UNIVERSIDAD DE CANTABRIA

DPTO. DE ELECTRÓNICA Y COMPUTADORES.

IMPACTO DEL SUBSISTEMA DE COMUNICACIÓN EN EL RENDIMIENTO DE LOS COMPUTADORES PARALELOS: DESDE EL HARDWARE HASTA LAS APLICACIONES.

Presentada por:

Valentin Puente Varona

Dirigida por:

Ramón Beivide Palacio.

SANTANDER, OCTUBRE DE 1999

Capítulo 3

Encaminamiento: una Nueva Propuesta Para Balancear Latencia y Throughput.

El objetivo central de este capítulo persigue la evaluación de diversos mecanismos de encaminamiento. Se pone especial énfasis en la evaluación de rendimiento de mecanismos de encaminamiento adaptativo sobre redes directas. En esta línea se propone un nuevo algoritmo de encaminamiento general que permite evitar el interbloqueo con un coste inferior a otros métodos preexistentes. En concreto, se ha estudiado su comportamiento, comparándolo con otras alternativas al uso, tanto deterministas como adaptativas, sobre redes k-ary n-cube.

3.1 Introducción.

Uno de los parámetros que influye de forma más decisiva en el rendimiento de las redes de interconexión es el algoritmo de encaminamiento. Este es el encargado de dirigir cada uno de los mensajes que circulan por la red desde su origen a su destino. Es conveniente que el mecanismo permita una utilización óptima de los recursos *hardware* que aporta la red sin dar lugar a anomalías o funcionamientos incorrectos que puedan limitar o afectar gravemente a la fiabilidad del sistema global.

Existe un número, dificil de cuantificar, de mecanismos de encaminamiento propuestos a lo largo de la bibliografía, que dependen fuertemente del tipo de topología, número de destinatarios, etc. El análisis presentado en este capítulo se centra en estudiar un aspecto clave del encaminamiento como es la adaptatividad. En particular, analizaremos el problema para el caso de las redes directas con tráfico *unicast*.

En primera aproximación, los mecanismos de encaminamiento pueden clasificarse en deterministas y adaptativos dependiendo si se contempla que los paquetes en el camino hacia su nodo destino, puedan evitar o no las zonas de congestión de la red.

3.1.1 Encaminamiento Determinista.

Los mecanismos de encaminamiento deterministas establecen el camino que han de seguir los paquetes atendiendo únicamente a la dirección del destinatario. De esta forma, entre dos nodos cualesquiera de la red el camino seguido es siempre fijo, independientemente de las condiciones del tráfico. La principal virtud de este mecanismo de encaminamiento es que permite simplificar considerablemente la complejidad del encaminador desde el punto de vista de su implementación. Esto permite lograr estructuras *hardware* muy sencillas, lo que ha potenciado su uso en diversos sistemas reales ya sean comerciales como el *Intel Paragon* [67], el *Cray T3D* [74] o el *SGI Origin 2000* [82] o académicos como el *Stanford DASH* [85]. El uso de este mecanismo de encaminamiento es habitual cuando el control de flujo es de tipo segmentado ya que facilita que las etapas del encaminador se encuentren adecuadamente balanceadas. Además de la sencillez de implementación de este mecanismo de encaminamiento otra ventaja adicional es que, en determinadas topologías, evita los interbloqueos sin necesidad de recurrir a mecanismos adicionales. Los casos más típicos son, la malla donde el encaminamiento en orden de dimensión o DOR (*Dimension Order Routing*) y el hipercubo con encaminamiento e-cube[122] eliminan la posibilidad de que se produzca *deadlock*.

En el caso de otras redes directas *k-ary n-cube*, como el toro, las simplicidad de implementación se mantiene, pero es necesario añadir mecanismos adicionales para la evitación del *deadlock*. En el caso de la malla y el hipercubo las dependencias entre canales son acíclicas con los mecanismos de encaminamiento deterministas pero sin embargo en el toro se forman ciclos de dependencias. Como se muestra en [40] este hecho puede dar lugar que la red no sea libre de interbloqueos. En ese sentido existen mecanismos que permiten solventar el problema eliminando los ciclos mediante el empleo de canales virtuales, como en la metodología propuesta por *Dally* y *Seitz* en [40].

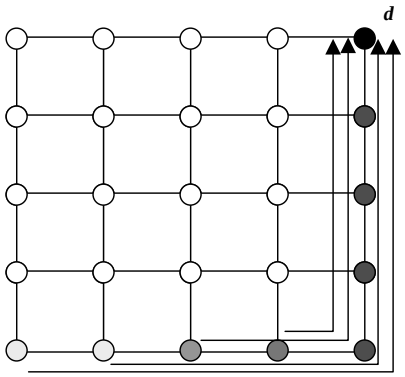


Figura 3-1. Infrautilización de los recursos con mecanismos de encaminamiento deterministas.

Como contrapartida a la sencillez, el encaminamiento determinista tiende a infrautilizar los recursos de la red. En consecuencia su *throughput* máximo puede llegar a verse seriamente comprometido. Este tipo de pérdidas de rendimiento se produce generalmente cuando el patrón de comunicación que ha de soportar la red no es uniforme. Un ejemplo típico es el que se muestra en la Figura 3-1. En este caso, todos los nodos de la fila inferior están enviando información al nodo etiquetado como *d*. Al establecer el camino de forma estática (en este caso, en orden creciente de dimensión) la descompensación en la utilización de los recursos es extrema ya que, de toda la red, los únicos espacios de almacenamiento y enlaces que se están empleando son los de la fila inferior y la última columna. El resto de recursos no son aprovechados en absoluto lo que hace que el rendimiento ofrecido por la red sea extremadamente menor que el podríamos alcanzar de utilizar el resto de enlaces y espacios de almacenamiento.

3.1.2 Encaminamiento Adaptativo.

El modo de solucionar la infrautilización de recursos, característica de los mecanismos de encaminamiento deterministas, es aportar un cierto grado de adaptatividad al encaminar los mensajes en los nodos intermedios. En este caso, el mecanismo de decisión no actúa de forma estática sino que tiene en cuenta el estado de la red a la hora de decidir por qué canal ha de ser retrans-

mitido el mensaje en cada nodo intermedio. Esta decisión puede ser tomada en base a información local (en función del estado de los canales de comunicación con los encaminadores vecinos) o en base a información global.

Además de la lógica ganancia en prestaciones, otra ventaja del encaminamiento adaptativo es que aporta un cierto grado de tolerancia a fallos. En el caso de que se produzca la rotura de uno o más enlaces de la red, es posible que el sistema siga funcionando al estar contemplados varios caminos alternativos para los mensajes.

Este tipo de mecanismos logra mejorar considerablemente el rendimiento en términos de *throughput* frente al caso del encaminamiento determinista, pero implican una complicación en el *hardware* del encaminador, incrementando de esta forma su coste. Este ha sido un argumento muy tenido en cuenta a la hora de construir muchos sistemas reales decantando la elección hacia el encaminamiento determinista. Aunque existen sistemas reales como el *Cray T3E* [114], *Servernet-II* [57] o el *S3.mp* [94] que si emplean este tipo de encaminamiento, también es cierto que son mayoría los casos donde se emplea encaminamiento determinista. Sin duda, la complejidad y coste asociado son las principales razones que han limitado la extensión de esta metodología en los sistemas reales. Además de la complejidad añadida de los mecanismos de decisión y arbitrio, el encaminamiento adaptativo pueda dar lugar a problemas de interbloqueo en situaciones en las que no sucedería para el caso del encaminamiento determinista. Esto hace necesario incorporar nuevos recursos en forma, por ejemplo, de canales virtuales adicionales, lo que siempre tiende a incrementar el coste del encaminador.

El mecanismo de decisión en este tipo de encaminamiento está basado en dos etapas distintas. En primer lugar, la *Función de Encaminamiento* aporta cuáles son los canales de salida permitidos para un mensaje dado. En segundo lugar, la *Función de Selección* es la encargada de elegir uno solo de los caminos o canales alternativos facilitados por la función de encaminamiento. El mensaje sera enviado definitivamente por este canal.

3.1.2.1 Función de Encaminamiento.

La función de encaminamiento evalúa el destino del paquete en el instante que éste ha logrado el arbitrio para avanzar y en función del destino y la posición del nodo actual establece cuales son los caminos permitidos en el avance del mensaje. Atendiendo a este subconjunto de canales, podemos clasificar los algoritmos de encaminamiento adaptativos de la siguiente forma:

Encaminamiento Parcialmente Adaptativo.

El subconjunto de canales de salida ofrecido por la función de encaminamiento no es completo, en el sentido de que se eliminan determinados canales con el fin de evitar posibles anomalías en la red o intentar compensar el coste introducido por el mecanismo de encaminamiento. Algunos ejemplos típicos son el *Turn-Model*[59] y el PAR (*Planar-Adaptive Routing*) [31]. En el primer caso se suprime la posibilidad de que se produzcan determinados giros logrando, de esta forma, que no aparezcan ciclos de dependencia entre canales. En el segundo caso, la idea es reducir el número de recursos necesarios para aplicar la adaptatividad restringiendo las dimensiones en que puede ser aplicada a únicamente dos, es decir, la adaptatividad puede ser aplicada únicamente dentro de un plano para redes con un número de dimensiones superior.

Encaminamiento completamente Adaptativo.

En los mecanismos de encaminamiento completamente adaptativos, la función de encaminamiento no restringe artificialmente los posibles canales de salida para ningún paquete. Generalmente los encaminadores son más costosos que los que emplean adaptatividad parcial, pero logran incrementar considerablemente el nivel de utilización de los recursos.

Por otro lado, en función del camino seguido por los mensajes hacia su destino podemos clasificar los mecanismos de encaminamiento en mínimos y no mínimos. Esta clasificación es también aplicable a los algoritmos de encaminamiento deterministas en el caso de plantear o no sistemas de tolerancia a fallos. En el contexto de los algoritmos de encaminamiento adaptativo sus características fundamentales son:

Encaminamiento Mínimo.

El subconjunto de posibles canales facilitado por la función de selección es tal que todos ellos siempre acercan el paquete hacia su destino. Restringe sensiblemente la tolerancia a fallos, pero logra mejores rendimientos que los sistemas no mínimos.

Encaminamiento no Mínimo.

En este caso, una parte de los canales facilitados por la función de encaminamiento alejan el paquete de su destino. La ideas perseguidas con este tipo de metodologías son evitar problemas de *deadlock* como en el *Chaos Router* [20], incrementar la tolerancia a fallos en el sistema o mejorar el balanceo de la carga [53]. Como contrapartida pueden aparecer problemas de *livelock*. En general el rendimiento obtenido por estos sistemas suele ser inferior a los ofrecidos por los que emplean encaminamiento mínimo [45].

3.1.2.2 Función de Selección

La función de selección es la encargada de elegir uno de entre todos los canales de salida facilitados por la función de encaminamiento. La elección, generalmente, se basa en criterios dinámicos en el sentido de que el estado de la red influye en la selección del canal definitivo. Algunas de las funciones de selección más habituales se señalan a continuación:

Zig-Zag

La estrategia de esta función de selección es intentar mantener el número de caminos alternativos del paquete en su avance al destino [9]. De esta manera, en caso de producirse contención, se maximizará la utilización de caminos alternativos hasta el salto previo al destino. Para lograr esto, por ejemplo en el caso de una red *k-ary 2-cube* como se puede ver en la Figura 3-2.a con encaminamiento mínimo, la idea es seleccionar el camino por la dimensión que, en ausencia de contención, esté más alejada del destino.

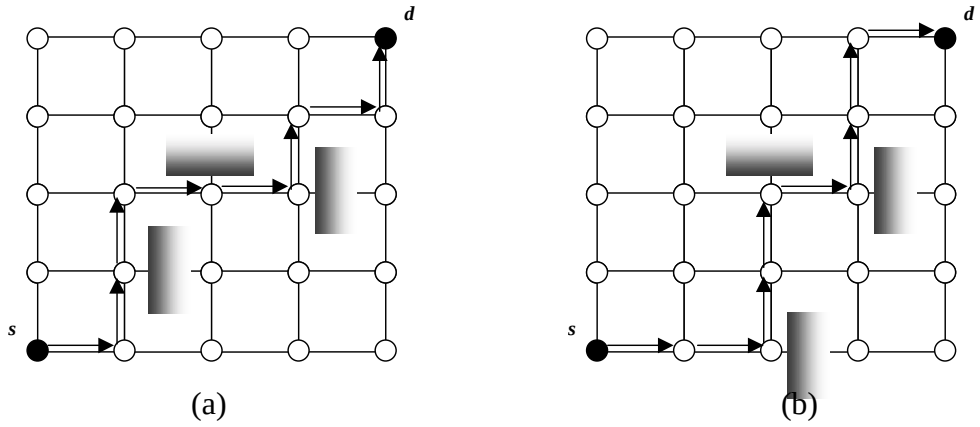


Figura 3-2. (a) Función de selección en Zig-Zag. (b) Función de selección X/Y dinámica.

X/Y Dinámica.

En esta función de selección la elección del canal de salida se hace exclusivamente en términos del estado de la red. Se puede entender como una extensión del encaminamiento en orden de dimensión en el sentido de que el paquete comienza su recorrido a lo largo de la dimensión inferior y solo cambiará de dimensión en el caso de encontrar un bloqueo. En ese caso, pasará a viajar por la dimensión inmediatamente superior hasta agotarla o encontrar de nuevo una congestión en la red. Aparentemente, esta función de selección es menos evolucionada que la anterior pero sin embargo, como se muestra en [7], permite alcanzar mejor rendimiento en la mayoría de los casos. En la Figura 3-2.b se muestra el modo en que opera esta función de selección para el caso de una red *k-ary 2-cube*.

Basadas en Históricos.

Existe otro conjunto de funciones de selección más evolucionadas que las anteriores y que no solo basan la toma de decisión en términos de la información aportada por los encaminadores vecinos en el momento de avanzar el mensaje sino en registros históricos que mantienen el estado de los enlaces en instantes previos. La idea que se persigue con este tipo de funciones es especular en qué estado se encuentra la red más allá del nodo vecino. Algunas de estas funciones de selección son *LRU*, *LFU* [131] ó *MAX-CREDITS* [55]. En el caso de *LFU* el canal seleccionado es aquel que menos ha sido utilizado en el pasado. En el caso de *LRU* el canal seleccionado será el que más tiempo haga que no ha sido utilizado. En ambos casos es preciso mantener un contador por canal de salida en el que se mantenga el número de paquetes enviados o el tiempo que ha transcurrido desde que se envió un paquete por ese canal. Por otro lado, *MAX-CREDITS* es posible emplearla cuando el control de flujo está basado en créditos y la idea es utilizar el canal que mayor número de créditos disponibles tenga. En todos los casos la elección esta basada en heurísticas y como se demuestra en [131] los resultados obtenidos son realmente buenos. Sin embargo, en ese trabajo a la hora de valorar estas funciones de selección se obvian sus implicaciones tecnológicas. Es claro que la complejidad que implican (contadores, comparadores, sumadores, etc...) en la máquina de estados encargada de gestionar el encaminamiento de los mensajes va a ser muy superior a cualquiera de las dos precedentes. En consecuencia, no está claro en qué medida las mejoras de rendimiento alcanzadas desde el punto de vista funcional se trasladan a la realidad. Bajo nuestro punto de vista, recurrir a este tipo de estrategias incrementará considerablemente tanto el área del encaminador como su tiempo de paso ya que la función de selección se encuentra en el camino crítico del encaminador.

3.2 Algoritmos de Encaminamiento Adaptativos para Redes *k-ary n-cube*. Anomalías asociadas.

En el punto precedente hemos esbozado de forma sencilla cuales son los aspectos clave a tener en cuenta a la hora de proponer y analizar un mecanismo de encaminamiento. En este punto nos centraremos en el caso de las redes *k-ary n-cube*. Este tipo de redes han sido empleadas extensamente en diversos sistemas reales dadas sus buenas características (Ver Sección 2.2.1). Por lo tanto, representan un entorno adecuado a la hora de analizar diversas soluciones o alternativas en los mecanismos de encaminamiento. Dentro de estas redes hemos optado por aquellas con bajo número de dimensiones puesto que permiten alcanzar mejor rendimiento que redes con un número superior de dimensiones, como se muestra en [2][41].

En este contexto, como se ha señalado, los mecanismos de encaminamiento adaptativos introducen un coste sobre el encaminamiento determinista por dos motivos. Por un lado, la adapta-

tividad incrementa las posibilidades de que se produzca un interbloqueo entre los mensajes que se encuentran en la red, esto hace que sea necesario el proponer sistemas de evitación adicionales, incrementando el coste del encaminador. Por otro lado, los mecanismos de arbitraje son substancialmente más complejos que en el caso determinista.

Para las redes *k-ary n-cube* con encaminamiento determinista, la aparición de *deadlock* esta causada por la existencia de enlaces periféricos que dan lugar a ciclos de dependencia entre recursos. Por esta razón, una malla es libre de *deadlock* con este tipo de encaminamiento. En la Figura 3-3 se puede apreciar la formación de un interbloqueo en una red *3-ary 1-cube* unidireccional. En este caso se aprecia como cada uno de los mensajes no va a poder avanzar nunca ya que, indirectamente, los recursos que necesita para progresar se encuentran ocupados y no pueden ser liberados hasta que no se liberen los que ocupan cada uno de ellos. Por lo tanto, se ha formado una dependencia cíclica que da lugar a un bloqueo indefinido de los mensajes. En el caso de una red bidireccional con mayor número de dimensiones la aparición de interbloqueos está originada por la misma causa.

En el caso del encaminamiento adaptativo los ciclos de dependencia no solo se forman como consecuencia de los enlaces periféricos sino que pueden aparecer incluso entre nodos que no están en un mismo anillo topológico. Esto hace que, hasta en redes como la malla, se pueda producir *deadlock* cuando el encaminamiento determinista logra evitarlo. En la Figura 3-4.(a) se puede apreciar la causa típica de *deadlock* con encaminamiento determinista y en la Figura 3-4.(b) un caso causado por un encaminamiento adaptativo para una red *5-ary 2-cube*. El segundo caso jamás se puede dar si empleamos un encaminamiento en orden de dimensión dado que los giros desde una dimensión superior a una inferior no son posibles.

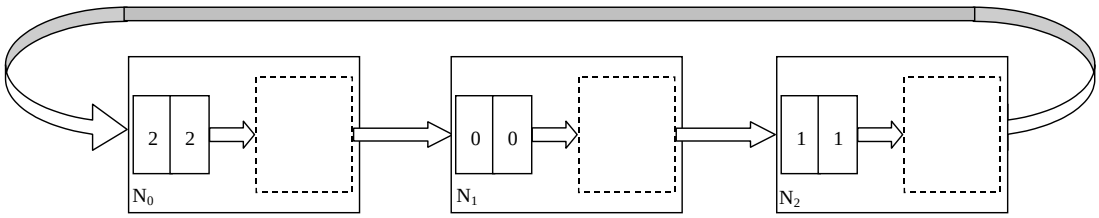


Figura 3-3. Dependencia cíclica de recursos en un anillo

.

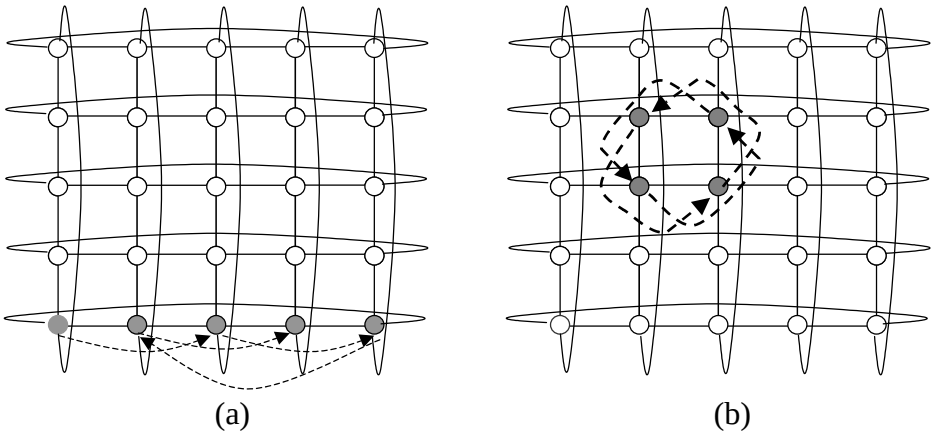


Figura 3-4. Producción de deadlock con encaminamiento determinista (a) y con adaptativo (b) en un Toro 2-D.

