All-to-All Personalized Communication on a Wraparound Mesh

Abstract

Application programs on distributed memory parallel computers often require an all-to-all personalized communication pattern, where each processor sends different messages to every other processor. We propose optimal data movement algorithms for all-to-all personalized communication on wraparound (torus) meshes, assuming circuit-switched or wormhole routing and a one-port communication model. Using this algorithm, the number of phases on wraparound meshes is shown to be half that of meshes. This algorithm can also apply to wraparound multi-dimensional and rectangular meshes. We also analyze the all-to-all personalized communication performance of k-ary n-cube networks. Results of experiments on the AP1000 show that this algorithm provides efficient all-to-all personalized communication.

1 Introduction

Effective use of communication resources is crucial to good overall performance in highly parallel systems with interconnection networks such as hypercubes, meshes, and torus networks. Four well known, frequently used data movement operations are: (1) one-to-all broadcasting, where a processor sends the same message to all processors; (2) one-to-all personalized communication, where a processor sends a different message to every other processor; (3) all-to-all broadcasting, where each processor sends a message to all other processors; and (4) all-to-all personalized communication, where each processor sends private messages to every other processor [1]. Of these four, all-to-all personalized communication has the largest bandwidth requirements. This communication is used in matrix transposition, the conversion between different data structures, and FFT algorithms.

Several algorithms for all-to-all personalized communication on hypercubes have been studied [1, 2, 4, 5, 6, 7]. Johnsson proposed optimal algorithms for all-to-all personalized communication on hypercubes using packet-switched (store-and-forward) routing [1]. However, store-and-forward routing is not efficient because of communication overhead. To utilize all channels, his algorithm assumes multi-port communication in which a processor can send and receive messages at the same time, but systems implementing this communication are difficult to build. Hence, one-port communication, in which a processor can send or receive one message at a time, is more common.

Circuit-switched and wormhole routing have been developed and used to improve store-and-forward communication [8]. The latency in these routing methods is proportional to message size, not to the distance between processors. Second-generation message-passing machines such as iPSC/2 [16], iPSC/860, and nCUBE/2 [17] have hypercube networks using the circuit-switched [14] and wormhole routing. Take and Boppana proposed optimal all-to-all personalized algorithms on hypercube networks with these routing methods [6, 7]. If the number of processors is P, hypercube networks need P phases. Ho and Bokhari have proposed algorithms to balance data transfer and start-up times. These have been implemented on iPSC/2 and iPSC/860 [2, 4, 5].

Some parallel machines have begun using mesh topologies rather than hypercubes. Dally has shown that low-dimensional networks have lower latency and higher hot-spot throughput than high-dimensional networks [9]. The J-Machine [10] uses a three-dimensional mesh, and the Ametek [15] and the Touchstone project use two-dimensional meshes. All three use wormhole routing. Scott has proposed an optimal all-to-all communication algorithm on a mesh topology [3].

In this paper, we propose some optimal all-to-all communication algorithms on wraparound (torus) networks. The torus network is twice as quick as the mesh. All-to-all personalized communication on a two-dimensional torus needs $P^{\frac{3}{2}}/8$ phases, for example, while two-dimensional meshes need $P^{\frac{3}{2}}/4$ phases [3]. Our algorithm can be applied to higher dimensional torus networks. We also present algorithms on torus networks with both unidirectional and bidirectional channels. We assume one-port communication between processors and networks.

It is important to choose a network topology that fits the requirements of certain classes of communications [18]. Dally analyzed the random traffic throughput performance of *k-ary n-cube* networks assuming a constant bisection width [9], i.e., the number of wires cut when the network is divided into two equal halves. We will analyze all-to-all personalized communication performance of *k-ary n-cube* networks assuming a constant bisection width and constant wiring connected to each processor and board on which several processors are mounted.

No data on experiments with all-to-all personalized communication on real machines with torus networks has been reported yet. The AP1000 [11] and iWARP [13] are now available as parallel machines with torus networks. The AP1000, on which we implemented and evaluated our algorithm, has a two-dimensional torus topology and uses wormhole routing.

The outline of the paper is as follows. In Section 2 we presents optimal algorithms for all-to-all personalized communication on torus networks. In Section 3 we analyze all-to-all personalized communication performance of *k-ary n-cube* networks. Section 4 presents implementation results of all-to-all personalized communication on the AP1000. Conclusion follows in Section 5.

2 All-to-All Personalized Communication

2.1 Communication Model

We assume that a processor can send and receive one message at a time (one-port communication). We consider two channel models: (1) bidirectional, in which a channel can transmit data in both directions at the same time, and (2) unidirectional, in which a channel can transmit data in only one direction at a time. We also assume circuit-switched or wormhole routing.

