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 5

Dependencias Topológicas

El tipo de topología empleada en la red de interconexión es uno de los factores clave que puede llegar a afectar al rendimiento del sistema global de forma significativa. En este capítulo analizaremos varios tipos de topologías y evaluaremos su influencia en el rendimiento de los computadores paralelos. Estudiaremos las restricciones que impone el hardware subyacente, en función de cada topología e introduciremos la utilización de una estructura nunca empleada en ningún sistema y cuyas características topológicas son óptimas.

5.1 Introducción.

Entre los diversos factores que influyen en el rendimiento del subsistema de comunicación de los computadores paralelos, uno de los más importantes es la topología empleada en la red de interconexión. Existe un elevado número de trabajos que abordan el análisis de este tipo de aspectos desde un punto de vista puramente teórico. Sin embargo, no es inmediata la traslación de las conclusiones extraídas en estos niveles de abstracción al campo de las aplicaciones reales. En primer lugar, es preciso conocer el impacto final de las implicaciones tecnológicas de cada tipo de topología en la complejidad del encaminador. En este caso, las restricciones *hardware* impuestas por la topología pueden limitar su rendimiento. En este capítulo se pretende aproximar ciertos aspectos teóricos sobre la topología de las redes de interconexión y su influencia sobre las aplicaciones reales, pasando por las consecuencias *hardware* que implica cada uno de los tipos de red a considerar.

Dentro de la amplia variedad de topologías propuestas en el área de las redes de interconexión nos centraremos en las redes de interconexión directas [115]. Este tipo de redes se basan en enlaces punto a punto entre los nodos de proceso del sistema. Su característica fundamental es que la latencia de los mensajes es dependiente del par origen-destino. Poseen unas características topológicas que permiten explotar la localidad espacial de los datos, característica común de la mayoría de las aplicaciones paralelas. Esto y sus características de coste, hacen que este tipo de topologías estén actualmente siendo más empleadas en los sistemas reales que las de otro tipo, como las redes indirectas.

Por tanto, nos centraremos en el análisis de redes directas y de bajo grado. Trabajos clásicos ([3], [41]) demuestran como este tipo de redes logran alcanzar, desde el punto de vista teórico, mejor rendimiento que redes de grado superior en situaciones de bisección constante. El término bisección constante indica que el análisis supone que el número de líneas de datos que cruza la bisección es fijo e independiente del número de dimensiones de la red. Básicamente, esto equivale a considerar que el número total de líneas de comunicación de cada encaminador se mantiene constante para cualquier grado de la red. La principal conclusión extraída de estos trabajos era que, desde el punto de vista teórico, es preferible tener una red de bajo grado con canales de comunicación anchos, que no una red de elevado número de dimensiones con canales de comunicación estrechos. A la vista de estos resultados, el estudio llevado a cabo en este capítulo se centra en redes de grado cuatro o bidimensionales. Las dos primeras son la *Malla* y el *Toro.* Estas redes han sido extensamente analizadas y empleadas en diversos sistemas comerciales y prototipos. Además, se introduce el análisis de una red de interconexión toroidal denominada *Midimew (MInimal DIstance MEsh with Wrap-arounds)* [17]. Sobre esta red, pese a que se pro-

puso originalmente hace más de una década, no se ha realizado anteriormente un estudio como el presentado en este trabajo. Posee unas propiedades topológicas óptimas, en el sentido de que no existe ninguna red de interconexión directa, de grado cuatro y simétrica, con diámetro ni distancia media inferior. Esta red nunca ha sido empleada en ningún sistema real dado que, pese a sus buenas características topológicas, existían un serie de aspectos que complicaban su empleo. En este trabajo se aportan soluciones válidas para los problemas críticos de la red. Las soluciones propuestas permiten competir directamente con la sencillez de implementación del toro o la malla y sin incorporar coste adicional.

5.2 Redes evaluadas. Redes directas de bajo grado.

En esta sección se introducirán las características más reseñables de los tres tipos de red de interconexión evaluados. En todas las topologías consideradas, como se ha comentado previamente, el grado se ha limitado a cuatro.

5.2.1 Malla

Se trata de una topología bien conocida y ampliamente utilizada en diversas máquinas reales. El rango de sistemas en el que se ha empleado va desde los prototipos experimentales como el *DASH* [85] hasta sistemas de muy alto rendimiento pertenecientes al programa *ASCI* del DOE como el *Option Red* [83]. Sus principales ventajas, frente a otro tipo de topología es su bajo coste comparado con redes de interconexión en las que aparecen enlaces periféricos, como es el caso del toro *n*-dimensional.

Tradicionalmente su bajo coste está directamente relacionado con la no existencia de interbloqueos cuando se emplean algoritmos de routing deterministas. Esto hace que no sea necesario incorporar mecanismos de evitación de interbloqueo, aspecto que, indirectamente, puede incrementar la complejidad de los encaminadores y por tanto degradar el rendimiento de la red. Además, el tipo de conectividad que presenta permite que los sistemas basados en esta red tengan una buena escalabilidad desde el punto de vista físico. Es muy fácil añadir nuevos nodos de proceso, sin ser necesario alterar la estructura preexistente. Además, la no existencia de enlaces periféricos hace que la implementación física de la red sea relativamente más sencilla y menos costosa que en otro tipo de topologías.

Sin embargo, esta red de interconexión adolece de varios inconvenientes. En primer lugar, las características topológicas de la *malla*, en lo referente a diámetro y distancia media, son bastante más pobres que los alcanzados por otras topologías. De la misma forma, la no existencia de enlaces periféricos hace que el nivel de productividad alcanzado por esta red esté también por

debajo del logrado por otras. Como consecuencia de este hecho, la escalabilidad en el ancho de banda es bastante más pobre. Otro aspecto negativo para esta topología es la falta de simetría. Generalmente, desde el punto de vista del programador o del sistema operativo, es más fácil distribuir los datos a manejar por la aplicación paralela o adaptar el algoritmo de la aplicación en una red simétrica que en una que no lo es.

5.2.2 Toro

Se trata de un caso especial de la familia de los grafos *k-ary n-cubes*. Dado que en nuestro caso el estudio se centra en las redes en las que cada nodo tiene solo 4 vecinos, nos restringiremos al análisis a los grafos *k-ary 2 cube*. Como es conocido, su estructura es muy similar a la de la malla incorporando enlaces periféricos, también denominados *wrap-around links*.

La incorporación de los enlaces periféricos permite que las características topológicas del toro mejoren de forma considerable con respecto a las de la malla. Estos enlaces adicionales suponen una importante reducción del diámetro y distancia media. Además, la escalabilidad en términos de productividad es superior a la malla, dado que el ancho de la bisección es el doble. Por otro lado, se trata de una red simétrica lo que resulta ventajoso para la mayoría de los algoritmos que necesitan cálculo paralelo.

Sin embargo, el empleo de los *wrap-arounds* implica la existencia de ciclos de dependencia estática entre canales. Como es conocido [40], esta clase de dependencias entre canales pueden provocar la aparición de interbloqueos, incluso empleando algoritmos de encaminamiento completamente deterministas. Frente a este problema, la solución adoptada usualmente se basa en emplear al menos dos canales virtuales por canal físico [40]. Como hemos analizado en el Capítulo 3, esto implica un incremento de complejidad en la arquitectura de los encaminadores de la red frente a la estructura que requiere la malla. Como consecuencia de este hecho, el tiempo de paso de los encaminadores se incrementa. Sin embargo, existen mecanismos de evitación de *deadlock* que eliminan la aparición de interbloqueos introduciendo el concepto de dependencias dinámicas entre canales. Como se mostró en el Capítulo 3, uno de los algoritmos que solucionan el problema, impidiendo la aparición de ciclos de dependencia dinámica, es el método de la *Burbuja*. Con el empleo de este algoritmo de evitación de interbloqueos se elimina la necesidad de canales virtuales adicionales sobre el caso de la malla, por lo tanto, desde este punto de vista, el coste añadido del toro es prácticamente despreciable frente al de la malla.

Tomando en cuenta estas consideraciones, la arquitectura de los encaminadores empleados en la malla y en el toro va a ser prácticamente idéntica, diferenciándose exclusivamente en las restricciones en inyección y cambio de dimensión que implica el método de la *Burbuja*. Por lo

tanto, el coste *hardware* de ambos encaminadores es el mismo. La pequeña diferencia funcional que existe entre ambos es muy simple y no va a repercutir en el *hardware*.

Hecha esta puntualización, parece claro que, desde el punto de vista del coste de implementación, el *toro* introduce una pequeña penalización debida al superior número de enlaces necesarios, pero añadiendo, desde el punto de vista funcional, notables ventajas sobre la *malla*.

5.2.3 Midimew

Las redes Midimew son también un tipo de red de bajo grado, muy similares a las redes *k-ary 2-cube*, pero con unas características topológicas muy interesantes. Esta red fue propuesta originalmente en [16]. Son isomorfas a una familia de los grafos circulantes de grado 4, $C_N(a,b)$ [19]. En los grafos circulantes, cada nodo *i* se encuentra conectado a los nodos $(i \pm a) Mod\ N$ y $(i \pm b) Mod\ N$ siendo *N* el número total de nodos de la red. Dentro de esta clase de grafos de grado 4 vértice-simétricos, la red Midimew corresponde a la familia de los grafos circulantes que verifican:

$$C_N\left(\left\lceil \sqrt{\frac{N}{2}} \right\rceil - 1, \left\lceil \sqrt{\frac{N}{2}} \right\rceil\right) \qquad \textbf{[5-1].}$$

donde $\lceil \bullet \rceil$ representa la función techo. El grafo circulante resultante de esta expresión es isomorfo a una estructura tipo malla con enlaces periféricos. En realidad, el aspecto de la red es toroidal y se asemeja bastante a la del toro bidimensional. Para una red de 24 nodos el aspecto que presenta el grafo circulante asociado es el que aparece en la Figura 5-1(a). Siguiendo el criterio de construcción descrito en [17], el aspecto definitivo de la red en forma de malla con enlaces periféricos es la que se muestra en la Figura 5-1(b).

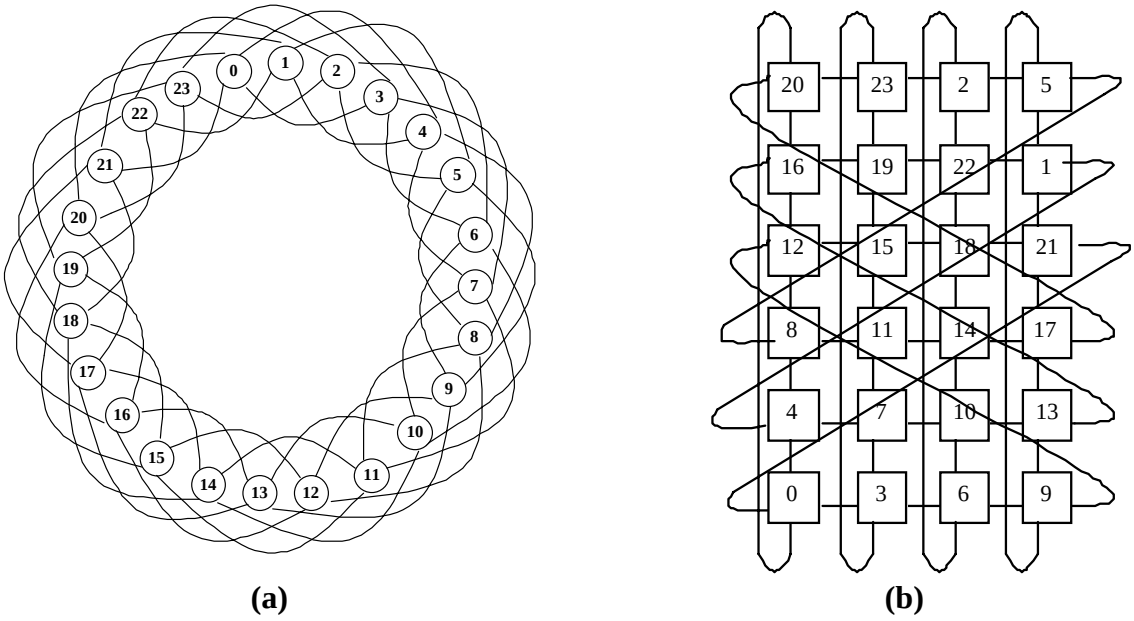


Figura 5-1. (a) Grafo circulante con N=24 y b=4 and (b) su transformación en una malla con enlaces periféricos.

