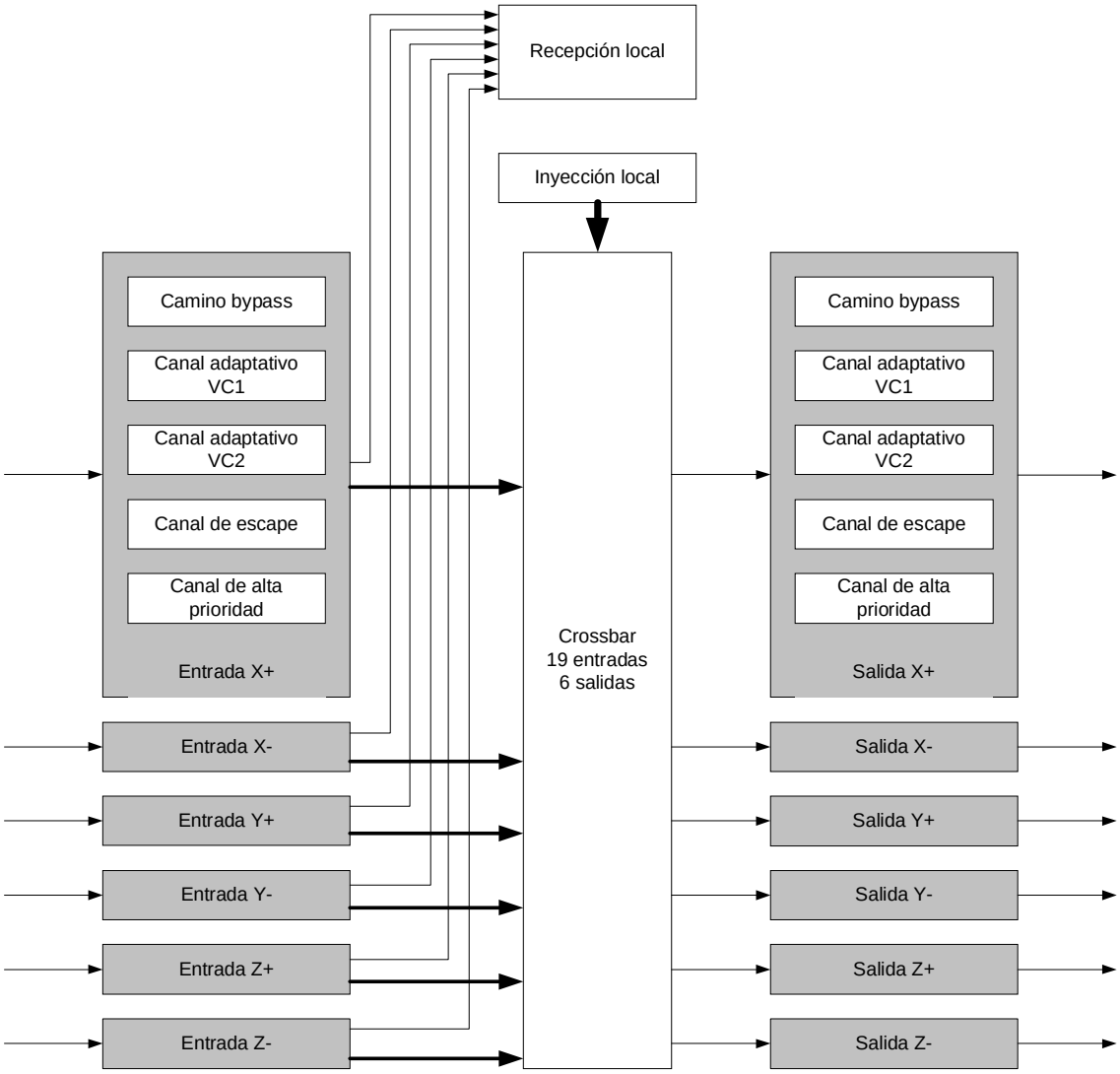

Figura 16. La red toroidal del BG/L. Figura realizada a partir de esquemas que aparecen en [ADI 2002, HB 2002]. Nótese cómo cada canal físico lo comparten cinco canales virtuales.

La estructura del encaminador es la representada en la Figura 16. Algunos comentarios sobre ella:

- Cada enlace interno entre una entrada y el crossbar es doble (2 bytes). El enlace entre el puerto de inyección y el crossbar es de anchura 7. Por eso el crossbar es de 19 (6x2 + 7) por 6
- El resto de los enlaces internos (de entrada a recepción, de crossbar a salida) son de un byte
- No representado, hay un enlace que, sin pasar por el crossbar, une cada entrada con su correspondiente salida[17]
- Los enlaces de entrada / salida son serie

El diseño está pensado para que la mayor parte del tráfico utilice los dos canales virtuales adaptativos (VC1, VC2). Un sistema de tokens sirve para prevenir desbordamientos de las colas de los nodos vecinos. En canal de escape se usa encaminamiento estático, con el método de la burbuja para prevenir interbloqueos.

[17] Se supone que este enlace lo utilizan los canales virtuales etiquetados con "camino bypass".

Otro canal, de alta prioridad sirve para mensajes entre los núcleos del SO de los nodos (las aplicaciones no pueden acceder a ese canal). En cuanto al almacenamiento, cada CV tiene un búfer de 1 Kbyte.

Estos son algunos datos adicionales sobre el encaminamiento adaptativo empleado con los canales virtuales VC1 y VC2.