The total time required for all-to-all personalized communication is given as

$$T_{aac} = L(C_S + \frac{m}{C_T}), \tag{1}$$

where C_S is the transfer start-up time, C_T is the data transfer rate, m is the message size, and L is the number of transfer phases. We can drop LC_S for large m, so, in this section and the next one, we only consider $L\frac{m}{C_T}$.

2.2 Lower Bound

To calculate the lower bound of transfer phases for all-to-all personalized communications, we make the transfer distance between adjacent processors one. The bandwidth required to complete all-to-all personalized communication is given by multiplying the mean distances between processors, D, by the number of processors, P, and the number of messages to be sent by each processor, S. The lower bound is given by the ratio of required bandwidth to the maximum available bandwidth N[1]. The lower bound of transfer phases for all-to-all personalized communication is

$$L = \frac{PDS}{N}. \tag{2}$$

[1]We consider a unidirectional channel as one and a bidirectional channel as two.

2.3 One-Dimensional Torus Networks

2.3.1 Lower bound

We calculate the lower bound of all-to-all personalized communication on one-dimensional torus networks as in Section 2.2. If the number of processors is a, the mean distance between processors is $\frac{a}{4}$ for even a and $\frac{a^2-1}{4a}$ for odd a. N is a for a unidirectional channel and $2a$ for a bidirectional channel. The lower bound on a unidirectional channel is

$$L_{1u} = \frac{a \times \frac{a}{4} \times a}{a} = \frac{a^2}{4}. \tag{3}$$

The lower bound on a bidirectional channel is

$$L_{1b} = \frac{a \times \frac{a}{4} \times a}{2a} = \frac{a^2}{8}. \tag{4}$$

2.3.2 Algorithm on one-dimensional torus networks

In an algorithm we found that achieves the lower bound, a is an even number greater than seven. Except for special phases described below, each phase is a one-directional cycle that consists of four processors, A, B, C, and D. a processors can be divided into two groups, G_1 and G_2. A and B are chosen from group G_1, while C and D are chosen from group G_2 so that C and D are $\frac{a}{2}$ away from A and B.

We represent the cycle (A→B→C→D→A) as

$$\alpha : \beta : \alpha : \beta, \tag{5}$$

where α and β are distances between processors. For example, the cycle in Figure 1 is represented as $1:3:1:3$.

If a is a multiple of four, we can consider the following $\frac{a}{4}-1$ representations,

$$\begin{cases} 1 : \frac{a}{2}-1 : 1 : \frac{a}{2}-1 \\ 2 : \frac{a}{2}-2 : 2 : \frac{a}{2}-2 \\ 3 : \frac{a}{2}-3 : 3 : \frac{a}{2}-3 \\ \qquad \cdots \\ \frac{a}{4}-1 : \frac{a}{4}+1 : \frac{a}{4}-1 : \frac{a}{4}+1. \end{cases} \tag{6}$$

If a is not a multiple of four, we can consider the following $\frac{a-2}{4}$ representations,

$$\begin{cases} 1 : \frac{a}{2}-1 : 1 : \frac{a}{2}-1 \\ 2 : \frac{a}{2}-2 : 2 : \frac{a}{2}-2 \\ 3 : \frac{a}{2}-3 : 3 : \frac{a}{2}-3 \\ \qquad \cdots \\ \frac{a-2}{4} : \frac{a+2}{4} : \frac{a-2}{4} : \frac{a+2}{4}. \end{cases} \tag{7}$$

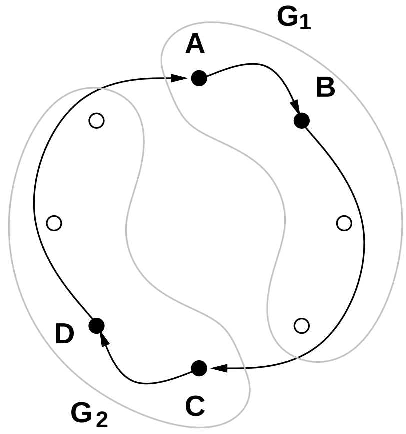

Figure 1: A cycle ($a = 8$)

In these cases, each representation has a phases.

In addition to the above representations, if a is a multiple of four, we must consider the following $\frac{a}{2}$ representation,

$$\frac{a}{4} : \frac{a}{4} : \frac{a}{4} : \frac{a}{4}. \tag{8}$$

All phases consisting of four processors are considered. There are other phases that consist of two processors:

$$\frac{a}{2} : \frac{a}{2} \tag{9}$$

In this case two of other processors send data to themselves. This has $\frac{a}{2}$ phases.

If a is a multiple of four, the total number of phases for (6), (8), and (9) is

$$a \times (\frac{a}{4} - 1) + \frac{a}{2} + \frac{a}{2} = \frac{a^2}{4}. \tag{10}$$

If a is not a multiple of four, the total number of phases for (7) and (9) is

$$a \times (\frac{a-2}{4}) + \frac{a}{2} = \frac{a^2}{4}. \tag{11}$$

In both cases, this algorithm can achieve lower bound L_{1u}. Figure 2 shows all phases on a one-dimensional torus with eight processors. Black points represent communication processors.