En general, su aspecto no tiene porque ser completamente rectangular sino que puede presentar algunas irregularidades en función del número de nodos que posea. En general, existen dos representaciones para este tipo de redes: la *Midimew* horizontal o MDWH y la representación *Midimew* vertical o MDWV. Ambas se apoyan en una estructura como la representada en la Figura 5-2. Los parámetros básicos de esta representación esta indicados en la Ecuación 5-2 para el caso de las redes horizontales y en la Ecuación 5-3 para el caso vertical.

$$r_H = \left\lceil \frac{N}{b} \right\rceil \cdot b - N \qquad \textbf{[5-2].}$$

$$h_H = (b + r_H)$$

$$v_H = \left\lceil \frac{N}{b} \right\rceil - r_H$$

$$r_V = \left\lceil \frac{N}{b-1} \right\rceil \cdot (b-1) - N \qquad \textbf{[5-3].}$$

$$h_V = (b - 1 + r_V)$$

$$v_V = \left\lceil \frac{N}{b-1} \right\rceil - r_V$$

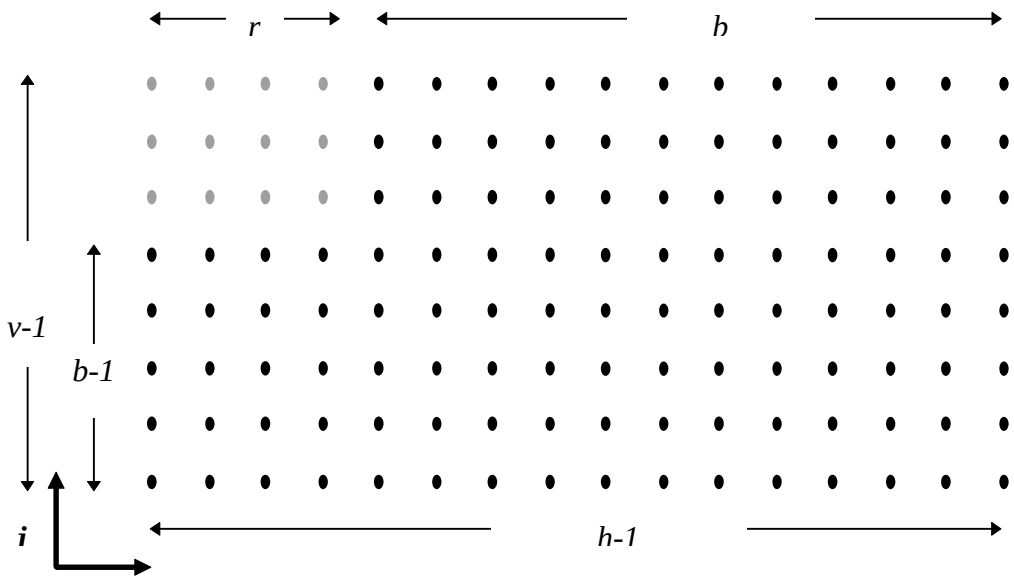


Figura 5-2. Construcción de la red *Midimew* a partir del grafo circulante isomorfo.

De acuerdo con esta estructura, siempre y cuando r_H o r_V sean cero, la red será rectangular y por tanto muy similar en su aspecto a un toro 2-D. Para que algunos de estos valores se anule es necesario que *N* sea múltiplo de *b* ó *b-1*. Si ocurre esto, como se puede apreciar en las expresiones previas, siempre el parámetro *r* será nulo. Por lo tanto, en función del valor de *b* y *b-1* en cada caso, será preciso emplear una u otra estructura. En cualquier caso, como es obvio, existirán configuraciones con un número de nodos tal que ninguna de las dos organizaciones alternativas permitirán lograr un red rectangular.

En el caso de las redes *Midimew* horizontales, cada nodo *(i,j)* estará etiquetado *f(i, j)=[i(b-1)+jb] Mod N* y se encuentra conectado a los nodos que determina el grafo circulante isomorfo, es decir los nodos vecino estarán etiquetados de acuerdo con:

$$\begin{aligned} f(i+1,\ j) &= [(i+1)(b-1) + jb] \text{ Mod } N = f(i,\ j) + (b-1) \text{Mod } N \\ f(i-1,\ j) &= [(i-1)(b-1) + jb] \text{ Mod } N = f(i,\ j) - (b-1) \text{Mod } N \\ f(i,\ j+1) &= [i(b-1) + (j+1)b] \text{ Mod } N = f(i,\ j) + b \text{ Mod } N \\ f(i,\ j-1) &= [i(b-1) + (j-1)b] \text{ Mod } N = f(i,\ j) - b \text{ Mod } N \end{aligned} \qquad \textbf{[5-4].}$$

En el caso de la *Midimew* vertical, cada nodo *(i,j)* estará etiquetado como *f(i, j)=[ib+j(b-1)] Mod N* y sus nodos vecinos estarán etiquetados como:

$$
\begin{aligned}
f(i + 1,\ j) &= f(i,\ j) + b \bmod N \\
f(i - 1,\ j) &= f(i,\ j) - b \bmod N \\
f(i,\ j + 1) &= f(i,\ j) + (b - 1) \bmod N \\
f(i,\ j - 1) &= f(i,\ j) - (b - 1) \bmod N
\end{aligned}
\qquad \textbf{[5-5].}
$$

A modo de ejemplo podemos construir ambas redes equivalentes para una red con 14 nodos. De acuerdo con la Ecuación 5-1, el grafo circulante que da lugar a la red es $C_{14}(3,2)$. Las dos organizaciones alternativas para esta red son las representadas en la Figura 5-3. Es importante señalar que, pese a que el aspecto de ambas redes difiera, las dos siguen siendo isomorfas al grafo circulante que las da lugar, luego son isomorfas entre ellas y por tanto siguen teniendo las mismas características topológicas.

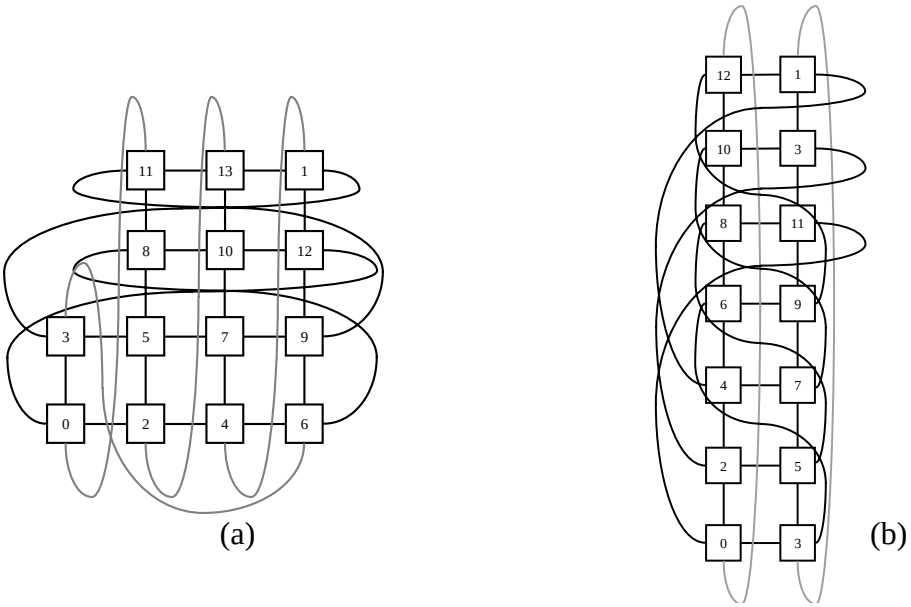


Figura 5-3. Redes *Midimew* (a) horizontal y (b) vertical para N=14.

5.2.4 Análisis Comparativo de Características Topológicas.

En primer término, vamos a analizar de forma cualitativa las tres topologías consideradas desde un punto de vista estrictamente topológico. Las características más reseñables de estas topologías se muestran en la Tabla 5-1.

Un primer dato relevante que se puede extraer de estas expresiones, es que la relación de diámetros entre el *toro* y la *Midimew* es aproximadamente:

$$\frac{d_{Toro}}{d_{Midimew}} \cong \sqrt{2} \qquad \textbf{[5-6].}$$

Por lo tanto, el diámetro de un toro bidimensional, con el mismo número de nodos, es superior

aproximadamente en un 42% al de esta red. Nótese que la función techo distorsiona de forma sensible este dato para redes con un número reducido de nodos. Por otro lado, la *Midimew* muestra una distancia media inferior al resto aunque con respecto al toro las diferencias no son tan elevadas como las mostradas en términos de diámetro. En la Figura 5-4 se ha representado cual es la tendencia que sigue tanto la distancia media como el diámetro, para las tres redes al variar el número de nodos.

Topología	***Número de Enlaces***	***Diámetro***	***Diámetro (k1=k2=k=√N)***	***Distancia Media (k (k_0 = bó $b-1$ **k**=√N)***
Malla 2-D $(k_1 \times k_2)=N$	$2k_1k_2 - k_1 - k_2 = 2\sqrt{N}(\sqrt{N}-1)$	$k_1 + k_2 - 2$	$2k-2$	$\frac{2k}{3}$
Toro 2-D $(k_1 \times k_2) = N$	$2k_1k_2 = 2N$	$\lceil \frac{k_1}{2} \rceil + \lceil \frac{k_2}{2} \rceil$	k	$2\left\lfloor \frac{k}{2} \right\rfloor \frac{k+1}{2} \frac{k}{k^2-1}$
Midimew con N Nodos $b = \lceil \sqrt{\frac{N}{2}} \rceil$	$2N$	b o $b-1$	$b = \lceil \frac{k}{\sqrt{2}} \rceil$	$k\left(1 - \frac{2(k_\bigcirc^2 - 1)}{3(N-1)}\right)$

Tabla 5-1. Comparación de las características topológicas de las tres topologías.

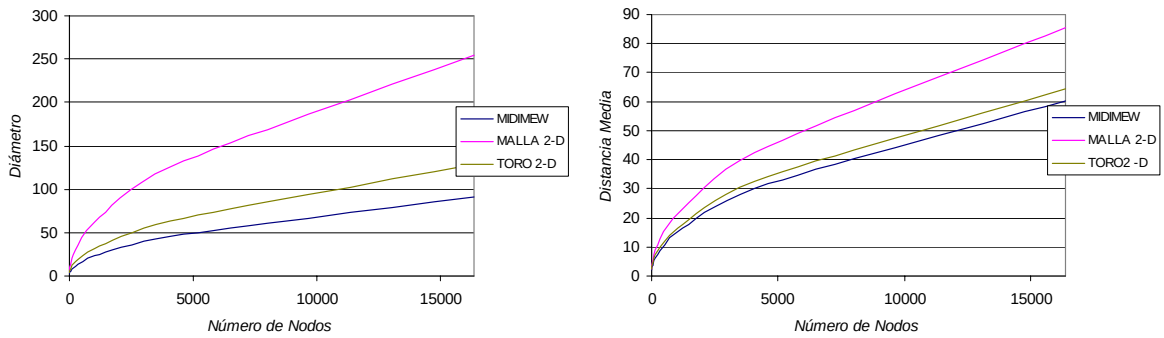


Figura 5-4. Evolución de las características topológicas de cada red, al variar el número de nodos.

