Decomposition of Total Exchange for Multidimensional Interconnects*

Vassilios V. Dimakopoulos *Nikitas J. Dimopoulos*

Department of Electrical and Computer Engineering
University of Victoria, P.O. Box 3055,
Victoria, B.C., CANADA, V8W 3P6
E-mail: {dimako, nikitas}@ece.uvic.ca

Abstract

Total exchange is an important collective communication problem in multiprocessor interconnection networks. It involves the dissemination of distinct messages from every node to every other node. We present a novel theory for solving the problem in any multidimensional (cartesian product) network. We construct a general algorithm and provide optimality conditions. It is seen that many of the popular topologies, including hypercubes, k-ary n-cubes and general tori satisfy these conditions. The results we present here apply to the single-port model.

1 Introduction

Multidimensional (or cartesian product) networks have prevailed the interconnection network design for distributed memory multiprocessors both in theory and in practice. Commercial machines like the Ncube, the Cray T3D, the Intel iPSC, Delta and Paragon, have a node interconnection structure based on multidimensional networks such as hypercubes, tori and meshes. These networks are based on simple basic dimensions: linear arrays in meshes [10], rings in k-ary n-cubes [5] and general tori, complete graphs in generalized hypercubes [3]. Structures with quite powerful dimensions have also been proposed, e.g. products of trees or products of graphs based on groups [11, 8].

One important issue related to multiprocessor interconnection networks is that of information dissemination. Collective communications for distributed-memory multiprocessors have received considerable attention, for example they are included in the Message Passing Interface standard and they support various constructs in High Performance Fortran.

Broadcasting, scattering, gathering, multinode broadcasting and total exchange constitute a set of representative collective communication problems. In *total exchange*, also known as *multiscattering* or *all-to-all personalized communication*, each node in a network sends distinct messages to all other nodes. Various data permutations occurring e.g. in parallel FFT and basic linear algebra algorithms can be viewed as instances of the total exchange problem [2].

In this work we present general theory for solving the total exchange problem in multidimensional networks. A multitude of quantities or properties in such networks can be decomposed to quantities and properties of the individual dimensions. We show here that the total exchange problem can also be decomposed to the simpler problem of performing total exchange in single dimensions. This is a major simplification to an inherently complex problem. We provide a general algorithm applicable to any multidimensional network given that we have total exchange algorithms for each dimension. Optimality conditions are given and it is seen that they are met for many popular networks, e.g. hypercubes, tori, generalized hypercubes etc.

The results presented here apply to *packet-switched* networks with *single-port* capabilities. This model is based on the following assumptions/restrictions:

- communication links are bidirectional and fully duplex
- a message requires one time unit (or step) to be transferred between two adjacent nodes
- a node can send at most one message and receive at most one message at each time unit.

Algorithms to solve the problem for certain networks and under a variety of assumptions have appeared in many recent works, mostly concentrating in hypercubes and two-dimensional tori (e.g. [12, 9, 13]). Under the single-port model an optimal algorithm for hypercubes is given in [2, pp. 81–83].

The paper is organized as follows. We formally introduce multidimensional networks in Section 2. Section 3 gives a lower bound on the time required for solving the total exchange problem under our model. In the same section we derive a new formula for this bound in the networks of interest. The result has its own merit as it also provides almost closed-form formulas for the *average distance* in networks for which no such formula was known up to now. In Section 4 we develop the total exchange algorithm and in Section 5 we give the optimality conditions. The results are summarized in Section 6.

*This research was supported in part through grants from NSERC and the University of Victoria.

0190-3918/96 $5.00 © 1996 IEEE

2 Multidimensional Networks

Let $G = (V, E)$ be an undirected graph[1] [4] with node set V and edge set E. This is the usual model of representing a multiprocessor interconnection network: processors correspond to nodes and communication links correspond to edges in the graph. The number of nodes in G is $n = |V|$.

A *path* in G from node v to node u, is denoted as $v \rightarrow u$. The *distance*, $dist(v, u)$, between v and u is the length of a shortest path between v and u. The *eccentricity* of v, $e(v)$, is the distance to a node farthest from v, i.e. $e(v) = \max_{u \in V} dist(v, u)$. The *diameter* of G is the maximum eccentricity in G.

