A Theory for Total Exchange in Multidimensional Interconnection Networks*

Vassilios V. Dimakopoulos   Nikitas J. Dimopoulos

Dept. of Electrical and Computer Engineering, University of Victoria

P.O. Box 3055, Victoria, B.C., CANADA, V8W 3P6.

Tel: (604) 721-8773, 721-8902, *Fax:* (604) 721-6052,

E-mail: {dimako, nikitas}@ece.uvic.ca

March 18, 1998

Abstract

Total exchange (or multiscattering) is one of the important collective communication problems 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. These networks have been adopted as cost-effective interconnection structures for distributed-memory multiprocessors. We construct a general algorithm for *single-port* networks and provide conditions under which it behaves optimally. It is seen that many of the popular topologies, including hypercubes, k-ary n-cubes and general tori satisfy these conditions. The algorithm is also extended to homogeneous networks with 2^k dimensions and with *multiport* capabilities. Optimality conditions are also given for this model. In the course of our analysis we also derive a formula for the *average distance* of nodes in multidimensional networks; it can be used to obtain almost closed-form results for many interesting networks.

Keywords: Collective communications, interconnection networks, multidimensional networks, packet-switched networks, total exchange

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

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 [16], rings in k-ary n-cubes [6] and general tori, complete graphs in generalized hypercubes [4]. Structures with quite powerful dimensions have also been proposed, e.g. products of trees or products of graphs based on groups etc. [23, 9, 12, 21, 26].

One important issue related to multiprocessor interconnection networks is that of information dissemination. Collective communications for distributed-memory multiprocessors have recently received considerable attention, as for example is evident from their inclusion in the Message Passing Interface standard [19] and from their support of various constructs in High Performance Fortran [13, 17]. This is easily justified by their frequent appearance in parallel numerical algorithms [11, 14, 3].

Broadcasting, scattering, gathering, multinode broadcasting and total exchange constitute a set of representative collective communication problems that have to be efficiently solved in order to maximize the performance of message-passing parallel programs. A general survey regarding such communications was given in [10]. In *total exchange*, which is also known as *multiscattering* or *all-to-all personalized communication*, each node in a network has a distinct message to send to every other node. Various data permutations occurring e.g. in parallel FFT and basic linear algebra algorithms can be viewed as instances of the total exchange problem [3].

The subject of this work is the development of a 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. For example, the degree of a node is the sum of the degrees in each of the 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 for inherently complex networks. We provide general algorithms 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 and generalized hypercubes to name a few.

The results presented here apply to *packet-switched* networks that follow the so-called *constant model* [10]. The assumptions pertaining the model we will follow are:

- communication links are bidirectional and fully duplex
- a message requires one time unit (or *step*) to be transferred between two nodes
- only adjacent nodes can exchange messages.

Another parameter of the model is that of port capabilities. Depending on whether a node can communicate with one or all of its neighbors at the same time unit, two basic possibilities arise:

Single-port: a node can send at most one message and receive at most one message at each step.

Multiport: a node can send and receive messages from all its neighbors simultaneously.

As discussed in [10], the above assumptions constitute the standard model when examining theoretical aspects of communications in packet-switched networks. Furthermore, results and conclusions under this model can form the basis of arguments for other models, such as the *linear* one which also quantifies the effect of message lengths. Many recent works focus exclusively on wormhole-routed networks (an excellent survey on collective communications for such machines was given in [18]). However, we believe that studies should not be limited to one particular type of architecture: "it is important to consider several types of communication models, given the broad variety of present and future communication hardware" [2]. In addition, since a circuit-switched or wormhole routed network can emulate a packet-switched network by performing only nearest-neighbor communications, the results also constitute a reference point for methods developed for the former type of networks.

Algorithms to solve the total exchange problem for specific networks and under a variety of assumptions have appeared in many recent works, mostly concentrating in hypercubes and two-dimensional tori (e.g. [24, 15, 2, 25]). Under the single-port model we know of two optimal algorithms, in [3, pp. 81–83] for hypercubes, and in [20] for star graphs. In contrast, our results are applicable not only to one particular structure but rather provide a general procedure for solving the problem in any multidimensional network.

This paper is organized as follows. We introduce formally multidimensional networks in the next section and we give some of their properties related to our study. Section 3 gives lower bounds on the time required for solving the total exchange problem under both port assumptions. In the same section we derive a new formula for the single-port bound as applied to 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 concentrate on single-port networks. We develop a total exchange algorithm and we give conditions under which it behaves optimally. We also review known results about simple dimensions and conclude that our method can be optimally applied to hypercubes, k-ary n-cubes and other popular interconnects. In Section 5 we modify the algorithm and adapt it to the multiport model. The extension works for networks which have 2^k ($k \geq 1$) identical dimensions (*homogeneous* networks). Again, we provide optimality conditions and observe that they are satisfied for a number of interesting topologies. The results are summarized in Section 6.

2 Multidimensional Networks

Let $G = (V, E)$ be an undirected graph[1] [5] with node (or vertex) set V and edge (or link) 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 $n = |V|$. An edge in E between nodes v and u is written as the unordered pair (v, u) and v and u are said to be *adjacent* to each other, or just *neighbors*.

[1] The terms 'graph' and 'network' are considered synonymous here.