Añadir mecanismos adicionales de evitación de interbloqueos implica incrementar el coste del encaminador al ser necesario incorporar más recursos que en el caso determinista. En este capítulo nos centraremos en repasar diversos métodos existentes y proponer un nuevo método, que permite minimizar el coste añadido de la adaptatividad.

3.3 Nueva Teoría de Evitación de Interbloqueos con Encaminamiento Completamente Adaptativo.

Ha quedado clara la problemática asociada al encaminamiento adaptativo, en el sentido de que es necesario contemplar mecanismos de evitación de bloqueo adicionales sobre los empleados en el encaminamiento determinista. A lo largo de la bibliografía existe una gran número de propuestas relacionadas con este hecho. Algunas de ellas son [20][31][38][43][44].

En general, las redes pueden sufrir tres tipos de anomalías: inanición, interbloqueo y *livelock*. De estas tres problemáticas, el interbloqueo o *deadlock* es la más compleja de solucionar. En general, a la hora de evitar el problema se suele recurrir a políticas o métodos conservadores que eliminan las posibilidad de que ocurra el interbloqueo ya sea previniéndolo o evitándolo. En el primer caso, se limita el empleo de determinados recursos de la red, independientemente del estado en que se encuentren. En el segundo caso la restricción depende del estado actual, es decir, antes de ocupar un recurso se analiza, en función del estado, si puede ser peligroso y en el caso de que sea así no se concede. Existe otro tipo de metodologías que abordan el problema recuperando la red en el caso de que ocurra un interbloqueo [117]. En este caso no se restringe el uso de los recursos de la red y exclusivamente si se detecta la aparición de un interbloqueo se pone en marcha un mecanismo de recuperación que permite a la red seguir operando correctamente.

En nuestro caso nos centraremos en mecanismos de evitación proponiendo una nueva estrategia general para cualquier tipo de red directa y después particularizaremos sobre la red de interconexión elegida. Nos centraremos en el análisis comparativo de dos propuestas: en primer lugar consideraremos un arquitectura basada en el mecanismo descrito en [43] y propondremos una nueva arquitectura basada en un mecanismo de evitación de *deadlock* desarrollado en este trabajo. En ambos casos, el análisis se complementará no solo comparando cada alternativa entre sí, sino que se incorporará en el estudio el análisis de encaminadores deterministas [40][24]. A continuación se indica de manera formalizada, el soporte teórico del mecanismo propuesto y posteriormente se indicará intuitivamente de qué manera opera.

3.3.1 Condiciones Básicas

Las condiciones básicas que definen el entorno de validez de nuestra propuesta son muy similares a las consideradas en [43]. Estas son las siguientes:

- Un nodo puede generar mensajes de longitud arbitraria, con cualquier destino en la red y a cualquier tasa de inyección. Los mensajes que superen el tamaño establecido por el control de flujo serán divididos en paquetes de longitud fija.
- El control de flujo empleado es *Virtual Cut-Through*.
- Cualquier paquete, al llegar a su nodo destino, será consumido en un tiempo finito.
- Cada canal tiene una única cola asociada, permitiendo el almacenamiento de un número finito de paquetes. Un canal solamente podrá aceptar un paquete si tiene espacio suficiente para almacenarlo completamente. El espacio en el buffer debe ser reservado antes de comenzar la transmisión del paquete. Las colas asociadas pueden encontrarse tanto en la entrada como en la salida del encaminador.
- El camino seguido por cualquier paquete depende de su destino, el nodo actual en que se encuentra, la cola en la que esta almacenado y el estado de los canales o colas solicitados.
- En el caso de que varios paquetes se encuentren esperando, porque todos los recursos solicitados se encuentren ocupados, son atendidos por un mecanismo de arbitrio que sigue una política *Round-Robin*. De esta manera se elimina la posibilidad de *starvation* en la planificación interna del encaminador.
- Las situaciones válidas para la red son aquellas que son alcanzables por el mecanismo de encaminamiento y control de flujo, partiendo siempre de una red vacía.

3.3.2 Conceptos Generales.

En esta sección se introducirán algunas definiciones básicas propuestas en [40] y [43] permitiendo así que el desarrollo teórico, que se describirá más tarde, sea autocontenido. Además, en contraste con la definición clásica de interbloqueo, redefiniremos el concepto de forma más general. Por otro lado, se introduce una nueva definición formal de control de flujo descrita en [25] y algunas otras definiciones menores que son necesarias para describir la propuesta presentada.

Definición 3-1. Una *red de interconexión, I*, puede representarse en términos de un grafo conexo *I=G(N,Q)* donde los vértices de grafo, *N*, representan el conjunto de procesadores o encaminadores que integran la red. Los arcos del grafo, *Q*, representan el conjunto de colas asociadas a los canales que los interconecta. Cada cola $q_i \quad Q$ posee una capacidad asociada $cap(q_i)$. El número de paquetes que tiene almacenados esa cola viene denotado por $size(q_i)$. Los nodos origen y destino de q_i vienen denotados por s_i y d_i, respectivamente. Por simplicidad, cualquier conjunto de colas dado puede ser enumerado como $q_1, q_2, q_3, \ldots, q_k$ ó $q_{i_1}, q_{i_2}, q_{i_3}, \ldots, q_{i_k}$, no implicando la numeración ninguna ordenación en las colas.

Cada nodo consiste en uno o más procesadores, una memoria y un encaminador. Los nodos están conectados, en general, por medio de enlaces bidireccionales. Cada enlace bidireccional consiste en dos colas de comunicación situadas en direcciones opuestas. El conjunto de colas de comunicación, también denominado colas de la red o colas externas, esta denotado por Q_N. Cada procesador se encuentra conectado a un encaminador por una o más colas de inyección y de consumo o entrega. El conjunto de colas de inyección y consumo viene denotado por Q_I y Q_D respectivamente.

En base a estos conjuntos de colas podemos definir Q_{ND} como la unión del conjunto formado por las colas de comunicación y las de consumo o entrega.

$$Q_{ND} = Q_N \quad Q_D \qquad \textbf{[3-1].}$$

Tomando en cuenta el conjunto de colas de inyección podemos definir Q_{IN} como:

$$Q_{IN} = Q_I \quad Q_N \qquad \textbf{[3-2].}$$

Finalmente, el conjunto de todas las colas de la red puede ser obtenido como:

$$Q = Q_{IN} \quad Q_{ND} \qquad \textbf{[3-3].}$$

Definición 3-2. Una *función de encaminamiento,* R: $N \times Q_{IN} \rightarrow P(Q_{ND})$, donde $P(Q_{ND})$ es un subconjunto de colas perteneciente a Q_{ND}, proporciona un conjunto de colas alternativas para encaminar el paquete p almacenado en la cola q_i. El conjunto de colas alternativo, facilitado por la función de encaminamiento, viene denotado por $R(q_i, n_d)$, donde n_d representa el nodo destino de *p.*

En el caso del encaminamiento determinista, la función *R* proporciona un único canal o cola. Sin embargo, en el caso adaptativo la función de encaminamiento proporciona un conjunto de uno o más canales de salida. Este conjunto puede ser restringido artificialmente de cara a la evitación de interbloqueos en la red.

Definición 3-3. Una *subfunción de encaminamiento,* R_s, para una función de routing dada *R*, es una función de encaminamiento definida en el mismo dominio que *R* pero su rango (conjunto de colas alternativas facilitadas) es un subconjunto restringido del original, es decir sobre un subconjunto $Q_{ND_s} \subseteq Q_{ND}$. Entonces la subfunción R_s vendrá dada por:

$$R_s: N \times Q_{IN} \rightarrow P(Q_{ND_s}) \qquad \textbf{[3-4].}$$

Un modo alternativo y útil para definir la subfunción de routing es:

$$R_s(q_i, n_d) = R(q_i, n_d) \bigcap Q_{ND_s}(\forall q_i \quad Q_{IN})(n_d \quad N) \qquad \textbf{[3-5].}$$

Definición 3-4. Un *función de control de flujo,* $F; Q_{IN} \times Q_{ND} \times Z \rightarrow \{true, false\}$, donde Z representa un conjunto de números enteros, determina el permiso de acceso por parte de un paquete p almacenado en el buffer asociado a la cola q_i al buffer asociado a la cola $q_j \quad R(q_i, n_d)$. El conjunto de números enteros del dominio de *F* representa el número de paquetes almacenado en el buffer asociado a q_j, es decir $size(q_j)$. De esta manera el paquete *p* solo podrá avanzar desde q_i hasta q_j si $F(q_i, q_j, size(q_j)) = true$. Por lo tanto, el valor que tome *F* en cada caso depende del estado de la red.

La función de control de flujo limita el conjunto de colas alternativas facilitadas por la función de encaminamiento. El conjunto de colas validas, $Q_{ND}^{v}(q_i, n_d)$, para un paquete *p* almacenado en la cola $q_i \quad Q_{IN}$ y con destino $n_d \quad N$, viene dado por:

$$Q_{ND}^{v}(q_i, n_d) = \{q_i \quad Q_{IN} \mid q_j \in R(q_i, n_d) \quad F(q_i, q_j, size(q_j)) = true\} \qquad \textbf{[3-6].}$$

Definición 3-5. Una *subfunción de control de flujo,* F_S, de una función de control de flujo $F: Q_{IN} \times Q_{ND} \times Z \rightarrow \{true, false\}$ viene dada por:

$$F_S: Q_{IN} \times Q_{ND_s} \times Z \rightarrow \{true, false\} \quad \textbf{[3-7].}$$

donde:

$$F_S(q_i, q_j, size(q_j)) = F(q_i, q_j, size(q_j)) \forall q_i \in Q_{IN}, \forall q_j \in Q_{ND_s} \quad \textbf{[3-8].}$$

Definición 3-6. Se define *H* como el conjunto de estados en que puede encontrarse una cola atendiendo al estado de ocupación del buffer asociado y que viene descrito por $\{llena, no-llena\}$

Definición 3-7. Una *función de selección* $S: Q^v_{ND}(q_i, n_d) \times H \rightarrow Q^v_{ND}$ escoge de entre el conjunto de colas válidas $Q^v_{ND}(q_i, n_d)$ una única cola a la que será encaminado el paquete *p* almacenado en la cola $q_i \in Q_{IN}$. El criterio para seleccionar la cola definitiva a la que será enviado el paquete puede estar basado en criterios dinámicas o estáticas.

Definición 3-8. Una *configuración* es un estado en el que se puede encontrar la red y que viene descrito por el número de paquetes que hay en que cada cola de la red. Si denotamos como $dest(p_j)$ el nodo destino del paquete p_j y si el primer paquete almacenado en la cola q_i tiene como destino n_d entonces $head(q_i) = n_d$. Bajo estas condiciones, una configuración es legal si y solo si:

$$\begin{array}{l} \forall q_i \in Q, size(q_i) \leq cap(q_i) \quad \forall q_i \in Q_{ND}, \\ \forall p_j \in q_j, \exists q_0 \in Q_I, \exists (q_1, q_2, \ldots, q_{i-1}) \in Q_N \\ \text{tal que } q_m \in R(q_{m-1}, head(q_m)), m = 1, \ldots, i \end{array} \quad \textbf{[3-9].}$$

Para cada cola, la capacidad no puede ser sobrepasada por los paquetes almacenados en los distintos bufferes asociados. Además, todos los paquetes han llegado a esa posición de acuerdo con la función de encaminamiento.

Definición 3-9. Una *configuración interbloqueada* para una red de interconexión dada *I*, una función de encaminamiento *R* y una función de control de flujo *F* es una configuración no vacía que verifica la siguiente condición:

$$\forall q_i \in Q_{IN}, \text{ tal que } size(q_i) > 0 \left\{ \begin{array}{c} head(q_i) \neq d_i \\ F(q_i, q_j, size(q_i)) \equiv false \quad \forall q_i \in R(q_i, head(q_i)) \end{array} \right. \quad \textbf{[3-10].}$$

En una configuración interbloqueada ningún paquete ha llegado a su destino y tampoco pueden avanzar en su camino dado que la función control de flujo no permite el uso de ninguna de las colas alternativas facilitadas por la función de encaminamiento.

Definición 3-10. Una red de interconexión *I, y* el par *(R,F),* donde *R* representa la función de encaminamiento y *F* la función control de flujo, es libre de interbloqueo si y solo si no existe ninguna configuración interbloqueada para ese par y esa red.

Definición 3-11. Una función de encaminamiento *R* para una red de interconexión *I* es *conexa* si y solo si para cualquier configuración legal se verifica:

$$\forall q_i \in Q_{IN}, \text{ tal que } size(q_i) > 0, \exists q_{i+1}, q_{i+2}, \ldots, q_{i+k-1} \in Q_N \ , \exists q_{i+k} \in Q_D \ , \quad \text{[3-11].}$$
$$\text{tal que } q_m \in R(q_{m-1}, head(q_i)), \quad m=i+1,\ldots, i+k$$

En otras palabras, es siempre posible establecer un camino para cualquier paquete almacenado en el buffer actual hacia su destino (cola de entrega o consumo). Notar que la configuración ha de ser necesariamente legal, de otro modo la función de encaminamiento puede no facilitar ninguna próxima cola.

Definición 3-12. Dada una red de interconexión *I*, una función de encaminamiento *R*, y un par de colas $q_i, q_j \in Q$, existe una *dependencia estática* entre q_i y q_j si existe alguna configuración legal con paquetes almacenados en el buffer asociado a q_i y:

$$q_j \in R(q_i, head(q_i)) \quad \text{[3-12].}$$

Es decir, q_j puede ser solicitada por paquetes almacenados en q_i en el caso de algunas configuraciones legales.

Definición 3-13. Dada una red de interconexión *I*, una función de encaminamiento *R* y una función de control de flujo *F*, existirá una *dependencia dinámica* entre un par de colas $q_i, q_j \in Q$ si y solo si existe una configuración legal tal que existen paquetes almacenados en el buffer asociado a q_i que verifican:

$$q_j \in R(q_i, head(q_i)) \wedge F(q_i, q_j, size(q_i)) \ = \ false \quad \text{[3-13].}$$

Es decir, q_j es una cola facilitada por la función de encaminamiento para el paquete almacenado en la cola q_i. Sin embargo, en un instante de tiempo dado, el paquete que esta solicitando

el recurso no puede obtenerlo dado que la función de control de flujo se lo deniega. Notar que, a diferencia de las dependencias estáticas, las dependencias dinámicas están sujetas al estado de la red y por lo tanto varían en cada instante de tiempo.

Definición 3-14. Un *Grafo de Dependencias Estáticas entre Recursos*, denotado por *D*, para una red de interconexión *I=G(N,Q)* y una función de encaminamiento *R* es un grafo dirigido *D=G(Q,E)*. Los vértices del grafo *D* son las colas de *I*. Los arcos de *D* están constituidos por los pares de colas (q_i, q_j) tales que exista una dependencia estática entre q_i y q_j.

Definición 3-15. Un *Grafo de Dependencias Dinámicas entre Recursos*, denotado por D_d, para una red de interconexión *I=G(N,Q)*, una función de encaminamiento *R* y una función de control de flujo *F*, es un grafo dirigido $D_d=G(Q,E_d)$. Los vértices del grafo D_d son las colas de *I*. Los arcos de D_d están constituidos por los pares de colas (q_i, q_j) tales que exista una dependencia dinámica entre q_i y q_j.

Cuando nos refiramos a una subfunción de encaminamiento R_1 solo será necesario considerar las colas que puede facilitar a la hora de establecer las dependencias entre canales. Por lo tanto, tanto el grafo de dependencias estático como el dinámico entre recursos asociado a dicha subfunción estarán restringidos en sus vértices al subconjunto de colas que puede retornar R_1.

3.3.3 Condiciones Necesarias Para una Función de Encaminamiento Libre de Interbloqueos.

Teorema 3- 1 *Dada una red de interconexión I=G(N,Q), el par (R,F), donde R es una función de encaminamiento conexa y F es una función de control de flujo, dicha red esta libre de interbloqueos si para cualquier configuración legal no existen ciclos en el grafo D_d de dependencias dinámicas entre canales.*

Demostración:

Supongamos que la red *I* ha alcanzado una configuración de *deadlock*. Entonces, de acuerdo con la Definición 3-9 de la página 93, tenemos que:

$$\forall q_i \in Q_{IN}, \text{ tal que } size(q_i) > 0 \begin{cases} head(q_i) \neq d_i \\ F(q_i, q_j, size(q_i)) \equiv false \quad \forall q_i \in R(q_i, head(q_i)) \end{cases} \quad \textbf{[3-14].}$$

Pero al garantizar la no existencia de ciclos de dependencia dinámica entre canales, sabemos que no existe ningún conjunto de canales $\{q_{x0}, q_{x1}, \ldots, q_{xn}\} \in Q_{IN}$ tal que:

$$q_{x(i+1) mod\ n} \in R(q_{xi}, n_d) \quad \text{y} \quad F(q_{xi}, q_{x(i+1) mod\ n}, size(q_{x(i+1) mod\ n})) = False \forall i = 1, \ldots, n \qquad \textbf{[3-15].}$$

Es decir, bajo la hipótesis de partida, sabemos que en cualquier instante de tiempo *t* existe al menos un par de colas $q_{xk}, q_{x(k+1) mod\ n} \in \{q_{x0}, q_{x1}, \ldots, q_{xn}\}$ tales que:

$$q_{x(k+1) mod\ n} \in R(q_{xk}, n_d) \quad \text{y} \quad F(q_{xk}, q_{x(k+1) mod\ n}, size(q_{x(k+1) mod\ n})) = true \qquad \textbf{[3-16].}$$

Bajo estas condiciones al mensaje almacenado en la cabeza de q_k le estará permitido el avance hacia el próximo canal, luego al contrario de lo que se supuso originalmente la red no se encuentra en un configuración interbloqueada ya que existe al menos un paquete al que le está permitido el avance.

Teorema 3- 2 *Dada una red de interconexión I=G(N,Q), el par (R,F), donde R es una función de encaminamiento conexa y F es una función de control de flujo, entonces la red será libre de interbloqueos si existe un subconjunto de colas $Q_{ND_1} \subseteq Q_{ND}$ sobre las que está definida una subfunción de encaminamiento $R_1(q_i, n_d) = R(q_i, n_d) \cap Q_{ND_1}$, $\forall q_i \in Q_{IN}, \forall n_d \in N$ y una subfunción de control de flujo $F_1(q_i, q_j, size(q_j)) = F(q_i, q_j, size(q_j))$, $\forall q_i \in Q_{IN}, \forall q_j \in Q_{ND_1}$, tal que R_1 es conexa y para cualquier configuración legal el par (R_1,F_1) no posee ciclos en el grafo de dependencias dinámicas entre recursos D_d.*

Demostración:

(a).- Sean Q_{ND} y Q_{ND_1} tales que $Q_{ND_1} = Q_{ND}$, lo que equivale a decir que $\forall q_i \in Q_{IN}, \forall n_d \in N$ tenemos que $R_1(q_i, n_d) = R(q_i, n_d)$ y $F_1(q_i, q_j, size(q_j)) = F(q_i, q_j, size(q_j))$. Entonces si (R_1, F_1) no posee ciclos en el grafo dinámico de dependencia entre canales, de acuerdo con el Teorema 3- 1, *I* es libre de interbloqueos.

(b) Si $Q_{ND_1} \subset Q_{ND}$, entonces $R_1(q_i, n_d) \subset R(q_i, n_d)$ $\forall q_i \in Q_{IN}, \forall n_d \in N$.

Sea D_d el grafo dinámico de dependencias entre recursos, definido por $D_d=G(Q,E_d)$ en un instante de tiempo *t*, donde E_d esta formado por pares $q_i, q_j \in Q_{ND}$ tal que:

$$q_j \in R(q_i, head(q_i)) \wedge F(q_i, q_j, size(q_i)) = false \quad \textbf{[3-17].}$$

De la misma manera se define el subgrafo de dependencias dinámicas entre recursos $D_{d1}=G(Q,E_{d1})$ donde E_{d1} está formado por aquellos pares $q_i, q_j \in Q_{ND_1}$ tales que:

$$q_j \in R_1(q_i, head(q_i)) \wedge F_1(q_i, q_j, size(q_i)) = false \quad \textbf{[3-18].}$$

Supongamos que la red de interconexión *I*, ha llegado a una situación de *deadlock* para la función de encaminamiento *R* y la función control de flujo *F*.

