Proc. 8th International Parallel Processing Symposium, Cancun, Mexico, April 1994

All-to-All Communication on Meshes with Wormhole Routing *

Rajeev Thakur and Alok Choudhary*
Northeast Parallel Architectures Center
111 College Place, Rm 3-228
Syracuse University, Syracuse NY 13244
thakur, choudhar @npac.syr.edu

Abstract

This paper describes several algorithms to perform all-to-all communication on a two-dimensional mesh connected computer with wormhole routing. We discuss both direct algorithms, in which data is sent directly from source to destination processor, and indirect algorithms in which data is sent through one or more intermediate processors. We propose algorithms for both power-of-two and non power-of-two meshes as well as an algorithm which works for any arbitrary mesh. We have developed analytical models to estimate the performance of the algorithms on the basis of system parameters. Performance results obtained on the Intel Touchstone Delta are compared with the estimated values.

1 Introduction

The need for scalable parallel computers has resulted in the mesh emerging as a popular interconnection network for distributed memory multicomputers. The Intel Touchstone Delta, the Intel Paragon and the Symult 2010 use a two-dimensional mesh while the MIT J-machine and the Mosaic computer developed at Caltech use a three-dimensional mesh [5]. All these machines use *wormhole routing*, an important feature of which is that the network latency is almost independent of the path length when there is no link contention and the packet size is large. In this paper, we discuss several algorithms to perform all-to-all communication on a mesh connected computer with wormhole routing. The all-to-all communication pattern (also known as *complete exchange*) is one in which all processors need to communicate with all other processors. It occurs in many applications like parallel quicksort, some implementations of the 2D FFT, matrix transpose, array redistribution etc. It is the densest form of communication which can result in a lot of link contention. Hence it is necessary to use efficient algorithms to perform complete exchange.

Complete exchange algorithms for hypercube and fat tree architectures are described in [2, 6]. These algorithms assume that the number of processors is a power-of-two, which is a valid assumption for those architectures. The mesh architecture introduces different problems because of high contention and the fact that the user can allocate a mesh size which need not be a power-of-two and may even be an odd number (eg. 5×5). Previous work on complete exchange algorithms for a mesh assumes that the number of processors is a power-of-two [3, 7]. In this paper, we discuss algorithms for both power-of-two and non power-of-two meshes. We have developed analytical models to estimate the performance of the algorithms. We present performance results on the Intel Touchstone Delta and compare them with the predicted values.

Section 2 describes the architecture of the Delta and the performance model used for the algorithms. *Direct* algorithms, in which data is sent directly from source to destination processor, are described in Section 3. Section 4 discusses *indirect* algorithms, in which data is sent from source to destination through one or more intermediate processors. The performance of the algorithms on the Delta is discussed in Section 5 followed by Conclusions in Section 6.

2 Architecture and Performance Model

The Intel Touchstone Delta is a 16×32 mesh of computational nodes, each of which is an Intel i860/XR microprocessor. The two-dimensional mesh interconnection network has bidirectional links with wormhole routing. It uses deterministic XY routing in which packets are first sent along the X dimension and then along the Y dimension. In wormhole routing, a packet is divided into a number of *flits* (flow control digits) for transmission. The size of a flit is typically the same as the channel width. The header flit of a packet determines the route and remaining flits follow in a pipeline fashion. The network latency for wormhole routing is $(L_f/B)D + L/B$, where L_f is the length of each flit, B is the channel bandwidth, D is the path length, and

*The authors are also with the Dept. of Electrical and Computer Engineering, Syracuse University

L is the length of the message. Thus, if $L_f << L$, the path length D will not significantly affect the network latency provided there is no link contention. Details of wormhole routing techniques can be found in [5].

To model the performance of the algorithms, we use an approach similar to that used by Barnett et al in [1]. The following notations are used in our models

α	startup time per message
β_{ex}	transfer time per byte for an exchange with no link conflicts
β_{sr}	transfer time per byte to send to and receive from different processors with no link conflicts
β_{sat}	transfer time per byte on a saturated link
L	number of bytes to be exchanged per processor pair
$f(i)$	maximum number of messages contending for a saturated link at step i
r	number of rows in the mesh
c	number of columns in the mesh
p	total number of processors $= r \times c$

The time taken for an exchange operation may be different from the time to send to and receive from different processors, because in the latter case the incoming and outgoing messages may traverse links with different amount of contention. Hence, we use β_{ex} or β_{sr} depending on the algorithm. We assume that the time taken is independent of distance, a property of wormhole routing. Thus, the time required for an exchange step i is given by

$$T = \alpha + L\, max(\beta_{ex}, f(i)\beta_{sat})$$

We assume that conflicting messages share the bandwidth of a network link and that there exists some positive integer γ such that $\beta_{ex} = 2^{\gamma}\beta_{sat}$. For the Delta, $\gamma = 1$ is a good approximation [1]. In other words, even if two messages contend for a link, there is no increase in communication time. Note that since the Delta has bidirectional links, two messages contend for a link only if they need to travel in the same direction simultaneously.