A *path* in G from node v to node u, denoted as $v \to u$, is a sequence of nodes $v = v_0, v_1, \dots, v_\ell = u$, such that all vertices are distinct and for all $0 \le i \le \ell$, $(v_i, v_{i+1}) \in E$. We say that the *length* of a path is ℓ if it contains ℓ vertices apart from v. The *distance*, $dist(v, u)$, between vertices v and u is the length of a shortest path between v and u. Finally, 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 maximum eccentricity in G is known as the *diameter* of G.

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

$$V = \Big\{(v_1, \dots, v_k) \mid v_i \in V_i, i = 1, \dots, k\Big\}$$
$$E = \Big\{((v_1, \dots, v_k), (u_1, \dots, u_k)) \mid \exists j \text{ s.t. } (v_j, u_j) \in E_j \text{ and } v_i = u_i \text{ for all } i \ne j\Big\}.$$

We will call such products of graphs *multidimensional* graphs and G_i will be called 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*. The definition of E above in simple words states that two nodes are adjacent if they differ in exactly one address digit. Their differing coordinates should be adjacent in the corresponding dimension. An example is given in Fig. 1. Dimension 1 is a graph consisting of a two-node path with $V_1 = \{a, b\}$ while dimension 2 consists of a three-node ring with $V_2 = \{1, 2, 3\}$. Their product has node set

$$V = \Big\{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\Big\}.$$

According to the definition, node $(a, 1)$ has the following neighbors: since node a is adjacent to node b in the first dimension, node $(a, 1)$ will be adjacent to node $(b, 1)$; since node 1 is adjacent to both nodes 2 and 3 in the second dimension, node $(a, 1)$ will also be adjacent to nodes $(a, 2)$ and $(a, 3)$.

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 [6]. Meshes are products of linear arrays [16]. Generalized hypercubes are products of complete graphs [4]. If all dimensions G_i, $i = 1, 2, \dots, k$, are identical then the network is characterized as *homogeneous*.

Multidimensional graphs have $n = |V_1||V_2|\cdots|V_k|$ nodes, where $|V_i|$ is the number of nodes in G_i, $i = 1, 2, \ldots, k$. It is also known that if $dist_i(v_i, u_i)$ is the distance between v_i and u_i in G_i then the distance between $v = (v_1, \ldots, v_k)$ and $u = (u_1, \ldots, u_k)$ in G is

$$dist(v, u) = \sum_{i=1}^{k} dist_i(v_i, u_i). \tag{1}$$

It will be convenient to use the *don't care* symbol '$*$' as a shorthand notation for a set of addresses. An appearance of this symbol at an element of an address tuple represents all legal values of this element. In the previous example, $(a, *) = \{(a, 1), (a, 2), (a, 3)\}$, $(*, 1) = \{(a, 1), (b, 1)\}$ while $(*, *)$ denotes the whole node set of the graph.

3 Lower Bounds for Total Exchange

In the total exchange problem, a node v has to send $n - 1$ distinct messages, one for each of the other nodes in an n-node network. Let us first assume that the single-port model is in effect. If there exist n_d nodes in distance d from v, where $d = 1, 2, \ldots, e(v)$, then the messages sent by v must cross

$$s(v) = \sum_{d=1}^{e(v)} dn_d \tag{2}$$

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* [5] of node v.

Every time a message is communicated between adjacent nodes one link traversal occurs. Under the single-port model nodes are allowed to 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, so that a lower bound on total exchange time is

$$TE_{\text{sp}} \geq \frac{S_G}{n} = AS(G). \tag{3}$$

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

For multiport networks tighter bounds are obtained through *cuts* of the network. Partition the vertex set V in two disjoint sets V_1 and V_2 such that $V_1 \cup V_2 = V$. Let $C_{V_1V_2}$ be the number of edges in E joining the two parts, i.e. edges $e = (v, u)$ such that $v \in V_1$ and $u \in V_2$. Messages from nodes in V_1 destined for nodes in V_2 must cross these $C_{V_1V_2}$ edges. The total number of such messages is $|V_1||V_2|$. Since only $C_{V_1V_2}$ messages are able to pass from V_1 to V_2 at a time, we obtain the following lower bound for total exchange time:

$$TE_{\text{mp}} \geq \frac{|V_1||V_2|}{C_{V_1V_2}}. \tag{4}$$

We are of course interested in maximizing the fraction in the right-hand side by selecting V_1 and V_2 appropriately so that the tightest possible bound results. In many cases a *bisection* of the graph is the most appropriate choice, although any sensible partition will yield quite tight bounds.

3.1 Status in multidimensional networks

In the course of our analysis on the single-port model we will need to compare the time needed for total exchange with the lower bound of (3). We present here a formula for the status and the average status of vertices in multidimensional graphs, as required by (3). The results are based on the status of vertices in individual dimensions.