Given k graphs $G_i = (V_i, E_i)$, $i = 1, 2, \ldots, k$, their product is the graph $G = G_1 \times \cdots \times G_k = (V, E)$ whose nodes are labeled by a k-tuple $(v_1, \ldots, v_k)$ and

$$\begin{aligned} V &= \{(v_1, \ldots, v_k) \mid v_i \in V_i, i = 1, \ldots, k\} \\ E &= \{((v_1, \ldots, v_k), (u_1, \ldots, u_k)) \mid \\ &\quad \exists j : (v_j, u_j) \in E_j \text{ and } v_i = u_i \text{ for all } i \neq j\}. \end{aligned}$$

We call such products of graphs *multidimensional* graphs and G_i is the ith *dimension* of the product. The ith component of the address tuple of a node will be called the ith *address digit* or the ith *coordinate*.

Multidimensional graphs have $n = |V_1||V_2| \cdots |V_k|$ nodes. Hypercubes are products of two-node linear arrays (or rings), tori are products of rings. If all dimensions of the torus consist of the same ring, we obtain k-ary n-cubes [5]. Meshes are products of linear arrays [10]. Generalized hypercubes are products of complete graphs [3].

The *don't care* symbol '$*$' is used as a shorthand notation for a set of addresses. It represents all legal values of the element of an address tuple it replaces.

3 Lower Bound for Total Exchange

In the total exchange problem, a node v has to send $n-1$ distinct messages, one to each of the other nodes in an n-node network. Given a node v in the network, if there exist n_d nodes in distance d from v; $d = 1, 2, \ldots, e(v)$, then the messages sent by v must cross

$$s(v) = \sum_{d=1}^{e(v)} d n_d$$

links in total. For all messages to be exchanged, the total number of link traversals must be

$$S_G = \sum_{v \in V} s(v).$$

The quantity $s(v)$ is known as the *total distance* or the *status* [4] of node v.

[1] The terms 'graph' and 'network' are considered synonymous here. We also follow the usual graph definitions and notation

Every time a message is communicated between adjacent nodes one link traversal occurs. Under the single-port model nodes transmit only one message per step, so that the maximum number of link traversals in a single step is at most n. Consequently, we can at best subtract n units from S_G in each step; thus a lower bound on total exchange time is

$$T \geq \frac{S_G}{n} = AS(G). \tag{1}$$

In other words, total exchange requires time bounded below by the *average status*, $AS(G)$, of the vertices.

3.1 Status in multidimensional networks

In this section we present a formula for the status of vertices in multidimensional graphs, as required by the lower bound of (1). The results are based on the status of vertices in individual dimensions. For formal proofs the reader is referred to [7].

Theorem 1 *Let $G = G_1 \times G_2 \times \cdots \times G_k$. If $s_i(v_i)$ is the status of v_i in G_i, $i = 1, 2, \ldots, k$, then the status of $v = (v_1, v_2, \ldots, v_k)$ in G is*

$$s(v) = n \sum_{i=1}^{k} \frac{s_i(v_i)}{|V_i|}. \qquad \square$$

The quantity $s(v)/(n-1)$ is known as the *average distance* of node v, giving the average number of links that have to be traversed by a message departing from v. Hence, Theorem 1 can also be used to calculate the average distance of vertices in many graphs for which no closed-form formula was known up to now. As an example, in generalized hypercubes [3] each dimension is a complete graph with m_i vertices, $i = 1, 2, \ldots, k$. In a complete graph all nodes are adjacent to each other, so that $s_i(v_i) = m_i - 1$. Consequently, the average distance in generalized hypercubes is

$$\frac{n}{n-1} \sum_{i=1}^{k} \frac{m_i - 1}{m_i} = \frac{n}{n-1} \left(k - \sum_{i=1}^{k} \frac{1}{m_i} \right).$$

In [3] it was possible to derive a formula only for the case where all m_i are equal to each other.

In the context of the total exchange problem we are interested in the average status of the nodes in the network. Let $AS(G_i)$ be the average status of G_i, defined in (1) as $AS(G_i) = \sum_{v_i \in G_i} s_i(v_i)/|V_i|$. We have the following corollary.

