Improving Parallel System Performance by Changing the Arrangement of the Network Links*

V. Puente, C. Izu†, J.A. Gregorio, R. Beivide, J. M. Prellezo and F. Vallejo

University of Cantabria
39005 Santander, Spain
{vpuente,ja,mon,prellezo,fernando}@atc.unican.es

†University of Adelaide
SA 5005 Australia
cruz@cs.adelaide.edu.au

Abstract

The Midimew network is an excellent contender for implementing the communication subsystem of a high performance computer. This network is an optimal 2D topology in the sense there are no other symmetric direct networks of degree 4 with a lower average distance or diameter. In fact, it reduces the diameter of the well known torus network by approximately $\sqrt{2}$. Although the topology was proposed and analyzed a decade ago, the lack of simple deadlock avoidance mechanisms prevented its utilization up to date. This study solved this drawback by applying the Bubble switching mechanism, a low cost deadlock-avoidance strategy developed by the authors. Moreover, by using routing tables we can configure our Virtual Cut-Through adaptive router to implement either a torus or a Midimew network. Thus, we can exploit the topological advantages of Midimew networks by simply changing the disposition of the wrap-around connections of its torus counterpart, without increasing the network implementation cost. To prove this assertion, we have carried out a thorough evaluation, from the hardware cost of the router to the parallel system performance under real loads.

1. Introduction

The topology is one of the key design factors of an interconnection network. There is a large body of theoretical research on optimal topologies, based on graph theory metrics such as average distance, network diameter, and bisection bandwidth, among others. These parameters have a direct impact on network performance. For example, average distance is reflected on the average network latency, network diameter determines the largest base latency, and bisection bandwidth reflects the communication capacity between any two halves of the network when dealing with random or non-local traffic.

Other implementation factors such as wiring density and chip pin-outs should be also taken into account when searching for an optimal topology. In particular, a milestone performance study [5] showed that lower dimensional networks, such as 2D and 3D tori, outperform higher dimensional ones such as the hypercube under constant wiring density constrains.

* *This work is supported in part by TIC98-1162-C02-01*

The Midimew (MInimal DIstance Mesh with Wrap-around links) [2] is an optimal topology in the sense that there is no direct symmetric network of degree 4 with lower average distance or diameter. Consequently, the Midimew seems an optimal candidate for building interconnection subsystems, provided that the complexity of its hardware implementation does not overtake the topological gains. However, the Midimew is not deadlock-free even for DOR (Dimension Order Routing) routing and up to now, there was not an easy mechanism to prevent deadlock for this topology.

In this paper we resolve this problem by adapting the mechanism recently proposed in [15]. Firstly, we provide the first hardware description for a Midimew router. Furthermore, by using routing tables, a viable solution under current VLSI technology, we provide a Virtual Cut-Through (VCT) router which can be configured to implement the most common planar topologies: mesh and torus, as well as the Midimew. In other words, we can rearrange the wrap-around connections of the torus to implement a Midimew at will. Obviously, the latter will be our preferred choice so that parallel applications can take advantage of the characteristics of this optimal topology.

Secondly, we used a thorough methodology to evaluate the advantages of choosing the Midimew interconnect, from the hardware cost of the router to the parallel system performance under real loads. The results from this evaluation will prove the practical benefits of the Midimew topology, when used to build the interconnection network of a cc-NUMA system. This work will show that the Midimew network improves the system performance when compared to that using a torus network, without added costs.

The rest of the paper is organized as follows. Section 2 reviews the characteristics of the Midimew network. Section 3 introduces our solution to deadlock in Midimew networks. Section 4 describes its router implementation. Section 5 details the evaluation of this topology under a cc-NUMA platform running parallel application benchmarks and discusses the results. Finally, in Section 6 we draw some conclusions.

2. The Midimew Network

The Minimal Distance Mesh with Wrap-around links, or Midimew for short, belongs to the family of circulant graphs of degree 4, $C_N(a,b)$, in which every node i in the graph is connected

to nodes *(i ± a)Mod N* and *(i ± b)Mod N*, being *N* the size of the network. The Midimew corresponds to the family:

$$C_N\left(\left\lceil\sqrt{\frac{N}{2}}\right\rceil-1,\left\lceil\sqrt{\frac{N}{2}}\right\rceil\right)$$

This family has the lowest diameter and average distance of any regular and symmetric degree-4 graph [2]. (With $\lceil\bullet\rceil$ representing the ceiling function).

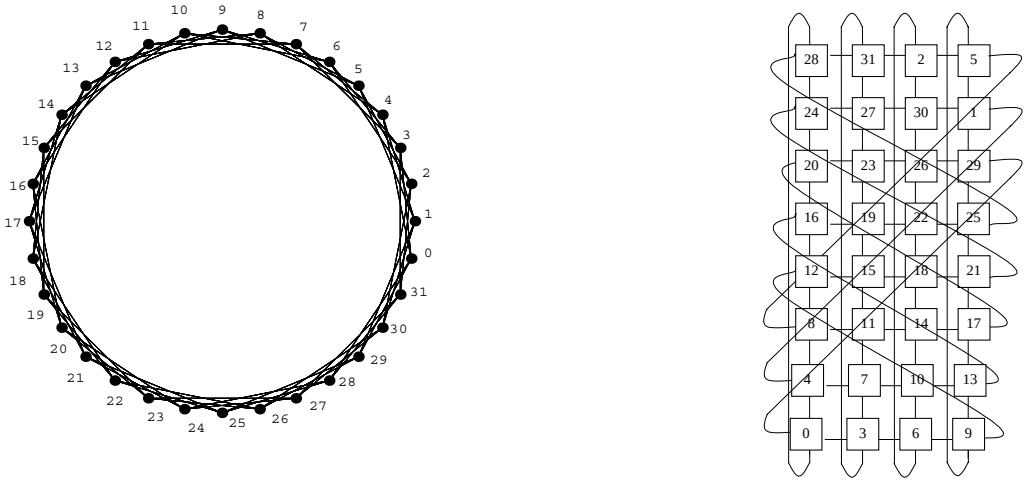
This circulant graph can be transformed into a mesh-connected network with wrap-around links as shown in Figure 1. This transformation provides good embeddability for parallel applications and a planar representation with a minimum number of crossing links. The figure shows a rectangular Midimew, in which the network size ***N*** must be a multiple of either *a* (which is *b-1*) or *b*. An algebraic methodology for the construction of Midimew networks of any size can be consulted in [2].