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*

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

Proof. The status of node v can be calculated through (2) or by using the equivalent formula:

$$s(v) = \sum_{u \in G} dist(v, u), \tag{5}$$

where $dist(v, u)$ is the distance between v and u. Hence, the status of v_i in G_i can be written as

$$s_i(v_i) = \sum_{u_i \in G_i} dist_i(v_i, u_i). \tag{6}$$

We know that in a multidimensional network the distance between two vertices is equal to the sum of distances between the corresponding coordinates (Eq. (1)). Consequently, from (5) we obtain

$$
\begin{aligned}
s(v) &= \sum_{(u_1,\ldots,u_k)\in G} dist((v_1,\ldots,v_k),(u_1,\ldots,u_k)) \\
&= \sum_{u_1\in G_1}\cdots\sum_{u_k\in G_k}\sum_{i=1}^{k} dist_i(v_i,u_i) \\
&= \sum_{i=1}^{k}\Big\{\sum_{u_1\in G_1}\cdots\sum_{u_{i-1}\in G_{i-1}}\ \sum_{u_{i+1}\in G_{i+1}}\cdots\sum_{u_k\in G_k}\sum_{u_i\in G_i} dist_i(v_i,u_i)\Big\} \\
&\overset{(6)}{=} \sum_{i=1}^{k}\Big\{\sum_{u_1\in G_1}\cdots\sum_{u_{i-1}\in G_{i-1}}\ \sum_{u_{i+1}\in G_{i+1}}\cdots\sum_{u_k\in G_k} s_i(v_i)\Big\} \\
&= \sum_{i=1}^{k}\Big\{\frac{n}{|V_i|}s_i(v_i)\Big\},
\end{aligned}
$$

as claimed. □

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. It is an important performance measure of the network since under uniform addressing distributions it is directly linked with the average delay a message experiences before reaching its destination [22]. The following corollary follows directly from Theorem 1:

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

$$AD(v) = \frac{n}{n-1}\sum_{i=1}^{k}(1 - \frac{1}{|V_i|})AD_i(v_i).$$

Corollary 1 can 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 [4] 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 $AD_i(v_i) = 1$. Consequently, the average distance in generalized hypercubes is

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

In [4] 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 (3) as $AS(G_i) = \sum_{v_i \in G_i} s_i(v_i)/|V_i|$. We have the following corollary.

Corollary 2 *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*

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

Proof. From Theorem 1 we obtain

$$\begin{aligned}
\sum_{v \in G} s(v) &= n \sum_{(v_1, \ldots, v_k) \in G} \sum_{i=1}^{k} \frac{s_i(v_i)}{|V_i|} \\
&= n \sum_{v_1 \in G_1} \cdots \sum_{v_k \in G_k} \sum_{i=1}^{k} \frac{s_i(v_i)}{|V_i|} \\
&= n \sum_{i=1}^{k} \sum_{v_1 \in G_1} \cdots \sum_{v_{i-1} \in G_{i-1}} \sum_{v_{i+1} \in G_{i+1}} \cdots \sum_{v_k \in G_k} \sum_{v_i \in G_i} \frac{s_i(v_i)}{|V_i|} \\
&= n \sum_{i=1}^{k} \sum_{v_1 \in G_1} \cdots \sum_{v_{i-1} \in G_{i-1}} \sum_{v_{i+1} \in G_{i+1}} \cdots \sum_{v_k \in G_k} AS(G_i) \\
&= n \sum_{i=1}^{k} \frac{n}{|V_i|} AS(G_i),
\end{aligned}$$

which, divided by n, gives the required result. □

4 Single-port 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. 2.

We will develop the basic idea behind our algorithm through the example in Fig. 2. 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. Somehow, these messages have to travel 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 have arrived 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 will be identical to the procedure we followed for messages meant for A_2. Finally, the remaining messages in A_1 are destined for A_4 and one more repetition of the above procedure will complete the task. Notice that what we did for messages originating at nodes of A_1 has to be done also for messages originating at the other copies of A, i.e. 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 every 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.

To recapitulate, we can solve the total exchange problem in $G = A \times B$ using the algortihm shown below.

1 For every $i = 1, 2, \dots, n_1$
2   For every $j = 1, 2, \dots, n_2$
3   For every $k = 1, 2, \dots, n_2$, $k \neq i$
4   Transfer messages $m_{(v_i,u_j)}(*,u_k)$ to node (v_i,u_k);
5 For every $k = 1, 2, \dots, n_2$
6   Do in parallel for all A_j, $j = 1, 2, \dots, n_2$
7   In A_j perform total exchange of messages $m_{(*,u_k)}(*,u_j)$
        (messages $m_{(v_i,u_k)}(*,u_j)$ reside in node (v_i,u_j));

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. 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 originating at nodes (v_i,u_k), $k = 1, 2, \dots, n_2$, 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.

The algorithm solves the total exchange problem but lines 1–4 do not show how the transfer of messages is exactly implemented. First of all, there may exist path collisions between transfers from (v_i,u_j) to (v_i,u_k) and transfers from $(v_{i'},u_j)$ to $(v_{i'},u_k)$, $i \neq i'$, if we try to do them simultaneously. A straightforward way of avoiding collisions and at the same time parallelizing the transfers (line 1) is the following: when transferring messages from (v_i, u_j) to (v_i, u_k) we only allow use of links in the second dimension. In other words, the allowable paths $(v_i, u_j) \to (v_i, u_k)$ involve only nodes $(v_i, *)$ of B_i. Then if $v_{i'} \neq v_i$, paths $(v_i, u_j) \to (v_i, u_k)$ and $(v_{i'}, u_j) \to (v_{i'}, u_k)$ have no node in common. Consequently, lines 1–4 can be rewritten in the improved form:

```
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;
```

We may still improve matters by further parallelizing lines 1–3. 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. 2 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: every node has one message for every other node 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 A1, appearing in Fig. 3. Algorithm A1 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 A1 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, we are in a position to systematically construct algorithms for tori, based on algorithms for rings.

We now proceed to determine the time requirements of the algorithm and the conditions under which it behaves optimally.

4.1 Optimality conditions

It is not very hard to calculate the time required for Algorithm A1. This is because it is written in a form suitable for the single-port model: every node participates in one total exchange operation at a time. When each total exchange is performed under the single-port model, in effect no node sends/receives more than one message at a time.

