CALCULATING OPTIMAL FLIT SIZE AND UPPER LIMIT ON THE PERFORMANCE OF WORMHOLE ROUTING

ANTHONY SYMONS AND V. LAKSHMI NARASIMHAN[1]

***Abstract*—The emergence of large scale distributed memory computers has brought with it a variety of interconnection networks. As it is not feasible to fully interconnect the processors with a diameter of one, routing of messages is necessary. Wormhole routing is an efficient method of routing and in this paper, we derive a model for wormhole routing. Using this model, we determine the optimal buffer size, and hence show that for medium to large messages, the communication time using wormhole routing is dependent on the square root of the network diameter.**

***Keywords*—Parallel Distributed Computing, Hypercube, Mesh, Performance Parameters, Wormhole routing**

***Source of Publication*—The paper was presented to the ICA^{3}PP-97 conference, Melbourne, December 1997 and appeared in the proceedings.**

1 INTRODUCTION

The computational requirements of today's scientific applications, such as Fast Fourier Transform (FFT) [1], fluid dynamics, equation of state calculations and dual simplex integer programming problems has outstripped the power of uniprocessor systems. As a consequence, these applications have migrated onto parallel computers. A major class of parallel computers is distributed memory systems which rely on message passing for synchronization. Examples of message passing parallel computers are Transputer Hypercube [2], Transputer mesh, cluster of workstations connected via an ethernet and the IBM - SP2.

In distributed memory architectures, the memory is distributed amongst all of the processors, i.e., each processor has its own memory. This means that there is no longer any contention for memory accesses (the memory is local), however, for a processor to access an object in another processor's memory, it must be passed in the form of a message.

To allow messages to be passed between processors, some form of communication is necessary, we call this the interconnection network. Some measures of the goodness of the interconnection network are the number of communication links and the diameter of the interconnection network. We define the diameter of the network as the minimum number of communication links between any two processors which are the farthest apart. Reducing the number of communication links reduces the the hardware cost of the system, yet, we shall see that there are trade offs between a small diameter and a small number of communication links.

The question then arises—why can't we fully interconnect all of the processors with a diameter of one? To achieve this, each processor would require a communication link to each other processor, thus for N processors, the number of communication links is of the order of N^2, which is neither economical nor physically feasible for large numbers of processors. This interconnection complexity can be reduced in two ways. Firstly a multistage interconnection network can be used. In this case, a path can be made between any two processors at the expense of going through a number of switching stages, effectively increasing the diameter to $O(\log N)$. Secondly, the processors are not fully interconnected, but rather they are connected to only a few neighbouring processors or switches based on their interconnection topology. Messages to destination processors not directly connected to the source processor now must be routed through intermediate processors or switches.

One method for message routing is termed Store and Forward routing (see Figure 1). Messages on intermediate nodes are read into a single buffer, and are only then output. It can be seen that the communication latency is proportional to the product of message size and network diameter.

Improvement in performance can be gained through the use of *wormhole routing* [3]. In this method, messages are divided into a number of packets, termed *flits*. At any intermediate node, once one flit is read, it can be output, thereby reducing the communication latency. This type of routing is termed Wormhole routing,