3 Direct Algorithms

Scott [7] has shown that $a^3/4$ is the lower bound on the number of phases required to perform a complete exchange on an $a \times a$ mesh such that there is no link contention in any phase. However, if we allow link contention to exist, the operation can be performed in fewer steps. We have adopted this approach of allowing a small amount of link contention to exist, thereby reducing the number of steps and keeping all processors active at every step. This approach takes advantage of the fact that in machines like the Delta and the Paragon, the links have excess bandwidth, so that a small number of contending messages will not significantly affect the communication time. We first discuss three direct algorithms for complete exchange on a mesh. In direct algorithms, data is sent directly from source processor to destination processor.

3.1 Pairwise Exchange for Power-of-Two Mesh (PEX)

The best algorithm for a hypercube architecture is the pairwise exchange algorithm described in [2, 7], as it guarantees no link contention in the hypercube at every step. This algorithm has also been shown to perform well on the fat tree architecture of the CM-5 [6]. It requires $p-1$ steps and the communication schedule is as follows. In step i, $1 \leq i \leq p-1$, each processor exchanges data with the processor determined by taking the exclusive-or of its processor number with i. Therefore, this algorithm has the property that the entire communication pattern is decomposed into a sequence of pairwise exchanges. Figure 1 shows the communication pattern of PEX on a 2×4 mesh. The complete exchange requires seven steps. In steps 1, 4 and 5 there is no contention. In steps 2, 3, 6 and 7, there are two messages contending for a link.

Since each step of PEX involves an exchange between pairs of processors, the maximum number of messages contending for a link at any step is limited by $max(r, c)/2$. An exact expression for the maximum number of messages contending for a link at step i can be obtained as

$$f(i) = 2^{\lfloor lg\{max(mod(i,c),i/c)\}\rfloor}$$

Hence, the time taken for step i is

$$T(i) = \alpha + L\, max(\beta_{ex}, f(i)\beta_{sat})$$

The cost of PEX can be determined by summing over all steps of the algorithm :

$$\begin{aligned} T_{PEX} &= \textstyle\sum_{i=1}^{p-1}[\alpha + L\, max(\beta_{ex}, f(i)\beta_{sat})] \\ &= (p-1)\alpha + L \textstyle\sum_{i=1}^{p-1} max(\beta_{ex}, f(i)\beta_{sat}) \end{aligned}$$

3.2 Pairwise Exchange for General Mesh (PEX-GEN)

The PEX algorithm cannot be directly used if the number of processors is not a power-of-two as the exclusive-or function will not create all the required processor pairs in $p-1$ steps. The Pairwise Exchange for General Mesh (PEX-GEN) algorithm is an extension of PEX for non power-of-two meshes. The algorithm first finds the smallest power-of-two (say q) greater than the number of processors and uses this number to schedule $q-1$ steps of the pairwise exchange. In each step, every processor checks to see if the calculated destination processor number is less than the actual number of processors. If so, it exchanges data with the processor, else it goes ahead to the next step. Thus, the algorithm requires $q-1$ steps where q is the nearest power-of-two larger than