Theorem 2 *If single-port total exchange algorithms for graphs A and B take T_A and T_B steps correspondingly then Algorithm A1 for $G = A \times B$ requires*

$$T = n_1 T_B + n_2 T_A$$

time units.

Proof. The result is straightforward: 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. □

Corollary 3 *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 $n = |V_1||V_2| \cdots |V_k|$.

Proof. The proof is by induction. If we only have one dimension then the corollary is trivially true. Assume as an induction hypothesis that it holds for up to $k-1$ dimensions. Then we must have

$$T' = n' \sum_{i=1}^{k-1} \frac{T_i}{|V_i|},$$

where T' is the time needed for total exchange in $G' = G_1 \times \cdots \times G_{k-1}$ and $n' = |V_1| \cdots |V_{k-1}| = n/|V_k|$. If we let $A = G'$ and $B = G_k$, $n_1 = n'$ and $n_2 = |V_k|$, Theorem 2 gives

$$\begin{aligned} T &= n' T_k + |V_k| T' \\ &= \frac{n}{|V_k|} T_k + |V_k| \frac{n}{|V_k|} \sum_{i=1}^{k-1} \frac{T_i}{|V_i|} \\ &= n \sum_{i=1}^{k} \frac{T_i}{|V_i|}, \end{aligned}$$

as required. □

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 (3) then the same is true for G.*

Proof. If in G_i total exchange can be performed in time equal to the lower bound of (3) then $T_i = AS(G_i)$. From Corollary 3, we must have

$$T = n \sum_{i=1}^{k} \frac{AS(G_i)}{|V_i|},$$

which, combined with Corollary 2, shows that $T = AS(G)$ and the algorithm is thus optimal. □

The last theorem provides the main optimality condition for Algorithm A1. If we have total exchange algorithms for every dimension and these algorithms achieve the bound of (3) then Algorithm A1 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 [8] that there exist algorithms that need time equal to (3) 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 A1 solves optimally the total exchange problem in k-ary n-cubes, general tori and generalized hypercubes.

5 Multiport Algorithm

In this section we will modify Algorithm A1 to work better under the multiport model. In its present form, Algorithm A1 is not particularly efficient under this model. This is because lines 4–6 are executed after lines 1–3 have finished. During execution of lines 1–3 only edges of the second dimension (B) are used while lines 4–6 use only edges of the first dimension (A). In the multiport model we try to keep as many edges busy as possible and the behavior of Algorithm A1 does not contribute to that effect. We seek, consequently, to transfer messages in both dimensions simultaneously. In other words we will reconstruct the algorithm such that lines 1–3 overlap in time as much as possible with lines 4–6.

The theory we present here applies to homogeneous networks. We recall that a multidimensional network is homogeneous when all its dimensions are identical. Thus, $G = H \times H \times \cdots \times H = H^k$ for some graph H. We will only consider the two-dimensional case, i.e. $G = H^2$, but it will also be seen that the algorithm we derive is applicable when the dimensionality of the graph is in general a power of 2, i.e. $G = H^{2^k}$.

Let $G = A \times B = (V, E)$ where $A = B = H$. Also, let $n = |V_H|$ that is, G has n^2 nodes. The network in Fig. 4 will be used as an example for our arguments. For node $(1, 1)$ we give the messages it will distribute in Table 2. The messages in the first column are meant for the other nodes in A_1. A total exchange within A_1 may thus begin immediately to distribute such messages. Since this total exchange uses only links in the first dimension, node (1,1) is also available to participate in some total exchange in the second dimension (i.e. in B_1). In a general network, node (v_i, u_j) in A_j can participate in a total exchange within B_i as soon as the first total exchange in A_j starts. Within A_j the transferred messages are $m_{(v_i,u_j)}(*, u_j)$, as given in column j of Table 1.

Let us see what messages will be involved in the first total exchange within B_i. Our objective is the following: we want every node (v_i, u_j) in B_i to receive $n-1$ appropriate messages so that after this total exchange in B_i is done, another total exchange can be initiated within A_j. Consequently, we seek to arrange the transfers so that (v_i, u_j) receives one message for each node in A_j, i.e. receive messages with destinations $(*, u_j)$. Notice that any node (v_i, u_j) will receive $n-1$ messages through a total exchange in B_i: since A_j has n nodes (including (v_i, u_j)), all the $n-1$ receptions of (v_i, u_j) should be meant for nodes other than (v_i, u_j) itself.

In the network in Fig. 4, we let for example node (1,1) send $m_{(1,1)}(2,2)$. This message will at some point be received by node (1,2) and it will provide one message for the forthcoming total exchange in A_2. If (1,2) sends $m_{(1,2)}(2,3)$ then node (1,3) will also be provided with one message for total exchange in A_3. Similarly, (1,3) sends $m_{(1,3)}(2,1)$, needed by node (1,1).

We define the following operators:

$$x \oplus_n y \stackrel{\text{def}}{=} [(x + y - 1) \bmod n] + 1$$
$$x \ominus_n y \stackrel{\text{def}}{=} [(x - y - 1) \bmod n] + 1.$$

These operators work like addition/subtraction modulo n but produce numbers ranging from 1 to n instead of 0 to $n-1$, and are better suited for our purposes here. Based on this operator and the preceding discussion, we see that one effective scheduling is to let node (v_i, u_j) (for all i and all j) send, among other messages, message $m_{(v_i,u_j)}(v_{i\oplus_n 1}, u_{j\oplus_n 1})$. Hence, this node will also receive $m_{(v_i,u_{j\ominus_n 1})}(v_{i\oplus_n 1}, u_j)$ from node $(v_i, u_{j\ominus_n 1})$ which it will use for the next total exchange in A_j.