2.1-Topological Comparison

We compare the Midimew with other planar topologies such as the 2D mesh and the 2D torus. Their topological properties are shown in Table 1. For an exhaustive comparison, please refer to [2].

For large networks, the diameter ratio between the torus and the Midimew is approximately the square root of 2. In other words, by only changing the disposition of the wrap-around connections the diameter is reduced up to 30% in relation to the torus network. The average distance is also reduced, although not as dramatically. For example, in a 256-node network the average distance decreases from 8.3 in the torus to 7.51 in the Midimew.

The presence of wrap-around connections in both the 2D torus and the Midimew tangles its mapping into a 2D layout. Solutions to the Midimew case have been proposed having a minor impact on cost and channel delay [8].



(a) **(b)**

Figure 1. (a) Circulant graph $C_{32}(4,3)$ and (b) its transformation into a mesh with wrap-around links.

Topology	***Number of links***	***Diameter***	***Diameter($k_1=k_2=k=\sqrt{N}$)***	***Average Distance ($k_1=k_2=k=\sqrt{N}$)***
Mesh $(k_1 \times k_2)=N$	$2k_1k_2-k_1-k_2=$ $2\sqrt{N}(\sqrt{N}-1)$	k_1+k_2-2	*2k-2*	$\frac{2k}{3}$
Torus $(k_1 \times k_2)=N$	$2k_1k_2=2N$	$\left\lceil\frac{k_1}{2}\right\rceil+\left\lceil\frac{k_2}{2}\right\rceil$	k	$2\left\lfloor\frac{k}{2}\right\rfloor\left\lfloor\frac{k+1}{2}\right\rfloor\frac{k}{k^2-1}$
Midimew of size N $b=\left\lceil\sqrt{\frac{N}{2}}\right\rceil$	$2N$	b or $b-1$	$k_0=\left\lceil\frac{k}{\sqrt{2}}\right\rceil$	$k_0\left(1-\frac{2\left(k_0^{\,2}-1\right)}{3(N-1)}\right)$

Table 1. Comparison of planar topologies ($\lceil\bullet\rceil$ ceiling function and $\lfloor\bullet\rfloor$ floor function).

In terms of scalability, all networks can be easily scaled up by adding additional rows or columns. In the case of the torus and Midimew such addition only requires a change on the wrap-around connections. Moreover, in the case of the Midimew it can be possible to add *any* number of nodes, being this a clear advantage over meshes and tori.

2.2 Routing in Midimew Networks.

In a planar network, we can express the path from a given source to its destination as a pair *(Δx, Δy)* of signed displacements which represent the number of hops required in each dimension to reach the destination node. The pair *(Δx, Δy)* is usually called the Routing Record (RR) and it constitutes the packet header. When the packet arrives at a router, the header is examined in order to select the next output channel, and then the router will update the RR to reflect the current distance to its destination. A packet with a header value (0,0) has completed its path, thus being at the destination node.

(a) N-node Midimew

```
b := ceill(sqrt(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)
      Δy := y0;
      Δx := x0;
else
       Δy := y1;
       Δx := x1;
end if;
```

(b) (k_1 x k_2)Mesh

```
Δx:=(source mod k1)-(dest mod k2);
Δy:=(source div k2)-(dest div k2);
```

(c) (k_1 x k_2) Torus

```
   Δx   := (source mod k2) -
(dest mod k2);
   If (Δx > k1/2)
         Δx := k1 - Δx;
   else if (Δx < -k1/2)
         Δx := k1  + Δx;
   end if;
   Δy   := (source div k1) -
(dest div k2);
   If (Δy > k2/2)
         Δy := k2 - Δy;
   else if (Δy < -k2/2)
         Δy := k2  + Δy;
   end if;
```

Table 2. Routing record calculation for mesh, torus and Midimew.

Table 2 shows the algorithm that derives these displacements from the values *source* and *dest* for the midimew, mesh and torus topologies. The algorithm for the Midimew relies on the node enumeration of the circulant graph as shown in Figure 1. The other two rely on the network enumeration in which the node at row *i* and column *j* is labeled with the number *(i+ k_2*j)*.

This algorithm must be implemented in hardware in order to provide a fast packet generation. Thus, the complexity of the algorithm may increase the time needed to compose the header. The operations *mod* and *div*, if the divisor is not a power of 2, represent the higher cost (tens of cycles) compared to the basic operations such as add, subtract or compare (one cycle each). The values k_1, k_2, *b* and *source* are fixed when the network is configured, thus some calculations are not required for each new packet, but all need to execute one *mod* and one *div* operations. Thus, although the Midimew has the longest RR algorithm, the time differences between the three topologies are not significant. However, if the system enumerates the nodes in a different way, the complexity of calculating the RR will change.