En el grafo dinámico de dependencias pueden en principio existir ciclos, pero en el subgrafo dinámico de dependencia de canales D_{d1} no existen ciclos, dado que éste viene descrito por la subfunción de encaminamiento R_1y la subfunción control de flujo F_1.

Dado que la red se encuentra en una configuración interbloqueada, tenemos que $\forall q_i \in Q_{IN}$ se ha de cumplir que:

$$\forall q_j \in R(q_i, n_d), \text{ siempre } F(q_i, q_j, size(q_i)) \equiv false \quad \text{y donde } q_i \notin n_d \quad \textbf{[3-19].}$$

Dado que no existen dependencias cíclicas en el subgrafo dinámico de dependencias de canales D_{d1}, podemos tomar tres canales $q_1, q_2, q_3 \in Q_{ND_1}$ de tal forma que:

$$\begin{cases} q_2 \in R_1(q_1, n_{d2}) \text{ y } F_1(q_1, q_2, size(q_2)) \equiv False \\ q_3 \in R_1(q_2, n_{d3}) \text{ y } F_1(q_2, q_3, size(q_3)) \equiv True \end{cases} \quad \textbf{[3-20].}$$

Dado que $Q_{ND_1} \subset Q_{ND}$ tenemos que

$$R_1(q_2, n_{d3}) \subset R(q_2, n_{d3}) \text{ y } F_1(q_2, q_3, size(q_3)) = F_1(q_2, q_3, size(q_3)) = True \quad \textbf{[3-21].}$$

luego existe al menos un par de canales $q_2, q_3 \in Q_{ND_1}$ entre los que no existe dependencia dinámica, como consecuencia de que la función control de flujo entre esos dos canales toma valor *True.* Por lo tanto, contrariamente a la condición de *deadlock* supuesta inicialmente, al menos existirá un paquete al que le está permitido el avance, luego la red no está en una configuración interbloqueada.

Si repetimos el mismo razonamiento en cualquier instante de tiempo $t^* \neq t$ llegaremos a una conclusión similar, es decir que la existencia de un subgrafo dinámico de dependencia de canales sin ciclos asegura la no alcanzabilidad de una configuración interbloqueada para la red de interconexión *I*.

3.3.4 Encaminamiento Adaptativo Para Redes *k-ary n-cube*.

De acuerdo con el Teorema 3- 2 podemos desarrollar un nuevo algoritmo de encaminamiento completamente adaptativo libre de interbloqueos aplicando esa teoría en el caso de redes *k-ary n-cube*. Podemos anticipar que el número de colas virtuales requeridas por canal físico va a ser inferior al requerido por mecanismos como los propuestos en [40] y [43]. Esta reducción en el número de colas o canales virtuales requeridos está relacionada con el hecho de que la subfunción de routing requerida para verificar las condiciones expuestas en el Teorema 3- 2 solo requiere un canal virtual.

Sea *I* una red *k-ary n-cube* con enlaces bidireccionales. Cada nodo de la red esta denotado por $n_{\vec{p}}$ donde $\vec{p} = (p_0, p_1, \ldots, p_{n-1})$ y donde $p_i \in \{0, \ldots, k-1\}$. Cada uno de los elementos p_d del vector representa la posición del nodo en la dimensión *d*. Por otro lado, dados dos nodos n_{p_1} y n_{p_2}, donde $p_i = p_j \forall i \neq d$, existe una única cola que une ambos nodos desde n_{p_1} hasta n_{p_2} asociada a la dimensión *d* y que viene denotada por $q_{d, p1, p2}$. De esta forma, la notación determina unívocamente la dirección del canal asociado a la cola, la dimensión en que se encuentra y los nodos que conecta.

3.3.4.1 Subfunción de encaminamiento libre de interbloqueos: control de flujo *Burbuja.*

De acuerdo con el Teorema 3- 2, la subfunción de encaminamiento asociada la elegiremos de tal forma que junto a la subfunción de control de flujo no dé lugar a ciclos en el grafo de dependencias dinámicas entre recursos[24]. Una subfunción que permite cumplir estos requisitos para el caso de las redes *k-ary n-cube* es el encaminamiento en orden de dimensión (DOR). Formalmente $R_{DOR}(q_{ixy}, n_d) = q_{lyz}$ donde:

$$\begin{cases} l = i \text{ si } n_d \text{ no se encuentra en la dimensión } i \\ l = (1+i) \text{ si } n_d \text{ se encuentra en la dimensión } l \end{cases} \quad \textbf{[3-22].}$$

para este tipo de redes de interconexión y teniendo en cuenta el carácter bidireccional de los enlaces, el número de dependencias estáticas entre canales que puede tener lugar es *2(n x k)*.

De acuerdo con el Teorema 3- 1, este tipo de redes será libre de interbloqueo si no existe ninguna dependencia cíclica en el grafo de dependencias dinámicas entre recursos. Por lo tanto, la red puede ser libre de interbloqueos pese a existir dependencias estáticas. Con el objetivo de cumplir esta condición se ha propuesto una función de control de flujo F_{CT^*}. Esta función elimina la posibilidad dependencias dinámicas que sean cíclicas restringiendo por un lado el proceso de inyección de los paquetes en la red y por otro los cambios de dimensión. Si denotamos *L* como la longitud del paquete, entonces esta función de control de flujo se define como:

$F_{CT^*}(q_{ixy}, q_{lyz}, size(q_{lyz})) = true$ si y solo si **[3-23].**

$$\begin{cases} size(q_{lyz}) \leq cap(q_{lyz}) - L \text{ si } \; i = l \\ size(q_{lyz}) \leq cap(q_{lyz}) - 2L \text{ si } \; i \neq l \text{ ó } q_{ixy} \in Q_I \end{cases}$$

Esta definición para la función control de flujo, hace necesario tener acceso a información adicional del nodo destino cuando el espacio de almacenamiento está en el puerto de entrada. Esto implica un incremento en el número de líneas de control por canal. Es posible eliminar este problema accediendo a la información local del nodo sin que aparezcan ciclos en el grafo de dependencias dinámicas entre recursos. Sea q_{ixy} el canal en el que se encuentra almacenado el paquete en el nodo *y*, y sea q_{lwz} el canal equivalente a q_{lyz} pero localizado en el nodo local. De acuerdo con esta notación se define la nueva función control de flujo F_{CT^*local} como:

$F_{CT^*local}(q_{ixy}, q_{lyz}, size(q_{lyz})) = true$ si y solo si **[3-24].**

$$\begin{cases} size(q_{lyz}) \leq cap(q_{lyz}) - L \text{ si } \; i = l \\ size(q_{lwz}) \leq cap(q_{lwz}) - 2L \text{ y } size(q_{lyz}) \leq cap(q_{lyz}) - L \text{ si } \; i = l \;\; \text{si} \;\; i \neq l \text{ ó } q_{ixy} \in Q_I \end{cases}$$

Tanto si utilizamos F_{CT^*} como F_{CT^*local} el grafo de dependencias dinámicas entre recursos estará exento de ciclos al emplear como función de encaminamiento R_{DOR}. Por lo tanto, teniendo en cuenta el Teorema 3- 1 la red será libre de interbloqueos. De hecho si suponemos que bajo estas condiciones se está produciendo un interbloqueo a lo largo de la dimensión *d*, necesariamente algún paquete se ha incorporado a la dimensión sin verificar la función F_{CT^*} o F_{CT^*local}, dado que bajo la definición de estas dos funciones no es posible llegar a un situación de interbloqueo en la que $\forall q_i \in \{q_{d,\,x0,\,x1}, q_{d,\,x1,\,x2}, \ldots, q_{d,\,xk-1,\,x0}\}$ siempre se ha de verificar, de acuerdo con la definición de interbloqueo y el modo en que están construidos estas dos funciones de control de flujo, $cap(q_i)=size(q_i)$.

A este control de flujo lo hemos denominado *Burbuja* y opera como sigue. Consideraremos un anillo con enlaces unidireccionales para una de las direcciones en una dimensión *d* de un *k-ary n-cube* y con espacios de almacenamiento en la entrada. En este caso la función de control de flujo F_{CT^*local} esta representada en la Figura 3-5 y F_{CT^*} en la Figura 3-6. Los paquetes (zonas sombreadas de cada buffer) pueden avanzar (flechas sombreadas) desde la cola en la que se encuentran a la siguiente cola del anillo si se verifica la condición de la *Burbuja*, es decir, después de inyectar el paquete se ha de garantizar que queda aún un espacio libre en la cola de avance y por tanto en el anillo. En este caso, para la primera función de control de flujo será suficiente con permitir la inyección si existen al menos dos huecos en el siguiente buffer. En el

segundo caso, al no tener acceso a la información remota, será suficiente con permitir la inyección si en la siguiente cola existe hueco para un paquete (recordar que esta es la condición del control de flujo VCT) y que en la cola del encaminador local que tiene la misma dimensión y dirección que la de la cola destino del paquete, al menos existen dos huecos. El número de huecos mínimo que se requiere en este caso es dos dado que con tan solo uno, al tener el espacio de almacenamiento de los encaminadores localizado en la entrada y no poder bloquearlo al inyectar un nuevo paquete, cualquier paquete en tránsito desde el encaminador previo puede hacer que el anillo se llene completamente provocando de esta manera que la probabilidad de interbloqueo no sea nula. En el hipotético caso de que todas las colas tengan espacio para un solo paquete es claro que si relajamos la restricción a un solo paquete entonces una inyección masiva en todos los nodos del anillo puede dar lugar a un bloqueo generalizado.

De la misma forma, cuando un paquete pretende avanzar hacia la próxima dimensión, solamente se le estará permitido el cambio de dimensión si al menos existen dos huecos en la dimensión requerida (ya sea en el buffer local o en el remoto). Por lo tanto, un cambio de dimensión puede ser considerado como una inyección desde el punto de vista del próximo anillo.

Intuitivamente, podemos mostrar el modo de operación del modo siguiente. Consideremos un conjunto de paquetes moviéndose a lo largo de un anillo unidireccional. Siempre que exista un hueco libre en las colas asociadas al anillo (una *burbuja*) nunca se llegará a producir un ciclo en el grafo de dependencias dinámicas entre recursos. Si no se incorporan paquetes nuevos, entonces al menos un paquete podrá avanzar y en pasos sucesivos todos los paquetes podrán alcanzar su destino. De esta manera, la clave para evitar la aparición del interbloqueo es garantizar que al menos exista un hueco libre en cada uno de los anillos de la red y en cada una de sus direcciones para el caso de enlaces bidireccionales. Esto puede ser logrado, de forma sencilla, empleando información local y restringiendo la entrada de nuevos paquetes si esa inyección puede provocar la desaparición de la burbuja.

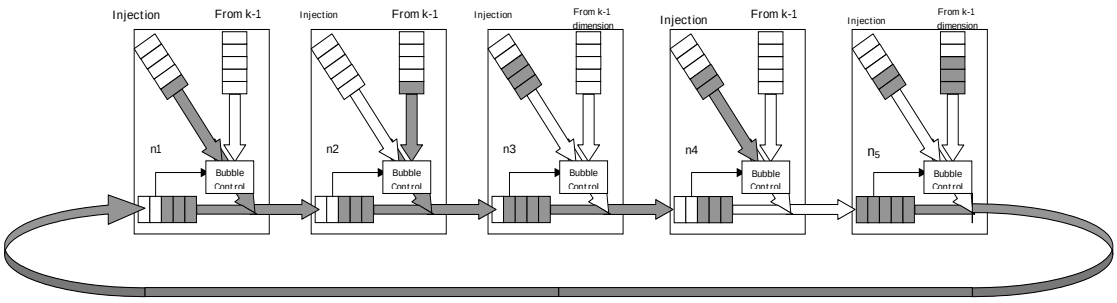


Figura 3-5. Control de flujo *burbuja* con información local.

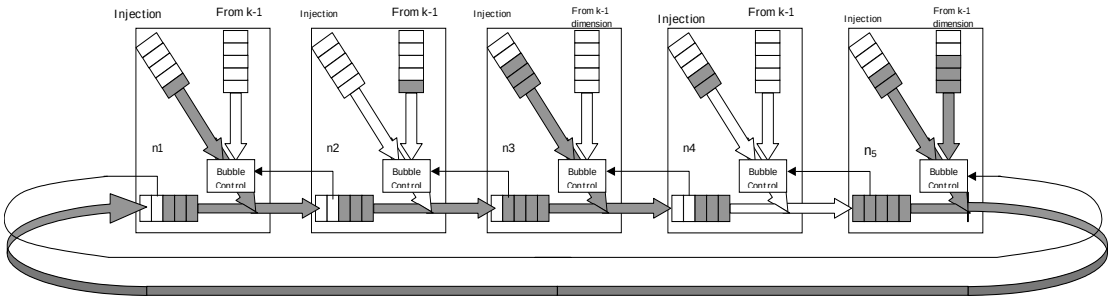


Figura 3-6. Control de flujo *burbuja* con información remota.

3.3.4.2 Función de Encaminamiento Adaptativa.

Teniendo en cuenta el Teorema 3- 2 podemos construir una subfunción de encaminamiento R_1 con un control de flujo adecuado, de tal forma que podamos incorporar un encaminamiento completamente adaptativo en *R* sin llevar a la red a una situación de interbloqueo. Tomando en consideración la notación previa, la función de control de flujo es:

$F(q_{ixy}, q_{lyz}, size(q_{lyz})) = true$ si y solo si **[3-25].**
$size(q_{lyz}) \leq cap(q_{lyz}) - L$ si $(q_{ixy}, q_{lyz} \in Q_{ND_1} \wedge i = l) \vee q_{lyz} \notin Q_{ND_1}$
$size(q_{lyz}) \leq cap(q_{lyz}) - 2L$ si $(q_{ixy}, q_{lyz} \in Q_{ND_1} \wedge i \neq l) \vee (q_{lyz} \in Q_{ND_1} \wedge q_{ixy} \notin Q_{ND_1})$

o de forma alternativa también:

$F_{local}(q_{ixy}, q_{lyz}, size(q_{lyz})) = true$ si y solo si **[3-26].**
$size(q_{lyz}) \leq cap(q_{lyz}) - L$ si $(q_{ixy}, q_{lyz} \in Q_{ND_1} \wedge i = l) \vee q_{lyz} \notin Q_{ND_1}$
$size(q_{lwz}) \leq cap(q_{lwz}) - 2L$ y $size(q_{lyz}) \leq cap(q_{lyz}) - L$ si
$(q_{ixy}, q_{lyz} \in Q_{ND_1} \wedge i \neq l) \vee (q_{lyz} \in Q_{ND_1} \wedge q_{ixy} \notin Q_{ND_1})$

En función del carácter de la cola de destino, la función de control de flujo puede dividirse en dos subfunciones:

Si $q_{lyz} \in Q_{ND_1}$ entonces:

$F_1(q_{ixy}, q_{lyz}, size(q_{lyz})) = true$ si y solo si **[3-27].**

$$\begin{cases} size(q_{lyz}) \leq cap(q_{lyz}) - L \text{ si } i = l \\ size(q_{lyz}) \leq cap(q_{lyz}) - 2L \text{ si } i \neq l \text{ ó } q_{ixy} \notin Q_{ND_1} \end{cases}$$

o:

$F_{1local}(q_{ixy}, q_{lyz}, size(q_{lyz})) = true$ si y solo si **[3-28].**

$$\begin{cases} size(q_{lyz}) \le cap(q_{lyz}) - L \text{ si } \; i = l \\ size(q_{lwz}) \le cap(q_{lwz}) - 2L \text{ y } size(q_{lyz}) \le cap(q_{lyz}) - L \quad \text{si} \quad i \ne l \text{ ó } q_{ixy} \notin Q_{ND_1} \end{cases}$$

y si $q_{lyz} \notin Q_{ND_1}$ entonces:

$$F_a(q_{ixy}, q_{lyz}, size(q_{lyz})) = true \text{ si } \; size(q_{lyz}) \le cap(q_{lyz}) - L \qquad \textbf{[3-29].}$$

Se puede anticipar que en el diseño del encaminador analizado en secciones posteriores, se han empleado colas en la entrada del encaminador. Bajo estas condiciones, hemos utilizado como subfunción control de flujo F_{1local} con el fin de mantener el *pin-count* del encaminador y adelantar el proceso de selección del próximo canal.

La ecuaciones indican, en pocas palabras, que al avanzar los paquetes por los canales adaptativos el control de flujo se reduce al VCT típico. En el caso de avanzar sobre un canal determinista el control de flujo aplicado es el de la *Burbuja*.

La función de encaminamiento asociada a las colas $q_i \notin Q_{ND_1}$, proporciona un conjunto de colas alternativas que dependen de la posición relativa del nodo actual y del nodo destino pero siempre siguiendo un camino que minimice la distancia a recorrer por el paquete. Por último, la función de selección será la encargada de elegir un único canal de los facilitados. En la implementación final se detallará cuál es la empleada en la propuesta definitiva.

En definitiva, existen dos grupos de colas, donde el algoritmo de la *Burbuja* es aplicado a un subconjunto de ellas (colas de escape) y al restante no se aplica ninguna restricción adicional (colas adaptativas). Siempre se prioriza la selección de una cola adaptativa, independientemente de si la cola actual es determinista o adaptativa. De acuerdo con esto, la restricción de la *Burbuja* es aplicada exclusivamente en el caso de que la siguiente cola sea una de escape. Es decir, los saltos desde colas deterministas hacia colas adaptativas se hacen sin ninguna restricción pero en sentido contrario se aplica la misma restricción que en la inyección. Por lo tanto, desde el punto de vista de las colas de escape, el último tipo de saltos es considerado como una inyección y desde el punto de vista de las colas adaptativas se ve de la misma forma que un consumo, por lo tanto, la red es libre de interbloqueos independientemente de las colas adaptativas.

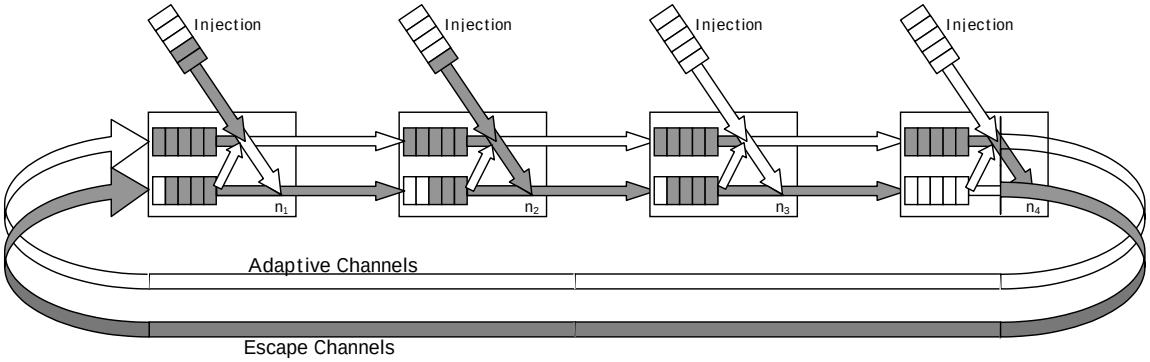


Figura 3-7. Estado del anillo unidireccional cuando todos las colas adaptativas están bloqueados.