Let us see what other messages will be sent during this first total exchange in B_i. In our example it is seen that since node (1,1) decided to send $m_{(1,1)}(2,2)$, it cannot send another message to node (1,2). Thus it has to send a message to node (1,3). Since this node will receive $m_{(1,2)}(2,3)$, which covers one destination in A_3, the only choice for (1,1) is to send $m_{(1,1)}(3,3)$. This message completes the set of messages needed by (1,3) for the next total exchange in A_3 since all other vertices in A_3 are now covered. Similarly, (1,2) and (1,3) must send $m_{(1,2)}(3,1)$ and $m_{(1,3)}(3,2)$ and all three nodes will have a complete set of messages, suitable for total exchanges within A_1, A_2 and A_3.

In general, the second message that node (v_i, u_j) will send is $m_{(v_i,u_j)}(v_{i\oplus_n 2}, u_{j\oplus_n 2})$. This will provide node $(v_i, u_{j\oplus_n 2})$ with a second message for the total exchange in $A_{j\oplus_n 2}$. The pattern should now be clear: during the first total exchange in B_i, every node (v_i, u_j), $j = 1, 2, \ldots, n$, sends the following messages:

$$m_{(v_i,u_j)}(v_{i\oplus_n 1}, u_{j\oplus_n 1}),\ m_{(v_i,u_j)}(v_{i\oplus_n 2}, u_{j\oplus_n 2}),\ \ldots,$$
$$m_{(v_i,u_j)}(v_{i\oplus_n (n-1)}, u_{j\oplus_n (n-1)}),$$

or, in a compact form:

$$\left\{m_{(v_i,u_j)}(v_{i\oplus_n \ell}, u_{j\oplus_n \ell}) \mid \ell = 1, 2, \ldots, n-1\right\}.$$

This node will provide node $(v_i, u_{j\oplus_n \ell})$ with the ℓth message it needs (i.e. a message destined for node $(v_{i\oplus_n \ell}, u_{j\oplus_n \ell})$ in $A_{j\oplus_n \ell}$). Notice that the above set contains one message to be received by each node $(v_i, u_{j'})$, $j' \neq j$, i.e. it is a perfect set for participation in the first total exchange in B_i. Also, it should be clear that (v_i, u_j) will receive the following messages:

$$\left\{m_{(v_i,u_{j\ominus_n \ell})}(v_{i\oplus_n \ell}, u_j) \mid \ell = 1, 2, \ldots, n-1\right\}.$$

Again notice that this set contains one message for each node $(v_{i'}, u_j)$, $i' \neq i$, in A_j. Thus we achieved our goal: every node in B_i receives a full set of messages to be used for the subsequent total exchange in A_j.

Since $A = B$, the first total exchange in A_j finishes exactly when the first total exchange in B_i finishes. Thus the second total exchange in A_j can start immediately, using the newly acquired (through the exchange in B_i) messages. Then the story repeats itself: a second total exchange in B_i can be performed simultaneously with the second total exchange in A_j. Our goal for this total exchange in B_i remains the same: to distribute messages that can be used for a third total exchange in A_j.

The idea behind selecting a group of messages for this second total exchange in B_i is similar to the one in the first total exchange we saw above. Now, we let (v_i, u_j) send messages

$$m_{(v_i,u_j)}(v_{i\oplus_n 2}, u_{j\oplus_n 1}),\ m_{(v_i,u_j)}(v_{i\oplus_n 3}, u_{j\oplus_n 2}),\ \ldots,$$
$$m_{(v_i,u_j)}(v_{i\oplus_n (n-1)}, u_{j\oplus_n (n-2)}),\ m_{(v_i,u_j)}(v_{i\oplus_n 1}, u_{j\oplus_n (n-1)}).$$

The situation is repeated continuously. While the rth total exchange within A_j is in progress, the rth total exchange in B_i is also performed in order to provide nodes with messages for the next — $(r+1)$th — total exchange in A_j. During the rth exchange in B_i a node (v_i, u_j) sends the following messages:

$$\begin{array}{l} m_{(v_i,u_j)}(v_{i\oplus_n r}, \quad u_{j\oplus_n 1}) \\ m_{(v_i,u_j)}(v_{i\oplus_n (r+1)}, u_{j\oplus_n 2}) \\ \qquad \ldots \\ m_{(v_i,u_j)}(v_{i\oplus_n (n-1)}, u_{j\oplus_n (n-r)}) \\ m_{(v_i,u_j)}(v_{i\oplus_n 1}, \quad u_{j\oplus_n (n-r+1)}) \\ \qquad \ldots \\ m_{(v_i,u_j)}(v_{i\oplus_n (r-1)}, u_{j\oplus_n (n-1)}). \end{array}$$

Observe that the destinations $v_{i\oplus_n \ell}$ are in the order given by $\ell = r, r+1, \ldots, n-1, 1, \ldots, r-1$. That is, the natural sequence $1, 2, \ldots, n-1$ which we used in the first total exchange in B_i is left-rotated by r positions. Based on this observation, it is easy to verify that the above set of messages can be given in the compact form:

$$\left\{ m_{(v_i,u_j)}(v_{i\oplus_n ((r-1)\oplus_{n-1}\ell)}, u_{j\oplus_n \ell}) \mid \ell = 1, 2, \ldots, n-1 \right\}. \tag{7}$$

Similarly, it is seen that after the rth exchange in B_i, node (v_i, u_j) will have received messages

$$\left\{ m_{(v_i,u_{j\ominus_n ((r-1)\oplus_{n-1}\ell)})}(v_{i\oplus_n \ell}, u_j) \mid \ell = 1, 2, \ldots, n-1 \right\}, \tag{8}$$

which can be used during the $(r+1)$th total exchange in A_j.