Alternatively to the RR method, each router can use a routing table, which has an entry for each destination node in the network listing the output channel(s) that will get the message closer to its destination. In this case, the destination node, contained in the header, is used to index the routing table and obtain the output. This method presents an uniform routing cost for any network type; thus, we have chosen it in order to provide a fair comparison among the three topologies.

We need to initialize each node's routing table when the network starts up. Although this requires additional router-specific messages or some other mechanism for establishing the contents of the routing table, this turns out to be an advantage as we can also modify the table contents in the presence of network faults, making packets navigate around them. In fact, both the S3.mp [11] and the Cray T3E [16], among others, use full table routing.

The same RR algorithms that calculates the routing record for a packet going from *source* to *dest* (see Table 2), can also be used to generate the contents of the routing tables. Each table contains 4 bits per node entry that indicate the dimension and the direction of travel. For example, the entry 1010 indicates the message can use either *+x* or *+y*, 0011 indicates the *x* dimension is exhausted (first bit is 0) so it must advance using the *-y* output channel. The entry for its own node number is 0000, indicating the message has reached its destination.

The advances in VLSI technology allow for large routing tables (hundreds of entries with 4 to 8 bits per entry) to be easily incorporated into a router chip. Besides, the access to such tables has a similar delay that the decoding of the routing record, although it requires larger silicon area. The table's size may limit network scalability for networks with thousands of nodes, but this can be solved by using hierarchical tables that route packets from one subnet to the next [6].

3. Deadlock Freedom in Midimew Networks

Parallel architectures have not capitalized on the topological gains of the Midimew network because there was no simple deadlock-avoidance mechanism for this topology. In this section we will discuss the solutions to this problem.

It's well known that the wrap-around links of the torus double the bisection bandwidth in comparison to that of the mesh. By doing so, the network forfeits deadlock freedom, even for DOR routing, as the wrap-around links introduce dependency cycles. Such cycles also occur in a Midimew network so that deadlock must be prevented for both DOR and adaptive routing.

A common approach to eliminate these dependencies is to break the cycles by using two virtual channels as shown in [4]. In a Midimew network, though, channel dependencies are not limited to a single row or column as in the torus, but they vary in length and shape depending on the network size *N* and the parameter *b* derived from it. Figure 2 shows two examples for *N* equal to 16 and 18.

We can see that the number of *x*-rings and *y*-rings created by the wrap-around links varies considerably from one network size to the next. In the case of the rectangular Midimew shown in figure 2(a), the *y*-rings resemble those of the torus, while the single *x*-ring interconnect all rows, forming a Hamiltonian cycle. In figure 2(b), though, the x-rings resemble those of the torus, and the two y-rings interconnect 3 columns each. Thus, it is difficult to establish what is the equivalent to the wrap-around transition from one virtual channel to the next. Furthermore, when we scale the network up the number of *x*-rings and *y*-rings changes and it is difficult to predict where the wrap-around links will be.

Recently, we have proposed an extension to virtual cut-through called Bubble flow control [15] which avoids packet deadlock in dimensional rings. Instead of statically breaking the cyclic dependencies, it limits the entry of packets to any dimensional ring in such a way that the buffer space of this ring will never be completely exhausted. In other words, a dynamic cyclic dependency within a dimensional ring cannot occur because there is always a free buffer within that ring.

Figure 3 illustrates the use of Bubble flow control in a dimensional ring. Packets (shaded queue units) are allowed to move (shaded arrows) from one queue to another inside the ring as per virtual cut-through switching. However, packet injection is only allowed at a given router if there are at least two empty packet buffers in the dimension (and direction) requested by the packet. By doing so, we guarantee that there is always at least an empty packet buffer in the ring. That free buffer acts as a bubble, guarantying that at least one packet is able to progress.

This mechanism is applicable to a network if we can easily identify which routing decision causes a packet to *enter* a network ring, and apply bubble flow control at that point. For DOR routing, packets enter a new dimensional ring when they are injected into the network or when they change dimension from X to Y or vice versa. This condition remains unchanged for any size Midimew, independently of the length of its *x*-rings or *y*-rings. So, there is no problem to identify where to apply the deadlock avoidance mechanism. Besides, this condition is similar for both torus and Midimew topologies so they will have the same complexity.

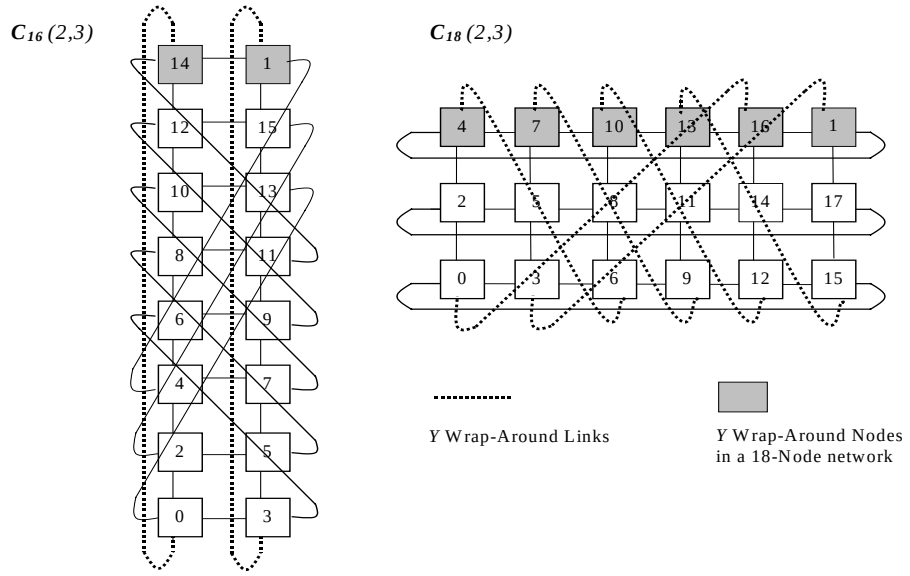


Figure 2.- Location of Y-wrap-around links for Midimew networks with 16 and 18 nodes