1. Head, Information Management Group, Information Technology Division, DSTO, SA 5108, Australia

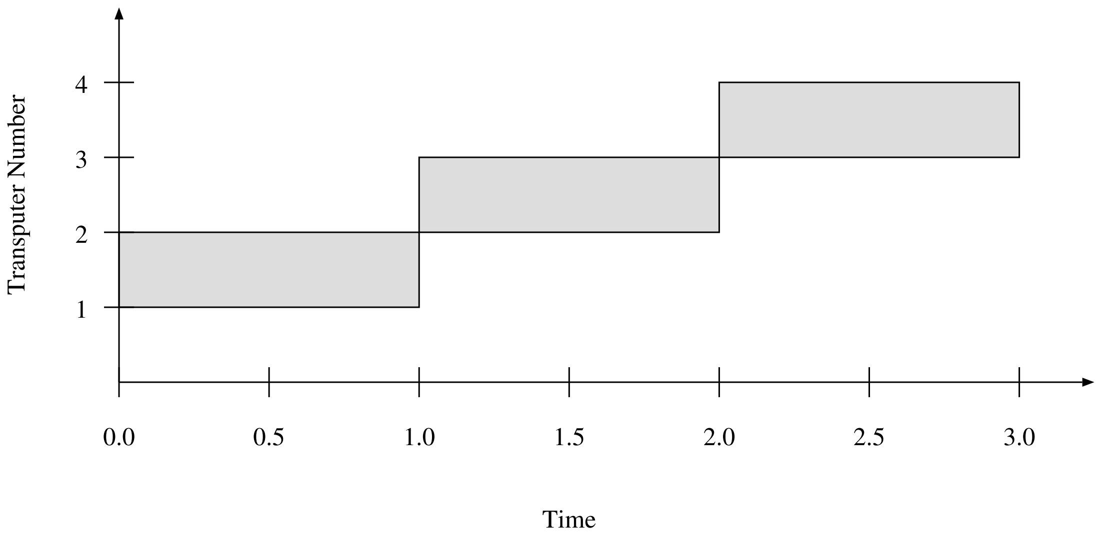

Figure 1: Store and Forward Communication

so called because the head of the message makes a path to the destination in which the tail duly follows. This process is shown in Figure 2.

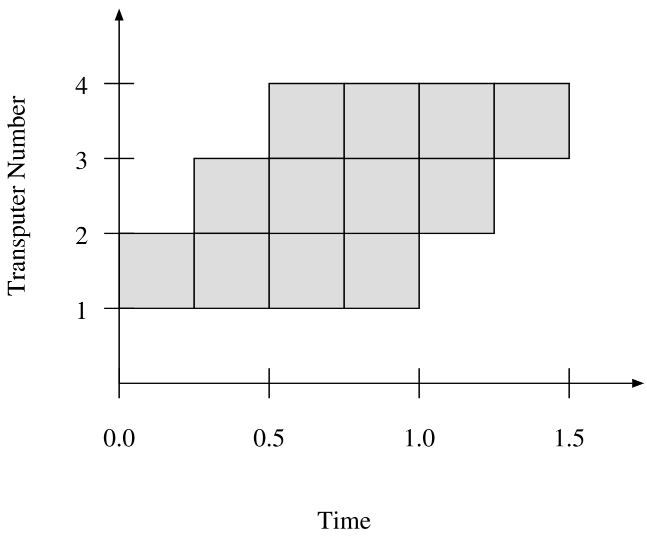

Figure 2: Wormhole Communication

Much research has been made into wormhole routing (see for example [4] and [5]), however, the effects of startup costs and overheads incurred through segmenting the data into flits has often been overlooked.

In this paper, we address this important issue by determining the optimal buffer size, and hence an upper limit on the performance of wormhole routing. The rest of this paper is organised as follows:- Section 2 introduces two common interconnection networks, namely, the binary hypercube and the two dimensional mesh. In section 3, the model for wormhole routing is developed. Section 4 compares the communication performance of the hypercube and mesh interconnection networks. The results in section 5 compare the performance of the interconnection networks for the Livermore Loops. A final summary is provided in section 6.

2 INTERCONNECTION NETWORKS

2.1 TRANSPUTERS

The transputer is used as the hardware basis for the simulated results in Section 5. The transputer [6] was first introduced by Inmos in 1985 as a commercial chip designed for distributed memory applications. The first series of transputers were the 16 bit T2 family. The first 32 bit transputer was the T414 which offered 10 MIPS and a floating point unit on chip. Several models of the transputer have since been released, notably the T800 with four communication links and T9000 series.

2.2 HYPERCUBE

Many parallel computers of the 1980s used a topology termed the *Hypercube*. In a three dimensional binary hypercube, as shown in Figure 3, a processor is placed at each vertex of a cube. An N-dimensional binary hypercube is an extension of this, with a processor placed at each vertex of the N-dimensional cube.