Corollary 1 *Let $G = G_1 \times G_2 \times \cdots \times G_k$. If $AS(G_i)$ is the average status of G_i, $i = 1, 2, \ldots, k$, then the average status of G is given by*

$$AS(G) = n \sum_{i=1}^{k} \frac{AS(G_i)}{|V_i|}. \qquad \square$$

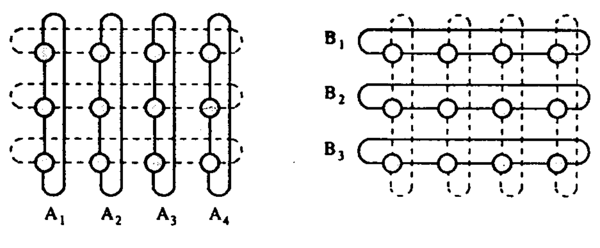

Figure 1. *A 4×3 torus as (a) four copies of a three-node ring or (b) three copies of a four-node ring*

4 Total Exchange Algorithm

Let $G = A \times B$. A k-dimensional network $G_1 \times \cdots \times G_k$ can still be expressed as the product of two graphs by taking $A = G_1 \times \cdots \times G_{k-1}$ and $B = G_k$, so we may consider two dimensions without loss of generality. Let $A = (V_A, E_A)$, $B = (V_B, E_B)$, $G = (V, E)$, $n_1 = |V_A|$, $n_2 = |V_B|$ and $n = n_1 n_2$. Finally, let

$$\begin{aligned} V_A &= \{v_i \mid i = 1, 2, \ldots, n_1\} \\ V_B &= \{u_i \mid i = 1, 2, \ldots, n_2\}. \end{aligned}$$

Graph G consists of n_2 (interconnected) copies of V_A. Let A_j be the jth copy of A with node set $(*, u_j)$, where $*$ takes all values in V_A. Similarly, G can be viewed as n_1 copies of B, and we let B_i be the ith copy of B with node set $(v_i, *)$. An example is shown in Fig. 1.

We will develop the basic idea behind our algorithm through the example in Fig. 1. Consider the top node of A_1. This node belongs to A_1 as well as B_1. All nodes in A_1 have, among other messages, messages destined for the rest of the nodes in A_1. These messages can be distributed by performing a total exchange within A_1. In addition, nodes in A_1 have messages for all nodes in A_2, A_3 and A_4 which must be delivered to their appropriate destinations. What we will do is the following: all messages of the top node of A_1 meant for the nodes in A_2 will be transferred to the top node of A_2. All messages of the middle node of A_1 destined for the nodes in A_2 will be transferred to the middle node of A_2. Similar will be the case for the bottom node of A_1. Once all these messages arrive in A_2, the only thing remaining is to perform a total exchange within A_2 and all these messages will be distributed to the correct destinations.

Next, nodes of A_1 have to transfer their messages meant for A_3 to nodes of A_3. The procedure is identical to the procedure followed for messages meant for A_2. The remaining messages in A_1 are destined for A_4 and one more repetition of the above procedure completes the task. Notice that the procedure outlined above for messages originating at nodes of A_1 must be repeated for messages originating at A_2, A_3 and A_4. We are now ready to formalize our arguments.

We are going to adopt the following notation: $m_{(v_i,u_j)}(v_k, u_l)$ will denote the message of node (v_i, u_j) destined for node (v_k, u_l). We will furthermore introduce the '$*$' symbol to denote a corresponding set of messages. For example, $m_{(v_i,u_j)}(*, u_l)$ denotes all messages of node (v_i, u_j) destined for the nodes of A_l, and $m_{(v_i,*)}(v_k, u_l)$ denotes all messages of B_i destined for node (v_k, u_l). Similarly, $m_{(v_i,u_j)}(*, *)$ denotes all messages of (v_i, u_j). Notice that this last set normally includes $m_{(v_i,u_j)}(v_i, u_j)$ since $(*, *)$ covers all nodes. Since no node sends messages to itself, it is always implied that *from any set of messages, we have removed message whose source and destination are the same.*