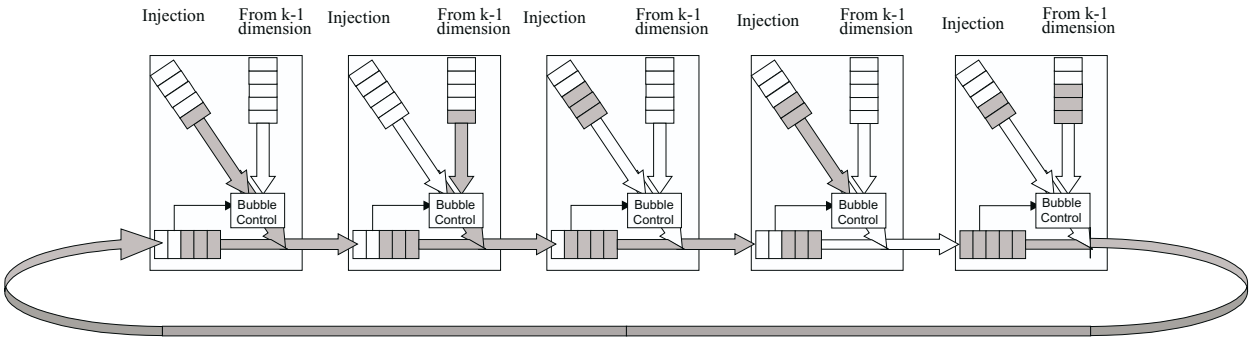


Figure 3.- Deadlock avoidance in a ring using Bubble Flow control.

4 Router implementation.

With the aim to show the feasibility of the Midimew network this section describes the hardware implementation of an adaptive router for this topology. This is the first hardware proposal of a Midimew router, thanks to the combination of Bubble flow control and routing tables.

In fact, by using routing tables we can implement a VCT router, which can be configured to build mesh, torus or Midimew networks; the differences in wrap-around connections and routing algorithms are only reflected in the contents of their routing tables. Thus, we can reuse the adaptive Bubble router described in [15] for torus networks to build meshes or Midimews as well.

Figure 4 shows the basic structure of this router. It has two virtual channels per link: a deterministic channel that applies DOR and an adaptive channel able to select any minimal path. The flow control is virtual cut-through (VCT) and channel multiplexing is performed at the packet level. It should be noted that deterministic channels require a minimum capacity of 2 packets in order to apply the Bubble mechanism [15] (except for the mesh case which has no wrap-around connections).

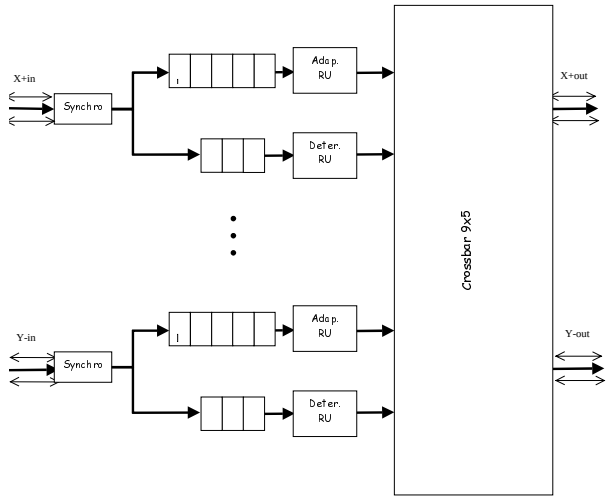


Figure 4. Architecture of the adaptive router used for the comparison of planar topologies.

We should point out that the cost of implementing the Bubble deadlock avoidance mechanism in the deterministic channel is negligible. It requires one occupancy status signal from each deterministic input to every routing unit. This signal indicates if there is a second free buffer on that dimensional ring, and if such is the case, the channel access can be granted.

The packet size has been fixed to 20 phits, and each phit has 16 bits. Router communication is asynchronous, so there is an extra bit (signaling the tail phit) that flags the transmission of data. Thus, the channel width is 17 bits.

Each router is self-timed and pipelined into 4 stages as shown in Figure 5. The first stage corresponds to the synchronization module for each input link. Then, the incoming packet is received and stored at the input FIFO. The header information is used at the routing unit to access the routing table and to select the output channel(s). The selected channel is requested to the arbiter; when the request is granted, the crossbar connection is established and the packet forwarded to the next node. For a detailed description of each module, please refer to [15].

We measured the router's implementation cost by generating its VHDL description, which was then fed into *Synopsys*, a high-level synthesis tool. This design was mapped into 0.70 μm technology with two metal layers from the ATMEL/ES2 foundry. The delays were calculated under the standard conditions of this technological library.

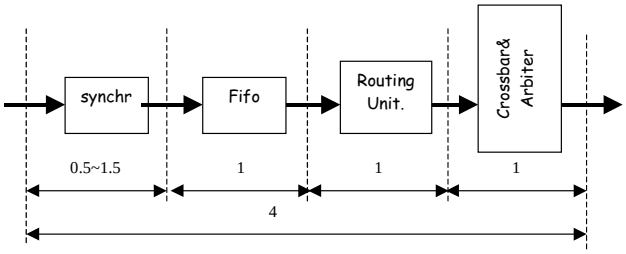


Figure 5. Router's 4-stage internal pipeline.