Otra propiedad topológica a tener en cuenta es la escalabilidad de la red. Las tres redes escalan correctamente. La más sencilla de escalar es, sin lugar a dudas, la malla dado que para añadir nuevos elementos de proceso es suficiente con añadir nuevas filas o columnas sin afectar ninguno de los enlaces existentes. En cambio, tanto en la *Midimew* como en el toro, el añadir nuevos nodos implica volver a conectar los enlaces periféricos. En el caso del toro los enlaces periféricos vendrán conectados a la nueva fila o columna añadida. Sin embargo, en la *Midimew* es probable que el añadir nuevos nodos implique alterar más enlaces de los afectados por los nodos añadidos. Sin embargo, la red *Midimew* posee una ventaja clara sobre las otras dos topologías ya que soporta el incremento en un número de nodos arbitrario (no necesariamente una

columna o fila completa) sin afectar las características topológicas de la red. Como es evidente, en la mayoría de los casos ni la malla ni el toro pueden ser escalados con un número arbitrario de nodos ya que se transformarían en irregularidades que complicarían considerablemente el modo de operación (evitación de *deadlock*, encaminamiento, etc...)

5.3 Limitaciones de las Redes Midimew y Soluciones Propuestas.

Se ha constatado las buenas características topológicas de la *Midimew* al compararla, desde el punto de vista topológico, con la malla o el toro. Sin embargo, esta red presentaba una serie de inconvenientes que han restringido considerablemente su utilización hasta el presente. Estos eran, básicamente, la complejidad del algoritmo de encaminamiento y la inexistencia de métodos de evitación de interbloqueo. En este punto proponemos soluciones eficaces para ambos problemas.

5.3.1 Evitación de Interbloqueos.

Como se ha comentado previamente, la incorporación de enlaces periféricos en la malla para dar lugar al toro, permiten mejorar considerablemente el rendimiento alcanzado por la red. Sin embargo, al añadir estos enlaces, indirectamente, aparecen ciclos de dependencia estáticos entre canales lo que provoca que la red no sea libre de interbloqueos. De la misma forma que en el toro, los enlaces periféricos de la *Midimew* dan lugar a ciclos cerrados en el grafo de dependencias estáticas entre canales. Por lo tanto, de forma similar al toro, la *Midimew* presentará problemas de interbloqueo. Los métodos de evitación tradicionales basados en el uso de canales virtuales no tienen una aplicación clara en el caso de la *Midimew*. La dificultad de su utilización se encuentra en que no es evidente cuando un enlaces es o no periférico. Esta dificultad aumenta cuando añadimos nuevos nodos a la red.

Sin embargo, el método de evitación de interbloqueo basado en la *Burbuja* es perfectamente aplicable en el caso de la *Midimew*. Dado que este mecanismo opera con información local a la hora de restringir la inyección o los cambios de dimensión, su uso en la *Midimew* es idéntico al del toro. En este sentido, la única diferencia entre el toro y la *Midimew* es el tamaño de los anillos. Dado que el mecanismo es local, sabremos que, independientemente del tamaño de los anillos si garantizamos la existencia de espacio para, al menos, un paquete tendremos asegurado que la red es libre de interbloqueo. Por lo tanto, en este tipo de red resulta evidente que el mecanismo cumple su cometido correctamente.

Por otro lado, de la misma forma que se extendió el algoritmo para el empleo de mecanismos de encaminamiento adaptativo en el caso del toro, el uso de la misma aproximación es perfec-

tamente aplicable aquí también. El Teorema 3- 2, es válido independiente de si la red se trata de una *k-ary n-cube* o no. Si aseguramos que no hay ciclos en el grafo de dependencias dinámicas entre canales, la red será libre de interbloqueos. Por ello, resulta inmediato aplicar dicho algoritmo al caso de la *Midimew* permitiendo así el empleo de mecanismos de encaminamiento adaptativo con solo dos canales virtuales por línea física.

5.3.2 Complejidad del Encaminamiento.

Los encaminadores pueden emplear básicamente tres mecanismos para determinar el canal de salida a partir de la información de destino contenida en la cabeza del paquete: aritmético o relativo, elección de puerto basada en fuente y consulta en tabla. En el caso relativo, el origen construye el *routing-record* (RR), el cuál contiene los desplazamientos que ha de recorrer el paquete en cada dimensión. En el caso de una red bidimensional, sus valores estarán formados por el par $(\Delta x, \Delta y)$. El RR es examinado en cada encaminador intermedio del camino y sus valores son incrementados/decrementados en función del puerto de salida elegido para acercar el paquete a su destino. De esta forma, el paquete habrá alcanzado su destino cuando el RR tome valor (0,0). Todos los encaminadores analizados previamente en este trabajo emplean esta aproximación para determinar los puertos de salida de cada paquete.

Para el caso de la malla, el método para calcular estos valores es extremadamente sencillo, como se puede ver en la Figura 5-5. Sin embargo, el toro implica un algoritmo más complejo. La dificultad del algoritmo para extraer el RR en la *Midimew* es aún substancíalmente más complejo. Como se puede apreciar, incorpora una operación módulo y una división entera. Desde el punto de vista de la interface de red (o el elemento encargado de generar la cabecera del paquete) esto puede conllevar una penalización importante de tiempo.

La segunda aproximación consiste en emplear encaminamiento basado en fuente, en el que el nodo origen incorpora, en la cabeza del mensaje, los puertos de salida a seguir en cada nodo intermedio de la ruta. Cada encaminador solo tiene que retirar la porción de la cabeza que le corresponde y seleccionar el puerto adecuado. Es una estrategia habitualmente empleada en redes irregulares como *Myrinet* [13][106]. Sin embargo, en el contexto de las redes regulares este mecanismo no tiene sentido ya que los paquetes tienden a tener un elevado *overhead* cuando la red tiene un diámetro medio-alto. En el caso concreto de los sistemas ccNUMA esta aproximación, como es lógico, introduciría una notable perdida de rendimiento.

(a) Malla 2-D

```
Dx  :=(source mod k)-(dest mod k);
Dy  :=(source div k)-(dest div k);
```

(b) Toro 2-D

```
Dx  :=(source mod k)-(dest mod k);
If (Dx > k/2)
    Dx := k - Dx;
else if (Dx < -k/2)
    Dx := k  + Dx;
end if;
Dy  :=(source div k)-(dest div k);
If (Dy > k/2)
    Dy := k - Dy;
else if (Dy < -k/2)
    Dy := k  + Dy;
end if;
```

(c) **Midimew**

```
b := ceill(N/2);
m := abs(source-dest);
If (dest => source)
    Sign:= -1;
else
    sign:= 1;
end if;
If (m> N div 2)
    Sign:= -sign;
    m = N-m;
end if;
y0 := (m mod b);
x0 := (m div b) - y0;
y1 := b + y0;
x1 := x0 - (b-1);
If (y0 = 0 or x0 < y1)
    Dy := y0;
    Dx := x0;
else
    Dy := y1;
    Dx := x1;
end if;
```

Figura 5-5. Cálculo de *Routing-Record* para la malla, el toro y la midimew .

La mejor alternativa para la *Midimew* es el encaminamiento basado en "consulta en tabla". Bajo esta aproximación, cada encaminador mantiene una tabla de rutas en la que cada posible nodo destino posee una entrada que contiene los canales de salida que acercan cada paquete a destino desde el encaminador actual. En este caso, la cabeza del paquete únicamente contiene el nodo destino, usándose éste para indexar la tabla de rutas y obtener los puertos de salida oportunos. En el caso de que se esté empleando encaminamiento adaptativo, la función de selección, en función del estado de ocupación de los puertos de salida, y la función control de flujo serán las encargadas de establecer el puerto definitivo a partir de los datos contenidos en la entrada de la tabla.

Dado que estamos en una red bidimensional, cada entrada de la tabla debe contener cuatro bits por entrada. Estos indicarán las dimensiones y direcciones que acercan el paquete a destino *(x±y±)*. Por ejemplo, la entrada 1010 indicaría que el paquete puede viajar tanto por $x+$ como por $y+$ para acercarse a destino, 0011 indicaría que el mensaje ha agotado su viaje a lo largo de la dimensión x y que únicamente puede salir por el puerto $y-$. Cuando la entrada tome valor 0000 el mensaje habrá alcanzado su destino.

Los valores de cada una de las tablas pueden ser inicializados cuando el sistema se pone en marcha mediante mensajes específicos. Los mensajes de inicialización de las tablas se pueden distinguir de los convencionales mediante líneas adicionales o mediante códigos especiales en la cabecera del mensaje. En este último caso, el overhead introducido es nulo, si bien es cierto que será válido siempre y cuando el número de nodos de la red sea inferior al número de posibles valores que pueda tomar un *phit*. En tal caso, es posible emplear uno de los bits de la cabeza para indicar que se trata de un mensaje de inicialización. En cuanto al coste de las tablas, los constantes avances en VLSI permiten que sea perfectamente abordable su implementación en forma de pequeñas memorias, aún para un número de nodos considerable.

La principal ventaja de esta aproximación es que se mejora notablemente la tolerancia a fallos del sistema ya que, en cualquier momento, podremos redistribuir el tráfico para evitar uno o más enlaces en fallo. La desventaja más clara es la escalabilidad para sistemas con un número de nodos muy alto. De cualquier modo, encaminadores de sistemas como el *S3.mp*[94] o el *Cray T3E* [112] emplean tablas de rutas completas. Otros sistemas, como el *SPIDER*, utilizan tablas de rutas jerárquicas. En este sentido, existen varias alternativas para minimizar el impacto en la escalabilidad del sistema, como las propuestas en [131].

Bajo este punto de vista, si empleamos esta estrategia, claramente las diferencias en la complejidad del algoritmo de cálculo de RR entre las tres topologias es irrelevante ya que el coste de la tabla de rutas es idéntica para cualquiera de ellas.

5.4 Implementaciones Hardware.

Tomando en consideración las puntualizaciones hechas en la sección anterior, es claro que el mecanismo de evitación de *deadlock,* basado en el algoritmo de la *Burbuja,* implica que todos los encaminadores necesarios en las tres redes son idénticos desde el punto de vista del coste *hardware*. Estructuralmente, la única diferencia será la aplicación del algoritmo de evitación de bloqueo tanto en el caso del *toro* como en la *Midimew*.

Con el fin de hacer una comparativa justa y sobre todo medir el impacto real de cada topología en la ejecución de las aplicaciones paralelas, el router que se ha empleado en las simulaciones es la implementación *hardware* que se propuso en el Capítulo 3. Dicho router emplea, para el caso de la *Midimew* y el toro, la *Burbuja* como mecanismo de evitación de *deadlock* y se trata de un router completamente adaptativo con dos canales virtuales por línea física. Existe un canal virtual determinista que emplea un algoritmo de routing en orden de dimensión y un canal completamente adaptativo sin ningún tipo de restricción. Un esquema de dicho router aparece en la Figura 5-6.