Cuando todas las colas adaptativas estas ocupadas y por tanto los paquetes no pueden avanzar a través de ellas, el anillo unidireccional queda representado como aparece en la Figura 3-7, o de un modo equivalente en la Figura 3-8

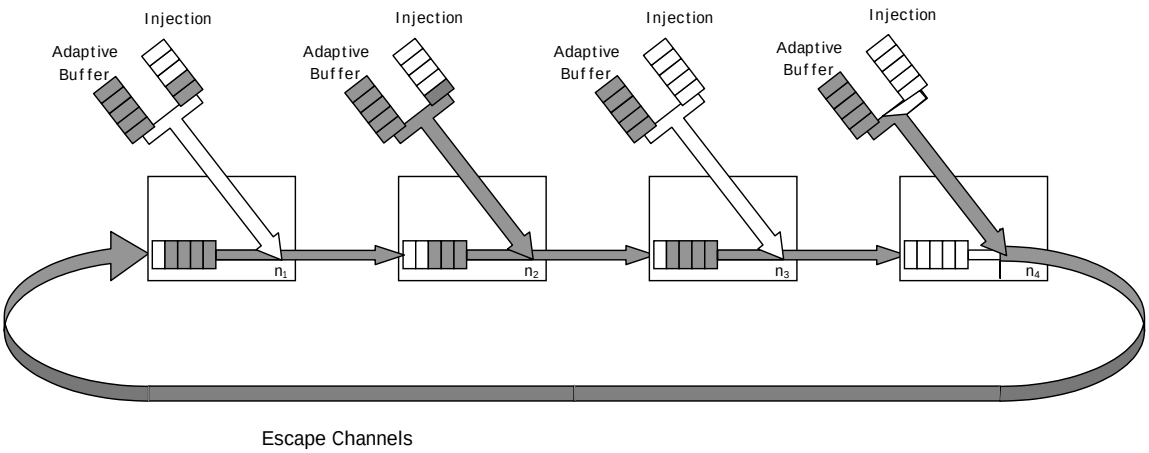


Figura 3-8. Representación equivalente para la situación mostrada en la Figura 3-7.

En este caso es claro que las colas deterministas pueden seguir siendo utilizadas como vías de escape para los paquetes almacenados en las colas adaptativas, permitiendo evitar cualquier situación de interbloqueo empleado únicamente dos canales virtuales por enlace físico.

3.3.4.3 Inanición

El control de inyección y cambios de dimensión aplicado en la *Burbuja* es una técnica eficiente a la hora de evitar los interbloqueos. Sin embargo, bajo determinadas condiciones, esta técnica puede provocar otra anomalía: inanición. Este problema existe debido a que la prioridad de acceso a algunos recursos es diferente. Por ejemplo, los paquetes que se mueven a lo largo de

la misma dimensión tienen mayor prioridad para avanzar en un momento dado que los que intentan entrar en ella (ver Figura 3-9).

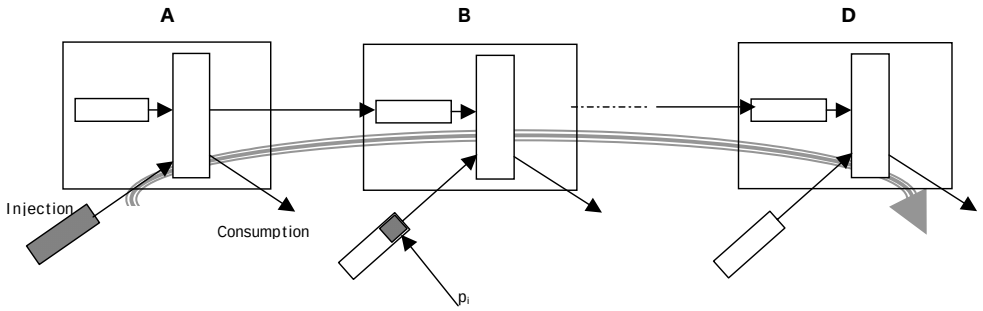


Figura 3-9. Ejemplo de producción de inanición en el caso de un solo canal Determinista con control de flujo basado en la Burbuja.

Los paquetes que van desde el encaminador *A* al *D* tienen mayor prioridad que los paquetes que pretenden ser inyectados en el encaminador *B*. Por lo tanto, si el tráfico entre *A* a *D* no se ve interrumpido, entonces los paquetes almacenados en la cola de inyección de *B* pueden estar esperando a entrar en la red durante un tiempo indefinido. Existen diversas soluciones que permiten eliminar esta anomalía incorporando controles adicionales como se muestra en [25].

Sin embargo, en el caso completamente adaptativo este problema no existe dado que el acceso a las colas adaptativas tiene prioridad uniforme independientemente de la situación del paquete. Por lo tanto, aunque las colas adaptativas estén completamente llenas, en un tiempo finito una de las colas adaptativas tendrá espacio para avanzar el nuevo paquete, dado que es libre de *deadlock*. De acuerdo con esto, el problema de la inanición queda solucionado en este caso. Para profundizar más en este razonamiento nos podríamos plantear que ninguno de los paquetes que está en el anillo puede avanzar hacia las colas deterministas debido a un problema de inanición generalizado en todo el anillo adaptativo lo que a la larga provocaría un problema de *deadlock* en todos los canales adaptativos. Sin embargo, esto no es posible que ocurra ya que en el nodo o nodos en que se está produciendo la inyección masiva que provoca la inanición, el canal adaptativo tiene la misma prioridad que las colas de inyección para incorporar un paquete en el canal de escape. Por lo tanto, está garantizado que en un tiempo finito, sí la política de arbitraje es justa, al menos un paquete almacenado en el canal adaptativo podrá avanzar hacia el canal de escape y por lo tanto se elimina la posibilidad de interbloqueo, lo que de nuevo conduce a asegurar que la red es libre de inanición.

3.4 Implicaciones Tecnológicas.

El algoritmo propuesto reduce a dos el número mínimo de canales virtuales requeridos para garantizar que la red es libre de interbloqueo. Como se muestra en [30] el coste añadido, desde el punto de vista *hardware,* por los canales virtuales puede ser crucial a la hora de obtener un rendimiento adecuado de la red de interconexión. Por ello, es de esperar que los encaminadores basados en esta propuesta permitan mejorar el rendimiento de la red.

Con el objetivo de validar la propuesta hecha desde el punto de vista teórico, se ha planteado un análisis comparativo entre ella y otras alternativas desde un enfoque más práctico. Para cuantificar el impacto en el rendimiento de cada alternativa considerada, se han diseñado en *hardware* los diferentes encaminadores analizados. Mediante este estudio podremos conocer de forma precisa cuál es realmente la posición, en lo que respecta a latencia exhibida como productividad de la red, de nuestra propuesta con respecto a los encaminadores deterministas y encaminadores completamente adaptativos basados en mecanismos previos de evitación de *deadlock*. El estudio se ha realizado para toros bidimensionales, si bien los resultados son perfectamente extrapolables para redes con un número de dimensiones superior u otro tipo de redes toroidales.

A continuación pasaremos a describir pormenorizadamente el encaminador propuesto sobre la base teórica introducida previamente. Posteriormente indicaremos el coste para el diseño obtenido por la herramienta de síntesis. La implementación de este encaminador se ha denominado *Bubble Router*. Además de la base teórica en que se apoya, incorpora varios detalles en su implementación que han permitido reducir aún más su coste. Posteriormente se expondrán el resto de encaminadores considerados.

3.4.1 Implementación Hardware del *Bubble Router.*

Partiendo de las descripciones VHDL de los encaminadores, hemos llegado a la implementación lógica alcanzada por la herramienta de síntesis *Synopsys*. La librería tecnológica empleada en este análisis es ECPD07 de ES2/ATMEL (0.7 μm de canal con dos niveles de metalización). Una descripción detallada de la metodología de diseño que se ha seguido se puede ver en la Sección 2.2.4. Por otro lado, como ya se comentó en la Sección 2.2.2 los diferentes diseños evaluados son *self-timed,* permitiendo eliminar de este modo cualquier problema relacionado con el *skew* del reloj.

Los principales módulos del encaminador propuesto se muestran en la Figura 3-10 (para el caso de una red *k-ary 2-cube*). Como se puede observar, cada enlace físico posee dos canales virtuales asociados que, cómo se ha indicado, es el número mínimo de canales virtuales requeridos

para que la red sea libre de interbloqueo. Los superiores corresponden a los canales completamente adaptativos del encaminador y los inferiores a los canales deterministas o de escape. Las colas asociadas a ambos canales son de tipo FIFO dado que habitualmente son las que menos complejidad implican.

Por otro lado, cada uno de los canales físicos de entrada posee un módulo de sincronización dado que el modo de operar de los canales de comunicación es asíncrono. Este módulo es el encargado de sincronizar los datos y las señales de petición asociadas. En cuanto a los canales de salida no es necesario emplear un módulo adicional de sincronización en la recepción de la señal de reconocimiento que proviene del router vecino, ya que el protocolo de comunicación junto con el control de flujo permite evitarlo (Ver Sección 2.2.2).

La conexión entre las colas de entrada y los canales de salida se realiza por medio de un crossbar. Este incorpora un sistema de arbitrio basado en *Round-Robin*. De esta manera, cuando dos o más canales de entrada intentan acceder a un mismo puerto de salida, el sistema de arbitrio interno del crossbar resuelve empleando una política justa que evita cualquier problema de *starvation.* Una descripción más detallada acerca el modo de arbitrio de este crossbar puede encontrarse en la Sección 4.2.1, pero, en pocas palabras, podemos decir que sus características son muy parecidas a las que implicaría un encaminamiento determinista, aún siendo éste adaptativo.

Los dos canales virtuales empleados por línea física son multiplexados a nivel de paquete, evitando así la necesidad de incorporar un módulo de multiplexación y control del canal. El control de flujo facilita esta simplificación en el encaminador al permitir que podamos avanzar el paquete completo sin problemas de bloqueo del canal físico, dado que las señales de protocolo indican si existe o no espacio para un paquete más. Es sencillo incorporar esta función en el propio crossbar ya que la gestión en este tipo de multiplexación es bastante simple. Notar que con un control de flujo tipo *wormhole* esto no podría ser resuelto de este modo dado que si multiplexamos el canal paquete a paquete, corremos el riesgo de que el paquete completo no logre entrar en el siguiente buffer. Esto conduciría a un bloqueo de los dos canales virtuales asociados a la línea física, pudiendo llegar a producirse *deadlock* en la red.

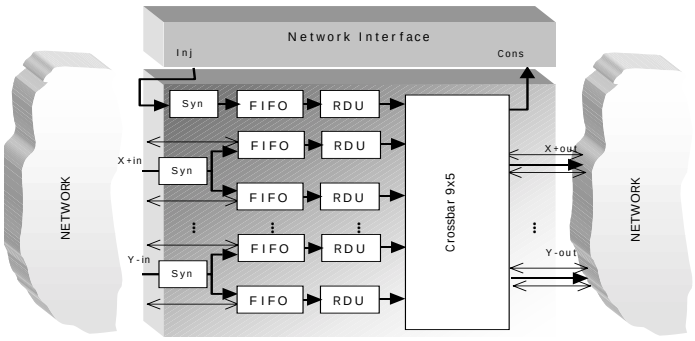


Figura 3-10. *Bubble Router*: estructura del encaminador adaptativo con mecanismo de evitación de interbloqueo basado en la burbuja (caso de un toro bidimensional).

Las líneas de comunicación entre encaminadores vecinos en esta implementación están constituidas por enlaces de 16 bits de datos más un bit para señalar el paso de la cola del paquete. Esta anchura de canal esta dentro de lo que es usual en muchos encaminadores empleados en varios sistemas comerciales, lo que indica que es una cifra perfectamente viable desde el punto de vista tecnológico. De hecho se ha elegido como termino medio entre el SGI SPIDER, con 20 bits de canal y el encaminador del Cray T3E, con 14 líneas de datos. La codificación del destino del paquete se realiza en base a encaminamiento relativo, conteniendo el primer *phit* del paquete ambos *offsets* del *Routing-Record*.

Por otro lado, la unidad de decisión de routing RDU (*Routing Decision Unit*) es el módulo encargado de encaminar los mensajes a través del router. El planteamiento seguido en la implementación del encaminador, lo convierte en uno de los componentes cruciales. Su cometido es interpretar el destino del paquete, establecer cuales son los puertos de salida que acerca el paquete a destino y solicitar su concesión al árbitro del encaminador. Pese a no ser el módulo de mayor coste del encaminador, en términos de tiempo o área, la complejidad y dificultad en su desarrollo ha sido, con diferencia, la más elevada.

Dado que los paquetes son encaminados de forma relativa, la primera acción a tomar cuando el primer *phit* del mensaje llega a este módulo, es comparar con cero para determinar si la dimensión actual se ha agotado. En función del resultado de la comparación y signo, la máquina de estados de la unidad de control obrará en consecuencia. (Ver Figura 3-12)

La máquina de estados realiza de forma secuencial las peticiones de canal de salida al crossbar, de acuerdo con la función de selección elegida. El crossbar gestiona todas las peticiones que le envían las diferentes RDU de los canales de entrada y el árbitro interno es el encargado de con-

ceder los canales de salida siguiendo una política de asignación tipo *Round-Robin*. Una vez concedido el canal, la unidad de control de la RDU avanza los datos en el orden correcto, decrementando en uno la parte del *phit* de cabeza correspondiente a la dimensión por la que va a avanzar el paquete. Con el fin de no incorporar el decrementador en el camino crítico la actualización del *phit* de cabeza se realiza en paralelo con el proceso de petición que lanza la RDU. Tanto el resultado del decrementador como el valor original son registrados en la unidad de datos de la RDU.

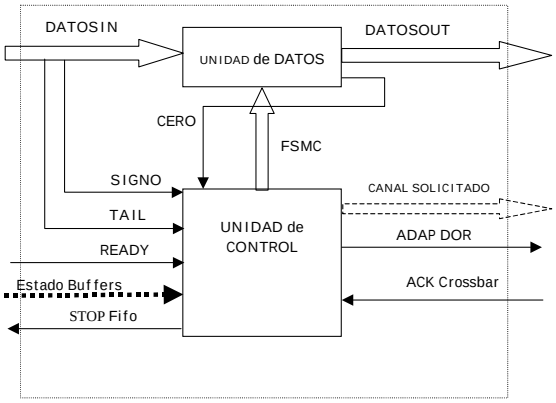


Figura 3-11. Estructura de la Unidad de Decisión de Encaminamiento (RDU).

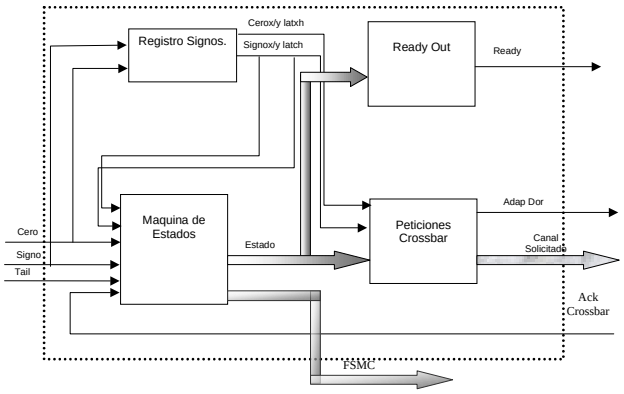


Figura 3-12. Unidad de Control de la RDU.

Básicamente el esquema de secuenciado en las peticiones logra reducir considerablemente la complejidad del arbitraje del router. Aunque, a priori, puede pensarse que este sistema puede acarrear una caída de productividad en el sistema, en el Capítulo 4 se verá que no es así.

El modo en el que se realizan las peticiones por parte de la RDU hacia el crossbar, suponiendo que el paquete se encuentra almacenado en una cola adaptativa o en la cola de inyección, es la siguiente.

1. En primer lugar se solicita el canal correspondiente a la cola adaptativa en el encaminador vecino a lo largo de la dimensión actual si el *offset* correspondiente a la dimensión actual no es nulo.
2. En segundo lugar se solicita el canal adaptativo del nodo vecino situado en la siguiente dimensión, si el *offset* de la siguiente dimensión es distinto de cero.
3. En tercer lugar se solicita la cola de escape en el encaminador vecino a lo largo de la dimensión entrante siempre y cuando se verifique la condición de la *Burbuja* en el buffer local. Obviamente, en este caso solo se produce una petición a un canal, que será el *x* determinista si el *offset* de la dimensión *x* es distinto de cero y el canal *y* si el *offset* de *x* es nulo. Esto es condición indispensable para que no se produzca un interbloqueo en los canales de escape ya que es la condición impuesta por el encaminamiento en orden de dimensión.

En el caso de que el paquete se encuentre situado en una canal de escape, la secuencia es idéntica a la expuesta previamente pero, obviamente, a la hora de solicitar el canal determinista en la misma dimensión no es necesario verificar la condición de la *Burbuja*. Por ello, hemos considerado adecuado emplear el mismo módulo *hardware* para todos los canales del encaminador, sean deterministas o adaptativos y en el caso de los primeros forzar siempre a *válida* la señal que indica la condición de la burbuja en el buffer asociado a los avances por misma dimensión.

Dada la complejidad de la máquina de estados (posee 22 estados), optamos por representarla en forma de red de Petri coloreada (Ver la Figura 3-13) [129].

La secuencia es cíclica dado que, si ninguno de los canales solicitados es concedido, el ciclo de petición comienza otra vez desde el primer estado. Por lo tanto, aunque extremadamente poco probable, es posible que un paquete intente indefinidamente acceder a los puertos de salida forzado por las condiciones de tráfico del resto de canales. Una política para evitar este problema es parar indefinidamente en el punto tres de la secuencia comentada previamente, después de un número de ciclos de petición preestablecido. El número de ciclos de petición se puede fijar a un número alto que permita evitar el problema de inanición asociado sin afectar significativamente al rendimiento.

En definitiva, la RDU consume, en una red vacía, un solo ciclo, tanto si el camino a seguir por el paquete es en la misma dimensión o si es preciso un cambio de dimensión. Por otro lado, la función de selección empleada es X/Y dinámica.

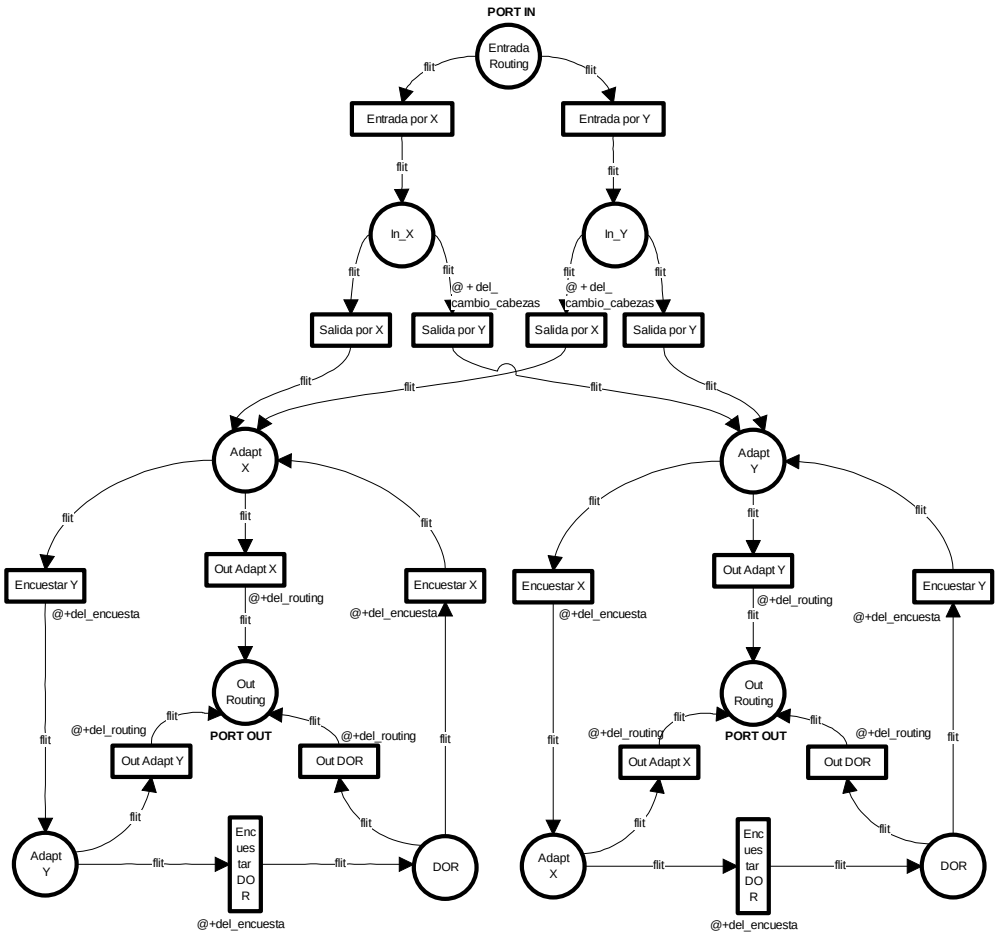


Figura 3-13. Modelo CPN de la máquina de estados de la RDU [129].