Although the router will exhibit the same delay under any topology, it is important to quantify its hardware cost. Then, higher level simulations can incorporate this cost and provide network performance in bytes per nanosecond instead of a generic phits per cycle, where both units may vary from one implementation to another.

	Critical Path (ns.)	Area (mm^2)
Synchronization	3.53	0.55(x5)
Input FIFO (4 packets = 80 phits)	5.22	1.10(x9)
Routing Unit	5.64	1.06(x9)
Crossbar & Arbiter	**5.65**	**8.98(x1)**
Clock Cycle/ Total Area	5.65	31.17

Table 3. Time and area characteristics of each router module.

The characteristics of the router elements in terms of delay and area are shown in Table 3. The clock frequency is set by the slowest stage of the pipeline that corresponds to the *crossbar & arbiter* unit. Once the critical module was identified, the other modules were synthesized under the new time restrictions in order to optimize their area demands.

5. Performance Analysis.

The theoretical advantages of the Midimew topology were clear [2]. Now that we are able to physically implement this network, we can also verify that these advantages are going to translate into better network and system performance when compared with other planar topologies. Hence, our next step is to compare the performance of the network, based on the router described above, under each topology. Although our main comparison is with the torus network we have considered the mesh network as the baseline planar topology due to its simplicity.

This task was carried out using a register-transfer level simulator called *SICOSYS* [14], which takes into account the key parameters of the low-level implementation and obtains results which are very close to those of a VHDL simulator but with a lower computational cost.

We considered two types of workloads. Firstly, we applied synthetic loads that allow us to heavily *stress* the network using a range of message distribution patterns. Secondly, we test the network under real loads generated by applications running under a DSM architecture with cache coherence protocol (ccNUMA)

5.1. Network Performance under Synthetic Loads

We have tested network performance under uniform traffic with either a fixed or a bimodal (short and long messages) length distribution. We have also considered three destination patterns based on some permutations: *matrix transpose*, *bit-reversal* and *perfect-shuffle*. Message length is fixed to 20 phits (a single packet) except for the 10% of long messages (200-phits split into 10 packets) in the bimodal distribution.

We studied three network sizes, 16, 64 and 256 nodes. Mesh and torus networks are 4x4, 8x8 and 16x16 respectively. Midimew configurations are $C_{16}(2,3)$, $C_{64}(5,6)$ $C_{256}(11,12)$ for each respective size. By varying the network size we can see the performance trends for each topology as system size increases.

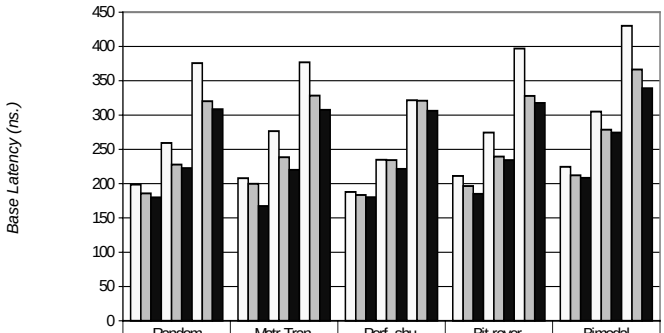
Figure 6 shows the base latency for each possible network and traffic combination. As all routers have the same node delays, the variations result from the different average distances of each particular message distribution under the three topologies. Although average distance varies from pattern to pattern, all exhibit the lowest value, and therefore the lowest base latency, for the Midimew case. As we increase network size, so they do the differences in average distance and their respective network latencies.

Figure 7 shows the maximum throughput (bytes/nanosecond) achieved by each simulated network under the different traffic patterns. We can observe that the Midimew network achieves the highest throughput for any traffic pattern, and that these gains increase with network size. Besides, the differences between the torus and the Midimew are greater for permutation patterns.

Average distance plays an important role on the time a message spends using the network resources. Shorter paths lead to a lower resource usage and therefore a higher message delivery rate. The Midimew gains over the torus topology are more significant for larger networks, because the reduction in average distance is also greater.

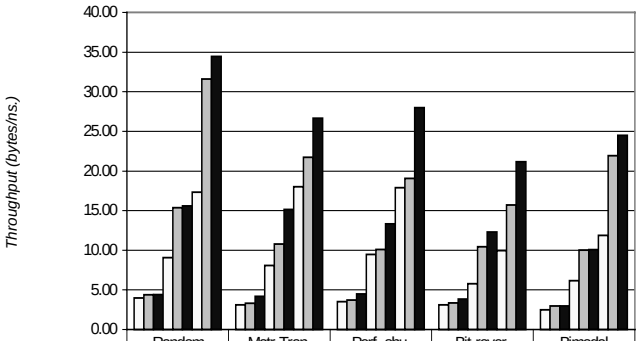
Another factor is the distribution of the traffic load over the network channels. Torus and Midimew, being symmetric networks, have a balanced channel utilization under uniform traffic. Thus, the throughput gains of the Midimew in relation to the torus for random and bimodal traffic patterns are due to its lower average distance. The mesh topology, with its lower bisection bandwidth and network asymmetry, achieves the lowest throughput for any traffic pattern.