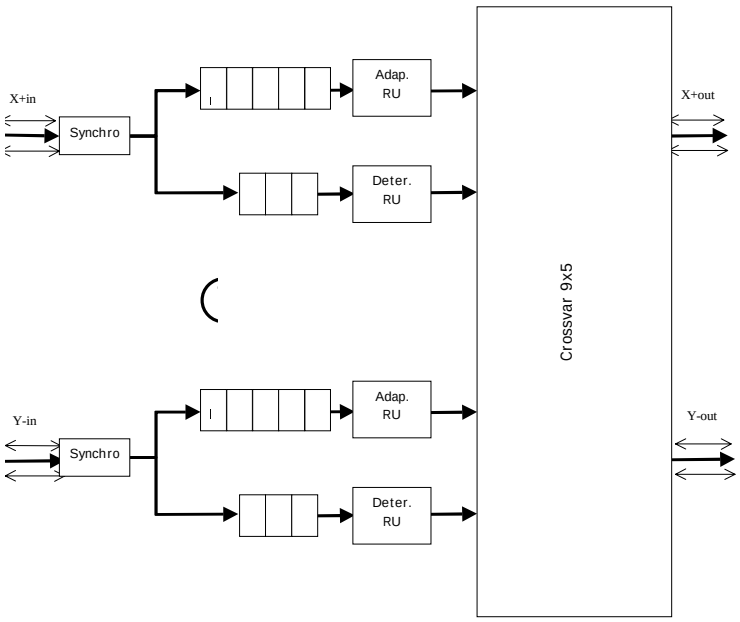


Figura 5-6. Router empleado en la comparativa de las 3 topologías evaluadas.

A diferencia del encaminador originalmente descrito en la Sección 3.4.1, en éste se utiliza encaminamiento basado en tabla. Por ello ha sido necesario eliminar los módulos encargados de comparación, decremento y almacenamiento de las cabeceras de la unidad de decisión de encaminamiento. De esta forma, la estructura de la nueva *RDU* es la mostrada en la Figura 5-7. Como se puede apreciar, dado que cada canal de entrada puede solicitar de forma simultánea paso al crossbar, hemos considerado oportuno emplear una tabla de rutas independiente por cada canal virtual. La implementación lógica de la tabla corresponde a una memoria RAM estática con puerto de lectura asíncrono. La implementación empleada proviene de la librería de componentes *Synopsys Design Ware* [124]. Con el fin de poder seguir realizando el proceso de petición en un solo ciclo, como se hacía con el encaminamiento basado en *routing-record,* ha sido preciso reducir la profundidad de la memoria. En este caso, cada destino necesita una entrada de cuatro bits para indicar los canales de salida que acercan el paquete a destino. En nuestro caso hemos dimensionado la tabla para un total de 64 nodos, por lo que, en principio, con una profundidad de 64 datos nos serviría. No obstante, pretender acceder a la memoria para extraer los datos de interés y actuar en el mismo ciclo con la petición al crossbar, resulta lento y hace que la *RDU* tenga un camino crítico muy superior al del crossbar. Para que no se degraden las prestaciones del encaminador, la solución adoptada ha sido paralelizar en grupos de 4 destinos cada entrada de la tabla. De esta forma, la tabla tendrá una profundidad de tan solo 16 bits. La parte de la palabra que nos interesa se selecciona con los dos bits menos significativos del *phit* asociado al destino. Como se podrá comprobar en puntos sucesivos, alguna de las configuraciones empleadas en la evaluación de rendimiento sobrepasan el número máximo de

nodos de la tabla. Hemos considerado oportuno manejar un única implementación *hardware* para la configuración base en lugar de emplear una diferente para cada número de nodos. De cualquier forma, la comparativa sigue siendo justa ya que este aspecto influye en los encaminadores de las tres redes de la misma forma.

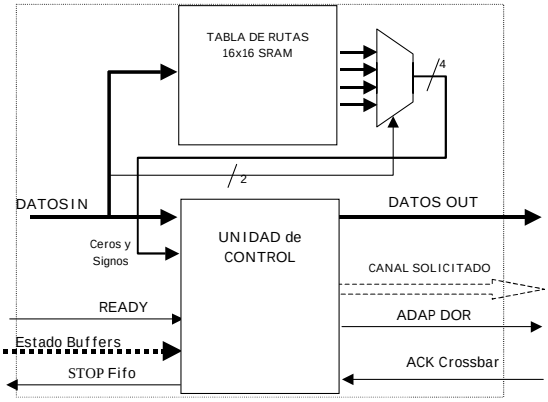


Figura 5-7. Estructura de la *RDU* con encaminamiento basado en tabla.

La estructura del pipeline de los encaminadores propuestos es el que aparece en la Figura 5-8.

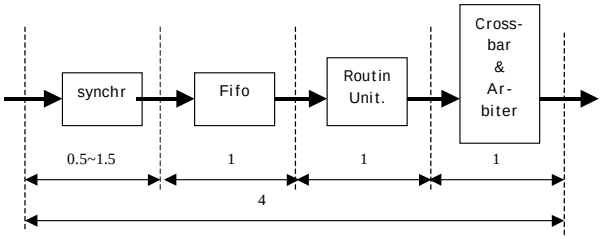


Figura 5-8. Estructura del pipeline de los encaminadores empleados.

Tanto la nueva RDU como el resto de módulos están sintetizados sobre la librería de ES2 de 0.7µm. Bajo estas condiciones y para un ancho de los canales de comunicación de 17 bits los módulos obtenidos por la herramienta de síntesis presentan las características de tiempo y área que se resumen en la Tabla 5-2

.

	Critical Path (ns.)	Area (mm^2)
Sincronizador	3.53	0.55(x5)
FIFOs Entrada Deterministas (4 paq.–80phits)	5.22	1.10(x4)
FIFOs Entrada Adaptativas (4 paq.– 80 phits)	5.22	1.10(x4)
Unidades de Routing Deterministas	5.64	2.130(x5)
Unidades de Routing Adaptativas	5.64	2.130 (x4)
Crossbar & Arbitro	**5.65**	**8.98(x1)**
Total	5.65	31.17

Tabla 5-2. Características de área y tiempo para los encaminadores.

En cuanto al coste *hardware* que introduce el mecanismo de evitación de *deadlock* basado en la *Burbuja* es nulo. La única diferencia esta en la aplicación de la limitación de inyección tanto en el caso del toro como en la *Midimew*. La aplicación del algoritmo implica la utilización de una línea adicional de ocupación desde la colas deterministas hacia la unidad de routing y una puerta AND con la línea de permiso avance que le llega a la unidad de routing desde el crossbar. Dentro de la complejidad, tanto de las colas *FIFO* como de la unidad de routing, el coste incorporado es totalmente despreciable y por ello las diferencias en las características de tiempo y área entre la malla y el resto de redes son nulas.

5.5 Análisis de Rendimiento.

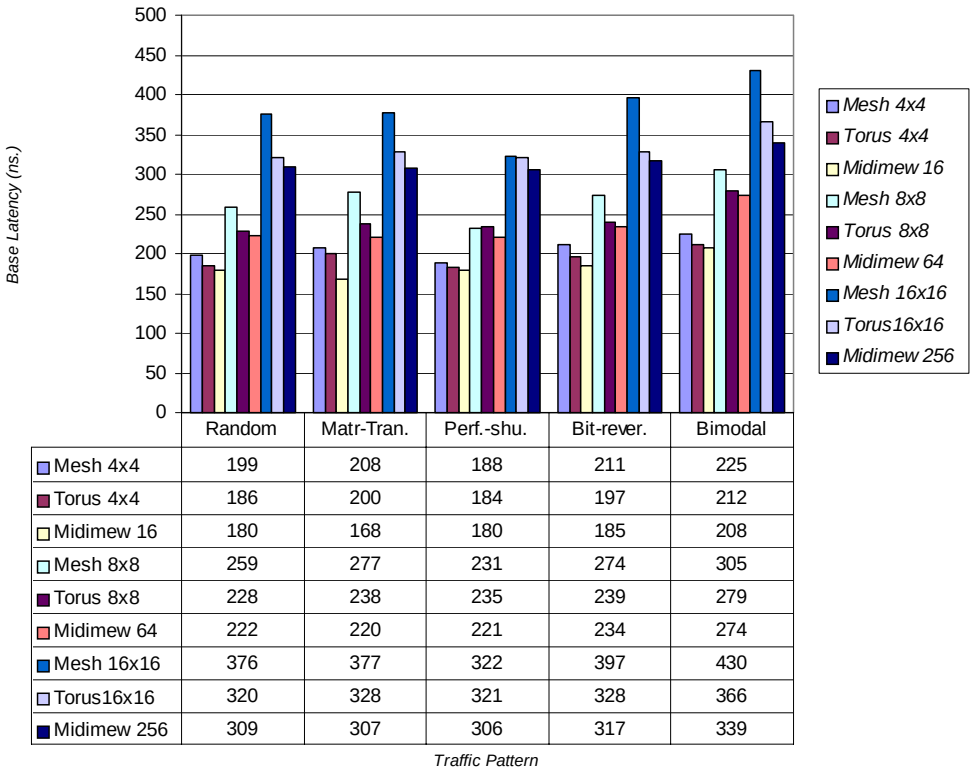
A la vista de los resultados mostrados para la implementación del encaminador, el siguiente paso es evaluar, de forma comparativa, el rendimiento ofrecido por cada una de las topologías analizadas. De forma similar a capítulos anteriores, el estudio se ha realizado bajo condiciones de tráfico sintético y tráfico real correspondiente a una arquitectura ccNUMA.

5.5.1 Cargas de Trabajo Sintéticas.

Las pruebas realizadas en este apartado incluyen el análisis de comportamiento para patrones de tráfico de tipo uniforme con longitudes de mensaje fija y otro, también uniforme, en el que las longitudes de los mensajes tienen una distribución bimodal. Por otro lado, se han considerado 3 tipos de permutaciones especificas: *matriz transpuesta*, *bit-reversal* y *perfect-shufle*. La longitud de los paquetes, en todos los casos probados, es de 20 *phits*. En el caso del tráfico bimodal los paquetes largos son de 100 *phits* (segmentados en 10 paquetes de 20 *phits*) y la probabilidad de generación, con respecto a los paquetes cortos, es de 0.1.

Los tamaños de red seleccionados para cada topología son tres: 16 nodos en configuración 4x4 para la malla y el toro, 64 nodos en configuración 8x8 para la malla y el toro y 256 en configuración 16x16 para la malla y el toro. Con el estudio de estos tres tamaños de red podremos intuir la tendencia en el comportamiento de las tres topologías al incrementar el número de nodos del sistema.

En la Figura 5-9 se muestran los datos de latencia base obtenidos para las diferentes redes y tamaños considerados. Las medidas están expresadas en nanosegundos, después de haber tenido en cuenta los resultados de la síntesis de los módulos del encaminador. Estos resultados dependen implícitamente de la características topológicas de cada red, en el sentido de que son función directa de la distancia media, aunque también son función del tipo de tráfico empleado.

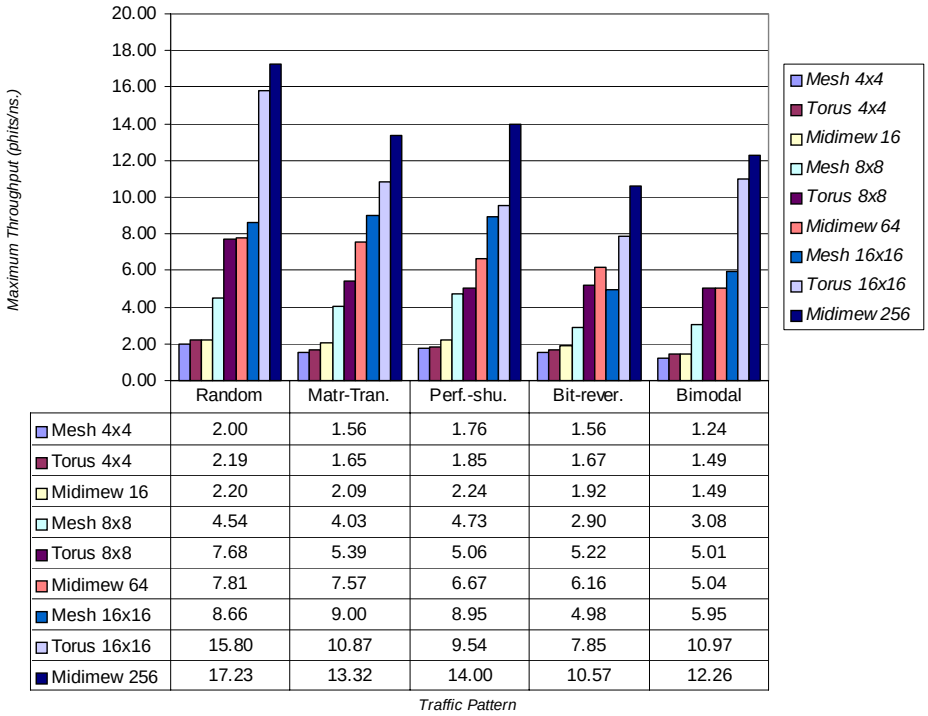


	Random	Matr-Tran.	Perf.-shu.	Bit-rever.	Bimodal
Mesh 4x4	199	208	188	211	225
Torus 4x4	186	200	184	197	212
Midimew 16	180	168	180	185	208
Mesh 8x8	259	277	231	274	305
Torus 8x8	228	238	235	239	279
Midimew 64	222	220	221	234	274
Mesh 16x16	376	377	322	397	430
Torus16x16	320	328	321	328	366
Midimew 256	309	307	306	317	339

Figura 5-9. Latencia base en nanosegundos (con una carga de 0.05% normalizada con respecto a la bisección).