Let us recapitulate. During the first total exchange in A_j, (v_i, u_j) uses $m_{(v_i,u_j)}(*, u_j)$. Simultaneously, total exchanges in B_i start. During the rth exchange in B_i the same node sends the set of messages given in (7), and receives the set given in (8). This set will be used for the $(r+1)$th exchange in A_j. This will occur for all $r = 1, 2, \ldots, n-1$. All total exchanges in B_i are performed in parallel with the total exchanges in A_j.

The last (nth) total exchange in A_j will involve the messages received during the $(n-1)$th total exchange in B_i. It can be noticed that (v_i, u_j) has sent all its messages meant for nodes in all other copies of A, A_k $(k \neq j)$, *except* for nodes (v_i, u_k). In the example of Fig. 4, we saw that during the first two exchanges in B_1, node (1,1) sent all its messages with the exception of messages $m_{(1,1)}(1,2)$ and $m_{(1,1)}(1,3)$ which are destined for node (1,2) and (1,3). The situation is similar for nodes (1,2) and (1,3). In conclusion, messages $m_{(v_i,u_j)}(v_i, *)$ of node (v_i, u_j) are the only messages remaining to be sent. Observe that this is a perfect set of messages for a (final) total exchange in B_i. This nth exchange can be performed while the nth exchange in A_j occurs.

What we have described up to now is formulated as Algorithm A2 in Fig. 5. The total exchanges in the copies of A and B are completely parallelized, hence lines 1–3. Lines 4–8 perform the transfers we described above in B_i. Lines 9–13 perform the total exchanges in A_j. Notice how simple lines 11–13 are: whatever was sent through the rth exchange in B_i is used during the $(r+1)$th exchange in A_j.

As it is, the algorithm works for any two-dimensional homogeneous network. Extension to more than two dimensions seems rather difficult because the homogeneity will be lost, in the sense that A could be different than B. For example, if $G = H^3$, G can be written as $G = A \times B$ only if $A = H^2$ and $B = H$ or vice versa.

However, it is easy to see that the algorithm is applicable if the dimensionality is *a power of 2*. If $G = H^{2^k}$ then we let $A = H^{2^{k-1}}$ and $B = H^{2^{k-1}}$. The algorithm can then be applied recursively for A and B, by e.g. setting $A = H^{2^{k-2}} \times H^{2^{k-2}}$, and so on.

We proceed now to determine the time requirements of Algorithm A2 and to give optimality conditions.

5.1 Optimality conditions

Theorem 4 *If H has n nodes and total exchange in H requires T_H steps then Algorithm A2 in $G = H \times H$ requires time equal to:*

$$T = nT_H.$$

Proof. Procedure TEA() performs n total exchanges in A_j (for all $j = 1, 2, \ldots, n$ in parallel), thus requiring nT_H steps. Similarly, TEB() also requires nT_H steps. The algorithm finishes when both procedures have finished, i.e. at time $T = nT_H$. □

By the recursive application of the algorithm for networks where the dimensionality is a power of 2 we have the following corollary.

Corollary 4 *Let $G = H^d$, where $d = 2^k$. If total exchange in H requires T_H time units, then total exchange in G can be performed in*

$$T = n^{d-1}T_H$$

steps.

Proof. The proof is by induction. The case of two dimensions was covered in Theorem 4. If, as an induction hypothesis, for $G' = H^{d/2}$ we need time $T' = n^{d/2-1}T_H$ then set $G = G' \times G'$ and apply Theorem 4 with G' treated as H, T' treated as T_H, and $n^{d/2}$ treated as n. It is then seen that $T = n^{d/2}T' = n^{d-1}T_H$, as claimed. □

Theorem 5 *Let $G = H^d$, where $d = 2^k$. If total exchange in H can be performed in time equal to the lower bound of (4) then the same is true for G.*

Proof. From Corollary 4, total exchange in G requires $T = n^{d-1}T_H$ time units, where $n = |V_H|$. If T_H achieves the lower bound in (4) then there exists a partition V_{H_1}, V_{H_2} of the node set of H such that

$$T_H = \frac{|V_{H_1}||V_{H_2}|}{C_{V_{H_1}V_{H_2}}},$$

where $C_{V_{H_1}V_{H_2}}$ is the number of links separating the two parts.