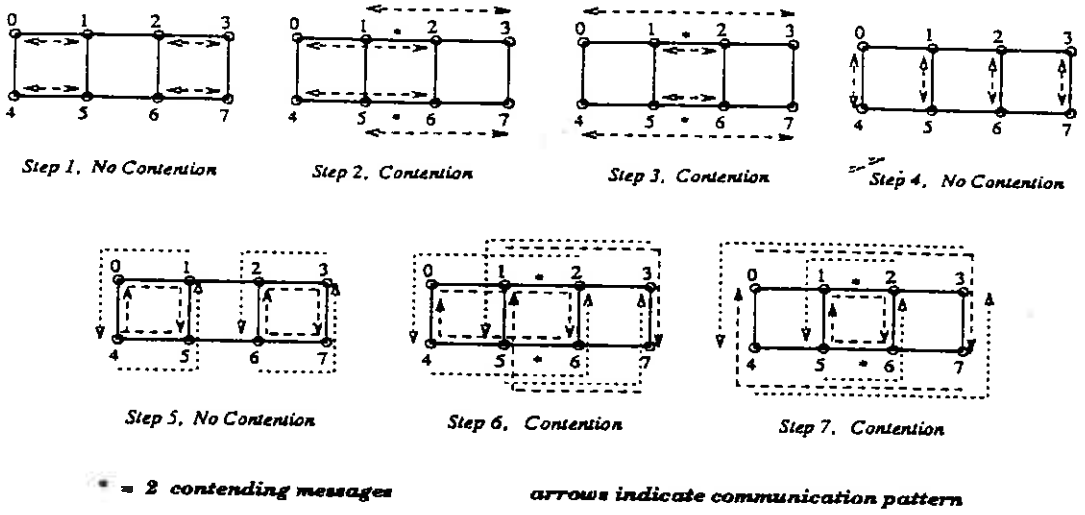

Figure 1: PEX on 2 × 4 mesh

the number of processors. The maximum contention in each step is upper bounded by that in the PEX algorithm.

3.3 General Algorithm for any Mesh (GEN)

For non power-of-two meshes, it would be advantageous to have an algorithm which requires only $p-1$ steps for any value of p. In the General Algorithm for any Mesh (GEN), processor pairs do not exchange with each other. Instead, at step i, a processor j sends data to processor $mod(j+i,p)$ and receives data from processor $mod(j-i+p,p)$. Clearly, this algorithm will require only $p-1$ steps, for any value of p. The maximum number of messages contenting for a link at step i can be obtained as

$$f(i) = min[mod(i,c), c-mod(i,c)]+min[i/c,(p-i)/c]$$

Hence the total time for all steps is :

$$T_{GEN} = \sum_{i=1}^{p-1}[\alpha + Lmax(\beta_{sr}, f(i)\beta_{sat})] = (p-1)\alpha + L\sum_{i=1}^{p-1} max(\beta_{sr}, f(i)\beta_{sat})$$

4 Indirect or Store-and-Forward Algorithms

In *indirect* or *store-and-forward* algorithms, a message is sent from source processor to destination processor through one or more intermediate processors. Indirect algorithms either reduce the number of communication steps by increasing the message size per step, or increase the number of communication steps in order to reduce link contention.

4.1 Recursive Exchange (REX)

In this algorithm, the mesh is first halved in the x direction and each processor sends all data destined for the other half of the mesh to its corresponding processor in the other half. The mesh is further halved recursively in the x direction and this process is repeated. This takes $\lg c$ steps. The mesh is then recursively halved in the y direction and messages are exchanged over each cut, which takes $\lg r$ steps. Thus, the total number of steps is $\lg c + \lg r = \lg p$ which is much less than $p-1$. However the message size in each step is larger ($Lp/2$). Each step incurs the additional overhead of reorganizing the data within each processor. This algorithm works only for power-of-two meshes.

Since the mesh is recursively divided by two and each processor in one partition communicates with its mirror image in the other partition, the maximum number of messages contending for a link at step i can be obtained as

$$f(i) = \begin{cases} c/2^i & \text{for } 1 \le i \le \lg c \\ \frac{r}{2^{i-\lg c}} & \text{for } \lg c < i \le \lg p \end{cases}$$

The cost of REX can be determined by summing over all steps of the algorithm :

$$T_{REX} = \sum_{i=1}^{\lg p}[\alpha + \frac{Lp}{2} max(\beta_{ex}, f(i)\beta_{sat})]$$

which can be expanded to

$$T_{REX} = \alpha \lg p + \frac{Lp}{2}\sum_{i=1}^{\lg c} max(\beta_{ex}, c/2^i\beta_{sat}) + \frac{Lp}{2}\sum_{i=\lg c+1}^{\lg p} max(\beta_{ex}, \frac{r}{2^{i-\lg c}}\beta_{sat})$$

4.2 Indirect Pairwise Exchange (IPEX)

The Indirect Pairwise Exchange (IPEX) algorithm aims at reducing link contention in the direct Pairwise

Exchange (PEX) algorithm. In IPEX, each processor communicates only with the processors in its row and column. Each exchange along a row is followed by a complete exchange along a column. During the row exchange, each processor sends Lr bytes of data to the destination processor, out of which $L(r-1)$ bytes are intended for other processors in the same column as the destination processor. This is followed by a complete exchange along the columns (involving messages of L bytes), in which the data received during the row exchange is sent to the appropriate processors in the same column. This entire operation requires $r(c-1)$ communication steps. Finally, an additional complete exchange is required along the columns for processors to exchange their own data directly with processors in the same column. In this phase, data is sent directly from source to destination, requiring $r-1$ exchange steps. Hence, the total number of steps required is $r(c-1)+(r-1) = rc-1 = p-1$.