For the bidirectional channel algorithm, we choose two cycles from phases in the unidirectional algorithm (Figure 3). These two cycles have opposite directions and all eight processors composing the two cycles must be different because we assume one-port communication. This algorithm assumes that a is greater than seven.

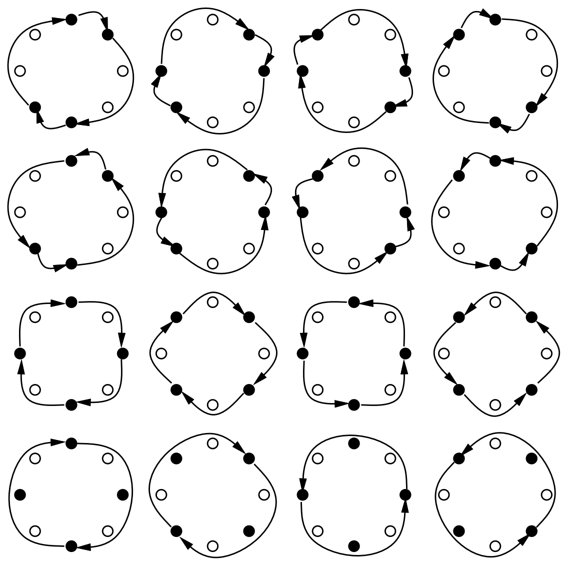

Figure 2: All phases on a one-dimensional torus with eight processors

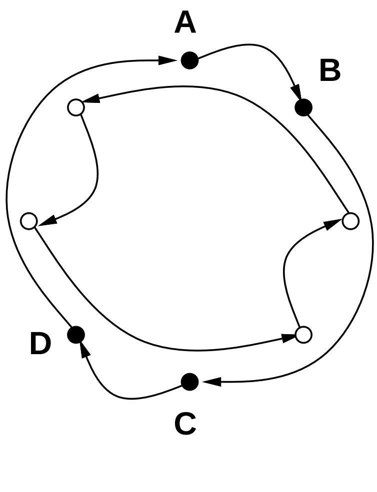

Figure 3: Two cycles

If a is a multiple of four, two cycles can be realized simultaneously for all phases in (6), (8) and (9). The total number of all phases can be halved:

$$\frac{1}{2} \times \frac{a^2}{4} = \frac{a^2}{8} \tag{12}$$

If a is not a multiple of four, all phases but one can be realized at the same time. This is because the number of cycles consisting of two processors, $\frac{a}{2}$, becomes odd. The number of all phases is

$$\frac{1}{2} \times a \times (\frac{a-2}{4}) + \frac{1}{2} \times (\frac{a}{2} - 1) + 1 = \frac{a^2+4}{8}. \tag{13}$$

If a is a multiple of four and greater than seven, this algorithm can implement lower bound L_{1b}.

2.4 Two-Dimensional Torus Networks

2.4.1 Lower bound

We want to find the lower bound for a two-dimensional wraparound mesh (torus) that consists of $a \times a$. The mean distance between processors in a two-dimensional torus is $\frac{a}{2}$ for even a. The number of processors is a^2. The maximum available bandwidth is $2a^2$ for unidirectional channels and is $4a^2$ for bidirectional channels. We can calculate each lower bound as follows:

$$L_{2u} = \frac{a^2 \times \frac{a}{2} \times a^2}{2a^2} = \frac{a^3}{4} \tag{14}$$

$$L_{2b} = \frac{a^2 \times \frac{a}{2} \times a^2}{4a^2} = \frac{a^3}{8} \tag{15}$$

2.4.2 Algorithm for a two-dimensional torus

The algorithm for a two-dimensional torus is implemented as the cross product [3] of the one-dimensional algorithm. We assume that messages are routed first horizontally and then vertically. Any path in a two-dimensional torus can be viewed as composed of one horizontal and one vertical message path, where the destination of the horizontal path is the source of the vertical path (Figure 4).