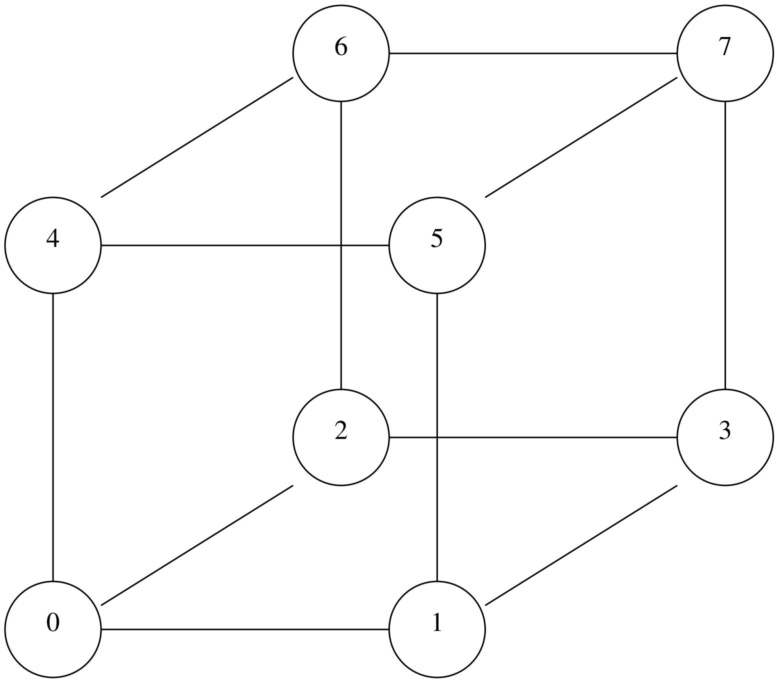

Figure 3: Hypercube Configuration

Definition 2.1 *An N-dimensional binary hypercube has $P=2^N$ processors and $N \times 2^{N-1}$ edges or communication links. Each node has N communication links. The diameter of the binary hypercube is N.*

Great interest was shown in the hypercube due to certain advantages over other topologies such as the ability to map mesh and tree topologies efficiently onto the hypercube, and the diameter of the hypercube increasing only as log*P* [7]. However, to achieve this interconnectivity, each processor must have log*P* links and the total number of communication links is of the order of *P*log*P*. For example, in a 1024 node hypercube, each processor must have 10 communication links. This requirement of links per processor can limit the size of the hypercube that can be implemented e.g., the Intel iPSC/860 is a hypercube system which can only support 128 processors.

2.3 2D MESH

The 2D mesh configuration (see Figure 4) overcomes the hypercube's problem of the number of links per processor increasing as the number of processors increases by using only four communication links per processor. As the number of links per processor is fixed, the mesh topology can scale to any number of processors.

Definition 2.2 *An $A \times B$ mesh has $P = A \times B$ processors and 2AB - A - B communication links. Each node has four bidirectional communication links connected to its nearest neighbours without wrap-around at the edges of the mesh. The diameter of a mesh*[1] *is A+B-2.*

1. The diameter of the mesh is calculated as follows :- The longest path is from the origin (0,0) to the most extreme processor (*A*,*B*). The message must be routed through *A*-1 links along the X axis and *B*-1 links along the Y axis—hence the total number of links is *A*+*B*-2

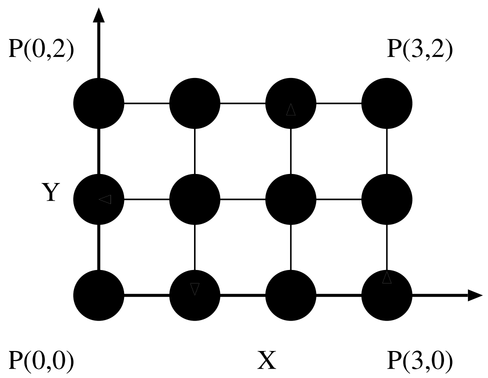

Figure 4: Mesh Configuration

To route messages amongst the processors, the mesh configuration can be considered as a matrix of R rows parallel to the X axis and C columns parallel to the Y axis. Three routings from slave to host are considered in this paper.

1. Column first (Y routing)—Messages are firstly routed down the columns.
2. Row first (X routing)—Messages are firstly routed across the rows.
3. Step—If the position of the slave (x,y) satisfies the equation y-x=2I where I is an integer, the message is routed through the row adjacent processor, else it is routed through the column adjacent processor.

The row and column first routing are the ordered dimensional routings analysed in [8] and are deadlock free. The performance of the dimensional routings for transputer networks is analysed in [9] and [10]. The step routing can be shown to not be deadlock free for general routing. However, if we define the host as the processor at the origin (0,0) of the mesh, then for communication to and from the host, the step routing can be shown to be deadlock free.