The maximum link contention at any step is the same as for pairwise exchange along a row or column which is $2^{\lfloor \lg i \rfloor}$, where i is the step number along the row or column. Hence, the total time required for IPEX is given by :

$$
\begin{aligned}
T_{IPEX} &= \sum_{i=1}^{c-1}[\alpha + L\,r\,max(\beta_{ex}, 2^{\lfloor \lg i \rfloor}\beta_{sat}) \\
&\quad + \sum_{j=1}^{r-1}\{\alpha + L\,max(\beta_{ex}, 2^{\lfloor \lg j \rfloor}\beta_{sat})\}] \\
&\quad + \sum_{i=1}^{r-1}[\alpha + L\,max(\beta_{ex}, 2^{\lfloor \lg i \rfloor}\beta_{sat})]
\end{aligned}
$$

5 Performance

We implemented all the algorithms on the Intel Touchstone Delta and studied their performance for different mesh configurations and message sizes. As suggested in [4], we use *forced* messages (which provide higher bandwidth but also higher startup cost) if the message size is greater than or equal to 1.5 Kbytes and *unforced* messages if the message size is less than 1.5 Kbytes.

Figure 2 shows the performance of the algorithms on a 16×32 mesh for different message sizes. Figure 3 shows the performance of the algorithms for different mesh sizes, keeping the message size constant at 16 Kbytes. We observe that REX performs the worst even though it requires only $\lg p$ steps. There are several reasons for this. First, there is a lot of link contention in each step. Second, the message size per step is increased to $L\,p/2$ instead of L in the direct algorithms. Third, the indirect form of communication requires a lot of data buffering and shuffling in order to send the appropriate data to the appropriate

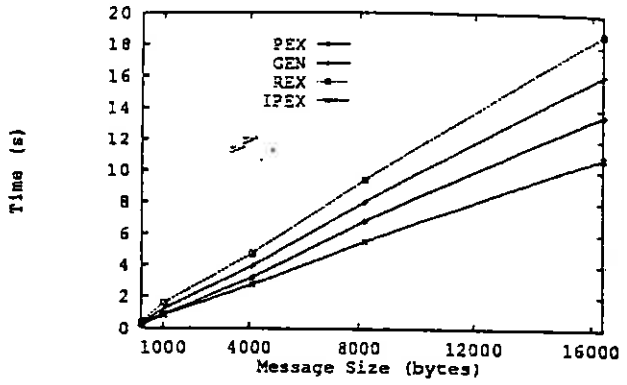

Figure 2: Performance of algorithms on a 16×32 mesh

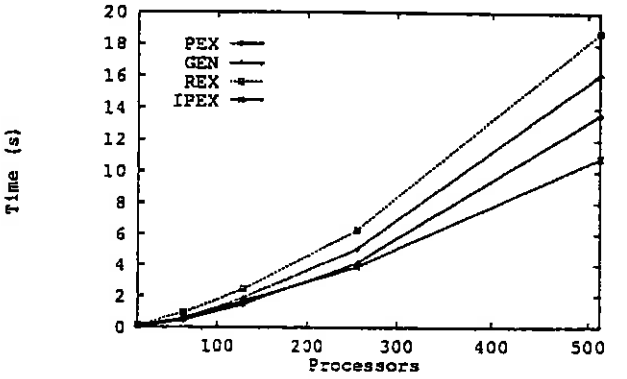

Figure 3: Performance for message size 16 Kbytes

node. This algorithm also has high memory requirements because the large message size requires more memory per node.

GEN performs better than PEX for small message sizes and small number of processors. However, for large number of processors (≥ 64) and large message sizes (> 1 Kbytes) PEX performs better. The GEN algorithm has a certain amount of asymmetry in the communication in the sense that each communication operation consists of a send to one processor and a receive from some other processor. Thus, the incoming and outgoing messages may traverse a different number of links with different amounts of contention, and the path which has the highest amount of contention adversely affects the communication time. On the other hand, in the PEX algorithm, processor pairs exchange with each other at every step, so the incoming and outgoing messages travel the same number of links with the same amount of contention. For large meshes and large message sizes, IPEX performs better than PEX. This is because in large meshes, direct