Each of the phases in the vertical direction can be matched with all phases in the horizontal direction. This gives a total of $(\frac{a^2}{4})^2 = \frac{a^4}{16}$ phases. This does not achieve the lower bound because only four processors are used in each direction. With proper selection it is possible to have more pairs that are independent in the same phase. If a is a multiple of four, we can overlap $\frac{a}{4}$ phases during one phase. Figure 5 shows an example of overlapping. (a) and (b),

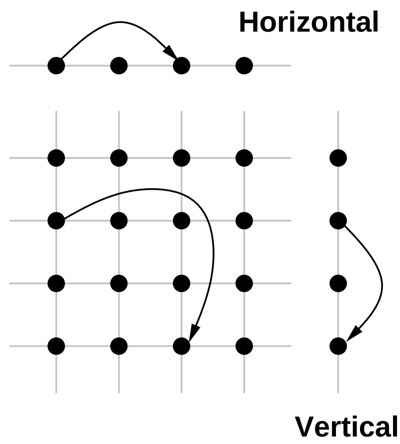

Figure 4: A cross product

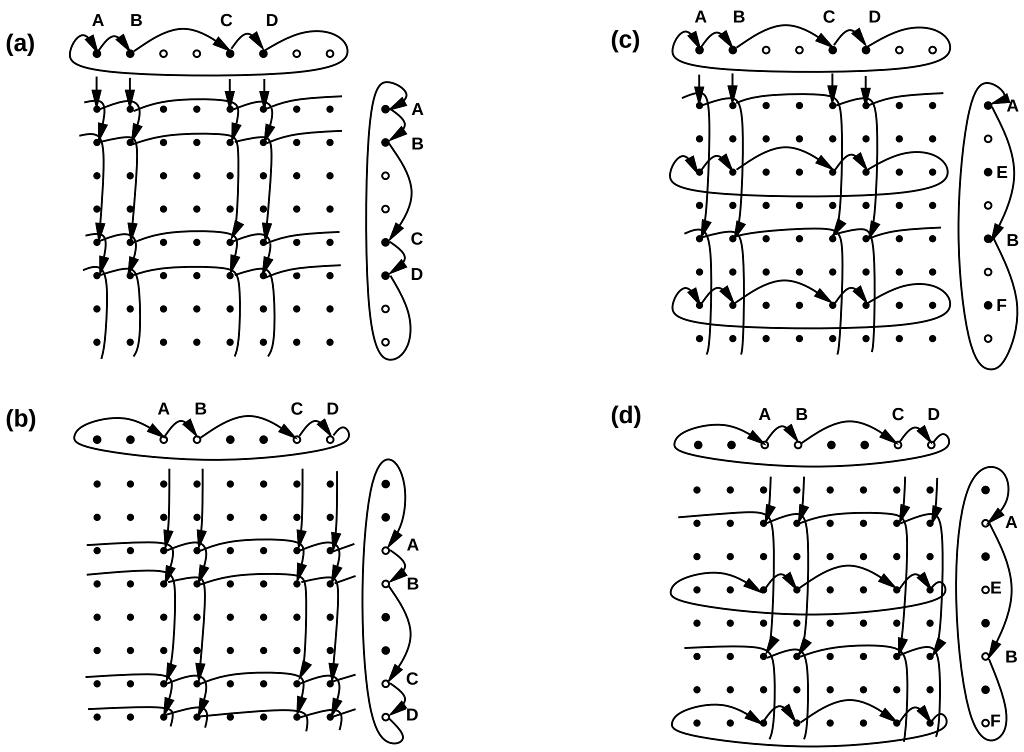

Figure 5: Overlaps

(c) and (d) can be overlapped in the same phase. The number of phases needed for all-to-all communication using unidirectional channels is

$$\frac{a^4}{16} \div \frac{a}{4} = \frac{a^3}{4}. \tag{16}$$

This is the same as lower bound L_{2u}.

We can apply this method to torus networks with bidirectional channels. In the one-dimensional algorithm, each direction uses eight processors. If a is a multiple of eight, the number of phases needed for all-to-all communication using bidirectional channels is

$$(\frac{a^2}{8})^2 \div \frac{a}{8} = \frac{a^3}{8}. \tag{17}$$

This is same as lower bound L_{2b}.

2.5 Rectangular Torus Networks

Our algorithm is optimal for rectangular torus topologies such as 128 (8×16) or 512 (16×32).

2.5.1 Lower bound

To calculate the lower bound for rectangular torus networks, $a \times b$ (a and b are even), divide one system into two groups, F_1 and F_2 (Figures 6 and 7). During all-to-all communication

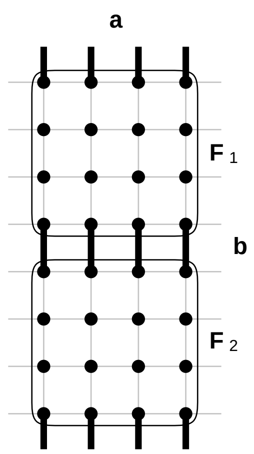

Figure 6: Divided rectangle (1)

between F_1 and F_2, $2 \times (\frac{ab}{2})^2$ messages are transferred through the channels shown as solid lines in Figure 6 and 7. For unidirectional channels, there are $2a$ channels in Figure 6 and $2b$ in Figure 7. All-to-all communication between F_1 and F_2 needs at least

$$2 \times (\frac{ab}{2})^2 \div 2a = \frac{ab^2}{4} \tag{18}$$

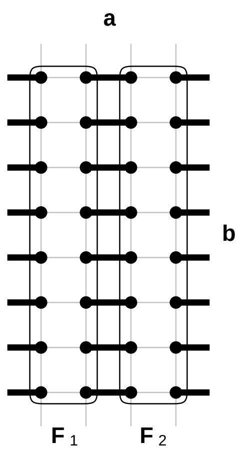

Figure 7: Divided rectangle (2)

phases in the case of Figure 6 and

$$2 \times (\frac{ab}{2})^2 \div 2b = \frac{a^2 b}{4} \tag{19}$$

phases in the case of Figure 7.