Por otro lado, en la Figura 5-10 se muestra la carga máxima aceptada para cada tipo de topología y tamaño de red, expresada en *phits* consumidos por nanosegundo. Las diferencias de productividad de cada una de las redes, como es conocido, depende del tiempo promedio que se encuentran los mensajes en la red, luego dependiendo de la distancia media de cada una de las topologías, el rendimiento será diferente. Además, es necesario tener en cuenta cada patrón de

tráfico. Claramente se observa como para las *Midimew*, en determinados tipo de tráfico, se alcanzan ganancias en *throughput* considerables.



	Random	Matr-Tran.	Perf.-shu.	Bit-rever.	Bimodal
Mesh 4x4	2.00	1.56	1.76	1.56	1.24
Torus 4x4	2.19	1.65	1.85	1.67	1.49
Midimew 16	2.20	2.09	2.24	1.92	1.49
Mesh 8x8	4.54	4.03	4.73	2.90	3.08
Torus 8x8	7.68	5.39	5.06	5.22	5.01
Midimew 64	7.81	7.57	6.67	6.16	5.04
Mesh 16x16	8.66	9.00	8.95	4.98	5.95
Torus 16x16	15.80	10.87	9.54	7.85	10.97
Midimew 256	17.23	13.32	14.00	10.57	12.26

Figura 5-10. *Throughput* máximo aceptado expresado en phits consumidos por nanosegundo.

Por ultimo y con la intención de clarificar estos resultados, en las Figura 5-11, 5-12 y 5-13 se han normalizado los datos previos, sobre el caso de la topología que muestra resultados más altos, para el caso de la latencia base y sobre la que más *throughput* permite alcanzar.

Los resultados de latencia muestran como el margen de diferencias se encuentra entre un 20% y un 30% para la malla con respecto a la *Midimew* y desde un 1% hasta un 10% en el caso del *toro*. Es importante notar que, como es lógico, la *malla* tiende a exhibir una latencia considerablemente superior a las otras dos topologías al incrementar el tamaño de la red. En la misma línea, el comportamiento de la *Midimew* tiende a mejorar con respecto al toro al incrementar el número de nodos. Topológicamente, este hecho se encuentra justificado porque las diferencias de distancia promedio entre esta red y el toro se incrementan al incrementar el número de nodos de la red, como se muestra en la Figura 5-4.

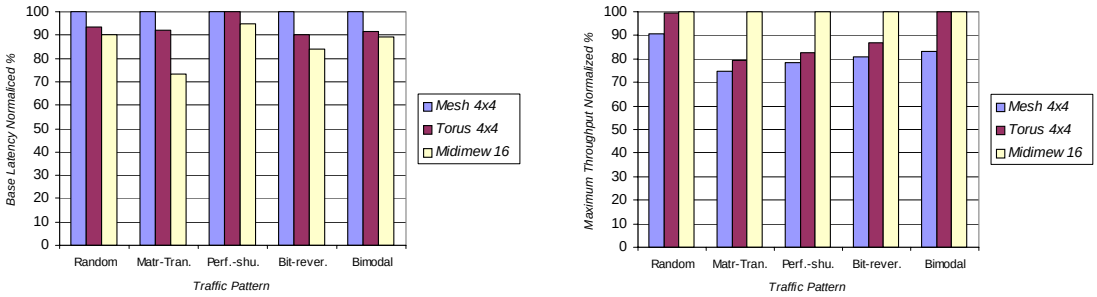


Figura 5-11. Comparativa de latencia base y carga máxima aceptada entre las tres topologías (Redes de 16 Nodos).

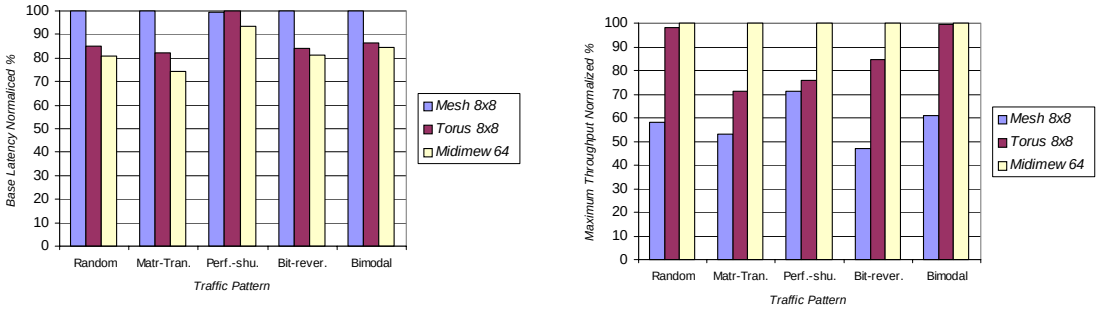


Figura 5-12. Comparativa de latencia base y carga máxima aceptada entre las tres topologías (Redes de 64 Nodos).

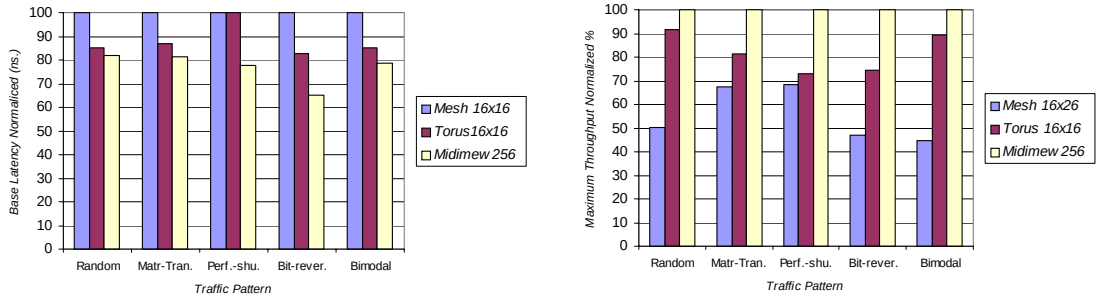


Figura 5-13. Comparativa de latencia base y carga máxima aceptada entre las tres topologías (Redes de 256 Nodos).

En cuanto a los resultados de *throughput*, los datos son, si cabe, más espectaculares que los de la latencia base. Como se puede observar, en algunos casos las diferencias entre las topologías alcanzan hasta un 60%. Claramente, las características de conectividad de la malla implican una pérdida muy importante en rendimiento con respecto a las demás topologías. Por otro lado, las diferencias de rendimiento entre el *toro* y la *Midimew* tienden a ser mínimas para el caso de los tráficos con patrón de destino uniforme para un número reducido de nodos, pero se incrementan

notablemente al pasar a patrones de tráfico no uniforme: por ejemplo, para el caso de la matriz transpuesta, las diferencias llegan a ser de hasta un 40% entre ambas topologías.

Con el fin de aclarar estos resultados, podemos observar en las figuras siguientes de qué modo se distribuye la distancia recorrida por los mensajes en cada una de las redes y para los diferentes patrones de tráfico considerados, en el caso de la redes de 256 nodos. Estas figuras nos permitirán interpretar los resultados obtenidos en throughput con cada uno de los tráficos de modo más preciso.

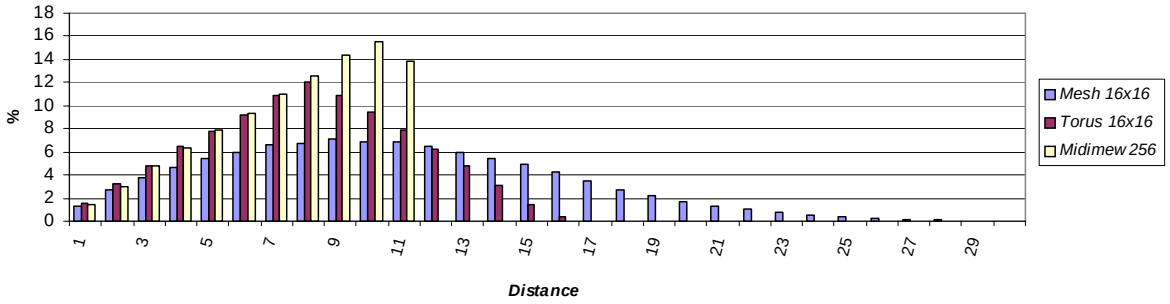


Figura 5-14. Histograma de distancias recorridas para tráfico Random.

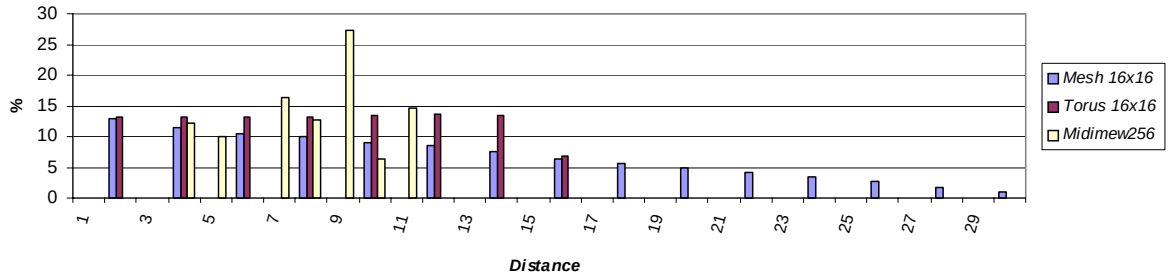


Figura 5-15. Histograma de distancias recorridas para tráfico Matriz Traspuesta.

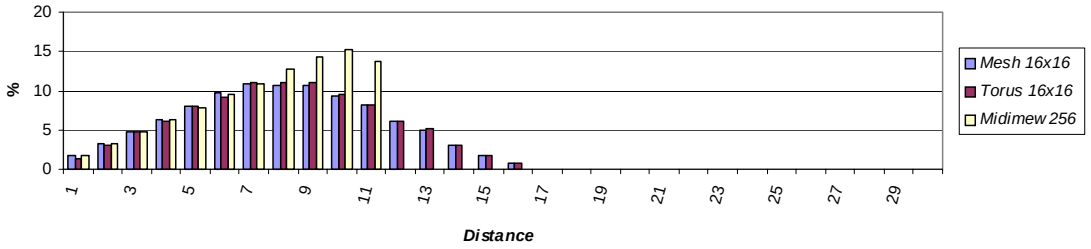


Figura 5-16. Histograma de distancias recorridas para tráfico Perfect Shuffle.

.