2.4 IBM SP2 HIGH PERFORMANCE SWITCH MIN

Recently, IBM has started installing its IBM PowerParallel SP2 which can be considered as a cluster of workstation nodes on a multistage interconnection network. Each of the SP2 nodes is based on the IBM RISC System/6000 workstation. The nodes are available as either *thin* or *wide* nodes. The wide nodes have greater memory bandwidth and a larger memory cache than the thin nodes. These nodes are particularly suited to floating point operations, with the wide nodes delivering a SPECfp92 rating of 259.7 and a Linpack DP rating of 131.8 [11].

The interconnection network is based on a *High performance switch logical frame* [12]. Each frame consists of a crossbar multistage interconnection network which interconnects 16 processors, and allows other frames to be connected. Up to 80 nodes can be connected via a single stage of frames. For 81 to 128 nodes, a second layer of frames is required for the interconnection. These frames are termed *Intermediate Switches*.

Each frame operates at 40MHz. Reported throughput figures for these frames are a latency of 42μ*s* [13], a uni-directional bandwidth of 35MBytes/s and a bi-directional bandwidth of 48MBytes/s [11].

3 WORMHOLE MODEL

Communication latency, the time delay between initiation of a message to the time when the data is available at the destination node, measures the performance of a message passing architecture. Communication latency can be split into three components.

1. Start up latency: Time to prepare a packet for transmission or reception.
2. Network latency: The time difference between the sending of the head of the message from the source processor and the reception of the message's tail on the destination processor.
3. Blocking latency: The time delays caused both by network traffic and contention for resources, i.e., the communication links.

A simple time line is shown in Figure 2 for the case when the message is four times the size of the buffer. Note now that the communication latency is the time to communicate the message between adjacent processors, plus three times the communication time of a buffer. In this case the latency is 1.5 units, a significant reduction from three units. The communication has thus become pipelined, i.e., the output of a buffer is overlaid with the input of another buffer. Thus the only delay in routing is the initial input of the buffer on each intermediate node. It would seem that the smaller the buffer size the better the performance.

However, an overhead is introduced due to the startup costs of sending approximately (message size ÷ buffer size) flits. As the buffer size is reduced, the number of flits is increased, thereby increasing the startup communication overheads. Thus there is a trade off between reducing latency due to smaller buffer sizes, and the increasing overheads due to increasing number of flits. We derive a model for wormhole routing as follows.

If we define D as the path length, i.e., the number of hops involved in the routing (the maximum path length for a network is given by its diameter), B as the buffer size in bytes and M as the message size in bytes then $N = \lfloor M/B \rfloor$ is the number of full buffers sent and $R = \mathrm{frac}(M/B)$ is the remaining data to be sent. Then the time to communicate a message between two adjacent and intermediate processors can be given by equation 1 where $C(X)$ is the communication function for sending a message of size X bytes over a communication link.

$$Com\ Time\ Adjacent = N \times C(B) + C(R) \tag{1}$$

If we generalize the communication function as $C(X) = \alpha X + \beta$, where α is the communication time proportional to the packet size (seconds per byte) and β is the fixed startup communication time (seconds), then equation 1 can be written as equation 2.

$$Com\ Time\ Adjacent = N \times (\alpha \times B + \beta\,) + \alpha \times R + \beta \tag{2}$$

By noting that $N \times B + R = M$, equation 2 can be simplified to give equation 3.

$$Com\ Time\ Adjacent = M \times \alpha + (N + 1) \times \beta \tag{3}$$

For each of the D-1 processors between the host processor and the destination processor, the delay between when the source processor starts to send the message and when the destination processor starts to receive the first buffer increases by one buffer communication time due to the delay caused by the previous processor having to input the first buffer before any output can occur. Thus, the total time between when the host initiates sending the message and destination processor receives the entire message can then be given by equation 4.

$$Total\ Com\ Time = M \times \alpha + (N + 1) \times \beta + (D\text{-}1) \times (\alpha \times B + \beta\,) \tag{4}$$

This equation can be further simplified if we can assume that $M \gg B$ such that $N + 1 \approx M/B$, thus giving equation 5

$$Total\ Com\ Time = \left(\qquad \right)(\qquad) \tag{5}$$

The second factor in equation 5 is the communication time of a buffer. The optimal buffer size can be found by differentiating equation 5 w.r.t. B, and setting equal to 0, giving the minimum at equation 6.