If $b \geq a$, $\frac{ab^2}{4} \geq \frac{a^2 b}{4}$, all-to-all personalized communication on rectangular torus networks needs at least $\frac{ab^2}{4}$ phases. Messages needed for all-to-all communications between two groups can be maximized when the number of processors in each groups is equal. The number of channels between groups can be minimized when divided as in Figure 6. Lower bound L_{2u}^r on rectangular torus networks with unidirectional channels is $\frac{ab^2}{4}$ and lower bound L_{2b}^r on rectangular torus networks with bidirectional channels is $\frac{ab^2}{8}$.

2.5.2 Algorithm for rectangular torus networks

The algorithm on square torus networks (Section 2.4.2) can be applied to rectangular torus networks with only slight modification. The difference is the number of phases overlapping. If $b \geq a$, we can overlap $\frac{a}{4}$ phases in the case of unidirectional channels. If a is a multiple of four, the number of phases is

$$\frac{a^2}{8} \times \frac{b^2}{8} \div \frac{a}{4} = \frac{ab^2}{4} \tag{20}$$

which is the same as lower bound L_{2u}^r.

Similarly, in the case of bidirectional channels, we can overlap $\frac{a}{8}$ phases. If a is a multiple of eight, the number of phases is

$$\frac{a^2}{4} \times \frac{b^2}{4} \div \frac{a}{8} = \frac{ab^2}{8} \tag{21}$$

which is the same as lower bound L_{2b}^r.

2.6 Higher Dimensional Torus Networks

In general, the lower bound of n-dimensional torus networks $a_1 \times a_2 \times \cdots \times a_n$ is

$$L_{nu}^r = \frac{a_m \prod_{i=1}^{n} a_i}{4} = \frac{a_m P}{4} \tag{22}$$

for bidirectional channels and

$$L_{nb}^r = \frac{a_m P}{8} \tag{23}$$

for unidirectional channels, where P is the number of processors and a_m is the maximum among $a_1, a_2, \cdots, a_n$.

In the case of unidirectional channels, the algorithm, which composes n one-dimensional phases and overlaps independent phases in one phase, needs $\frac{a_m P}{4}$ phases if $a_1, a_2, \cdots, a_n$ (except a_m) are multiples of four. This is the same as lower bound L_{nu}^r. In the case of bidirectional channels, $\frac{a_m P}{8} = L_{nb}^r$ can also be achieved if $a_1, a_2, \cdots, a_n$ (except a_m) are multiples of eight.

If $a_1 = a_2 = \cdots = a_n = a$ and a is a multiple of four, lower bound L_{nu} for all-to-all personalized communication on n-dimensional torus networks with unidirectional channels is

$$L_{nu} = \frac{P^{\frac{n+1}{n}}}{4}. \tag{24}$$

If a is a multiple of eight, lower bound L_{nb} for all-to-all personalized communication in n-dimensional torus networks with bidirectional channels, is

$$L_{nb} = \frac{P^{\frac{n+1}{n}}}{8}. \tag{25}$$

2.7 Comparison With Meshes

Scott has proposed an optimal all-to-all personalized communication algorithm on a mesh topology [3]. The number of phases for all-to-all personalized communication on multi-dimensional torus networks is $\frac{P^{\frac{n+1}{n}}}{2}$ for unidirectional channels and $\frac{P^{\frac{n+1}{n}}}{4}$ for bidirectional channels. This shows that a multi-dimensional torus is twice as fast as a multi-dimensional mesh.

3 Topology Comparison

All-to-all personalized communication has the highest bandwidth requirements. In this section, we analyze network topologies using all-to-all personalized communication performance. Here we consider only *k-ary n-cube* networks. We will first compare networks without considering network cost, and then compare networks under constant network cost.