Consider the set of messages $m_{(*,*)}(*, *)$. This set represents our total exchange problem: every node has one message for every other node. Next consider the set $m_{(*,u_j)}(*, u_j)$. This is the set of messages of nodes in A_j destined for the other nodes in A_j: they can be distributed by a total exchange operation within A_j. Finally, consider the set $m_{(v_i,u_j)}(*, u_k)$ of node (v_i, u_j) meant for the nodes of A_k. This set will be transferred to node (v_i, u_k). Thus, after such transfers, node (v_1, u_k) will have received $m_{(v_1,u_j)}(*, u_k)$, node (v_2, u_k) will have received $m_{(v_2,u_j)}(*, u_k)$, and so on. Notice that every node in A_k will have received messages meant for every node in A_k: these messages clearly can be distributed to the appropriate destinations through a total exchange operation within A_k.

The first problem we have with this approach is that there may exist path collisions when node (v_i, u_j) transfers messages to (v_i, u_k) and node $(v_{i'}, u_j)$ transfers messages to $(v_{i'}, u_k)$, $i \neq i'$. We can avoid these collisions if we only allow use of links in the second dimension (B). That is, the allowable paths $(v_i, u_j) \rightarrow (v_i, u_k)$ involve only nodes $(v_i, *)$ of B_i. Then if $v_{i'} \neq v_i$, paths $(v_i, u_j) \rightarrow (v_i, u_k)$ and $(v_{i'}, u_j) \rightarrow (v_{i'}, u_k)$ have no node in common. Let us consider again the example in Fig. 1. At some point all nodes in A_1 want to transfer their messages, say, for nodes in A_4. The top node of A_1 transfers its messages to the top node in A_4, the middle node of A_1 transfers its own messages to the middle node of A_4 and so on, without any interference between them if the paths used belong to the second dimension. That is, all the transfers of the top node of A_1 use links in B_1, all transfers from the bottom node of A_1 use links in B_3, etc.

Algorithm A1 shown in Fig. 2 solves the total exchange problem in $G = A \times B$. First we perform *all* the transfers we described above and then we perform the total exchanges within each A_j. The transfers correspond to lines 1–4 in Algorithm A1. After they are completed, every node (v_i, u_j), for every i, j, will have received all messages meant for the jth copy of A orig-

Table 1. *Messages to be transferred from node* (v_i, u_j).

	For A_1	$\cdots$	For A_k	$\cdots$	For A_{n_2}
R_1	$m_{(v_i,v_j)}(v_1, u_1)$	$\cdots$	$m_{(v_i,v_j)}(v_1, u_k)$	$\cdots$	$m_{(v_i,v_j)}(v_1, u_{n_2})$
R_2	$m_{(v_i,v_j)}(v_2, u_1)$	$\cdots$	$m_{(v_i,v_j)}(v_2, u_k)$	$\cdots$	$m_{(v_i,v_j)}(v_2, u_{n_2})$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$
R_{n_1}	$m_{(v_i,v_j)}(v_{n_1}, u_1)$	$\cdots$	$m_{(v_i,v_j)}(v_{n_1}, u_k)$	$\cdots$	$m_{(v_i,v_j)}(v_{n_1}, u_{n_2})$

```
1 Do in parallel for all v_i ∈ V_A (i = 1,2,...,n_1)
2   For every j = 1,2,...,n_2
3     For every k = 1,2,...,n_2, k ≠ i
4       Transfer messages m_(v_i,u_j)(*,u_k) to node
        (v_i,u_k) using links in B_i;
5 For every k = 1,2,...,n_2
6   Do in parallel for all A_j, j = 1,2,...,n_2
7     In A_j perform total exchange with node (v_i,u_j)
      sending messages m_(v_i,u_k)(*,u_j);
```

Figure 2. *Algorithm A1*

```
1 For r = 1,2,...,n_1
2   Do in parallel for all B_i, i = 1,2,...,n_1
3     In B_i perform total exchange with node (v_i,u_j)
      sending messages m_(v_i,u_j)(v_r,*), j = 1,2,...,n_2;
4 For every k = 1,2,...,n_2
5   Do in parallel for all A_j, j = 1,2,...,n_2
6     In A_j perform total exchange with node (v_i,u_j)
      sending messages m_(v_i,u_k)(*,u_j), i = 1,2,...,n_1;
```

Figure 3. *Algorithm A2*

inating at nodes (v_i, u_k), $k = 1, 2, \ldots, n_2$, i.e. all messages $m_{(v_i,u_k)}(*, u_j)$. Lines 5–7 of the algorithm distribute these messages to the correct vertices of A_j in n_2 rounds. In the kth round a total exchange is performed and the exchanged messages have originated from A_k.