Non-uniform traffic makes an unbalanced used of channels, so throughput depends heavily on the type of traffic distribution. In fact, mesh and torus networks exhibit the same message average distance for the perfect-shuffle permutation (8.05 in a 256-node system). Therefore, they present the same base latency. Throughput, though, differs because is not only affected by the number of channel used but also by their distribution, which is more balanced in the torus due to the presence of wrap-around links.



	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	235	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
	309	307	306	317	339

Figure 6. Base latency in nanoseconds (for 0.05% normalized[1] load).



	Random	Matr-Tran.	Perf.-shu.	Bit-rever.	Bimodal
Mesh 4x4	3.99	3.12	3.53	3.12	2.48
Torus 4x4	4.38	3.31	3.71	3.34	2.97
Midimew 16	4.40	4.17	4.49	3.85	2.97
Mesh 8x8	9.08	8.06	9.47	5.79	6.16
Torus 8x8	15.36	10.78	10.12	10.45	10.02
Midimew 64	15.62	15.14	13.34	12.32	10.08
Mesh 16x16	17.32	18.00	17.90	9.97	11.89
Torus 16x16	31.60	21.73	19.08	15.71	21.95
Midimew 256	34.46	26.65	28.00	21.15	24.51

Figure 7. Maximum Network Throughput in bytes delivered per nanosecond.

[1] to the bisection bandwidth limit

The Midimew achieves throughput gains of 19%, 32% and 26% in relation to the torus, in the case of 256-nodes, for the three non-uniform patterns: matrix-transpose, perfect-shuffle and bit-reversal respectively. The reduction in message path length for the three patterns is 8.5%, 8% and 5% respectively. The additional throughput gains under matrix-transpose and perfect-shuffle traffic are due the more balanced use of the channel resources. For example, in Figure 8 we show the buffer occupancy for the 16x16 torus and for the 256 Midimew under matrix transpose traffic beyond saturation point. Each one of the four squares in the Figure represents the set of adaptive channels for each topology. It can be noted that there is a more balanced buffer usage in the Midimew case, hence its channels become saturated at higher loads.

5.2 Network Performance under Real Loads

In the previous subsection we have proved that the Midimew increases the network performance of its torus counterpart. We complete this study by evaluating the impact that this topology will have on the whole system; in other words, how it will affect the execution time of parallel applications.

This evaluation was carried out by using a execution-driven simulator ED-SICOSYS[14] which has been derived from RSIM [12] by replacing the original RSIM's network module NETSIM [7] with our own simulator SICOSYS. This allows for a more precise simulation of the communication subsystem as part of a cc-NUMA machine.

RSIM emulates the behavior of a cc-NUMA machine with state-of-the-art ILP processors. Thus, our evaluation must consider parallel applications written under the distributed-shared memory paradigm. The three benchmark applications we are going to use belong to the SPLASH2 suite [18], and were selected because they are communication intensive, thus the network plays an important role in their performance. The applications are RADIX, FFT and LU, executed on a 64-node system with either a 8x8 mesh, a 8x8 torus or a $C_{64}(5,6)$ Midimew. In all cases, we used the default problem sizes.

TOPOLOGICAL ASPECTS OF DATA DISTRIBUTION

When a parallel application is compiled, the shared variables must be distributed amongst the processing nodes. Parallel applications must exploit data locality in order to minimize their communication demands. Thus, data distribution depends on both the number of nodes and the network topology [13].

The standard data distribution follows the locality pattern of the torus and mesh network in which node *i* is directly connected to nodes *i+1* and node *i-1*. Before we execute the application we should consider how to adapt this data distribution to exploit data locality in the Midimew network. We can take one of these two approaches:

- Make the application aware of the Midimew characteristics, including its particular node enumeration and let the programmer deal with it.
- Hide this difference from the application by constructing the routing tables based on the enumeration assumed by the application code. The relation between the enumeration at application level and the one at the network level follows the algorithm depicted in Figure 9.

In this study we opted for the second approach, so the application code is the same for the three topologies. The former solution, though, allows the programmer to explore other data distributions that may further exploit the characteristics of the underlying topology.

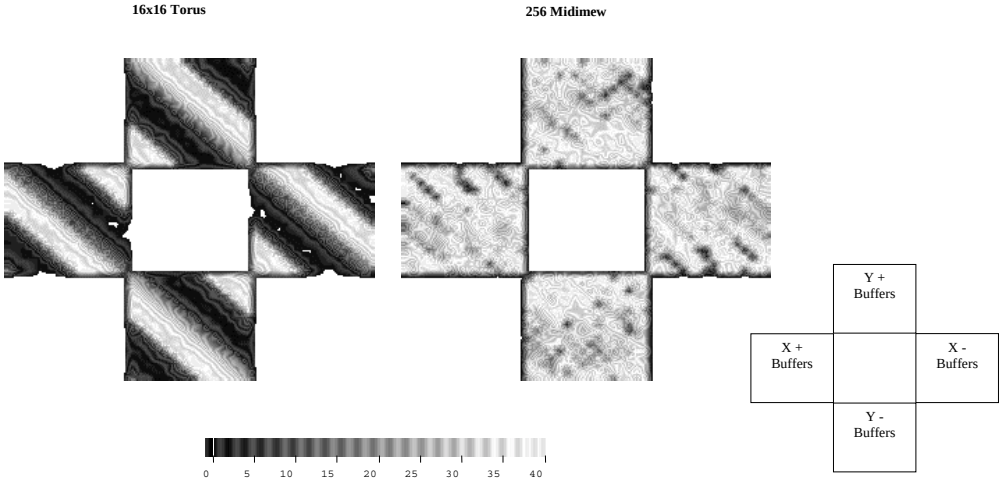
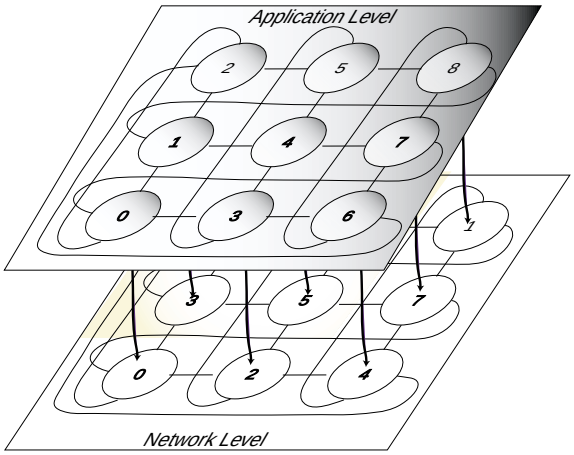