3.1 Comparison Without Considering Network Cost

Figure 8 compares the transfer time for a constant transfer rate with one channel. Hypercubes

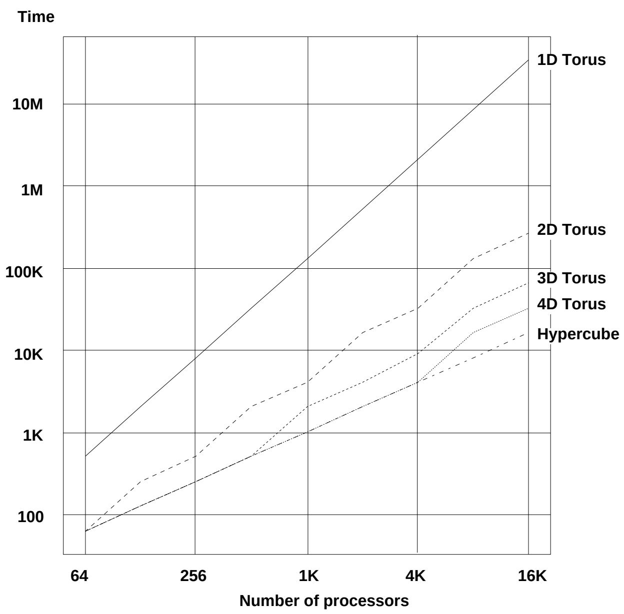

Figure 8: Topology comparison

can always perform all-to-all personalized communication in phases of the number of processors. Hypercubes implement the lower bound for this communication, but do not use all the channels in each phase. In accordance with Section 2.2, the lower bound on hypercubes is

$$L_{hypercube} = \frac{P\frac{\log P}{2}P}{P\log P} = \frac{P}{2}. \tag{26}$$

Using all channels on a hypercube violates the assumption of one-port communication. This applies to other topologies such as two-dimensional networks with fewer than 64 processors, three-dimensional networks with fewer than 512 processors, and four-dimensional networks with fewer than 4096 processors. The lower bound on these networks is bounded by the bandwidth between processor and network, not by the network bandwidth.

3.2 Comparison Considering Network Cost

To build networks, we must consider network cost. Dally considered *bisection width*, the number of wires cut when the network is divided into two equal halves [9]. This is the network cost when all processors are mounted on one board. We also consider the number of wires connected to one processor and one board as network costs. The assumption of a constant number of wires connected with one processor is the same as that of constant wires in a network.

Assuming 16 processors on one board, Table 1 shows the all-to-all personalized communication performance and network cost.

Table 1: All-to-all personalized communication (AAPC) and network cost

Network cost	1D	2D	3D	4D	Hypercube
AAPC	$P^2/8$	$P^{\frac{3}{2}}/8$	$P^{\frac{4}{3}}/8$	$P^{\frac{5}{4}}/8$	P
Bisection	4	$4P^{\frac{1}{2}}$	$4P^{\frac{2}{3}}$	$4P^{\frac{3}{4}}$	P
1 processor	4	8	12	16	$2\log P$
1 board	4	32	80	128	$32(\log P - 4)$

† P: number of processors

Under a constant *bisection width*, all topologies but hypercubes have the same performance. Hypercube performance is half that of the others because hypercubes have idle channels in all-to-all personalized communication. If we assume multi-port communication, all topologies have the same performance. This also applies to mesh topologies. The performance on meshes is half of that on torus networks. However, the *bisection width* is also half that of torus networks. Under a constant *bisection width*, torus networks and meshes have the same performance because all topologies have the same network capacity [9] and all-to-all personalized communication uses all channels.

Figure 9 shows the performance with a constant number of wires connected to each processor. Figure 10 shows the performance with a constant number of wires connected to each board.

Under these assumptions, no difference in performances exists among three-dimensional and four-dimensional torus networks and hypercubes with fewer than 16K processors. Network topologies are determined by applications and costs. For parallel systems often using all-to-all personalized communication, networks will require torus networks with more than two-dimensional.

4 Experiments

In previous sections, we considered network throughput alone. To implement all-to-all personalized communication, however, we must also consider transfer set-up time. We implemented our algorithm on the AP1000 to verify its effectiveness.

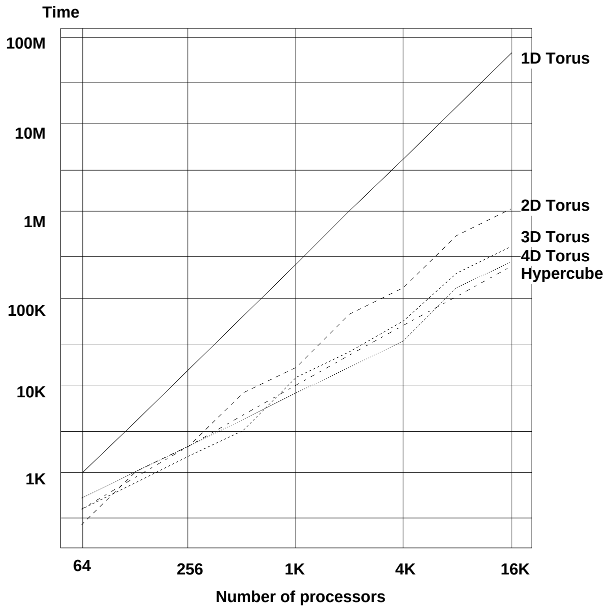

Figure 9: Topology comparison with a constant number of wires connected to each processor