$$B = \sqrt{\frac{M \times \beta}{(\qquad) \times \alpha}} \tag{6}$$

Thus we see that the optimum buffer size increases as the message size increases, yet decreases as the message path increases. The increase in optimum buffer size with increasing message size can be explained by considering the ratio of buffer size to message size. This ratio determines the number of buffers sent and hence the amount of overhead caused through start up latency. A reduction in this ratio (i.e., increasing the buffer size) reduces the start up latency overheads. The reduction of optimal buffer size with the increase in path length can be explained by noting that as the path length is increased, the delay caused by the propagation of the first buffer increases. As this delay is proportional to the buffer size, reduction of the buffer size reduces the propagation delay.

For the case of a four transputer[1] pipeline, i.e., $D = 3$, the values for α and β are $5.64 \times 10^{-7} s/byte$ and $1.76 \times 10^{-4} s$ respectively[2]. Substitution into equation 6 with a message size of 8000 bytes, gives an optimal buffer size of 1117 bytes, which is significantly smaller than the message size.

The assumption of $M \gg B$ implies from equation 6 that $M \gg \frac{\beta}{(D-1)\alpha}$. Using the transputer values for α and β, gives $M \gg \frac{312}{(D-1)}$, i.e., this assumption is satisfied by messages greater than one kilobyte.

The communication time for wormhole routing can be compared with the time for store and forward routing. In store and forward routing, the communication time is a factor of D greater than the time to send the message between adjacent processors, i.e., $D \times (\alpha M + \beta)$. For small β such that $(\alpha \times B) \gg \beta$, and large D such that $D - 1 \approx D$, the ratio of store and forward communication time (SFT) to wormhole communication time (WT) is given by equation 7.

$$\frac{SFT}{WT} = \frac{\alpha M \times D}{\alpha M + \alpha D \times B} = \frac{D}{1 + \frac{D \times B}{M}} \tag{7}$$

For large messages and small buffer sizes, this ratio approximates D, i.e., wormhole routing gives a performance improvement of D over store and forward routing.

Substituting the optimal buffer size equation 6 into equation 5 gives equation 8.

$$Total\ Com\ Time = \alpha M\left(1 + \sqrt{\frac{\beta}{\alpha}}\sqrt{\frac{D-1}{M}}\right)^2 \tag{8}$$

For large messages[3] such that $M \gg \frac{(D-1)\beta}{\alpha}$, the communication time is given approximately by equation 9.

$$Total\ Com\ Time \approx \alpha\left(M + 2\sqrt{\frac{\beta}{\alpha}}\sqrt{M(D-1)}\right) \tag{9}$$

Thus we see that the communication time for wormhole routing in the absence of link contention is $O(M + \sqrt{MD}$, i.e., the time is proportional to the message size and the square root of the product of the message size and network diameter. Hence, for the same message size on differing topologies, the communication latency is dependent on the square root of the diameter of the network.

4 TOPOLOGY COMPARISON

A square mesh has a diameter $2\sqrt{P}$-2, where P is the number of processors, and a binary hypercube has a diameter of $\log_2 P$. Thus the diameter of the mesh is greater than that for a hypercube with the same number of processors. This may be seen as a limitation for the mesh topology, however, to see the effect of the greater diameter, let us consider a 1024 processor hypercube and mesh. The diameters of the hypercube and mesh are 10 and 62 respectively. Therefore the diameter of the mesh is a factor of 6 larger than that of the hypercube. However, our analysis of wormhole routing shows that the communication latency is dependent on the square root of the diameter, thus the mesh is only a factor of 2.5 larger than the hypercube.

Further, as the number of communication links used in the mesh configuration is less than for the hypercube, the routing hardware and software used in the mesh can often be made much simpler and hence faster than for the hypercube. Therefore it is expected that the mesh will be the dominant topology for large message passing systems [3].

5 LIVERMORE LOOP RESULTS

The Livermore loop kernels are a common way to measure the performance of parallel systems [14]. These kernels were developed to cover a broad range of generic Fortran CPU limited computations to be applied to vector or parallel processors. It has been observed that some traditional benchmark tests were

1. A transputer based network is used for illustrative purposes.
2. These values of α and β are determined from actual measurements on a network of 20 MHz transputers.
3. Using transputer values, $M \gg 312\ (D-1)$, i.e., this assumption is satisfied by messages at least greater than ten kilobytes.

defective when applied to these type of processors which have CPU performance 10 to 100 times the performance of conventional uni-processors [15].