3.4.1.1 Resultados de la implementación.

Hemos procedido sobre todos los módulos descritos previamente, realizando una descripción VHDL y sintetizando, con ayuda de *Synopsys,* la implementación en puertas lógicas de cada uno. Aunque la precisión alcanzada por la herramienta no es completa, si es muy próxima a lo que ocurriría en una hipotética implementación real (Ver Sección 2.2.5).

En particular, el proceso de diseño seguido para las colas FIFO del encaminador es diferente ya que para alcanzar su implementación se ha empleado el generador de macroceldas del kit de diseño de la librería (*ES2 Synopsys Design Kit v5.2)*[49]. Esta herramienta permite generar a partir de los datos de profundidad, tamaño de los buses de acceso, etc... macroceldas en formato EDIF [48] que pueden ser integradas en cualquier diseño de forma sencilla. Los resultados obtenidos son muy buenos, tanto en área como en tiempo, ya que son celdas desarrolladas específi-

camente por la propia fundición. Los resultados alcanzados para diversas profundidades con 17 bits de datos en condiciones típicas de operación se muestran en la Tabla 3-1.

Capacidad	Critical Path (ns)	Area (mm^2)
40 phits	5.20	0.47
80 phits	5.23	0.73
160 phits	5.25	1.01

Tabla 3-1. Características de tiempo y área de las colas FIFO en condiciones típicas.

De entre estas tres colas hemos optado por una con capacidad de hasta cuatro paquetes dado que más allá de este valor la ganancia en *throughput* es prácticamente despreciable. Es decir, emplearemos colas con capacidad para 80 *phits* (considerando una longitud de paquete de 20 *phits*). En todo el análisis comparativo que efectuaremos a continuación se empleará siempre este tamaño para los paquetes.

En cuanto a la implementación del resto de los módulos del encaminador, en la Tabla 3-2 podemos apreciar las características de tiempo y área en condiciones típicas de operación para la implementación alcanzada por la herramienta de síntesis.

Módulo	Critical Path (ns)	Área (mm^2)
Sincronizador	2.16	0.55 (x5)
FIFOS (80 phits) (I, D,A)	5.23	0.73 (x 9)
RDU	5.64	0.85 (x 9)
Crossbar & Árbitro	**5.65**	**8.98** (x 1)
Tiempo de ciclo/ Area Total	***5.65***	***25.95***

Tabla 3-2. Características de todos los módulos del encaminador bajo condiciones típicas.

A tenor de estos datos, es evidente que el módulo que impone el periodo de reloj del encaminador es el crossbar junto con el árbitro, quedando fijado éste a *5.65* nanosegundos. Estos resultados, junto con el modo de operación del encaminador, da lugar a la estructura de pipeline del encaminador que aparece en la Figura 3-14. El sincronizador consume entre 0.5 y 1.5 ciclos de reloj en función de las diferencias de fase entre los relojes de los dos encaminadores adyacentes. Tanto la FIFO como la unidad de decisión de routing y el crossbar y árbitro requieren un ciclo. Por lo tanto, el número de ciclos que son necesarios para que, en ausencia de contención, un paquete alcance el puerto de destino desde su entrada en el encaminador es de 4 ciclos en promedio. Tendiendo en cuenta estos datos el tiempo de paso en promedio sera aproximadamente de *22.6* nanosegundos.

En estas condiciones y teniendo en cuenta el tamaño de los enlaces, el ancho de banda agregado de los cinco canales bidireccionales del encaminador será de 3.54 Gbytes/segundo.

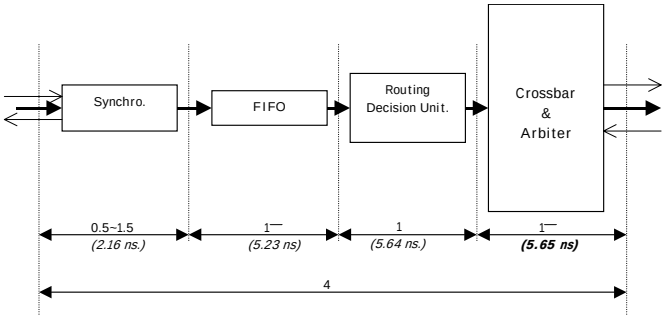


Figura 3-14. Estructura del pipeline del *Bubble Router.*

3.4.2 Implementaciones alternativas.

Una vez descrita la implementación del encaminador propuesto es necesario conocer su rendimiento relativo a otras arquitecturas alternativas. Con este propósito se han diseñado tres encaminadores que se basan en teorías previas. Dos de ellos son deterministas, lo que nos permitirá establecer cuál es el coste que introduce nuestra propuesta sobre encaminadores notablemente más sencillos, y uno más adaptativo.

Encaminador determinista con mecanismo de evitación de interbloqueo basado en la burbuja.

Este encaminador para redes *k-ary n-cube*, originalmente propuesto en [24], basa su mecanismo de evitación de interbloqueos en el Teorema 3- 1. El control de flujo empleado es por tanto una derivación de VCT como se muestra en la Ecuación 3-24 de la página 99. Los bloques básicos que componen el encaminador se muestran en la Figura 3-15. La principal diferencia con el *Bubble Router* es el crossbar y las unidades de encaminamiento. Al requerir un solo canal por línea física, el crossbar tan solo tienen 5 entradas. El esquema de arbitrio interno del crossbar es muy similar al empleado en el encaminador anterior. Sin embargo, las unidades de encaminamiento son substancialmente más sencillas. En este caso solamente es necesario establecer el único camino de salida en orden de dimensión para cada uno de los paquetes, lo que evita la complejidad de la unidad de routing del encaminador previo. Otra diferencia es que la profundidad de las colas de transito se han sobredimensionado con el fin de ofrecer una capacidad por canal físico similar al encaminador adaptativo. De esta forma se ha fijado su tamaño a 160 *phits*.

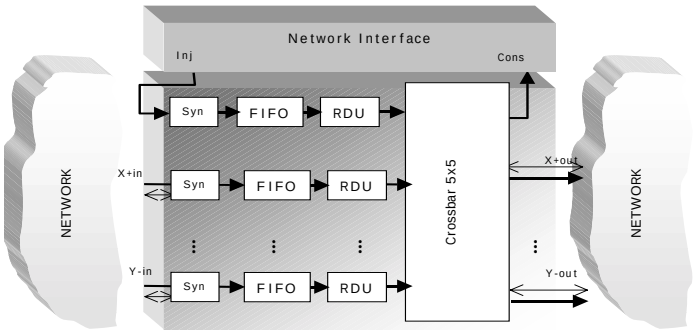


Figura 3-15. Encaminador determinista con mecanismo de evitación de interbloqueo basado en la burbuja.

Empleando la misma metodología de trabajo y sobre la misma librería tecnológica que en el caso anterior, los resultados alcanzados por la herramienta de síntesis se muestran en la Tabla 3-3. La estructura del *pipeline* del encaminador se muestra en la Figura 3-16. Como puede observarse, el tiempo de ciclo en este caso lo establecen las colas de almacenamiento en lugar del crossbar. El periodo del reloj queda definitivamente establecido en *5.25* nanosegundos. Dado que el número de etapas del pipeline es cuatro y que todas ellas consumen un solo ciclo de reloj en promedio, el tiempo de paso a través del encaminador es de *21* nanosegundos. Lo que indica que el encaminador, en ausencia de contención, es un 8% más rápido que el anterior.

Módulo	**Critical Path (ns)**	**Area (mm^2)**
Sincronizador	2.16	0.53 (x 5)
FIFOS (160 phits)	***5.25***	1.01 (x 5)
Unidad de Encam.	5.16	0.14(x 5)
Crossbar & Árbitro	5.25	2.62(x 1)
Tiempo de ciclo/ Area Total	***5.25***	***11.01***

Tabla 3-3. Características de todos los módulos del encaminador *burbuja-determinista* en condiciones típicas de operación.

Figura 3-16. Estructura del pipeline del encaminador *burbuja-determinista*.

Encaminador determinista con mecanismo de evitación de interbloqueo basado en canales virtuales.

De la misma forma que en el anterior, se ha implementado un encaminador DOR que emplea dos canales virtuales por línea física para evitar el interbloqueo en la red [40]. En este caso, el control de flujo empleado es *wormhole*. Se ha optado por emplear el esquema de utilización de los canales virtuales estático propuesto originalmente [40] en lugar de sistemas pseudo-adaptativos de utilización como el propuesto en [46]. La razón de diseñar de esta forma el encaminador es que esas técnicas avanzadas tienden a incrementar la complejidad del encaminador, lo que indirectamente afecta a la principal ventaja de los encaminadores deterministas: simplicidad y baja latencia de paso. En la Figura 3-17 aparece una representación básica de la estructura de este encaminador.

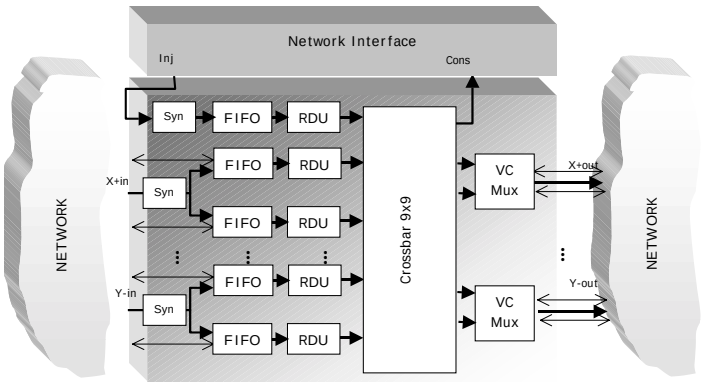


Figura 3-17. Encaminador determinista con mecanismo de evitación de interbloqueo basado en los canales virtuales.

La diferencia fundamental con el anterior es el número de canales virtuales requeridos y en consecuencia el tamaño del crossbar. En este caso, la funcionalidad es muy similar a los dos encaminadores anteriores en lo que respecta al crossbar pero tanto el número de entradas como salidas es nueve. Es importante señalar que en este caso no es posible realizar la multiplexación de los canales virtuales en el crossbar ya que, al emplear un control de flujo *wormhole*, la multiplexación ha de ser realizada *flit* a *flit*. Por esa razón, es necesario realizar la multiplexación en la etapa de salida, denominada *VC Mux*. Teniendo en cuenta estas consideraciones y empleando la misma metodología de trabajo que en los dos encaminadores anteriores, las características de tiempo y área de la implementación alcanzada por la herramienta de síntesis son las que se muestran en la Tabla 3-4. Por otro lado, en la Figura 3-18 se muestra la estructura del *pipeline* de este encaminador.

Módulo	**Critical Path (ns)**	**Area (mm^2)**
Sincronizador	2.16	0.53 (x 5)
FIFOS (80 phits)	5.23	0.73 (x 9)
Unidad de Encam.	5.02	0.17(x 9)
Crossbar & Arbitro	5.57	7.49(x 1)
VC Controller	5.06	0.21(x4)
Tiempo de ciclo/ Area Total	***5.57***	***19.08***

Tabla 3-4. Características de todos los módulos del encaminador *canales virtuales*-determinista bajo condiciones típicas.

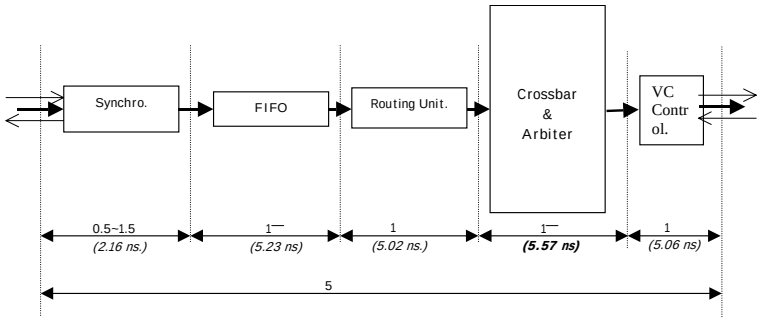


Figura 3-18. Estructura del pipeline del encaminador *canales virtuales*-determinista.

Estos datos indican que el tiempo de ciclo estimado es de *5.57* nanosegundos, siendo la etapa de arbitraje y crossbar la que marca el periodo de reloj del circuito. Aparentemente, comparando estos resultados con los que se muestran para el crossbar 9x5 en la Tabla 3-2, pese a tener un número de puertos de salida superior, el tiempo de ciclo es inferior. Sin embargo, como se analizará en el Capítulo 4, el tiempo de ciclo del crossbar depende exclusivamente del número de entradas que tiene que gestionar. La pequeña diferencia en el tiempo de ciclo es debida a que en el caso del *Bubble Router* el crossbar debe realizar la multiplexación de los canales que, pese a ser sencilla, introduce esa pequeña penalización de *0.08* nanosegundos. Teniendo en cuenta los datos relativos a la longitud del pipeline y periodo de reloj, el tiempo de paso por el router, en ausencia de contención, es de *27.85* nanosegundos, lo que representa un incremento de algo más de un 15% con respecto al *Bubble Router*.

Encaminador adaptativo con mecanismo de evitación de interbloqueo basado en canales virtuales.

Para completar este análisis, el tercer encaminador considerado es un encaminador completamente adaptativo con mecanismo de evitación de *deadlock* basado en la teoría desarrollada por *Duato* [46]. En este caso el control de flujo empleado es, como en el anterior, de tipo *wormhole*. La característica fundamental de este encaminador es que requiere tres canales virtuales por línea física para garantizar que la red sea libre de interbloqueo. Uno de los canales es completamente adaptativo y los otros dos son deterministas actuando, en conjunto, como vía de escape. En este caso sí se emplea un sistema de utilización pseudo-adaptativo para los dos canales de escape ya que el coste adicional que introduce es bajo.

Una representación de la estructura del encaminador se muestra en la Figura 3-19. Como se puede apreciar para un toro bidimensional el tamaño del crossbar requerido es de 13x13. De la misma forma que en el encaminador determinista basado en canales virtuales, la multiplexación ha de realizarse *flit* a *flit* para evitar bloqueos en el canal físico por desbordamiento de las colas de recepción, lo que hace necesario emplear un módulo de multiplexación externo al crossbar.

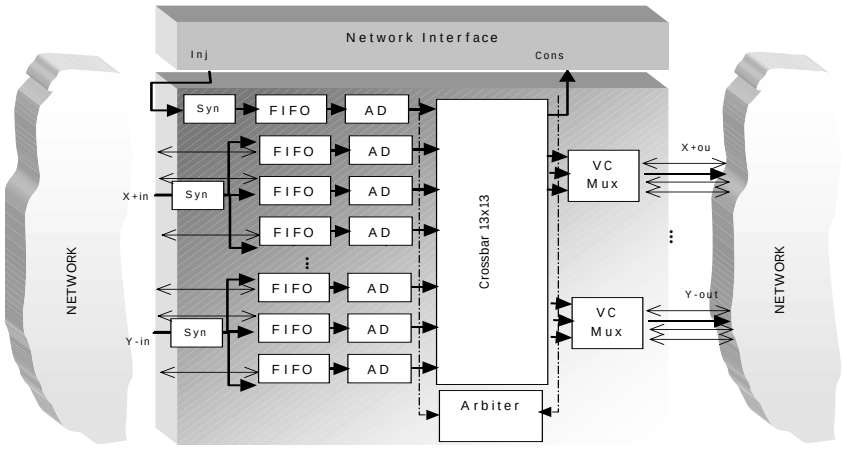


Figura 3-19. Encaminador adaptativo con mecanismo de evitación de interbloqueo basado en canales virtuales.

El tamaño de las diferentes colas empleadas se ha fijado de tal forma que la capacidad total por línea física sea la misma que en el encaminador original, es decir un total de 160 *flits*. En este caso se ha repartido en dos colas de 40 *flits* para los canales virtuales deterministas y 80 *flits* para la cola correspondiente al canal completamente adaptativo. Por otro lado, de forma similar al caso del encaminador de *Cray T3E*, el tamaño de los paquetes que puede manejar se ha restringido a un múltiplo de 20 *flits*, de esta manera la restricción [44] en el uso del canal adaptativo puede ser eliminada y por tanto esta permitida la presencia de varios paquetes en la misma cola adaptativa.

En cuanto a la estructura de arbitraje del encaminador, cabe reseñar el empleo de un esquema de arbitrio global en lugar de segmentado como el empleado en el *Bubble Router*. En la Sección 4.2.2 se puede encontrar una descripción detallada del mismo. Bajo estas condiciones, el módulo denominado *Address Decoder (AD)* es el encargado de determinar todos los canales de salida adecuados para el paquete. Los AD de cada canal lanzan una petición al árbitro global. Este, siguiendo una política de asignación *Round-Robin* y en función del estado de ocupación de cada canal, establece los caminos adecuados en el propio crossbar.

Bajo estas condiciones se ha procedido de la misma forma que con el resto de encaminadores y los resultados obtenidos para la implementación lograda por la herramienta de síntesis son las mostradas en la Tabla 3-5. Por otro lado, la estructura del *pipeline* del encaminador se muestra en la Figura 3-20.

Módulo	**Critical Path (ns)**	**Area (mm^2)**
Sincronizador	2.16	0.53 (x 5)
FIFOS (80 flits)	5.23	0.73 (x 5)
FIFOS (40 fliits)	5.20	0.47 (x 8)
Address Decoder.	7.45	1.05 (x 13)
Crossbar & Arbiter	7.32,**7.50**	11.08 (x 1)
VC Controller	6.69	2.45 (x4)
Tiempo de ciclo/ Area Total	***7.50***	***44.59***

Tabla 3-5. Características de todos los módulos del encaminador *canales virtuales*-adaptativo bajo condiciones típicas.

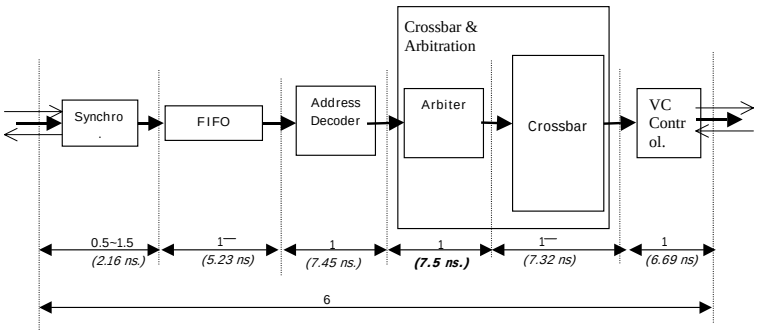


Figura 3-20. Estructura del pipeline del encaminador *canales virtuales*-adaptativo.

En esta caso el tiempo de ciclo viene marcado por la etapa de arbitraje y se sitúa en *7.50* nano-segundos. El incremento en el tiempo de ciclo viene motivado por el aumento en el número de

entradas a gestionar y por las características del tipo de arbitrio empleado. En este caso ha sido necesario segmentar el crossbar y árbitro en etapas diferentes con el fin de no incrementar aún más el tiempo de ciclo del router. Es cierto que, de haber utilizado el esquema de arbitrio empleado en el *Bubble Router*, las diferencias no serían tan acusadas, pero se ha implementado éste siguiendo esquemas de gestión empleados habitualmente en este tipo de routers como en encaminador del Cray T3E. En el Capítulo 4 se realizará un análisis mucho más detallado de este tipo de aspectos.

En la mayor parte de los encaminadores de alto rendimiento actuales (Cray T3E, Intel Cavallino entre otros) el coste de emplear un número alto de canales virtuales se intenta minimizar empleando crossbars multiplexados. En ese caso, la idea es retrasar la multiplexación de los canales virtuales a la entrada del crossbar. De esta forma, el número de entradas del crossbar se mantiene igual que el número de puertos físicos del encaminador independientemente del número de canales virtuales empleados. Esta estrategia permite reducir considerablemente el área con respecto a una implementación en la que se use un crossbar completo (como es nuestro caso). Sin embargo, esto no reduce el retraso del encaminador ya que se necesario emplear un multiplexor para seleccionar en la entrada del crossbar un único canal virtual. La suma del demultiplexor y el crossbar multiplexado viene a ser la misma, en lo que respecta a retraso, que cuando se emplea un crossbar completo. Sin embargo, en nuestro caso y dado el desbalanceo del pipeline sería posible incorporar la multiplexación de los canales virtuales en la etapa del *Address Decoder* sin degradar el tiempo de ciclo del encaminador. Frente a esto, es importante señalar que otro efecto colateral es que la multiplexación a nivel de *flit*, requerida por el control de flujo *wormhole* tradicional, hace que la complejidad del árbitro se incremente notablemente y el rendimiento de la red disminuya ya que el arbitraje a nivel de *flit* en el paso a través del crossbar reduce la utilización de los canales [113]. De cualquier modo, en el Capítulo 6, se aborda el análisis de este tipo de soluciones y los resultados alcanzados no son muy alentadores, fundamentalmente porque multiplexar los canales deterministas con los adaptativos puede afectar el rendimiento del encaminador, especialmente cuando se supera el punto de saturación de la red.