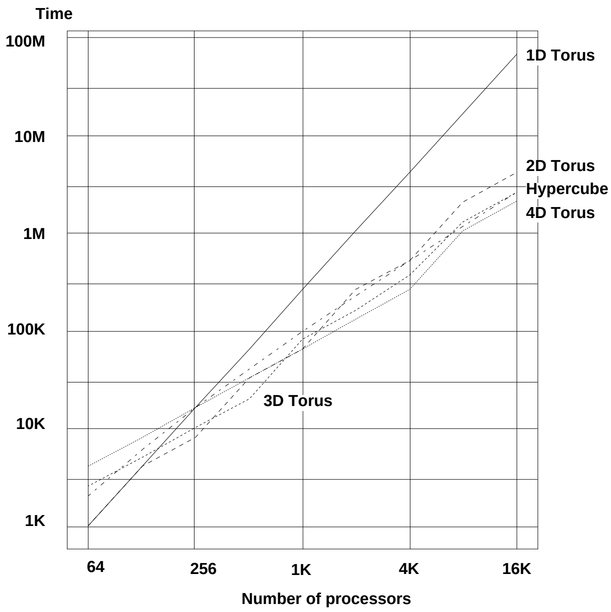

Figure 10: Topology comparison with a constant number of wires connected to one board

4.1 AP1000 Architecture

The AP1000 is a distributed-memory, message-passing, parallel computer [11]. It consists of 16 to 1024 processors connected by three independent networks called the torus network (T-Net), the broadcast network (B-Net), and the synchronization network (S-Net). A message controller (MSC) interfaces processors and networks to minimize message handling overhead. Figure 11 shows the AP1000 architecture.

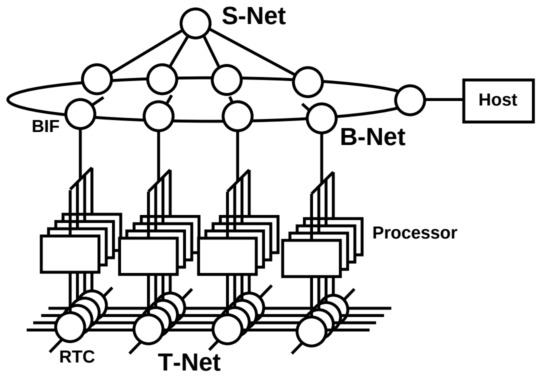

Figure 11: AP1000 Configuration

All-to-all personalized communication can be implemented efficiently using the T-Net. The T-Net has a two-dimensional torus topology and uses wormhole routing [12]. The T-Net consists of RTC routing chips and each RTC has four unidirectional channels. This means that each channel can transfer data in only one direction at a time.

Our algorithm needs barrier synchronization because not every processor sends messages in each transfer phase, and each phase must be isolated. The specialized S-Net network performs this function.

4.2 Experiments on the AP1000

To predict the time required for all-to-all personalized communication, we measured the time for point-to-point communication and barrier synchronization.

The time for a message of m bytes to be transferred between processors is $1.1 + m/17.1$ microseconds for unidirectional transfer and $1.1 + m/7.14$ microseconds for bidirectional transfer. The transfer rate of each channel for bidirectional transfer is less than half of that for unidirectional transfer because of the overhead for changing channel directions, which is why we implemented the unidirectional algorithm. The time for barrier synchronization, 10.5 microseconds, is independent of the number of processors. To implement all-to-all personalized communication, fast barrier synchronization is important to reduce the transfer start-up time, C_S. From these measurements, the predicted time for all-to-all personalized communication is $L_u(C_S + m/C_T) = L_u(11.6 + m/17.1)$ microseconds for unidirectional transfer. L_u is 128 for 64 processors and 1024 for 256 processors. This measurement shows that the the C_S including barrier synchronization is negligible for large messages.

Figure 12 shows the results for all-to-all personalized communication algorithms on the AP1000 with 64 (8×8) and 256 (16×16) processors. The dotted lines indicate predicted data and solid lines show actual measurements.