To compare the performance of the various topologies we introduce two measures, namely, Average Network Performance (ANP) and Relative Performance Metric (RPM). To allow the relative merits of the topologies and not the actual implementation technologies to be compared, these metrics are calculated using the same base processing speeds and communication link speeds for each topology.

As explained in Section 2.1, the transputer has only four communication links, thus only allowing binary hypercubes of up to 16 processors. To allow analyses of up to 1024 processors, we assume that each transputer has 10 communication links. The communication link values used for simulating transputer networks and networks based on the IBM SP2 High Performance Switch are shown in Table 1.

Link Type	Start up (s)	Transfer capacity (s/byte)
Transputer	1.76×10^{-4}	5.67×10^{-7}
IBM SP2 HPS	42.0×10^{-6}	28.57×10^{-9}

Table 1: Communication Link Values

Figure 5 shows the harmonic mean performance of the Livermore Loops on the 2D mesh topologies using the step, X and Y routing, binary hypercube, and a transputer based MIN. Each topology is simulated using the transputer processing nodes and transputer communication link values. These results were obtained by using Parsim [14], a message passing computer simulator with a buffer size for the wormhole routing of 1000 bytes.

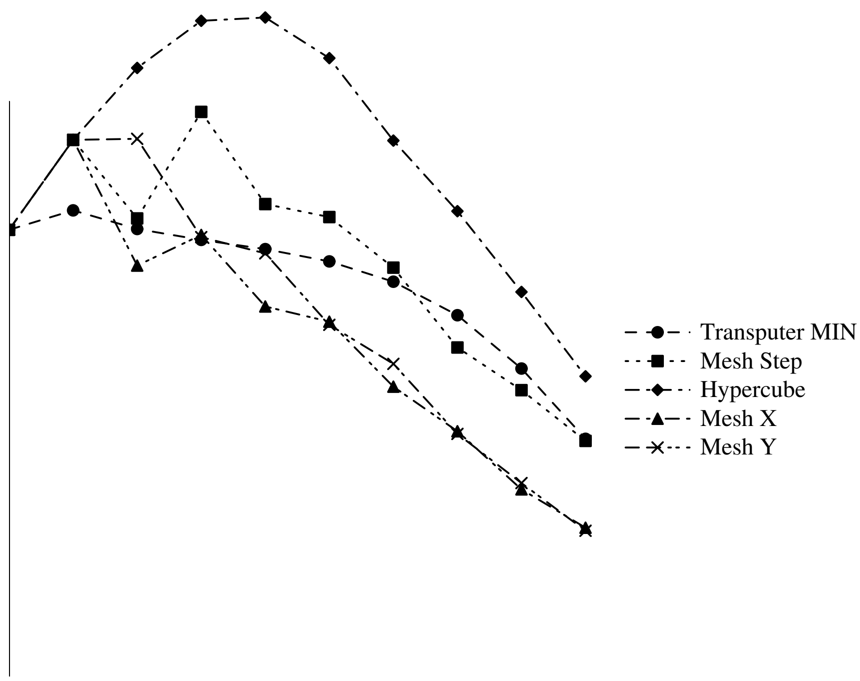

Figure 5: Harmonic Mean Topology Comparison

To allow a comparison between the various topologies, we define $HC(p)$ as the harmonic mean performance of the hypercube at p processors, and similarly $MS(p)$, $MX(p)$, $MY(p)$, and $MIN(p)$ as the harmonic mean performance of the mesh using step, X and Y routing and the performance of the transputer based MIN. We can then define the Average Topology Performance (ANP) as equation 10.

$$ANP(p) = \frac{HC(p) + MS(p) + MX(p) + MY(p) + MIN(p)}{5} \qquad (10)$$

We then define the Relative Performance Metric (RPM) for each topology as the performance of the topology relative to the average performance of all compared topologies. This can be calculated by equations 11-15.