Figure 8. Buffer occupation (expressed in phits) for matrix transpose traffic in 16x16 Torus and in 256-node Midimew




```
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;
              }
              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;
}
```

Figure 9. Node mapping between Application Level and Network Level.

PERFORMANCE ANALYSIS.

The parameters of the cc-NUMA machine simulated (cache coherence protocol, processor architecture, etc) have the default values set by RSIM [13] except for:

- The cache line size which is set to 32 bytes,
- The command size is 8 bytes,
- The processor speed is set to 650 MHz instead of the default 300 MHz.

Therefore, a data message contains 40 bytes or 20 phits (the channel width or phit size is 2 bytes). The control messages, request or invalidation, are 8 bytes or 4 phits.

The higher processor speed represents a viable parameter with the latest implementation technology. In line with this, the network speed is limited by the router speed which clock cycle was estimated in Section 3 to be *5.65 ns* for any of the three topologies. So the network operates at a clock speed of 177 MHz, and the relative speed of the processor in respect to the network is 3.67. We consider that there are two separate data and control networks, so there is no application deadlock due to limited consumption queues capacity.

	Average distance	***Radix***	***FFT***	***LU***
Mesh 8x8	5.33	4.55	5.29	4.88
Torus 8x8	4.06	3.47	3.99	4.00
Midimew 64	3.87	3.45	3.74	3.76

Table 4. Average message path length for each application and topology (64 processors).

The average distance traveled by messages for each application and topology is shown in Table 4. The first column, given as a reference point, represents the topological average distance. Obviously, the mesh shows the largest values for any application. Midimew and torus differ in the range of 1% to 8%. So, we will expect execution time for FFT and LU to be shorter for the Midimew in respect to the torus. The topology has a lesser impact on Radix because a large amount of traffic is related to synchronization tasks involving access to global variables (*home*) at a neighbor node.

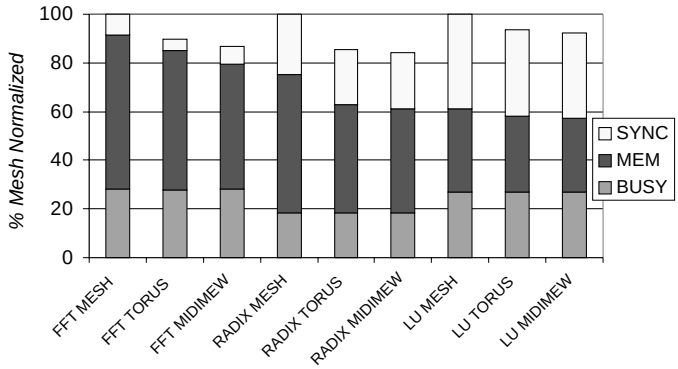


Figure 10. Execution time (normalized in relation to the mesh system) for each application executed over the different topologies.

Figures 10 and 11 show the execution time for each possible combination; results for a given application have been normalized to the result of a mesh-based system. We can see that the Midimew system is again showing the best results, although the differences between torus and Midimew systems are less significant than for synthetic traffic for the same size system. This is because the applications demands are quite

variable over the execution time in comparison to the *constant pressure* provided by synthetic traffic.

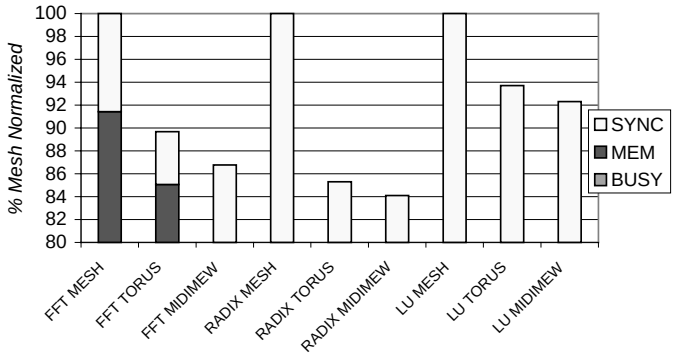


Figure 11. A close-up of the previous graph showing detailed execution times.

CHANGING THE NETWORK SIZE: FFT STUDY.

As with synthetic traffic, is interesting to study the trends in network performance as the network size increases, so we can forecast its performance for larger networks. Both Radix and LU do not properly scale over the 64-node size, so we focus on the FFT application and consider four system sizes: 16, 32, 64 and 128 nodes. All the other parameters are those described for the previous tests.

Figure 12 shows the results for each topology. The trend is similar to that observed for synthetic loads. As the network size increases, so it does the difference in performance between the three topologies. For the largest system, 128 nodes, the Midimew reduces execution time by 8% and 23% in respect to the torus and mesh respectively. We can conclude that a system with a large number of nodes will clearly benefit from the topological properties of the Midimew.