Algorithm A1 solves the total exchange problem but lines 1–4 do not show how the transfer of messages is exactly implemented. Within B_i we need to transfer messages $m_{(v_i,u_j)}(*, u_k)$ from *every* vertex u_j to *every* other vertex u_k. In Table 1 we list the messages to be transferred by some vertex (v_i, u_j) of A_j. Notice that we do not have to transfer messages meant for A_j anywhere, so the jth column of the table is actually unused (it will only be used for a total exchange within A_j). Column k contains all messages of (v_i, u_j) meant for A_k, to be transferred first to node (v_i, u_k).

Instead of transferring the messages column by column (i.e. transfer all messages in column 1 to A_1, then all messages in column 2 to A_2, etc.) we transfer them horizontally (row by row). The batch R_r of messages in row r contains all messages $m_{(v_i,u_j)}(v_r, *)$. We will transfer all of them, except of course for $m_{(v_i,u_j)}(v_r, u_j)$ in column j which is meant for a node of A_j. Let us consider again the network in Fig. 1 and assume that the bottom nodes of A_1, A_2, A_3 and A_4 want to transfer their first batch, R_1. The batch of the bottom node of A_1 contains one message for each of the bottom nodes of A_2, A_3 and A_4. Similarly, batch R_1 for the bottom node of A_2 contains one message for the other three nodes in question. It should be immediately clear that these messages constitute an instance of the total exchange problem in B_1.

In general, when every node (v_i, u_1), (v_i, u_2), $\ldots$, (v_i, u_{n_2}) in B_i transfers its own batch R_r of Table 1, a total exchange within B_i can distribute the messages appropriately. Consequently, all rows of Table 1 of every node will be transferred where they should by performing n_1 total exchanges in B_i: at the rth exchange all nodes $(v_i, *)$ transfer their rth batch of messages (rth row of the corresponding tables).

Based on the above discussion, and recalling that transfers within B_i do not interfere with transfers within $B_{i'}$, $i' \neq i$, we may express our total exchange algorithm in its final form, Algorithm A2, appearing in Fig. 3. Algorithm A2 is a general solution to the total exchange problem for any multidimensional network. If the network has $k > 2$ dimensions, $G = G_1 \times \cdots \times G_k$, Algorithm A2 can be used recursively, by taking $A = G_1 \times \cdots \times G_{k-1}$ and $B = G_k$. The total exchanges in A_j (lines 4–6) can be performed by invoking the algorithm with $A = G_1 \times \cdots \times G_{k-2}$ and $B = G_{k-1}$ and so forth.

The algorithm is in a highly desirable form: it only utilizes total exchange algorithms for each of the dimensions. The problem of total exchange in a complex network is now reduced to the simpler problem of devising total exchange algorithms for single dimensions. For example, algorithms for tori can be systematically constructed beased on algorithms for rings.

5 Optimality Conditions

It is not very hard to calculate the time required for Algorithm A2. Lines 1–3 perform n_1 total exchanges within B_i (for all $i = 1, 2, \ldots, n_1$ in parallel), each requiring T_B steps. Similarly, lines 4–6 perform n_2

total exchanges within A_j (for all $j = 1, 2, \ldots, n_2$ in parallel), each requiring T_A steps.