Consider the following partition of V, the node set of G:

$$\begin{aligned} V_1 &= \bigcup_{v_i \in V_{H_1}} (v_i, *, \ldots, *) \\ V_2 &= \bigcup_{v_i \in V_{H_2}} (v_i, *, \ldots, *). \end{aligned}$$

Then clearly, $|V_1| = |V_{H_1}|n^{d-1}$ and $|V_2| = |V_{H_2}|n^{d-1}$. Notice that G contains n^{d-1} copies of H and that in order to separate the two parts we only need to disconnect each copy of H, by removing links only in the first dimension. Since $C_{V_{H_1}V_{H_2}}$ links are needed to disconnect each copy of H, we obtain

$$C_{V_1V_2} = n^{d-1}C_{V_{H_1}V_{H_2}}.$$

Thus, V_1 and V_2 is a partition of G such that

$$\frac{|V_1||V_2|}{C_{V_1V_2}} = \frac{n^{d-1}|V_{H_1}|n^{d-1}|V_{H_2}|}{n^{d-1}C_{V_{H_1}V_{H_2}}} = n^{d-1}T_H,$$

which is equal to T, the time needed for total exchange in G. Thus the bound in (4) is tight for G, too. $\square$

Summarizing, Algorithm A2 is a multiport total exchange algorithm for homogeneous networks whose dimensionality is a power of 2. If total exchange in H can be performed in time equal to the lower bound of (4) then Algorithm A2 optimally solves the problem in G. For example, in [7] we have given algorithms that achieve this lower bound in linear arrays and rings. Consequently, Algorithm A2 leads to an optimal total exchange algorithm for homogeneous meshes and tori with 2^k $(k \geq 1)$ dimensions.

6 Discussion

We have given a systematic procedure for performing total exchange in multidimensional networks. The main contribution is probably the existence of a decomposition of the problem to simpler subproblems. Given that we have total exchange algorithms for single dimensions, we can synthesize an algorithm for the multidimensional structure. In contrast with all the other works on the problem, this approach is not limited to one particular network but to *any* graph that can be expressed as a cartesian product.

Except for the structured nature of our method, we also showed that it is optimal with respect to the number of communication steps for many popular networks. Under the single-port assumption, Algorithm A1 provides optimal solutions for hypercubes, k-ary n-cubes, general tori and actually any product of Cayley graphs. For most of these networks, this is the first optimal algorithm to appear in the literature.

Under the multiport assumption, we reached similar conclusions for homogeneous networks with 2^k dimensions: Algorithm A2 solves the problem in any such network. Optimality is also guaranteed if the single-dimension algorithm achieves the bound of (4). In particular, based on known results for linear arrays and rings, meshes and k-ary n-cubes with 2^k dimensions can optimally take advantage of our algorithm. We are currently studying the behavior of the algorithm in the case where the number of dimensions is not a power of two.

Algorithm A2 can be trivially adapted to work in the case of a graph $A \times B$ where $A \neq B$ (non-homogeneous graph) but both A and B have the same number of nodes. In this case it can be easily seen that the algorithm would require time equal to $n \max\{T_A, T_B\}$, where T_A, T_B are the corresponding total exchange times in the two dimensions. Optimality cannot be guaranteed though, even if T_A and T_B are optimal. Generalizing the algorithm to other non-homogeneous graphs seems more difficult except maybe for the case where the number of nodes in A is a multiple of the number of nodes in B (or vice versa). The problem lies in the fact that the number of messages transferred during a total exchange in one dimension may not be enough to perform a total exchange in the other dimension.

It would be interesting to study how the algorithms presented here apply to wormhole routed networks. We believe that our ideas could be useful in deriving good total exchange algorithms for these networks but in order to evaluate them we need different lower bounds due to the different assumptions that pertain this switching scheme. It seems quite an open research area as there are not too many total exchange algorithms known for this model.

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, C. Ozveren, G. D. Stamoulis, P. Tseng and J. N. Tsitsiklis, "Optimal communication algorithms for hypercubes," *J. Parallel Distrib. Comput.*, Vol. 11, pp. 263–275, 1991.

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

[4] 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.

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

[6] 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.

[7] V. V. Dimakopoulos and N. J. Dimopoulos, "Optimal total exchange in linear arrays and rings," in *Proc. ISPAN'94, Int'l Symp. Parall. Arch., Algor. and Networks*, Kanazawa, Japan, Dec. 1994, pp. 230–237.

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

[9] 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, Sept. 1995.

[10] P. Fraigniaud and E. Lazard, "Methods and problems of communication in usual networks," *Discrete Appl. Math.*, Vol. 53, pp. 79–133, 1994.

[11] D. B. Gannon and J. van Rosendale, "On the impact of communication complexity on the design of parallel numerical algorithms," *IEEE Trans. Comput.*, Vol. C-33, No. 12, pp. 1180–1194, Dec. 1984.

[12] E. Ganesan and D.K. Pradhan, "The hyper-deBruijn network: Scalable Versatile Architecture," *IEEE Trans. Parall. Distrib. Syst.*, Vol. 4, No. 9, pp. 962–978, Sept. 1993.

[13] S. Hiranandani, K. Kennedy and C. - W. Tseng, "Compiling Fortran D for MIMD distributed-memory machines," *Commun. ACM*, Vol. 35, No. 8, pp. 66–80, Aug. 1992.

[14] S. L. Johnsson, "Communication efficient basic linear algebra computations on hypercube architectures," *J. Parallel Distrib. Comput.*, Vol. 4, pp. 133–172, 1987.

[15] 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.

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

[17] D. B. Loveman, “High Performance Fortran,” *IEEE Parallel Distrib. Tech.*, Vol. 1, pp. 25–42, Feb. 1993.

[18] P. K. McKinley, Y. - jia Tsai and D. F. Robinson, “Collective communication in wormhole-routed massively parallel computers,” *IEEE Computer*, Vol. 28, No. 12, pp. 39–50, Dec. 1995.