3.4.3 Sumario.

A tenor de los resultados de síntesis para cada una de las implementaciones, en la Tabla 3-6 se muestra un sumario de los datos de tiempo de paso y área para todos los encaminadores considerados.

Encaminador	Número de Ciclos	Tamaño de Crossbar	Tamaño de las Colas *(phits)*	Tiempo de ciclo *(ns)*	Área *(mm^2)*	Tiempo de Salto *(ns.)*
Burbuja DOR	4	5x5	160	5.25	11.01	21.50
VC. DOR	5	9x9	80+80	5.57	19.08	27.85
VC Adapt.	6	13x13	80+40+40	7.50	44.59	45.00
Burbuja Adapt.	*4*	*9x5*	*80+80*	*5.65*	*25.95*	*22.6*

Tabla 3-6. Características de los diseños para redes bidimensionales.

Los resultados muestran claramente las diferencias en tiempo y área de todas las implementaciones. En ese sentido, el encaminador más rápido y de menor área es el determinista basado en la burbuja y el más lento y costoso en términos de área es el adaptativo basado en los canales virtuales. Nuestra propuesta se sitúa en un muy buen lugar con respecto al resto, mejorando incluso el tiempo de paso del encaminador determinista basado en canales virtuales y sacando una ventaja más que notable con respecto al otro encaminador adaptativo. Estos datos permiten intuir que el rendimiento que alcanzará el *Bubble Router* será superior a los dos encaminadores *wormhole,* tanto en latencia como en *throughput*.

3.5 Análisis de Rendimiento.

Una vez determinado el coste, desde el punto de vista *hardware,* de nuestra propuesta y del resto de encaminadores alternativos, el siguiente paso es estudiar el rendimiento alcanzado por cada uno en el seno de una red. Como se detalla en el Capítulo 2, este análisis se realiza desde dos puntos de vista: en primer lugar evaluaremos el rendimiento de la red de interconexión con cada uno de los encaminadores empleando únicamente tráfico sintético y posteriormente pasaremos a realizar el análisis bajo condiciones de carga real, empleando para ello una arquitectura tipo ccNUMA.

3.5.1 Cargas Sintéticas.

Esta tarea puede ser realizada empleando simuladores *VHDL* o simuladores funcionales. Como se muestra en la Sección A.2.2, este análisis será realizado con el simulador SICOSYS. La precisión alcanzada por este simulador es muy similar a la obtenida por los simuladores VHDL con un coste computacional en tiempo y recursos requeridos notablemente inferior. En el Apéndice A se puede encontrar una descripción más detallada de este simulador.

Por otro lado, a la hora de fijar el tamaño de la red a considerar, hemos optado por emplear toros de 64 nodos (*8-ary 2-cubes*) bajo dos patrones de destino distinto: uniforme y algunas permutaciones específicas. En el primer caso se han empleado tres distribuciones en la longitud de los paquetes: en el primer caso solo se han considerado mensajes de 20 *phits* y en el segundo caso se ha considerado una mezcla de mensajes de 20 *phits* con mensajes de 200 *phits*. La tasa de generación de los segundos es de un 10% con respecto a los primeros. En el caso de los encaminadores VCT los mensajes largos son fragmentados en 10 paquetes de 20 *phits* de tal manera que no se sobrepase en ningún caso el tamaño de paquete contemplado. El tercer patrón de longitudes considerado es el que hemos denominado bimodal corto: básicamente consiste en un tráfico bimodal en el que la longitud de referencia la marcan los mensajes largos, en este caso de 20 *phits*. La longitud de los mensajes cortos es siempre submúltiplo de esta longitud, en nuestro caso 4 *phits*. La idea es emular el comportamiento de un sistema ccNUMA de manera más eficaz que con el tráfico bimodal convencional. Los mensajes largos representarían mensajes de respuesta en forma de líneas de cache y los cortos, tráfico de coherencia. En este caso se considera que la tasa de generación de los mensajes largos es del orden del 80% sobre los mensajes cortos. Estos datos están bastante de acuerdo con las características exhibidas por las aplicaciones, especialmente cuando estamos empleando redes aisladas de peticiones y respuestas (dado que en todo el trabajo estamos empleando redes aisladas de peticiones y respuestas, hemos considerado oportuno modelar con éste tráfico únicamente la red de respuestas). En el caso de los encaminadores VCT el control de avance en las colas y la aplicación de la *Burbuja* se realiza en términos de los mensajes largos, es decir 20 *phits*. Bajo estas condiciones se introduce una ligera pérdida de prestaciones originada por una ligera caída en la utilización máxima de las colas de transito, pero se evita la fragmentación de los paquetes largos. Este tráfico permitirá cuantificar cuál es la caída de rendimiento.

En cuanto a las permutaciones empleadas hemos considerado la matriz transpuesta, *bit-reversal* y *perfect-shuffle*. La tasa temporal de generación es uniforme para todos los tráficos (Ver la Sección 2.3.1).

Latencia Base

Como ya se ha comentado, este es un factor a tener en cuenta dado el impacto que supone en muchas aplicaciones paralelas [132]. Además, cualquier aplicación posee fases en su ejecución en las que la carga aplicada sobre la red es relativamente baja. La latencia base, o en ausencia de contención, depende de la complejidad interna del router y las características de distancia media del tráfico. Por lo tanto, bajo estas condiciones, los encaminadores adaptativos tenderán a exhibir peor rendimiento en esta zona de la región de trabajo de la red. Este inconveniente se

ha intentado minimizar en nuestra propuesta de dos maneras: limitando a solo dos los canales virtuales por enlace físico y empleando un esquema de arbitrio simplificado. Estos dos hechos se muestran de forma clara en la Tabla 3-7. Los datos mostrados corresponden a la latencia promedio de los paquetes para cada uno de los tráficos evaluados y considerando una carga aplicada sobre la red de tan solo 0.05% de la capacidad máxima de la bisección. Por lo tanto, la probabilidad de inyección de cada encaminador por ciclo es de tan solo $1.56 \cdot 10^{-5}$. Los resultados, como no podía ser de otro modo, trasladan las características temporales de los encaminadores mostrados en la Tabla 3-6 y se observa como nuestra propuesta supera la latencia alcanzada por el encaminador más simple en tan solo un 9%. Mientras tanto el encaminador adaptativo *wormhole* alcanza unos rendimientos en latencia base peores que nuestra propuesta llegando a ser las diferencias de hasta un 50%. Es interesante indicar de nuevo como el *Bubble Router* incluso llega a mejorar el rendimiento en latencia del encaminador determinista *wormhole* como consecuencia directa de la etapa adicional que requiere este último para realizar la multiplexación de los canales virtuales.

Encaminador	Uniforme	Bimodal	Bimodal Short	Matriz Trans.	Bit Reversal	Perfect Shuffle
Burbuja DOR	212.9	330.5	192.6	221.4	225.2	212.0
VC. DOR	248.7	373.9	226.4	260.2	264.8	247.3
VC Adapt.	374.4	541.5	344.0	391.9	392.9	376.8
Burbuja Adapt.	229.5	350.0	208.3	238.3	239.0	230.4

Tabla 3-7. Latencia base (ns.): Toro 8x8.

Throughput Máximo Alcanzado.

La latencia base es un parámetro a considerar a la hora de analizar las diversas alternativas. Sin embargo, muchas aplicaciones paralelas en determinadas zonas de su ejecución someten a una gran presión a la red de interconexión, demandando, en ocasiones, más capacidad que la que puede ofrecer la red de por sí. Es por ello importante determinar este parámetro ya que influirá notablemente en las aplicaciones. A la hora de conocer la productividad máxima alcanzada por la red de interconexión nos centraremos en establecer cuál es el nivel máximo de carga soportado, es decir, cuál es volumen más alto de información que es capaz de manejar.

Para analizar este parámetro, es importante incorporar las dependencias tecnológicas de cada encaminador. Es usual que esto no se haga así, recurriendo a representaciones normalizadas poco realistas. En nuestro caso analizaremos el rendimiento de cada encaminador, tanto desde

el punto de vista estructural, como tecnológico. En la Tabla 3-8 se muestran estos datos para cada encaminador y tipo de tráfico considerado.

	Uniforme		**Bimodal**		**Bimodal Short**		**Matriz Trans.**		**Bit Reversal**		**Perfect Shuffle**	
Encaminador	Phits/ ciclo	Phits/ns	Phits/ ciclo	Phits/ns	Phits/ ciclo	Phits/ns	Phits/ ciclo	Phits/ns	Phits/ ciclo	Phits/ns	Phits/ ciclo	Phits/ns
Burbuja DOR	38.7	7.38	29.9	5.69	38.6	7.35	13.0	2.47	12.0	2.30	18.7	3.56
VC. DOR	36.72	6.59	28.1	5.05	36.0	6.49	14.7	2.64	12.4	2.25	20.6	3.70
VC Adapt.	39.4	5.24	34.7	4.63	39.2	5.22	27.3	3.64	32.7	4.36	29.1	3.87
Burbuja Adapt.	43.6	7.71	36.8	6.51	41.8	7.40	30.6	5.40	34.1	6.03	28.7	5.08

Tabla 3-8. Throughput máximo alcanzado por cada propuesta para un toro 8x8.

Los resultados, en términos estructurales, muestran una clara supremacía por parte de los encaminadores adaptativos sobre los encaminadores deterministas, como era de esperar. Para esta situación se observa un rendimiento muy parecido entre el encaminador adaptativo *wormhole* y nuestra propuesta. Sin embargo, a la hora de tomar en consideración las dependencias tecnológicas, teniendo en cuenta cada tiempo de ciclo y expresando el throughput en *phits*/nanosegundo, vemos como los resultados cambian claramente. Nuestra propuesta mejora el rendimiento de todos los encaminadores (incluido el adaptativo basado en canales virtuales) llegando a ser las diferencias, con respecto a los encaminadores deterministas, superiores en más de un 50% en algunos patrones de tráfico no uniformes. De la misma forma, se observa como el encaminador basado en canales virtuales reduce notablemente su rendimiento al tener en cuenta su tiempo de ciclo, quedando éste en muchos casos con un *throughput* máximo sostenido por debajo del encaminador determinista basado en la *Burbuja*.

Latencia y Throughput con Carga Variable.

Para completar el análisis es interesante conocer cual es comportamiento de la red para cada encaminador considerado bajo un ritmo de carga variable. Esto nos permitirá profundizar más en los costes que introduce la adaptatividad en el rango medio de cargas. Las Figuras siguientes muestran la representación normalizada T-CNF para cada uno de los tráficos considerados. En estos gráficos, es preciso recordar, que la latencia promedio incorpora el tiempo de espera en la cola de inyección.

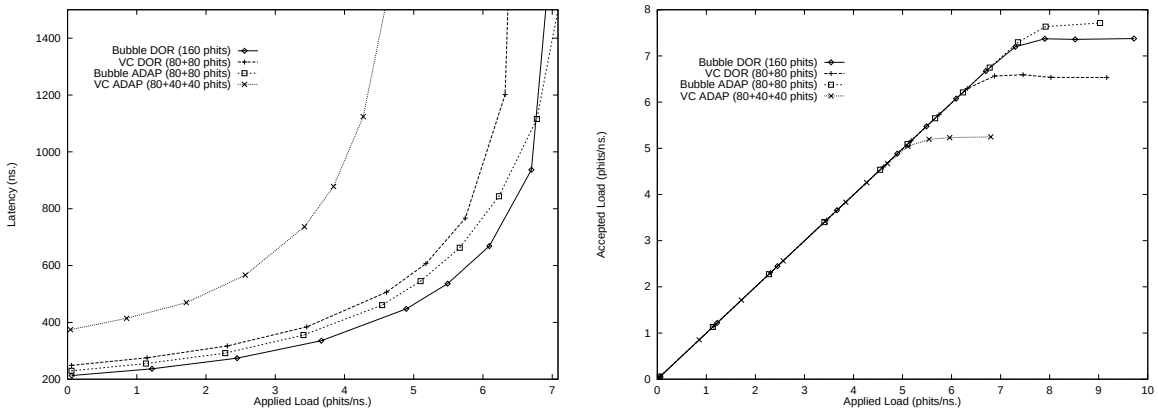


Figura 3-21. Latencia y Throughput para un Toro 8x8 bajo tráfico uniforme (L=20 phits).

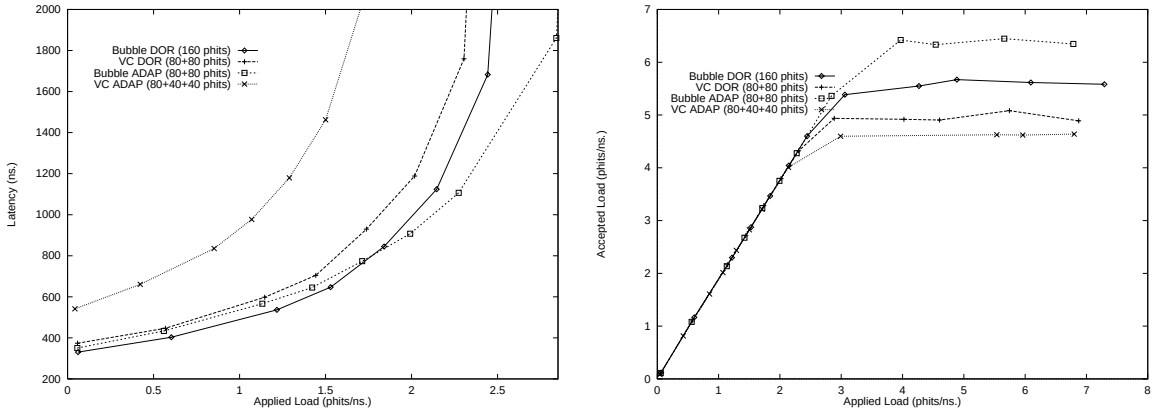


Figura 3-22. Latencia y Throughput para un Toro 8x8 bajo tráfico bimodal (L_{corto}=20 phits,L_{largo}=200 phits con probabilidad 0.1).

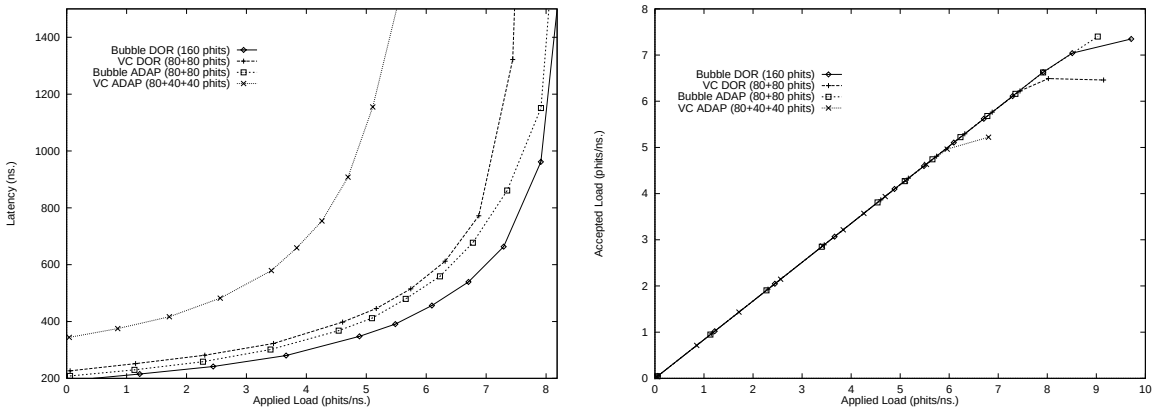


Figura 3-23. Latencia y Throughput para un Toro 8x8 bajo tráfico bimodal corto(L_{corto}=4 phits,L_{largo}=20 phits con probabilidad 0.8).

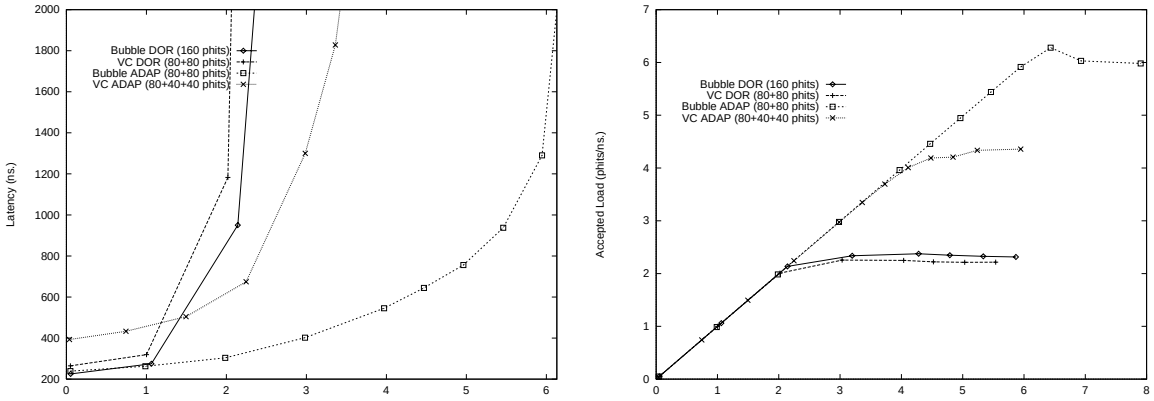


Figura 3-24. Latencia y Throughput para un Toro 8x8 bajo tráfico bit-reversal (L=20 phits).

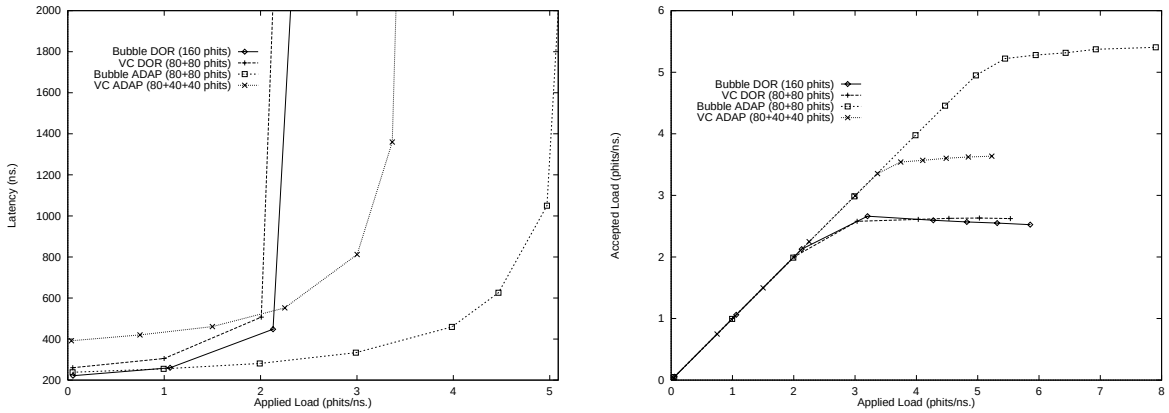


Figura 3-25. Latencia y Throughput para un Toro 8x8 bajo tráfico matriz transpuesta(L=20 phits).