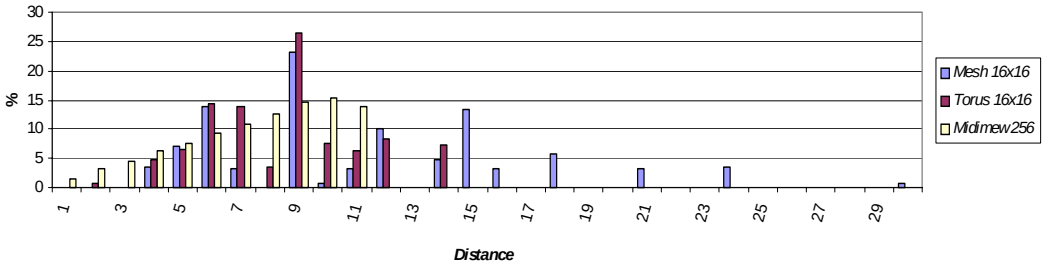


Figura 5-17. Histograma de distancias recorridas para tráfico Bit Reversal.

Se puede apreciar claramente como, en el caso de la red *Midimew*, las distancias recorridas por los mensajes se comportan de forma más uniforme que en el resto de topologías cuando los patrones de tráfico son no uniformes. Estas distribuciones permiten que la red *Midimew* muestre un comportamiento mejor. El efecto de esta distribución es una más uniforme utilización de los recursos de la red. En consecuencia, el tráfico estará distribuido más homogéneamente en la red *Midimew* que en el resto de topologías y en particular el toro. En el caso del tráfico matriz transpuesta, como se puede apreciar en la Figura 5-18, la tasa de ocupación de los bufferes adaptativos (que como sabemos es la más significativa) aparece más homogéneamente distribuida. En el toro se forman, en cada uno de los tipos de bufferes, dos diagonales escasamente utilizadas que al final producen una pérdida de rendimiento frente a la *Midimew*. Además de este hecho, la reducción de la latencia promedio y del diámetro de la red favorecen estas ganancias en productividad.

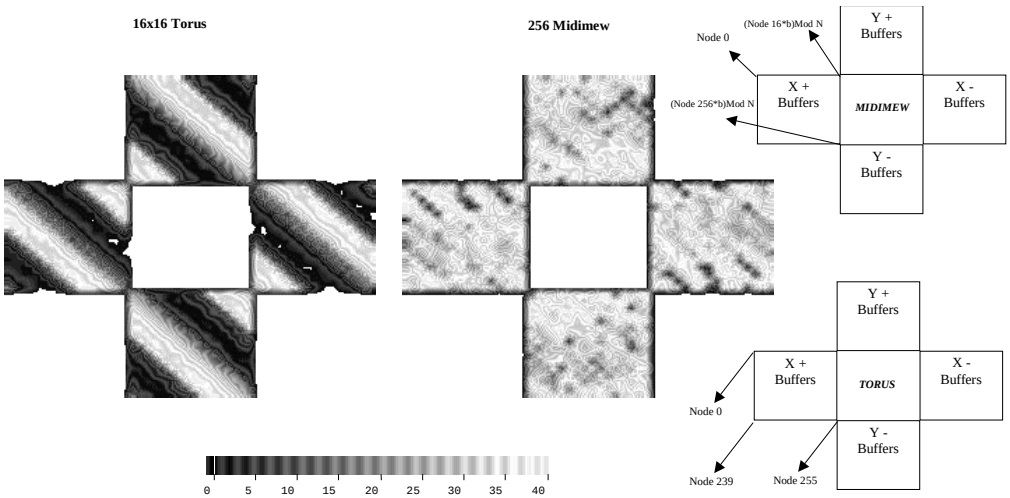


Figura 5-18. Mapas de ocupación de los bufferes adaptativos para el Toro 16x16 y la Midimew de 256 nodos con tráfico Matriz Transpuesta (expresada en *phits*).

5.5.2 Análisis de Rendimiento con Cargas Reales.

Para finalizar este estudio, utilizando ED-SICOSYS, hemos pasado a evaluar en qué modo afecta cada topología evaluada al rendimiento de las aplicaciones reales.

Como ya sabemos, el sistema emula el comportamiento de una arquitectura DSM con coherencia hardware ccNUMA. En cuanto al banco de pruebas seleccionado, seguiremos empleando las aplicaciones descritas en la Sección 2.3.3.1. Estas aplicaciones son RADIX, FFT y LU siendo ejecutadas, en principio, sobre 64 procesadores, es decir, sobre una *malla* 8x8, un *toro* 8x8 y una *Midimew* de 64 nodos, $C_{64}(6,5)$. En todas las aplicaciones se emplea el tamaño de problema por defecto de la aplicación, excepto en Radix donde para esta comparativa se ha considerado oportuno reducir el tamaño del problema a la mitad.

El sistema simulado corresponde a una ccNUMA con los parámetros del sistema superior (tamaño de las caches, protocolo de coherencia, arquitectura del procesador, etc...) descritos en la Tabla 3-9 en la página 126. Seguiremos empleando líneas de cache de 32 *bytes* y comandos de 8 *bytes*. Por lo tanto, el tamaño de los mensajes de datos es de 40 *bytes* y dado que la anchura del canal empleado en los encaminadores es de 2 bytes, la longitud de los paquetes es de 20 *phits*. De igual forma, el tamaño de los mensajes de comandos e invalidación es de 4 *phits*. La velocidad del procesador sigue siendo 650 MHz. De acuerdo con esto, la velocidad de la red se supone no dependiente de los enlaces, y es marcada exclusivamente por el encaminador. Como se ha comentado previamente, los costes *hardware* del encaminador, para cada tipo de red, son idénticos y como se muestra en la Tabla 5-2 su tiempo de ciclo en condiciones típicas de temperatura y voltaje es *5.65 ns.* Esto implica que la red puede llegar a operar hasta con un reloj de 177 MHz. Por lo tanto, la velocidad relativa entre la red y procesador es de 3.67. En lo que respecta a la evitación de *deadlock* causado por la capacidad limitada de los canales de consumo, se ha planteado el uso una única configuración en la que las redes de datos y peticiones están físicamente aisladas.

5.5.3 Distribución de Datos. Localidad espacial en la *Midimew.*

Antes de pasar a evaluar el rendimiento del sistema sobre cada tipo de topología para las cargas de trabajo propuestas, hemos analizado el comportamiento de las aplicaciones sobre cada una de las topologías de forma cualitativa. En este análisis previo se estudia cómo es el patrón de destinos de cada aplicación y se introduce un algoritmo de asignación o distribución de recursos para la *Midimew*. Este sistema permitirá evitar algunas suposiciones hechas en las aplicaciones y que al no verificarse en esta red, pueden llegar a perjudicar de forma grave el rendimiento del sistema.

Al emplear nuestro simulador conducido por ejecución, y compilar cada aplicación es necesario distribuir las variables compartidas por los diferentes procesos a ejecutar de acuerdo con el número de procesadores y la topología de la red de interconexión[97]. Las aplicaciones están pensadas para explotar, en cierta medida, las ventajas de localidad espacial entre nodos, realizando por ejemplo muchas tareas de sincronización siempre entre vecinos. Originalmente las aplicaciones suponen el empleo de redes en las que los nodos guardan una relación de adyacencia similar a la malla. Bajo estas circunstancias se supone siempre que el identificador de cada proceso es igual a el número del nodo sobre el que está corriendo. De esta forma, la distribución de datos es realizada de forma inmediata ya que la adyacencia entre identificadores y nodos de la red es la misma, lo que permite que, desde el punto de vista de la aplicación, resulte sencilla la distribución de las variables a lo largo de todo el sistema.

Sin embargo, la asignación o mapeo de variables cuando la topología empleada para la red de interconexión es la *Midimew*, es necesario modificarla teniendo en cuenta el etiquetado y localidad de esta topología. En este caso, la adyacencia al nivel de la aplicación no implica la adyacencia en el nivel de la red ya que procesos a distancia uno “a nivel de aplicación” pueden encontrarse corriendo en nodos situados a una distancia mucho mayor. Frente a este problema se puede optar por dos soluciones alternativas:

Redistribución de Variables Compartidas.

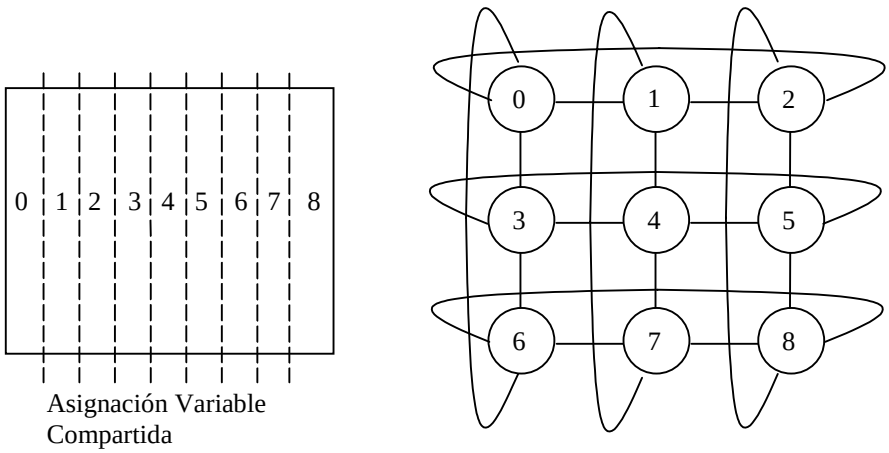


Figura 5-19. Asignación de una matriz compartida por columnas en Toro y Malla. Ejemplo de 9 procesos sobre un Toro 3x3 o Malla 3x3.

La primera solución consiste en redistribuir de forma correcta las variables compartidas. En las figuras siguientes se muestra un ejemplo para un matriz compartida distribuida por columnas en el caso de 9 procesos sobre una malla, un toro (Figura 5-19) y una *Midimew* (Figura 5-20). Por ejemplo, la FFT emplea variables compartidas de este tipo.

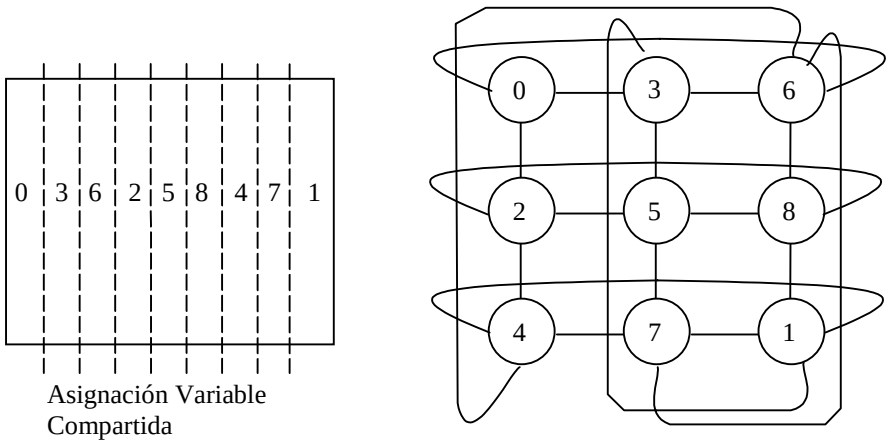


Figura 5-20. Asignación de una matriz compartida por columnas en Midimew. Ejemplo 9 procesos

La asignación lograda mediante este algoritmo es óptima en el sentido de que no se introducen incrementos en la distancia media de los mensajes generados por la aplicación, manteniéndose ésta siempre por debajo de la distancia media topológica de la *Midimew*. En el caso de hacer una distribución similar al toro o la malla, la distancia media recorrida por los mensajes supera, en la mayoría de los casos, la distancia media topológica. Además de la correcta distribución de los datos, es posible, desde el punto de vista algorítmico, proponer modificaciones en las propias aplicaciones que permitan explotar aún mejor las características intrínsecas de las *Midimew,* pero ese tipo de aspectos esta fuera del ámbito de este trabajo.