Theorem 2 *If single-port total exchange algorithms for graphs A and B take T_A and T_B steps correspondingly then Algorithm A2 for $G = A \times B$ requires*

$$T = n_1 T_B + n_2 T_A$$

time units. □

Using a simple induction the following is proven:

Corollary 2 *If $G = G_1 \times G_2 \times \cdots \times G_k$ and a single-port total exchange algorithm for G_i takes T_i time units, $i = 1, 2, \ldots, k$, total exchange in G under the single-port model can be performed in*

$$T = n \sum_{i=1}^{k} \frac{T_i}{|V_i|}$$

steps, where $n = |V_1||V_2| \cdots |V_k|$. □

Combining Corollary 1 with Corollary 2, we can prove the following:

Theorem 3 *If single-port total exchange for every dimension $i = 1, 2, \ldots, k$ of $G = G_1 \times G_2 \times \cdots \times G_k$ can be performed in time equal to the lower bound of (1) then the same is true for G.* □

The last theorem provides the main optimality condition for Algorithm A2. If we have total exchange algorithms for every dimension and these algorithms achieve the bound of (1) then Algorithm A2 also achieves this bound. For example, in hypercubes every dimension is a two-node graph. Trivially, in a two-node graph the time for total exchange is just one step, equal to the average status. Thus the optimality condition is met and the presented algorithm is an optimal algorithm for single-port hypercubes.

More generally, we have shown elsewhere [6] that there exist algorithms that need time equal to (1) for *any* Cayley [1] network. Consequently, the optimality condition is met for arbitrary products of Cayley networks. Rings and complete graphs are examples of Cayley networks and thus Algorithm A2 solves optimally the total exchange problem in k-ary n-cubes, general tori and generalized hypercubes.

6 Summary

In this paper we studied the total exchange problem in the context of multidimensional networks, under the single-port model. We showed that the problem can be decomposed into the simpler problems of devising total exchange algorithms in individual dimensions. Given that we have such algorithms that achieve the lower bound of (1) for each of the dimensions, we can synthesize optimal algorithms for the multidimensional network.

Many popular networks, including hypercubes, tori, generalized hypercubes and, in general, products of symmetric graphs in the Cayley class, consist of dimensions for which algorithms achieving the bound in (1) exist. Algorithm A2 is thus an optimal solution to the total exchange problem for the above networks.

A detailed exposition of this material, which also includes some extensions to the multiport model is available in [7] and can be obtained through the World Wide Web at `http://www-lapis.uvic.ca`.

References

[1] S. B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Trans. Comput.*, Vol. 38, No. 4, pp. 555–566, Apr. 1989.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewoods Cliffs, N.J.: Prentice - Hall, 1989.

[3] L. N. Bhuyan and D. P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Comput.*, Vol. C-33, No. 4, pp. 323–333, Apr. 1984.

[4] F. Buckley and F. Harary, *Distance in Graphs*. Reading, Mass.: Addison - Wesley, 1990.

[5] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, Vol. C-36, No. 5, pp. 547–553, May 1987.

[6] V. V. Dimakopoulos and N. J. Dimopoulos, "Optimal total exchange in Cayley graphs," Technical Report ECE-96-1, University of Victoria, Jan. 1996.

[7] V. V. Dimakopoulos and N. J. Dimopoulos, "A theory for total exchange in multidimensional interconnection networks," Technical Report ECE-96-2, University of Victoria, Jan. 1996.

[8] K. Efe and A. Fernández, "Products of networks with logarithmic diameter and fixed degree," *IEEE Trans. Parall. Distrib. Syst.*, Vol. 6, No. 9, pp. 963–975, 1995.

[9] S. L. Johnsson and C. - T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.*, Vol. 38, No. 9, pp. 1249–1268, 1989.

[10] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. San Diego, CA: Morgan Kaufmann, 1992.

[11] A. L. Rosenberg, "Product-shuffle networks: towards reconciling shuffles and butterflies," *Discrete Appl. Math.*, Vol. 37/38, pp. 465–488, July 1992.

[12] Y. Saad and M. H. Schultz, "Data communications in hypercubes," *J. Parallel Distrib. Comput.*, Vol. 6, pp. 115–135, 1989.

[13] E. A. Varvarigos and D. P. Bertsekas, "Communication algorithms for isotropic tasks in hypercubes and wraparound meshes," *Parallel Comput.*, Vol. 18, pp. 1233–1257, 1992.