$$HC^* = \sum_{i=1}^{N} \frac{HC(i)/ANP(i)}{N} \tag{11}$$

$$MS^* = \sum_{i=1}^{N} \frac{MS(i)/ANP(i)}{N} \tag{12}$$

$$MX^* = \sum_{i=1}^{N} \frac{MX(i)/ANP(i)}{N} \tag{13}$$

$$MY^* = \sum_{i=1}^{N} \frac{MY(i)/ANP(i)}{N} \tag{14}$$

$$MIN^* = \sum_{i=1}^{N} \frac{MIN(i)/ANP(i)}{N} \tag{15}$$

From Figure 5 these are calculated as shown in Table 2. The RPM indicates the suitability of a topology

HC*	1.29
MS*	1.03
MIN*	0.99
MY*	0.86
MX*	0.82

Table 2: Relative Performance Metrics

for the algorithm. The larger the performance metric, the more suitable the topology is for the algorithm. Therefore we see that the hypercube is the most suitable of the topologies compared for the Livermore Loops. This can be explained by the fact that the number of active links for the hypercube increases as $\log_2 P$ where P is the number of processors, thus reducing the communication bottlenecks.

6 CONCLUSIONS

In this paper, we derived a model for wormhole communication and calculate the optimum buffer size as

$$B = \sqrt{\frac{M \times \beta}{(D-1) \times \alpha}}$$

This routing method can then be applied to other families of distributed memory systems. For large messages, the communication time using wormhole routing increases as $\sqrt{D}$, where D is the diameter of the network.

The simulated results using the Livermore Loops shows that the hypercube topology does provide a 26% performance improvement over the mesh topology. However, this is at the expense of having 2.5 times the number of communication links. Further, even though the network diameter of the mesh is six times that of the hypercube, the use of wormhole routing reduces the impact of the larger diameter.

Using this fact, it is expected that even though the mesh topology has a larger diameter than an equivalent hypercube, the use of wormhole routing and fewer communication links will make the mesh the dominant topology for the future.

REFERENCES

[1] A. Averbuch, E. Gabber, B. Gordissky, and Y. Medan. A parallel FFT on a MIMD machine. *Parallel Computing*, 15:61–74, 1990.

[2] D. Mitchell et al. *Inside The Transputer*. Blackwell Scientific Publications, Melbourne, 1990.

[3] L. Ni et al. A survey of wormhole routing techniques in direct networks, *Computer*, pages 62–76, February 1993.

[4] C. Chang et al. The performance improvement of wormhole router for multicomputer systems. In IEEE Region 10 International Conference, editor, *Proceedings TENCON '93 : 1993 IEEE Region 10 Conference on computer, communication, control and power engineering*, pages 254–257, Beijing, China, October 1993. International Academic Publishers.

[5] R. V. Boppana and S. Chalasani. A comparison of adaptive wormhole routing algorithms. *Computer Architecture News*, 21:351–360, 1993.

[6] Inmos Ltd, Sydney. *Transputer Technical Notes*, 1989.

[7] E. L. Lafferty, M. J. Prelle, M. C. Michaud, and J. B. Goethert. *Parallel computing—An introduction*. Noyes Data Corporation, New Jersey, 1993.

[8] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Computers*, C-36:547–553, May 1987.

[9] C. Izu, A. Arruabarrena, and R. Beivide. Analysis and evaluation of message management functions for bidimensional transputer networks. *Microprocessors and Microsystems*, 16(6):301–309, 1992.

[10] D. Talia. Message routing systems for transputer based multicomputers. *IEEE Micro*, pages 62–72, June 1993.

[11] C. Stunkel et al. The SP2 communication subsystem. Technical report, IBM Thomas J Watson Research Center, August 1994.

[12] C. Stunkel, D. Shea, D. Grice, P. Hochschild, and M. Tsao. The SP1 High-Performance Switch. In *Proc. Scalable High Performance Computing Conference*, pages 779–804, Knoxville, Tennessee, May 1994.

[13] G. Wightwick et al. A numeric weather prediction model for the IBM SP2 parallel computer. In Dr V. L. Narasimhan, editor, *IEEE First ICA^{3}PP*, volume 1, pages 237–245, April 1995.

[14] A. Symons and V. L. Narasimhan. The design and application of PARSIM—a message PAssing computeR SIMulator. *IEE Proceedings: Digital Technology*, 144(1), January 1994.

[15] F. H. McMahon. The Livermore fortran kernels: A computer test of the numerical performance range. *Lawrence Livermore National Laboratory*, UCRL-53745, December 1986.