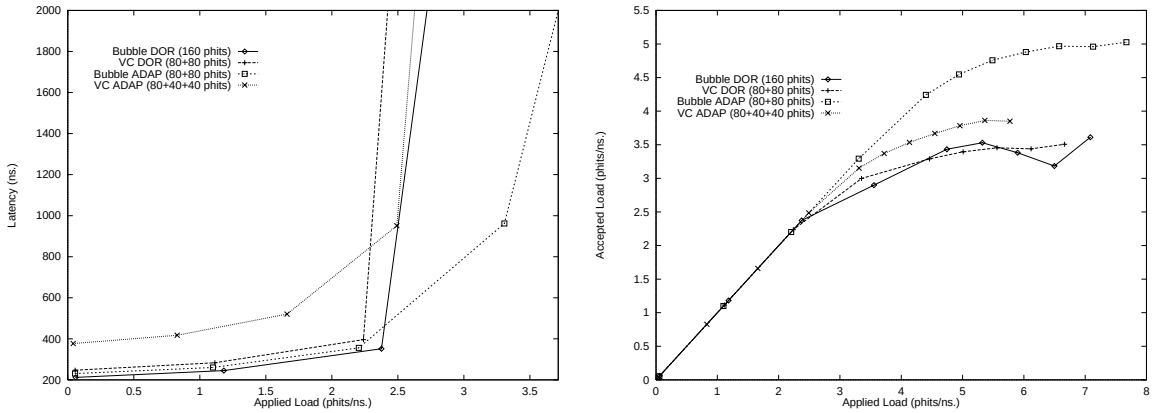


Figura 3-26. Latencia y Throughput para un Toro 8x8 bajo tráfico perfect-shuffle (L=20 phits).

Estos resultados muestran, claramente, el comportamiento superior de los dos encaminadores basados en la *Burbuja*. Las diferencias observadas en la latencia base se mantienen en rangos de carga considerables.

Obviamente, en el caso del tráfico uniforme, como es conocido, el encaminamiento adaptativo no saca excesiva ventaja sobre el encaminamiento determinista. Este hecho, junto a que el encaminador determinista basado en la burbuja tiene un tiempo de ciclo inferior, hace que su rendimiento sea ligeramente superior en *throughput* al del router adaptativo basado en la burbuja. De la misma forma, ambos routers mejoran considerablemente el rendimiento de los encaminadores *wormhole.*

En conclusión, nuestra propuesta mejora en términos generales el rendimiento de los demás encaminadores en lo que respecta al *throughput*, sin penalizar excesivamente la latencia en el nivel medio-bajo de cargas de trabajo. Por ello, es posible esperar que existan aplicaciones paralelas muy demandantes que mejoren con este encaminador, sin penalizar el rendimiento de otras aplicaciones, cuya demanda de tráfico hacia la red sea mucho más baja.

3.5.2 Cargas Reales.

En este punto se evalúa el comportamiento de cada uno de los encaminadores descritos previamente, bajo condiciones de tráfico real. Para abordar este análisis hemos empleado el simulador conducido por ejecución descrito en el Apéndice A. Dicho simulador utiliza como sistema de interconexión la misma estructura que SICOSYS, por lo que los resultados extraídos del mismo serán altamente fiables en lo que respecta al subsistema de comunicación.

El banco de pruebas empleado es el descrito en la Sección 2.3.3, que como ya se comentó, se trata de un subconjunto de la suite SPLASH 2 [137]. La configuración del sistema, en lo concerniente a características del procesador, jerarquía de memoria, etc..., son las indicadas en las Tablas 2-7 y 2-8 y cuyos parámetros más significativos son los mostrados en la Tabla 3-9.

La única diferencia, con respecto a la configuración empleada en el análisis de las aplicaciones, es que se ha optado por emplear caches con tamaño de bloque de 32 *bytes*. El resto de parámetros son los mismos. El tamaño de los comanados básicos empleado es de 8 *bytes*. De acuerdo con esto, la longitud de los paquetes de petición será de 4 *phits* y la de los de respuesta de 20 *phits*. Por otro lado, la elección de los tamaños de cache garantiza, según los desarrolladores de las aplicaciones, que los resultados de la ejecución serán significativos.

Procesador		Similar R10000 a 650 Mhz	
Parámetros Jerarquía de memoria.	L1	Tiempo de acceso	1 ciclo
		Asociatividad	1
		Tamaño de Bloque	32 bytes
		Tamaño	16 Kbytes
	L2	Tiempo de acceso (Tag/Dato)	(3 ciclos / 5 ciclos)
		Asociatividad	4
		Tamaño de Bloque	32 bytes
		Tamaño	64 Kbytes
	Protocolo de Coherencia		MESI
Otros Parámetros	Memoria	Acceso DRAM (sin bus ni caches)	18 ciclos
	Bus	Ancho	32 bytes
		Tiempo de ciclo	3 ciclos

Tabla 3-9. Resumen de los parámetros más significativos del sistema.

Por otro lado, dados los resultados de la síntesis de cada uno de los encaminadores, podemos establecer de forma precisa cuál es la velocidad relativa entre el procesador y la red del sistema. Tomando en cuenta los tiempos de ciclo que aparecen en la Tabla 3-6 y que los procesadores empleados operan a 650 MHz, las relaciones son las mostradas en la tabla siguiente.

Encaminador	**Frecuencia de reloj del Encaminador**	**Velocidad relativa procesador/red**
BDOR[a]	190 Mhz	3.41
BADA[b]	176 Mhz	3.67
CDOR[c]	179 Mhz	3.62
CADA[d]	133 MHz	4.87

a. Encaminador determinista con mecanismo de evitación de interbloqueo basado en la burbuja.

b. Encaminador adaptativo con mecanismo de evitación de interbloqueo basado en la burbuja.

c. Encaminador determinista con mecanismo de evitación de interbloqueo basado en los canales virtuales.

d. Encaminador adaptativo con mecanismo de evitación de interbloqueo basado en los canales virtuales.

Tabla 3-10. Velocidades relativas procesador/red en cada caso.

Bajo estas condiciones hemos evaluado el rendimiento ofrecido por cada encaminador y aplicación considerada, empleando en este análisis una configuración dual para las redes de peticiones y respuestas como método de evitar la aparición de un posible interbloqueo a causa de la capacidad limitada de las colas de consumo. El número de nodos considerado en todos los casos

es 64. De la misma forma que en la evaluación conducida por cargas sintéticas, en todos los casos emplearemos una red configurada como un toro 8*x*8.

FFT.

El tamaño de problema considerado para esta aplicación es el establecido por defecto, es decir, 64K dobles complejos. Los resultados de ejecución para cada uno de los encaminadores son los mostrados en la Figura 3-27.

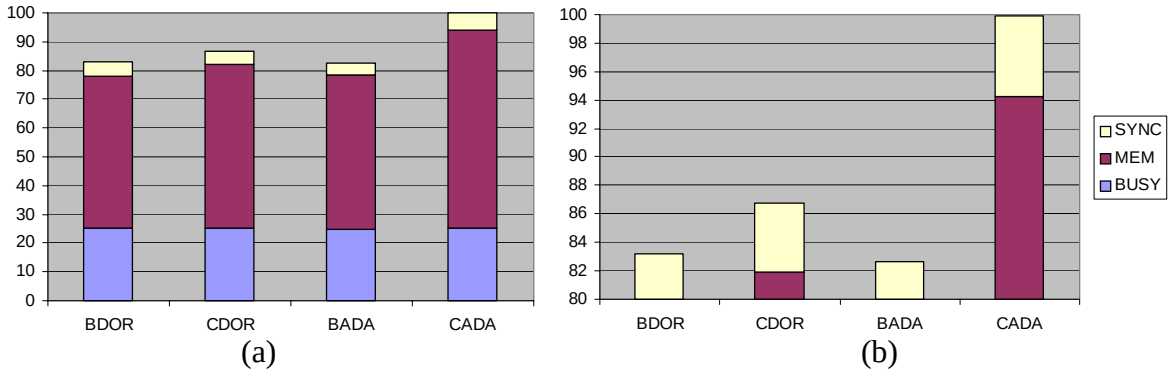


Figura 3-27. (a) Tiempo de ejecución normalizado para FFT con 64 procesadores, (b) Detalle.

Los resultados muestran como los dos encaminadores con mecanismo de evitación de interbloqueo basado en los canales virtuales exhiben un rendimiento peor que los basados en la burbuja. Los dos encaminadores basados en burbuja muestran un rendimiento muy próximo con diferencias en tiempos de ejecución a favor del adaptativo por debajo del 1%.

Para entender mejor estos resultados y la aparente discrepancia entre ellos y los exhibidos por cada encaminador bajo condiciones de tráfico sintético, en la Figuras 3-28 y 3-29 se muestra la traza de ejecución de la fase paralela[1] de la aplicación para la red de respuestas. En estas figuras se representa tanto la latencia promedio de los mensajes como la carga que ha de soportar la red a medida de que la aplicación se ejecuta. En la primera figura hemos optado por representar ambas medidas en términos de ciclos de red y en la segunda por ciclo de procesador. Al emplear dos criterios temporales en cada caso podemos entender mejor qué es lo que esta sucediendo en el sistema desde el punto de vista estructural, siendo el segundo caso el verdaderamente significativo ya que el eje de ordenadas se encuentra perfectamente normalizado. En la primera figura, al tener diferentes tiempos de ciclo los encaminadores considerados, es evidente que las medidas no se pueden comparar directamente aunque nos servirá para apreciar como afectan las

1. En los tiempos de ejecución considerados únicamente se considera el tiempo de ejecución de la fase central de la aplicación, obviándose las fases de inicialización y finalización. Este criterio se ha mantenido a lo largo de todo el trabajo.

diferencias estructurales entre los diferentes encaminadores. Sin embargo, en el segundo caso el tiempo de ciclo del procesador sí es idéntico a todas las redes.

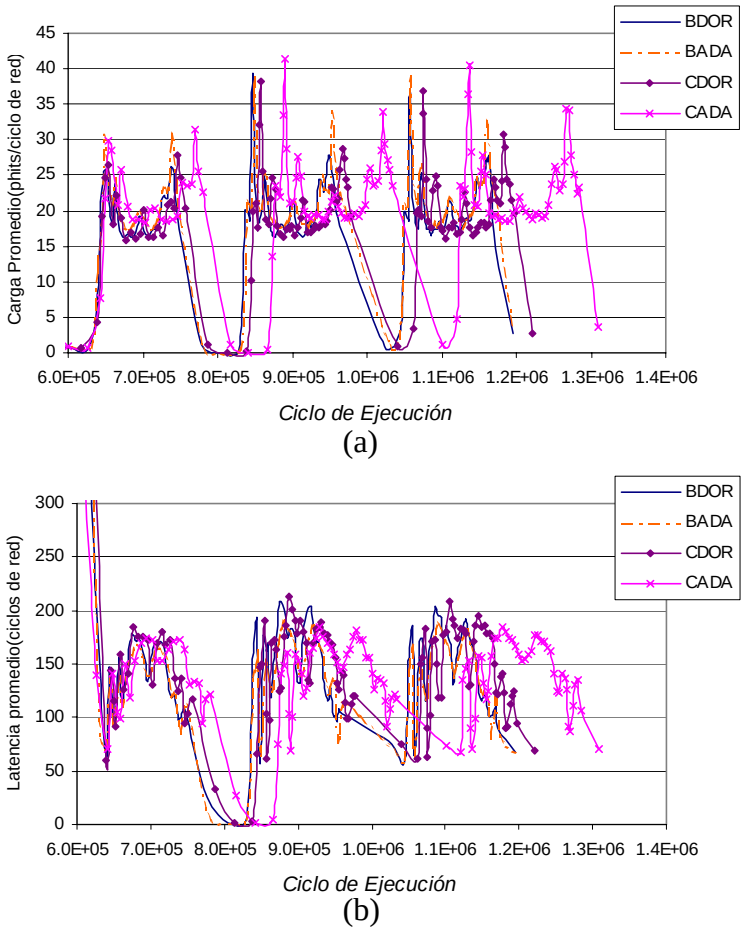
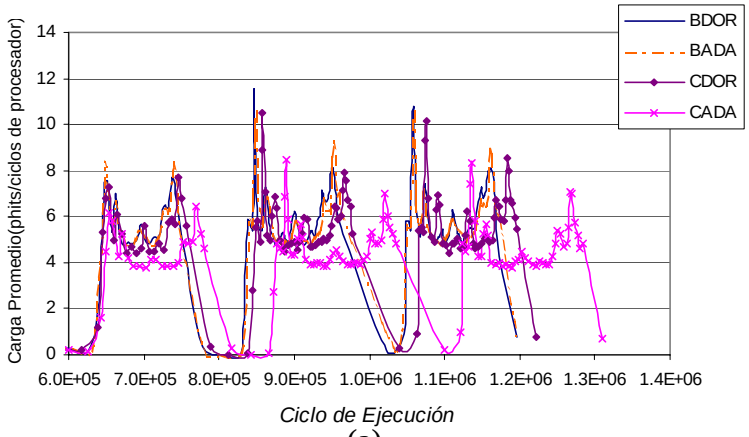


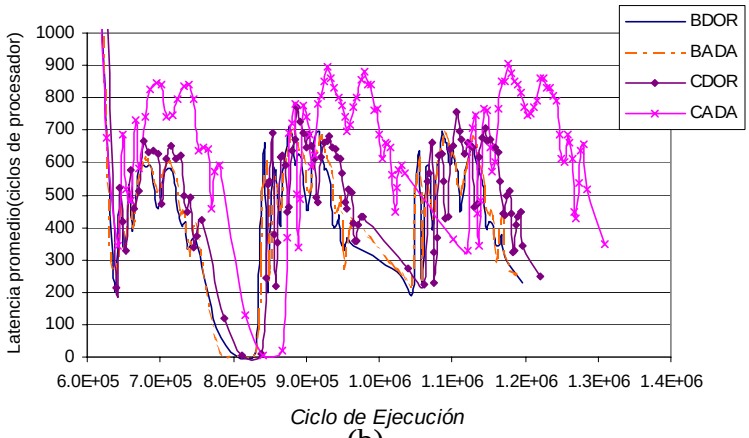
Figura 3-28. Traza de ejecución de la red de respuestas para FFT con 64 procesadores (en *términos estructurales)*, (a) Carga Promedio, (b) latencia promedio.

El primer hecho claro que se observa es que, con esta aplicación, el sistema no llega a aplicar una carga excesivamente alta a la red. En el peor de los casos, el máximo apenas supera los 30 *phits* por ciclo de red en la mayoría de los casos sobre un máximo teórico de 64 *phits*. Aun así, las características del tráfico de esta aplicación hace que se observen algunas diferencias en el volumen de tráfico aceptado por cada red. Se observa como el encaminador con canales virtuales adaptativo exhibe un comportamiento ligeramente mejor que cualquiera de las demás alternativas. El encaminador adaptativo basado en la burbuja le sigue de cerca y por debajo los dos encaminadores deterministas. La latencia, en ciclos de red, sigue aproximadamente la misma tendencia. Sin embargo, como ya sabemos, esta medida expresa únicamente el rendimiento desde el punto de vista estructural. Al considerar el periodo de reloj de cada encaminador,

expresando estas medidas en ciclos de procesador, podemos saber cual es el coste real de los accesos remotos y su impacto en el tiempo de ejecución de la aplicación. De esta forma, como se puede ver en la Figura 3-29, el elevado tiempo de ciclo del encaminador adaptativo basado en canales virtuales introduce una notable pérdida de rendimiento. Pese a mostrar mejor rendimiento estructural, el efecto del periodo de reloj hace que el *throughput* real empeore y la latencia de los mensajes llegue a ser considerablemente superior al resto de los casos. En conjunto, esto se traduce en un mayor tiempo de ejecución para la aplicación. En el resto de casos, los periodos de reloj de los encaminadores no llegan a ser determinantes. En las figuras previas podemos apreciar como la adaptatividad entra en juego permitiendo reducir, de forma sensible, la latencia de los mensajes en zonas de carga alta, provocando una mejora en el rendimiento final. Además, es importante indicar que en las zonas de carga baja los encaminadores mas rápidos logran recuperar una ligera ventaja con respecto a el adaptativo basado en la burbuja. En esta aplicación el porcentaje de ambas zonas esta equilibrado, lo que limita la ganacia absoluta del encaminador adaptativo.



(a)



(b)

Figura 3-29. Traza de ejecución de la red de respuestas para FFT con 64 procesadores (en *términos reales)*, (a) Carga Promedio, (b) latencia promedio.

Radix

El tamaño de problema seleccionado para esta aplicación es el marcado por defecto, es decir, tanto el número de claves enteras a ordenar como el radio de las mismas es de un millón. Los resultados de la ejecución con cada uno de los encaminadores analizados son los mostrados en la Figura 3-30. Como se puede apreciar, el comportamiento que muestra en cada caso la aplicación en muy similar a *FFT*. El encaminador adaptativo basado en burbuja sigue siendo el de menor tiempo de ejecución, llegando a aventajar hasta en un 28% al encaminador adaptativo basado en canales virtuales y un 7 % al encaminador determinista basado en canales virtuales. Con respecto a el encaminador determinista basado en burbuja la diferencia solo llega a ser algo más de 1.5%.

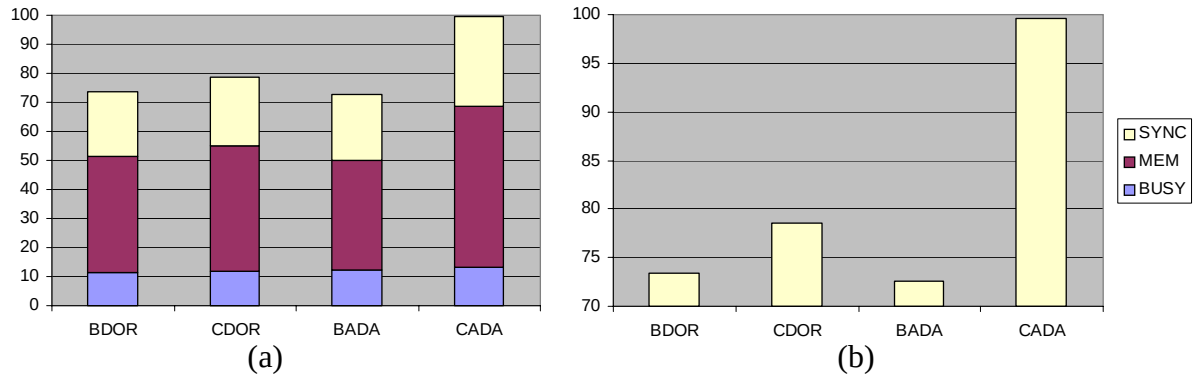


Figura 3-30. (a)Tiempo de ejecución normalizado para Radix con 64 procesadores, (b) Detalle

El comportamiento de los encaminadores *wormhole* con respecto a los VCT es acorde, a grandes rasgos, con los resultados obtenidos bajo cargas de trabajo sintéticas. Sin embargo, resulta interesantes las pequeñas diferencias que se observan en el rendimiento de los encaminadores VCT. Para analizar este hecho, de la misma forma que en la aplicación anterior, mostraremos como varía la demanda del sistema hacia la red de interconexión a medida que la aplicación se ejecuta. En las Figuras 3-31 y 3-32 se muestra el comportamiento de la aplicación tanto en términos estructurales como tecnológicos. Como se puede apreciar, existen dos zonas en las que la presión ejercida sobre la red es elevada, haciendo que, en determinados casos, se supere su límite de saturación. Se observa claramente cómo, en función del rendimiento que alcanza cada encaminador en términos estructurales, el máximo alcanzado es más o menos elevado. Esto es muestra inequívoca que la red se está convirtiendo en el cuello de botella. Pese a este hecho, las diferencias en tiempos de ejecución no son directamente proporcionales a las diferencias en el límite de saturación de cada red. La razón fundamental de este hecho es que existen zonas de carga más baja en las que los encaminadores más rápidos (deterministas) tienden a recuperar el tiempo perdido en las zonas altas de carga. Este hecho es claro en el caso del encaminador deter-

minista basado en burbuja con respecto a su contrapartida adaptativa. Se observa como la altura de las zonas de carga elevadas en el encaminador adaptativo son más elevadas, lo que hace que el tiempo requerido para efectuar el intercambio de claves sea menor que el determinista. Puesto que el área encerrada por esa zona tiene que ser aproximadamente constante (representa el número total de mensajes generado por la aplicación), el encaminador adaptativo permitiría alcanzar su finalización antes que el determinista.

Notar que el tráfico asociado a esta aplicación en esas zonas corresponde al intercambio de claves y dado que son generadas de forma aleatoria, el patrón asociado es prácticamente uniforme por lo que las diferencias no son muy acusadas. Sin embargo, en las zonas de carga baja el encaminador determinista tiende a recuperar el tiempo perdido previamente hasta casi reducir la ganancia alcanzada por el adaptativo en la zona anterior. Además, en el caso concreto de esta aplicación, el patrón de tráfico asociado a esas zonas es prácticamente local en su totalidad (corresponde a las pausas de sincronización de la aplicación que siempre se realizan entre nodos vecinos). El resultado final de este comportamiento es que las diferencias, en el tiempo de ejecución de la aplicación, no son tan acusadas como las observadas en el tráfico sintético.