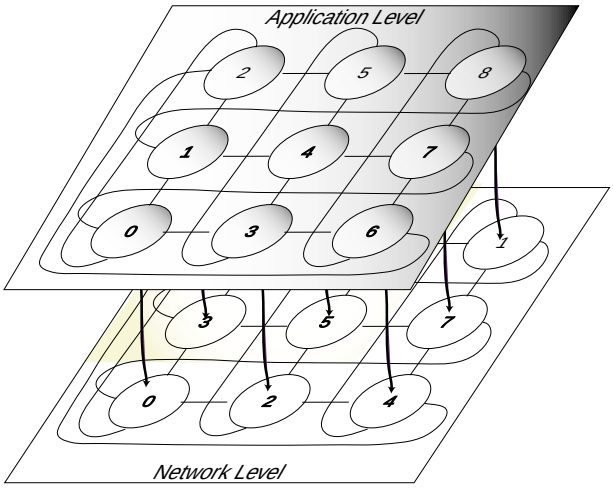
Reasignación de procesos.

Si bien es posible redistribuir los datos, en el caso de algunas aplicaciones la tarea puede llegar a complicarse considerablemente. Un ejemplo de este tipo de casos es Radix: en ésta es necesario redistribuir las claves a ordenar (de forma similar a las matrices manejadas en la FFT) y las variables globales que mantienen las pausas. Estas variables están íntimamente ligadas al algoritmo en sí, lo que se traduce en que no es inmediata la redistribución correcta de la variables asociadas a las pausas.

Sin embargo, existe una solución mucho más simple que la previamente mencionada. Consiste en hacer que el sistema superior, y por tanto la aplicación, vea un sistema de numeración consecutivo entre nodos. Por tanto, estas etiquetas serán distintas a las que se ven desde el nivel de la red. Recordar que, en el nivel de red, la numeración de los nodos es imprescindible a la hora de establecer el *Routing-Record* de cada mensaje. De esta forma, no es necesario, en absoluto, tocar la aplicación sino únicamente alterar las tablas de encaminamiento del encaminador de tal forma que ambas numeraciones coincidan. En el caso de no emplear encaminamiento basado

en tabla, la solución pasa por convertir cada origen y destino de los mensajes generados por la aplicación al correspondiente al nivel de la red.

En la Figura 5-21 se puede apreciar la relación existente entre el nivel de red y el nivel de aplicación, en lo que respecta a la etiquetación de nodos y procesos, para el caso de 9 procesos y nodos. En la misma figura se muestra el algoritmo que permite relacionar ambos niveles para un número arbitrario de nodos. Dado que la solución es siempre válida y sencilla de implementar, optaremos por emplear ésta en lugar de la previa. Notar que el coste es nulo ya que no es necesario realizar ningún cambio, ni en la aplicación ni el sistema, para llevarla a cabo.




```
int MidimewIndex(int NetworkIndex){
         int  b := ceill(sqrt(N/2));
         int  rest:=b;
         int  counter:=0;
         int  number;
        while(rest>0){
           number:=(b-1)*(b-rest);
           while(number<N){
                 if(NetworkIndex==counter)
                    ApplicationIndex:= number;
                    return ApplicationIndex;
                 end if;
                 counter++;
                 number=number+b;
            }
            number=rest;
            while(number<(b-1)*(b-rest)){
             if(NetworkIndex==counter){
                    ApplicationIndex:= number;
                    return ApplicationIndex;}
                 counter++;
                 number=number+b;
             }
          rest=rest-1;
        }
```

Figura 5-21. Relación entra la numeración de nodos en el nivel de red y en el nivel de aplicación.

Con el fin de comprobar que la asignación de recursos efectuada es correcta, pasaremos a analizar de modo cualitativo cómo es el histograma de distancias recorridas para cada aplicación en el caso de todas las topologías. En este sentido las características del tráfico generado para el caso de 64 procesadores se muestra en las figuras siguientes:

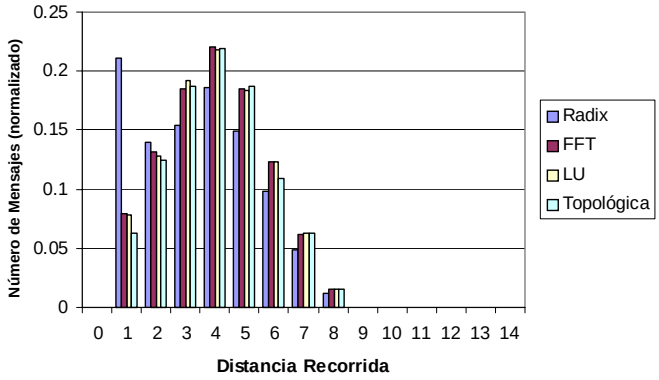


Figura 5-22. Histograma de distancias recorridas para cada aplicación en un Toro 8x8 (64 Procesadores)

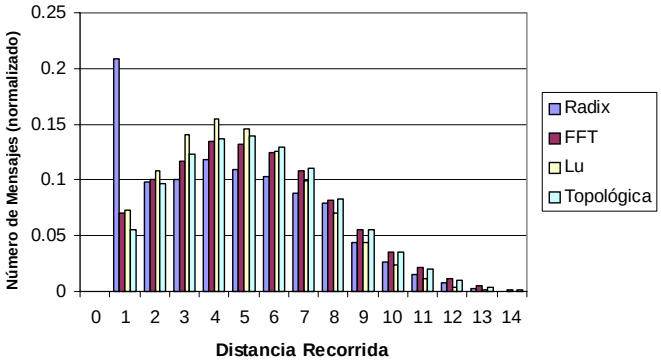


Figura 5-23. Histograma de distancias recorridas para cada aplicación en una Malla 8x8 (64 Procesadores).

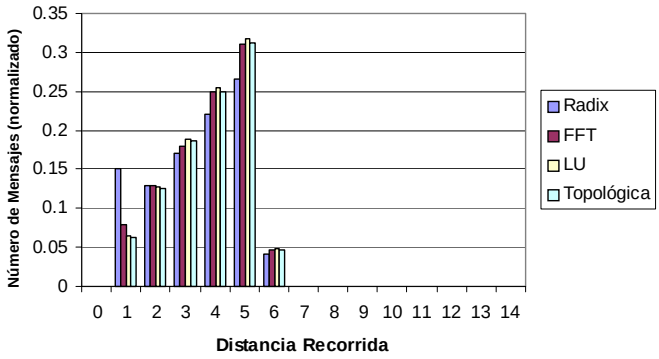


Figura 5-24. Histograma de distancias recorridas para cada aplicación en una Midimew 64 (64 Procesadores).

La serie etiquetada como *topológica* que aparece en las figuras, corresponde, en cada caso, al histograma de distancias que da lugar a la distancia promedio topológica de cada red. En otras palabras, correspondería a las distancias recorridas por los paquetes en el caso de un patrón de tráfico uniforme. En cualquier caso, el porcentaje de mensajes que viaja a distancias próximas al diámetro de la red se mantiene siempre acotado por debajo del límite topológico. Tomando en consideración estos resultados, los datos de distancia media recorrida por los mensajes, para cada una de las aplicaciones y con la distribución de datos definitiva, son los mostrados en la Tabla 5-3.

	d. m. topológica	***Radix***	***FFT***	***LU***
Malla 8x8	5.33	4.55	5.29	4.88
Toro 8x8	4.06	3.47	3.99	4.00
Midimew 64	3.87	3.45	3.74	3.76

Tabla 5-3. Distancias promedio recorridas por los mensajes para cada aplicación y topología.

Como se puede observar, el número de saltos promedio que han de recorrer los mensajes en cada caso oscila considerablemente entre las tres topologías. Este hecho es más notable especialmente al comparar la malla con las otras dos topologías. Por otro lado, las diferencias entre el toro y la *Midimew* oscilan entre un 8% para el caso de la FFT y menos de 1% para el caso de Radix. Por ello es lógico que las diferencias, en lo que respecta a tiempo de ejecución, seán más notables en el caso de FFT y prácticamente despreciables para el caso de Radix. La razón fundamental para que Radix presente esas características es que gran parte del tráfico está relacionado con el acceso a las pausas de sincronización entre procesos. El acceso es tal que cada nodo accede únicamente a la pausa que mantiene en su *home* uno de sus vecinos. Este hecho se puede constatar en los histogramas de distancias recorridas para esta aplicación en todas las topologías.

5.5.4 Evaluación de Prestaciones.

En esta primera parte del análisis nos centraremos en evaluar el comportamiento de cada aplicación y red para un número fijo de procesadores. En principio, el número de procesadores considerado es 64. Los resultados alcanzados son los que aparecen en la Figura 5-25 mostrándose un detalle de estos datos en la Figura 5-26.

Es claro como los resultados obtenidos con tráfico sintético se corresponden con los mostrados en la figura anterior, especialmente al comparar el rendimiento alcanzado por la malla y las demás topologías. Por otro lado, se observan ganancias de la *Midimew* con respecto al toro entre un 4% para el caso de FFT y de un 2% para las otras dos aplicaciones. Sin duda, las ganancias

son discretas, pero es necesario remarcar que no se ha introducido ninguna modificación en el código de las aplicaciones a la hora de efectuar el análisis. Es previsible que, de emplear algoritmos desarrollados específicamente para este tipo de red, las diferencias en tiempos de ejecución sean superiores para estos tamaños.

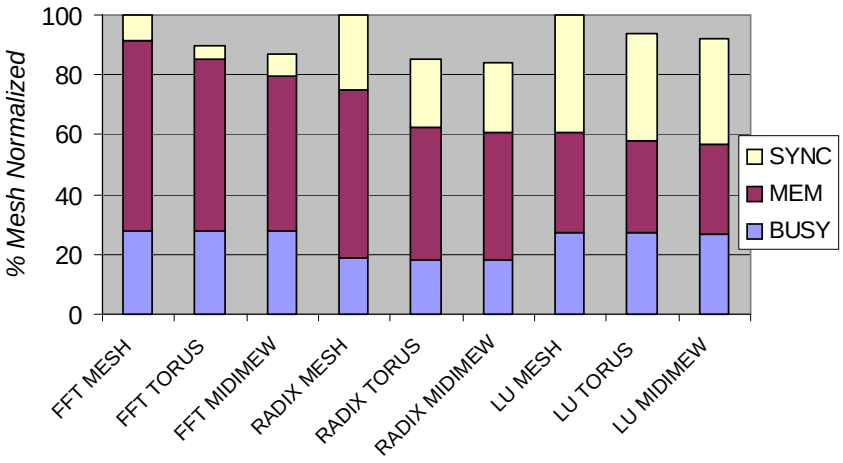


Figura 5-25. Tiempo de ejecución normalizado a la malla.

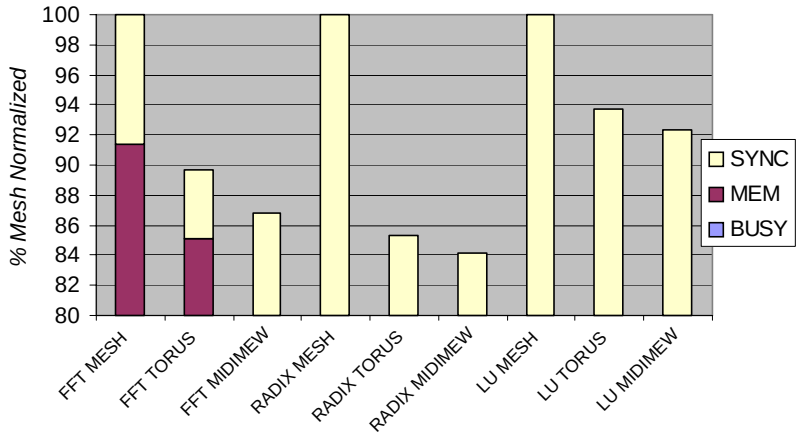


Figura 5-26. Detalle del tiempo de ejecución.