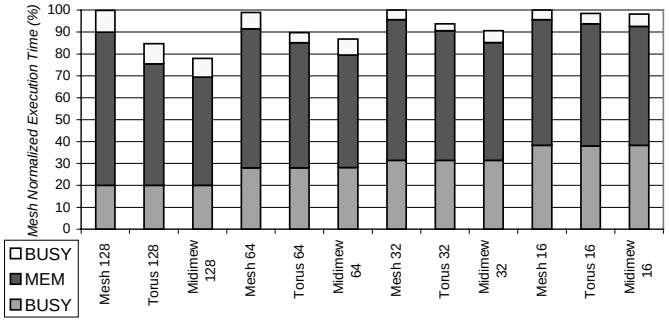


Figure 12. Execution time for FFT application for different system sizes.

6. Conclusions

The Midimew is an optimal 2D topology which reduces the average distance and diameter of its torus counterpart. Therefore, Midimew networks are good candidates for implementing the communication subsystem of a multiprocessor computer. Reducing the average message path length plays an important role in network performance: message latency is reduced and throughput increased.

As the Midimew can be decomposed into a set of dimensional rings, we can avoid deadlock, as we did for the torus network, by combining dimensional order routing and Bubble flow control. The differences in path selection disappear by using routing tables, so we can implement an adaptive router, which is identical for either torus, or Midimew networks. Therefore, we can exploit the topological advantages of the Midimew by simply changing the arrangement of the wrap-around connections of its torus counterpart and consequently, without any increase on the network implementation cost.

A thorough evaluation of both networks, both under synthetic loads and under real loads generated by benchmark applications running on a cc-NUMA architecture, has confirmed the superior performance of the Midimew network with no extra cost. It must be highlighted that higher performance is achieved when the network size increases. Therefore, Midimew networks are very good candidates for the design of truly massively parallel computers, as the ones used in the ASCI project.

The system evaluation presented above corresponds to the standard data distribution used for mesh and torus networks. We are currently exploring the benefits of other data distributions that exploit even further the topological properties of the Midimew network.

In short, all our evaluation showed that the improvements in terms of topological characteristics of the Midimew network are transportable to improvements in network performance under both synthetic and real application loads.

7. References

[1] A. Agarwal, “Limits on Interconnection Network Performance”, IEEE Trans. on Comp., vol. 2, no4, pp:398-412, October 1991.

[2] R. Beivide, E. Herrada, J. L. Balcazar, A. Arruabarrena, “Optimal Distance Networks of Low Degree for Parallel Computers”, IEEE Trans. on Comp., vol. 40, no. 10, pp. 1109-1123 November, 1991.

[3] A. Chien, "A cost and Speed Model for k-ary n-cube wormhole router", In Proc. of Hot Interconnects, August 1993.

[4] W. J. Dally and C. L. Seitz, “Deadlock-Free Message Routing in Multiprocessor Interconnection Networks”, IEEE Trans. on Comp., vol. C-36, 5, pp. 547-553, 1987.

[5] W.J. Dally, “Performance Analysis of k-ary n-cube Interconnection Networks”, IEEE Trans. On Comp., Vol 39,No. 6 pp. 775-785, June 1990.

[6] M.Galles, “Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI Spider Chip”, Proc. of Hot Interconnects IV, August 1996.

[7] J. R. Jump. "NETSIM Reference Manual". Rice University Electrical and Computer Engineering Department, March 1993.

[8] C.M. Lau and G. Chen, “Optimal Layouts of Midimew Networks”, IEEE Trans. on Parallel and Distributed Systems vol. 7, no. 9, pp. 954-961, September 1996.

[9] J. Laudon and D. Lenoski, “The SGI Origin: A cc-NUMA Highly Scalable Server”, Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA-97), June 1997.

[10] J Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, M.S. Lam “ The Stanford DASH Multiprocessor” IEEE Computer no. 25 vol 3, pp- 63-79, March 1992.

[11] A. Nowatzyk, G. Aybay, M. Browne, E. Kelly, M. Parkin, B. Radke, and S. Vishin “The S3.mp Scalable Shared Memory Multiprocessor”, Int Conf on Parallel Processing, August 1995.

[12] V. S. Pai, P. Ranganathan, S. Adve "Rsim: An execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors", IEEE TCCA Newsletter, pp. 1-10 October. 1997.

[13] V. S. Pai, “RSIM Reference Manual. Version 1.0”. Department of Electrical and Computer Engineering, Rice University. Technical Report 9705. July 1997.

[14] J.M. Prellezo, V. Puente, J.A. Gregorio, R. Beivide, "SICOSYS: an interconnection network simulator for parallel computers," available at http://www.atc.unican.es/REPORTS/TR-ATC2-UC98.pdf, June 1998.

[15] V. Puente, J.A. Gregorio, J. M. Prellezo, R.Beivide, J. Duato, and C. Izu, "Adaptive Bubble Router: a Design to Balance Latency and Throughput in Networks for Parallel Computers", Proc. of International Conference on Parallel Computing, pp. 58-67, September. 1999.

[16] S.Scott and G. Thorson, “The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus”, Hot Interconnects IV, August 1996.

[17] C.L.Seitz, “Concurrent VLSI architectures”, IEEE Trans. on Comp., C-33, pp. 1247-1265, December 1984.

[18] S. C. Woo et al., “The SPLASH-2 Programs: Characterization and Methodological Considerations”, In Proceedings of the 22nd International Symposium on Computer Architecture, pp. 24-36. June 1995.