- Se usa la técnica habitual de etiquetar los paquetes con su registro de encaminamiento en origen
- Unos bits (*hint-bits*) en la cabecera del paquete indican las direcciones posibles: 011000 indica que pueden realizarse movimientos en los ejes X- e Y+ (pero no en los X+, Y-, Z+, Z-). Estos bits se modifican conforme el paquete avanza por la red
- Política de selección: cuando un paquete va avanzando, de entre los canales virtuales posibles para el siguiente salto (teniendo en cuenta la dirección y sentido que lleva el paquete), se solicita aquel que lleva el paquete a la cola más corta de entre las viables (SHORTEST_REQ)
- Política de arbitraje: cuando hay varios paquetes, en diferentes colas, que quieren salir por el mismo puerto, se asigna dicho puerto al paquete que viene de la cola de entrada más larga (LONGEST_ARB)

Se nos indica también que se han incorporado al diseño mecanismos para tolerancia a fallos: el encaminador funciona aunque el resto del nodo falle y es posible establecer los hint-bits por software, para evitar nodos/enlaces rotos. Con esto, nos aseguran, se puede mantener conectividad total en la red con hasta 3 fallos (no co-lineares) en una partición.

Apéndice B: Visión general del Red Storm

En este apéndice se facilitan algunos datos de diseño que se van conociendo sobre el proyecto Red Storm, una iniciativa conjunta de Sandia National Laboratories y Cray Inc. La información de esta sección está extraída en su totalidad de [CT 2003].

El Red Storm será un procesador masivamente paralelo, que dará una imagen de sistema único (es decir, no se trata de un cluster poco integrado). Como la mayoría de los MPPs, consiste en un conjunto de nodos de cómputo, cada uno con su propia memoria, interconectados mediante una red de alto rendimiento: una malla 3D. Cada nodo tiene un acceso (bidireccional) a esa red.

El núcleo de cada nodo de cómputo lo formarán procesadores AMD 64 a 2 GHz. El número de nodos previsto es 10.368, organizados en una red de 27x16x24 nodos, y empaquetados en 108 armarios. La memoria total alcanzará los 10 TB (memoria DDR a 333 MHz).

El diseño es tal que permite su fácil particionado. Desde el principio, se organizará en tres áreas separadas mediante armarios especiales. Una cuarta parte de la máquina es "negra" (dedicada a trabajos no clasificados), otra cuarta parte es "roja" (dedicada a trabajos clasificados) y la mitad central es "blanca" (conmutable). En cada extremo están dispuestos, además, una colección de nodos de E/s, conectados al sistema de almacenamiento masivo (hasta 240 TB).

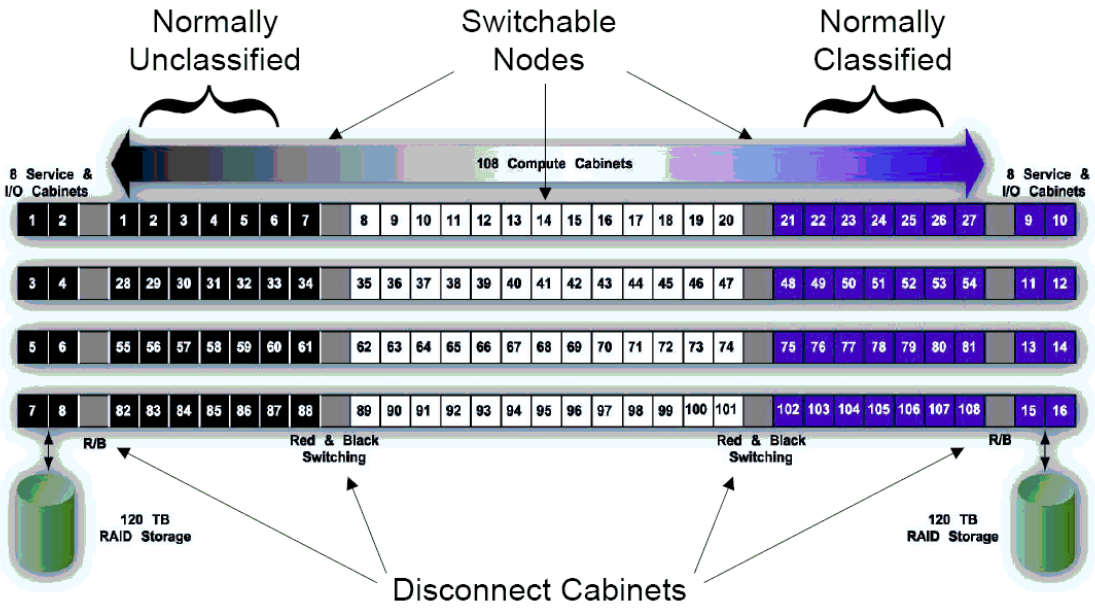

Figura 17. Distribución física de los armarios de un sistema Red Storm.

Los armarios de cómputo tienen esta organización:

- Cada armario tiene tres módulos de cómputo, además de sistemas de ventilación, alimentación y cableados diversos
- Un módulo de cómputo consta de 8 tarjetas
- Cada tarjeta cuenta con 4 procesadores y 4 encaminadores de red

La máquina estará en funcionamiento en 2004, y se espera que cumpla estas especificaciones de rendimiento:

- ~40 TeraFLOP/s de pico
- Ancho de banda agregado memoria/procesador: ~55 TB/s
- Interconexión
    - Ancho de banda bidireccional de los enlaces: ~6 GB/s pico, 4,1 GB/s sostenidos
    - Ancho de banda de la bisección: ~2,3 TB/s pico, 1,6 TB/s sostenidos
    - Latencia entre vecinos <2µs; latencia de extremo a extremo <5µs
- E/S
    - Ancho de banda del sistema de ficheros: 50 GB/s por cada color
    - Ancho de banda con la red exterior: 25 GB/s por cada color