Además, es importante indicar que el tiempo de ciclo de cada encaminador afecta de forma considerable en el volumen máximo de tráfico soportado por cada red. En el caso del encaminador adaptativo basado en la *burbuja*, su tiempo de ciclo más alto, con respecto a su contrapartida determinista, hace que las diferencias en *throughput* observadas, desde el punto de vista estructural, se reduzcan sensiblemente. En el caso del encaminador adaptativo basado en los canales virtuales, de la misma forma que en la *FFT*, su rendimiento se ve seriamente afectado por su elevado tiempo de ciclo, de tal forma que su nivel de productividad real es extremadamente pobre para esta aplicación, llegando a ser incluso inferior al de los encaminadores deterministas.

En cuanto a la latencia de los mensajes, se observa un comportamiento muy similar al caso del *throughput* en el sentido de que la latencia es inferior para el encaminador adaptativo en zonas elevadas de carga y en zonas bajas de la carga el comportamiento se invierte como consecuencia de un mayor tiempo de ciclo para los encaminadores adaptativos. Por otro lado, se observa como en las zonas bajas la latencia esta en torno a los 200 ciclos de procesador, lo que confirma que el tráfico que circula por la red en esa fase de la ejecución es prácticamente local.

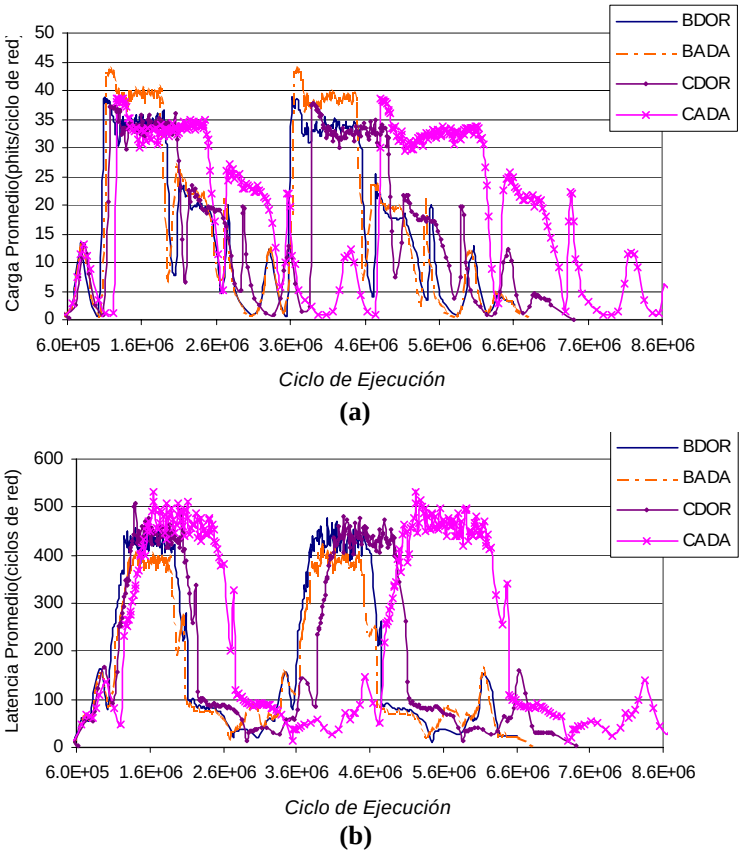


Figura 3-31. Traza de ejecución de la red de respuestas para Radix con 64 procesadores (en *términos estructurales)*, (a) Carga Promedio, (b) latencia promedio.

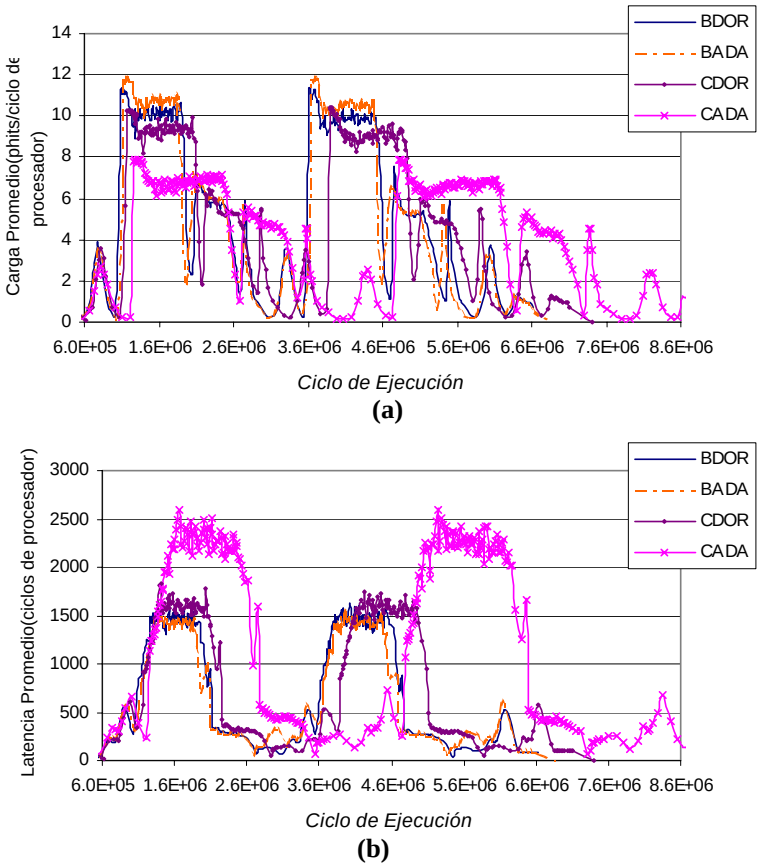


Figura 3-32. Traza de ejecución de la red de respuestas para Radix con 64 procesadores (en *términos reales)*, (a) Carga Promedio, (b) latencia promedio.

LU

A la hora de ejecutar la aplicación, en este caso se ha optado por emplear un tamaño de problema inferior al marcado por defecto. Dada la cantidad de arquitecturas alternativas a evaluar y lo demandante de esta aplicación, la reducción es prácticamente obligatoria (con el tamaño por defecto, con matrices de 512x512 elementos, el tiempo de simulación llega a superar en la mayoría de los casos las 150 horas). En este caso se han empleado matrices de 256x256 elementos y con un tamaño de bloque de 16x16. Aún con esta reducción en el tamaño del problema, la aplicación sigue escalando de forma correcta con los parámetros elegidos en el sistema. Los resultados de ejecución se muestran en la Figura 3-33. Como ha sido tónica habitual en las aplicaciones anteriores, cada uno de los encaminadores evaluados muestran unas diferencias claramente apreciables. Del mismo modo que en situaciones previas, el encaminador que muestra un rendimiento más pobre es el adaptativo basado en canales virtuales. Las diferencias con respecto al resto superan en muchos casos el 15%, llegando a ser más del 20% las diferencias con respecto al otro encaminador adaptativo. En este sentido, el encaminador adaptativo basado en

la burbuja muestra, para esta aplicación, el mejor tiempo de ejecución de todas las alternativas. Las diferencias con respecto a los otros dos encaminadores es de, aproximadamente, un 10% con respecto al determinista basado en canales virtuales y un 6% con respecto al encaminador determinista basado en burbuja.

Esta aplicación es un ejemplo claro de la utilidad del encaminador adaptativo basado en la burbuja, pese a tener una latencia base y tiempo de ciclo superior a los encaminadores deterministas, logra mejorar el tiempo de ejecución de la aplicación.

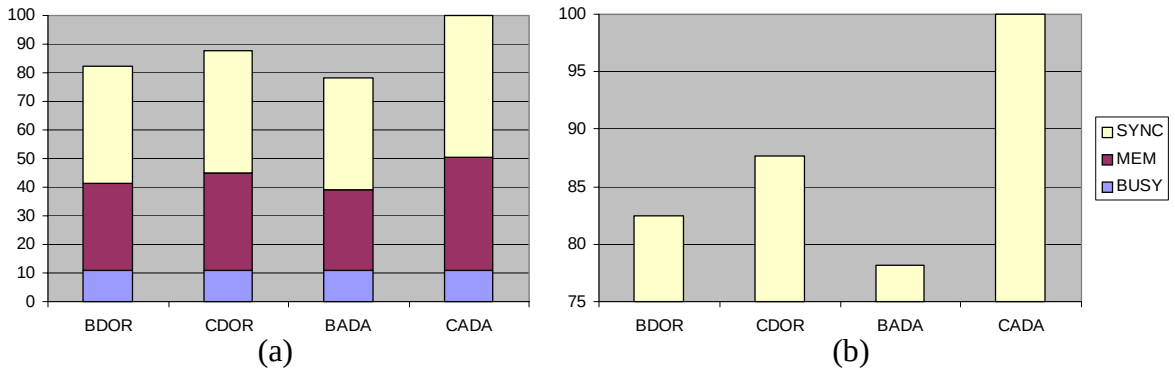


Figura 3-33. (a)Tiempo de ejecución normalizado para Lu con 64 procesadores, (b) Detalle.

Con el fin de comprender mejor qué es lo que esta sucediendo, en la Figura 3-34 se ha representado la traza de ejecución de la aplicación para la fase de interés (en este caso solo se representa el *throughput* en términos estructurales y la latencia en términos tecnológicos). Curiosamente, la presión que ejerce la aplicación sobre la red es relativamente baja en términos absolutos. Como se puede apreciar en la Figura 3-34.(a), en el peor de los casos, para cualquiera de las propuestas, apenas si supera los 10 *phits* por ciclo de red, lo que equivale a decir que solamente el 15% de los nodos del sistema llegan a inyectar información en la red de forma simultánea. Sin embargo la latencia promedio de las referencias remotas es más que considerable, llegando a superar en algunos casos los 1500 ciclos de procesador. Es indudable que el tipo de patrón de tráfico que subyace a esta aplicación es el causante de este efecto. Como se muestra en la Figura 2-14, este patrón es bastante proclive a la formación de puntos calientes de carga, especialmente a lo largo de la diagonal de la matriz a factorizar. Precisamente la adaptatividad logra minimizar el impacto de estas zonas altas de carga en la latencia, lo que se traduce en un menor tiempo de ejecución para la aplicación.

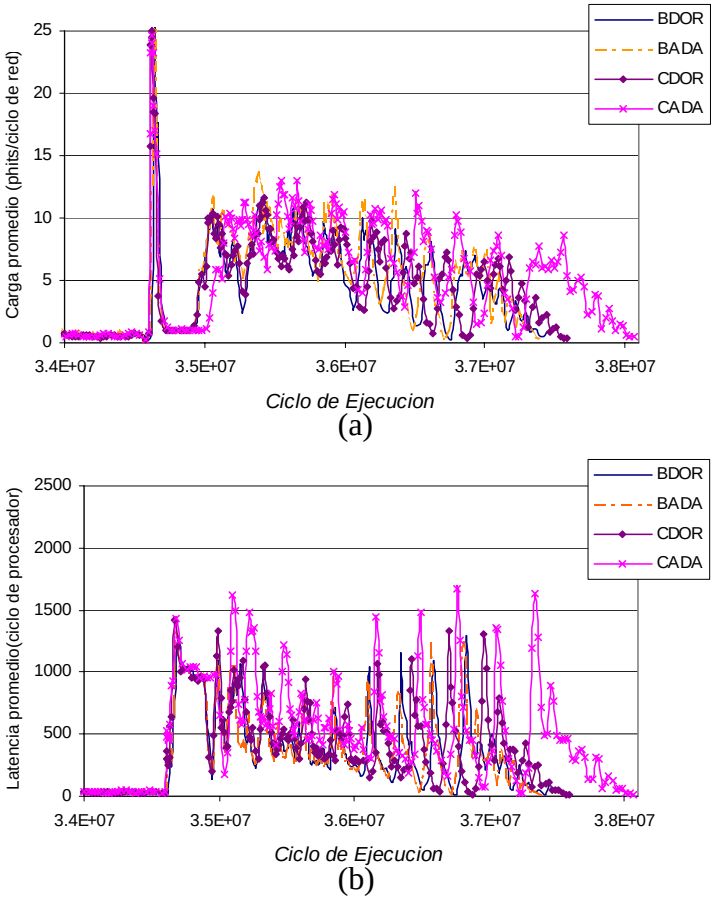


Figura 3-34. Traza de ejecución de la red de respuestas para LU con 64 procesadores, (a) Carga Promedio, (b) latencia promedio .

Con el fin de clarificar las anteriores afirmaciones pasaremos a realizar una ampliación de la traza de ejecución en la zona inicial de la fase de interés. En la Figura 3-35 se pueden apreciar estos datos. Ambos encaminadores adaptativos tienden a soportar un nivel de carga superior a los deterministas. Aunque la diferencia en este sentido no es elevada, lo que si es importante indicar es como la latencia del basado en la burbuja es notablemente inferior a cualquiera de las otras alternativas. Como ya sabemos, el mayor tiempo de ciclo del encaminador adaptativo basado en los canales virtuales hace que presente una latencia muy superior a cualquiera de las demás alternativas, por lo que, como ocurría en aplicaciones anteriores, el tiempo de ejecución de la aplicación en este caso es bastante superior a cualquiera de las restantes opciones. Un caso claro es el punto marcado en la Figura como A. Se observa como el adaptativo basado en burbuja llega a permitir mejorar la latencia en aproximadamente 200 ciclos de procesador. Esta mejora, junto con la constante presión por parte de la aplicación sobre la red (la fases de cálculo y comunicación no están tan marcadas como en *Radix* o *FFT*), hace que finalmente este encaminador logre mejorar apreciablemente el tiempo de ejecución de la aplicación.

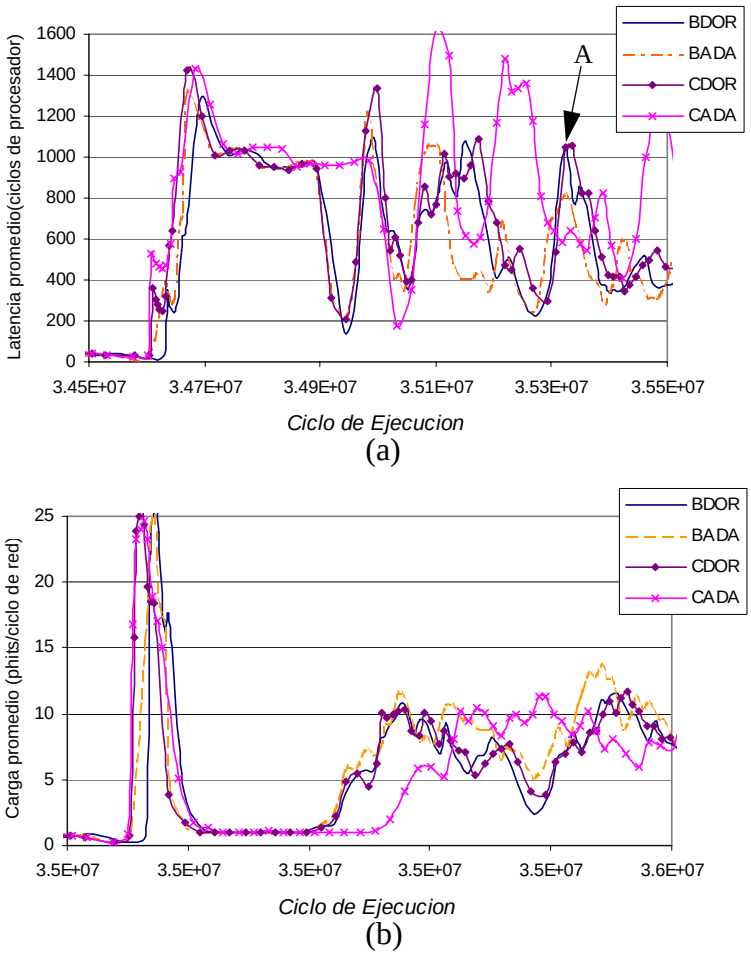


Figura 3-35. Detalle de la traza de ejecución para LU con 64 procesadores, (a)latencia promedio, (b) Carga Promedio.

3.6 Conclusiones

En este capítulo hemos presentado un mecanismo simple que permite implementar encaminadores completamente adaptativos con un coste reducido. La técnica de control de flujo empleada garantiza el continuo movimiento de los mensajes a lo largo de cualquier anillo de la red dado que garantiza que siempre hay espacio para, al menos, un paquete en las colas de tránsito asociadas al anillo. Las colas en las que se aplica este control de flujo corresponden a aquellas en las que los mensajes se encaminan siguiendo un criterio completamente determinista. Este conjunto de colas juegan el papel de vías de escape al combinarlas con otras completamente adaptativas. En consecuencia, el encaminador adaptativo basado en control de flujo burbuja permite a la red ser libre de interbloqueos con un coste más reducido que propuestas previas al requerir únicamente dos conjuntos de colas y por tanto dos canales virtuales por línea física.

El análisis comparativo realizado de nuestra propuesta frente a otros encaminadores deterministas y adaptativos, en el caso particular de los toros bidimensionales, muestra que éste encaminador alcanza notables mejoras en *throughput* frente a cualquiera de las demás alternativas y el incremento en latencia base apenas supera el 9% con respecto al encaminador más simple.

La simplicidad del propio algoritmo, junto con la implementación *hardware* empleada, permite al encaminador propuesto mejorar incluso el rendimiento del encaminador determinista basado en canales virtuales a cargas bajas. Por otra parte, la complejidad del encaminador adaptativo basado en canales virtuales hace que su tiempo de ciclo sea muy alto con respecto al basado en *Burbuja*. La consecuencia de este hecho es que su rendimiento, desde el punto de vista tecnológico, sea muy pobre, llegando en algunos casos a ser peor, tanto en latencia como en *throughput*, que los encaminadores deterministas.

Los efectos que muestran los datos de la implementación se trasladan directamente al rendimiento de la red de interconexión, observándose, en términos generales, que los encaminadores VCT basados en *Burbuja* logran mejor rendimiento que los encaminadores *wormhole*. Esto se observa tanto al emplear cargas de trabajo sintéticas como reales. En el segundo caso, pese a emplear un sistema ccNUMA, muy sensible a la latencia, el encaminador propuesto logra, en la mayoría de las aplicaciones evaluadas, sacar una considerable ventaja al resto de implementaciones. Por otro lado, el encaminador *wormhole* adaptativo tiende a exhibir un notable peor rendimiento en todas las aplicaciones frente al resto de encaminadores considerados.

A la vista de estos datos, pese a las limitaciones del entorno de evaluación, especialmente en lo que se refiere al reducido tamaño del sistema evaluado y problemas pequeños que son factibles de considerar, la tendencia que podemos observar muestra como la propuesta realizada tiende a favorecer el rendimiento del sistema global. Es de esperar que con otros tamaños de problema más grandes y sistemas de mayores dimensiones estas ganacias tiendan a incrementarse. Incluso, sin emplear configuraciones exóticas para el sistema, como varios procesadores por nodo, etc... las ganancias son claras, pese a que existen muchas referencias a lo largo de la bibliografía donde se indica lo contrario, incluso asumiendo en muchos de estos casos que el coste *hardware* introducido por la adaptatividad es nulo. Sin embargo, a nuestro parecer, tales afirmaciones se desprenden de sistemas de evaluación menos precisos que el empleado aquí. La precisión del simulador de la red de interconexión puede tener mucha influencia en estos datos, y en nuestro caso estamos empleando una infraestructura de simulación que emula el sistema de forma muy fiable. Como se puede ver en Sección A.3.2, las diferencias en los tiempos de ejecución de las aplicaciones al emplear un simulador más o menos preciso pueden ser notables. La baja precisión en la simulación se traduce en un rendimiento distorsionado para zonas altas

de carga y el error tiende a ser mayor cuando la complejidad del encaminador es elevada, como es el caso de los encaminadores adaptativos. Como se ha mostrado en el comportamiento dinámico de las aplicaciones, existen muchas zonas puntuales en las que el sistema ejerce una presión muy alta, llegando a superar en ocasiones el punto de saturación de la red. Es por ello que un modelado impreciso de las zonas altas de carga se traduzca en resultados erróneos para el tiempo de ejecución de la aplicación.