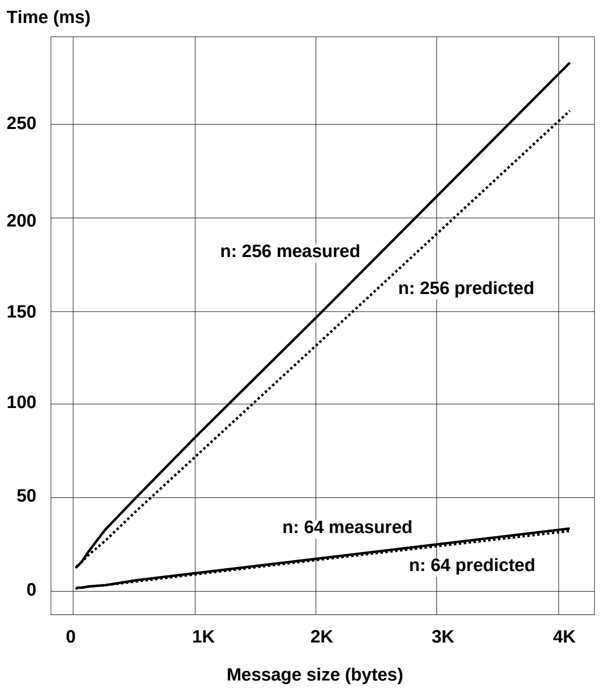

Figure 12: Experimental results

Predicted and observed run times agree well, indicating that our algorithm sustains a maximum transfer rate.

5 Conclusion

We have proposed optimal all-to-all personalized communication on torus networks. A torus network using this algorithm is twice as quick as a mesh. This algorithm can also apply to one-dimensional and multi-dimensional torus networks.

We compared networks for all-to-all personalized communication performance. With a constant number of wires connected to each processor and board, the performance among three-dimensional and four-dimensional torus networks and hypercubes is the same.

We implemented the proposed algorithms on the AP1000 and showed that the algorithm sustained maximum communication performance. Using the S-Net for barrier synchronization to isolate each transfer phase, we were able to efficiently implement all-to-all personalized communication on the AP1000.

We assumed that the time needed to transfer messages exceeds transfer set-up time, but, if the number of processors increases, this will no longer be true. We must also consider cases dealing with short messages.

References

[1] S. L. Johnsson and C. T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes," *IEEE Transaction on Computers*, Vol. 38, No. 9, 1989, pp. 1249–1268.

[2] C. T. Ho and M. T. Raghunath, "Efficient Communication Primitives on Circuit-Switched Hypercubes," *6th Distributed Memory Computing Conference*, 1991, pp. 390–397.

[3] D. S. Scott, "Efficient All-to-All Communication Patterns in Hypercubes and Mesh Topologies," *6th Distributed Memory Computing Conference*, 1991, pp. 398–403.

[4] S. H. Bokhari, "Multiphase Complete Exchange on a Circuit Switched Hypercube," *NASA Contractor Report 187499*, 1991.

[5] S. H. Bokhari, "Complete Exchange on the iPSC-860," *NASA Contractor Report 187498*, 1991.

[6] R. V. Boppana and C. S. Raghavendra, "All-to-All Personalized Communication on Circuit Switched Hypercubes," *Tech. Report CENG 90-20*, Dept. of EE-Systems, USC, 1990.

[7] R. Take, "An Optimal Routing Method of All to All Communication on Hypercube Networks," *35th IPSJ Conference*, 1987 (in Japanese).

[8] W. J. Dally, "A VLSI Architecture for Concurrent Data Structures," Kluwerr Academic Publishers, 1987.

[9] W. J. Dally, "Performance Analysis of k-ary n-cube Interconnection Networks," *IEEE Transactions on Computers*, 39 (6), June 1990, pp. 775–785.

[10] W. J. Dally, et. al., "The J-Machine: A Fine-Grain Concurrent Computer," *Information Processing*, 1989, pp. 1147–1153.

[11] H. Ishihata, T. Horie, S. Inano, T. Shimizu, and, S. Kato, "An Architecture of Highly Parallel Computer AP1000," *IEEE Pacific Rim Conf. on Communications, Computers and Signal processing*, May 1991, pp. 13–16.

[12] T. Horie, M. Ikesaka, H. Ishihata, "Routing Controller of Cellular Array Processor CAP-II," 90-ARC-83, July 1990, pp.223–228 (in Japanese).

[13] S. Borkar et al., "Supporting Systolic and Memory Communication in iWarp," *The 17th Annual International Symposium on Computer Architecture*, 1990, pp. 70–81.

[14] S. F. Nugent, "The iPSC/2 Direct-Connect Communication Technology," *Third Conference on Hypercube Concurrent Computers and Applications*, Jan. 1988, pp. 610–619.

[15] C. L. Seitz et al., "The architecture and programming of the Ametek Series 2010 multicomputer," *Third Conference on Hypercube Concurrent Computers and Applications*, Jan. 1988, pp. 33–36.

[16] R. Arlauskas, "iPSC/2 system: a second generation hypercube," *Third Conference on Hypercube Concurrent Computers and Applications*, Jan. 1988, pp. 38–42.

[17] "nCUBE2 Supercomputers," *Technical Overview*, 1990.

[18] D. A. Reed and D. C. Grunwald, "The Performance of Multicomputer Interconnection Networks," *IEEE Computer*, June 1987, pp. 63–73.