5.5.4.1 Dependencia con el Tamaño de la Red: Caso de FFT.

De forma similar que en el caso del tráfico sintético resulta interesante determinar cuales son las prestaciones de cada red de interconexión al variar el número de procesos. De esta forma podremos establecer como escala esta aplicación concreta para cada una de las redes consideradas. Dada la no escalabilidad de Radix y LU por encima de 64 procesadores, se ha realizado la comparativa exclusivamente con FFT. El estudio se centra en 4 configuraciones distintas de

red: 16 nodos, 32 nodos (*Toro y Malla, 8x4)*, 64 nodos *(Toro y Malla, 8x8)* y 128 nodos *(Toro y Malla, 16x8)*. La configuración del sistema, en lo que respecta a jerarquía de memoria, procesador y velocidad de la red, es idéntica a la empleada en el punto anterior. Los resultados de la Figura 5-27 muestran cómo las diferencias son coherentes con las obtenidas para tráfico sintético.

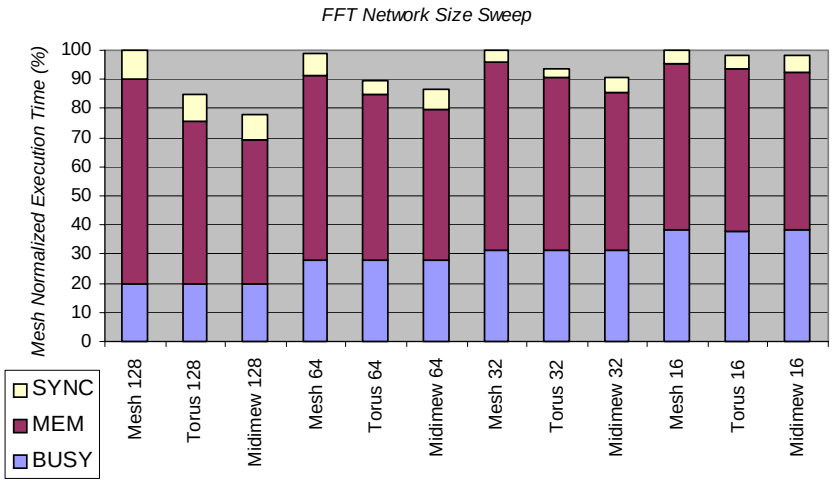


Figura 5-27. Evolución del tiempo de ejecución de FFT al variar el número de nodos de la red.

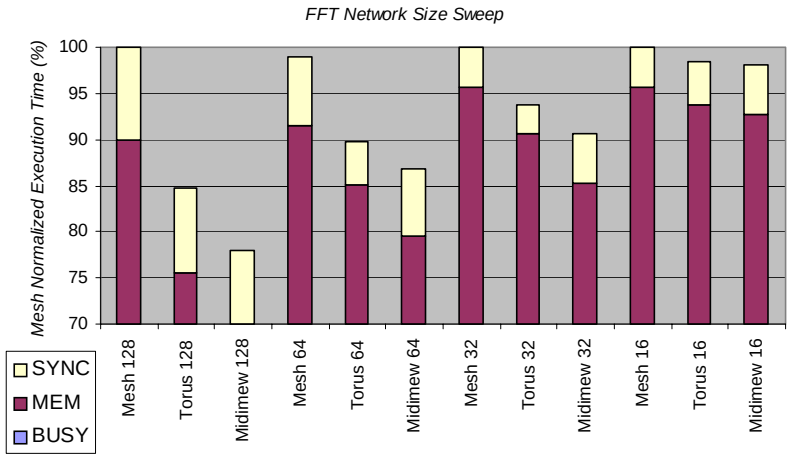


Figura 5-28. Detalle del tiempo de ejecución.

Las diferencias en el rendimiento alcanzado por la aplicación con cada red tienden a incrementarse claramente al elevar el número de elementos de proceso. Especialmente interesante es el comportamiento de la *Midimew*. El caso de 128 procesadores muestra diferencias con respecto al toro de aproximadamente un 9% y de casi un 25% con respecto a la malla. Claramente se

observan diferencias más acusadas a medida que el número de procesadores del sistema crece. Sin duda este comportamiento permite concluir que los sistemas con un número elevado de nodos sacarán mayor partido de las propiedades topológicas de la red *Midimew*. Por lo tanto, es claro que las características teóricas de la red son extrapolables tanto al campo de las prestaciones con cargas sintéticas como al campo de las aplicaciones reales.

5.5.4.2 Otro Tipo de Distribución de Datos: Caso FFT.

Los resultados previos corresponden a un re-mapeo en la *Midimew* aplicando el algoritmo descrito en la Figura 5-21. Como se ha comentado previamente, el empleo de ese tipo de distribución se ha elegido para intentar mantener una secuencia en la numeración de los nodos similar a la malla ó el toro. Sin embargo, es claro que pueden existir otras distribuciones no triviales que permitan explotar mejor las características topológicas de la *Midimew.*

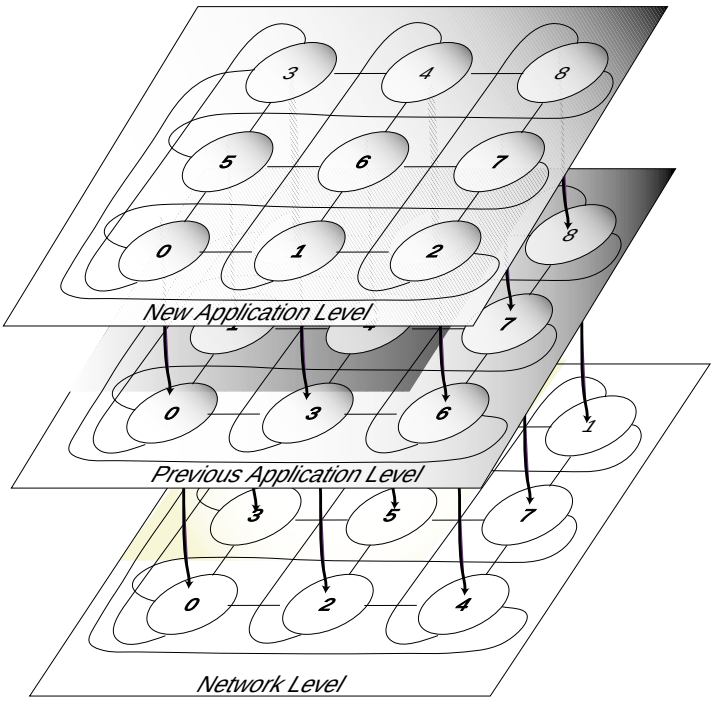


Figura 5-29. Doble reasignación de nodos para *FFT*

En este sentido se ha elegido una nueva distribución de procesos en el caso en la *FFT,* evaluando cuál ha sido el comportamiento de la aplicación en este caso. El nuevo sistema de distribución hace que no se conserven ni las adyacencias originales de la *Midimew* ni de la malla o el toro. El algoritmo es el equivalente a realizar dos veces la conversión descrita en la Figura 5-21. Por ejemplo, para el caso de una red de 9 nodos la asignación definitiva es la mostrada en la Figura 5-29.

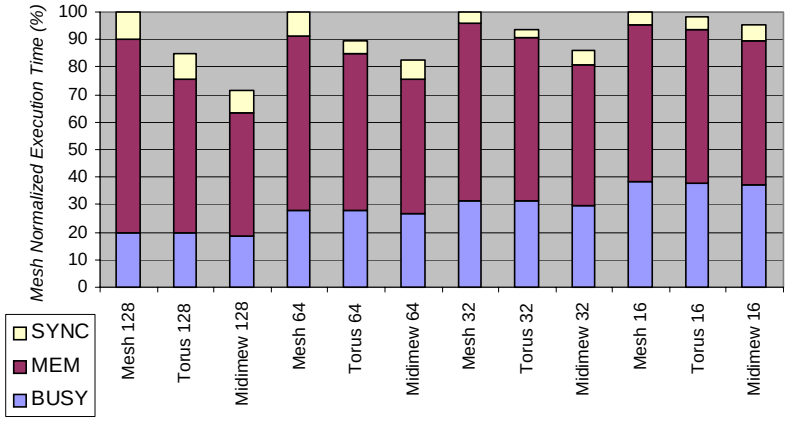


Figura 5-30. Resultados de ejecución con la nueva etiquetación de nodos.

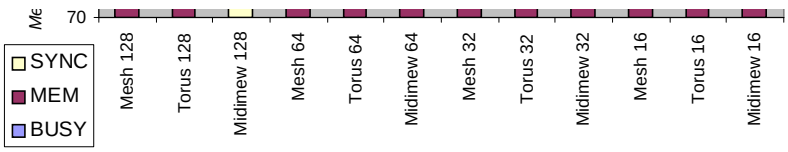


Figura 5-31. Resultados de ejecución con la nueva etiquetación de nodos. Detalle.

Los resultados alcanzados con esta nueva distribución de procesos muestran una ganancia considerable con respecto a la asignación empleada previamente (las diferencias crecen hasta un 16% con respecto al toro y casi un 30% con respecto a la malla). Aparentemente el patrón de comunicación de la aplicación (comunicaciones *all-to-all) se* beneficia considerablemente del nuevo criterio de etiquetación. Las razones de este comportamiento son realmente complejas y dependen mucho del algoritmo que está ejecutando la aplicación. Este problema de distribución de recursos puede llegar a ser difícil de resolver. Es importante señalar que la nueva disposición de los procesos es similar a algunas empleadas para esta aplicación en redes tipo toro como la descrita en [27] o en [42]. El encontrar las razones definitivas queda fuera del ámbito de este trabajo aunque el comportamiento en este caso particular deja claro que es posible optimizar las

aplicaciones de forma que logren explotar eficientemente las características de conectividad de la red, logrando una notable mejoría en el tiempo de ejecución.

5.6 Conclusiones

La red *Midimew* es una red simétrica 2-D óptima, con una menor distancia media y diámetro que el toro bidimensional. Por lo tanto, esta topología es una buena candidata para implementar el subsistema de comunicación de un sistema multiprocesador. La reducción en la distancia topológica juega un papel importante a la hora de mejorar el rendimiento de la red de interconexión desde dos puntos de vista: la latencia de los mensajes se reduce y el *throughput* se ve notablemente incrementado.

Frente a los problemas prácticos que presentaba la *Midimew*, combinando encaminamiento basado en tabla y el control de flujo *Burbuja*, hemos mostrado que es factible emplear un encaminador adaptativo que resulta completamente idéntico al empleado en el caso del toro. Así, podemos explotar las ventajas topológicas de la *Midimew* donde el único cambio introducido con respecto al toro se reduce a cambiar la disposición de los enlaces periféricos. De esta forma, sin incrementar, en modo alguno, el coste de la red logramos mejorar el rendimiento del sistema global.

La evaluación efectuada para las tres redes analizadas, tanto mediante el uso de cargas de trabajo sintéticas como bajo aplicaciones reales sobre un sistema ccNUMA, ha confirmado la superioridad de la *Midimew* sobre otras redes de bajo número de dimensiones como la malla y el toro. Por otro lado, se aprecia como la malla exhibe un comportamiento muy pobre con respecto tanto al toro como, por supuesto, a la *Midimew*.

Además, es necesario reseñar que las diferencias entre las tres redes se incrementan a medida que crece el tamaño del sistema. En este sentido, es claro observar como la red *Midimew* logra mejorar el rendimiento cuando el tamaño del sistema es mayor. Por tanto, este tipo de redes puede ser un buen candidato a emplear en computadores masivamente paralelos como los enmarcados dentro del proyecto *ASCI*.