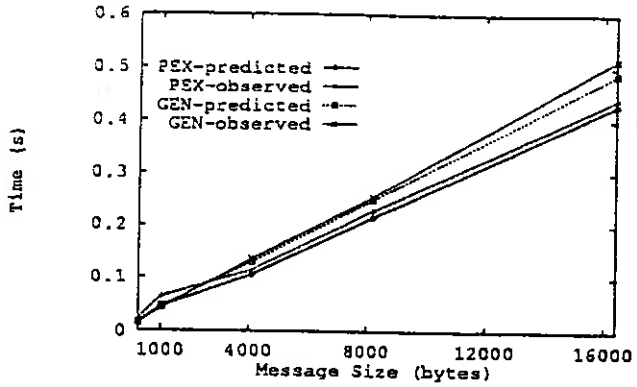

Figure 4: Observed and Predicted times (PEX, GEN)

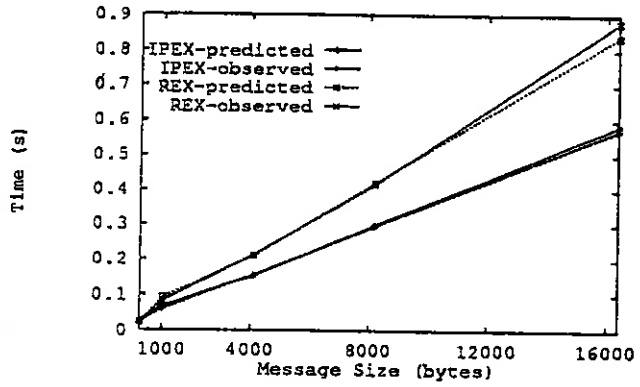

Figure 5: Observed and Predicted times (REX, IPEX)

algorithms result in a lot of link contention. The reduction in contention by IPEX is larger than the cost of sending messages indirectly, hence IPEX performs better.

We have validated the models developed to predict the performance of the algorithms by comparing the predicted times with the actual times observed on the Delta. For this purpose, we use typical values for the communication costs on the Delta [4, 1], namely for unforced messages $\alpha = 75\mu s$, $\beta_{ex} = 0.35\mu s$ and for forced messages $\alpha = 150\mu s$, $\beta_{ex} = 0.2\mu s$. We assume that $\beta_{sr} \approx \beta_{ex}$ and $2\beta_{sat} \approx \beta_{ex}$, ie. two messages can travel on a link in the same direction without conflict. Figures 4 and 5 show that the observed and predicted times agree very closely.

6 Conclusions

In this paper, we have discussed several algorithms for all-to-all communication on mesh connected computers with wormhole routing. Performance results on the Intel Touchstone Delta were presented. We observed that when the number of processors is small (< 64) and message size is small (< 1 Kbytes), the GEN algorithm performs the best. For larger message and mesh sizes, PEX performs better than GEN. For large meshes, direct algorithms result in a lot of link contention. IPEX performs the best in this case, as it reduces contention by sending messages indirectly.

The performance of the algorithms can be easily predicted from the mesh size, message length and some communication parameters of the system. These algorithms can be used on any mesh connected computer and the performance models enable us to choose the best algorithm under the circumstances.

Acknowledgments

This work was supported in part by ARPA under contract no. DABT63-91-C-0028. The content of the information does not necessarily reflect the position or policy of the Government and no official endorsement should be inferred. Alok Choudhary's research is also supported by an NSF Young Investigator Award CCR-9357840 and a grant from Intel SSD. This research was performed in part using the Intel Touchstone Delta System operated by California Institute of Technology on behalf of the Concurrent Supercomputing Consortium. Access to this facility was provided by the Center for Research on Parallel Computation.

References

[1] Barnett, M., Littlefield, R., Payne, D., and van de Geijn, R., "Global Combine on Mesh Architectures with Wormhole Routing", *Proc. of 7th Int. Parallel Proc. Symp.*, April 1993.

[2] Bokhari, S., "Complete Exchange on the iPSC/860", *ICASE Technical Report 91-4*, 1991.

[3] Bokhari, S., and Berryman, H., "Complete Exchange on a Circuit Switched Mesh", *Proc. of Scalable High Perf. Computing Conf.*, 1992, pp. 300–306.

[4] Littlefield, R., "Tuning Communication", *Proceedings of the Delta Advanced User Training Class Notes*, CCSF Technical Report CCSF-25-92, July 1992, pp. 99–119.

[5] Ni, L., and McKinley, P., "A Survey of Wormhole Routing Techniques in Direct Networks", *Computer*, February 1993, pp. 62–76.

[6] Ponnusamy, R., Thakur, R., Choudhary, A., and Fox G., "Scheduling Regular and Irregular Communication Patterns on the CM-5", *Proc. of Supercomputing 92*, November 1992, pp. 394–402.

[7] Scott, D., "Efficient All-to-All Communication Patterns in Hypercube and Mesh Topologies", *Proc. of 6th Distributed Memory Computing Conf.*, 1991, pp. 398–403.