[19] Message Passing Interface Forum, “MPI: A message-passing interface standard,” Technical Report CS-94-230, University of Tennessee, Apr. 1994.

[20] J. Misŝić and Z. Jovanović, “Communication aspects of the star graph interconnection network,” *IEEE Trans. Parall. Distrib. Syst.*, Vol. 5, No. 7, pp. 678–687, July 1994.

[21] S. Öhring and S. K. Das, “Folded petersen cube networks: new competitors for the hypercubes,” *IEEE Trans. Parall. Distrib. Syst.*, Vol. 7, No. 2, pp. 151–168, Feb. 1996.

[22] D. A. Reed and D. C. Grunwald, “The performance of multicomputer interconnection networks,” *IEEE Computer*, Vol. 20, No. 6, pp. 63–73, June 1987.

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

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

[25] 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.

[26] A. S. Youssef and B. Narahari, “The Banyan-hypercube networks,” *IEEE Trans. Parall. Distrib. Syst.*, Vol. 1, No. 2, pp. 160–169, Apr. 1990.

	For A_1	$\cdots$	For A_j	$\cdots$	For A_k	$\cdots$	For A_{n_2}
R_1	$m_s(v_1, u_1)$	$\cdots$	$m_s(v_1, u_j)$	$\cdots$	$m_s(v_1, u_k)$	$\cdots$	$m_s(v_1, u_{n_2})$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$
R_{i-1}	$m_s(v_{i-1}, u_1)$	$\cdots$	$m_s(v_{i-1}, u_j)$	$\cdots$	$m_s(v_{i-1}, u_k)$	$\cdots$	$m_s(v_{i-1}, u_{n_2})$
R_i	$m_s(v_i, u_1)$	$\cdots$	—	$\cdots$	$m_s(v_i, u_k)$	$\cdots$	$m_s(v_i, u_{n_2})$
R_{i+1}	$m_s(v_{i+1}, u_1)$	$\cdots$	$m_s(v_{i+1}, u_j)$	$\cdots$	$m_s(v_{i+1}, u_k)$	$\cdots$	$m_s(v_{i+1}, u_{n_2})$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$
R_{n_1}	$m_s(v_{n_1}, u_1)$	$\cdots$	$m_s(v_{n_1}, u_j)$	$\cdots$	$m_s(v_{n_1}, u_k)$	$\cdots$	$m_s(v_{n_1}, u_{n_2})$

Table 1: Messages to be transferred from node $s = (v_i, u_j)$. Column j is actually unused since messages of (v_i, u_j) for A_j do not have to be transferred to any other copy of A.

For A_1	For A_2	For A_3
–	$m_{(1,1)}(1,2)$	$m_{(1,1)}(1,3)$
$m_{(1,1)}(2,1)$	$m_{(1,1)}(2,2)$	$m_{(1,1)}(2,3)$
$m_{(1,1)}(3,1)$	$m_{(1,1)}(3,2)$	$m_{(1,1)}(2,3)$

Table 2: Messages to be transferred from node (1,1)

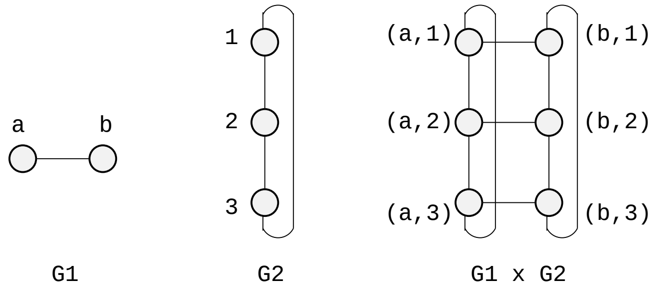

Figure 1: Cartesian product of two graphs

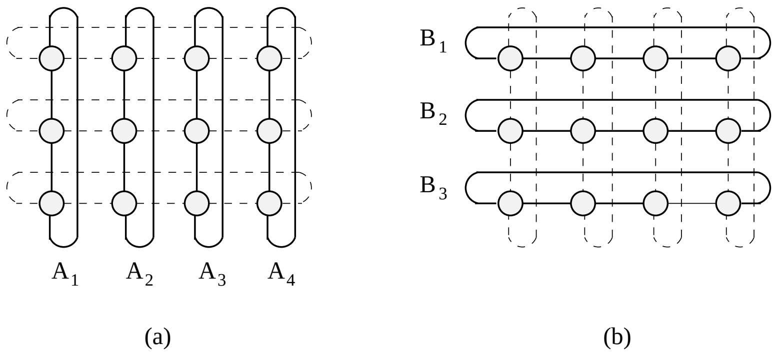

Figure 2: A 4×3 torus as (a) four copies of a three-node ring or (b) three copies of a four-node ring

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

Figure 3: Algorithm A1

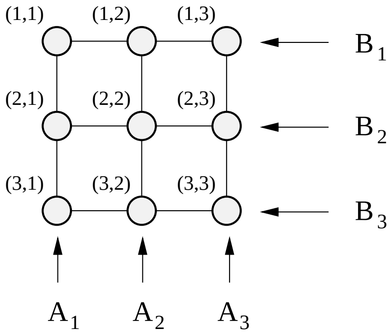

Figure 4: A 3×3 homogeneous mesh

1 Do in parallel
2   - TEA();
3   - TEB();

TEB()
4   For $r = 1, 2, \ldots, n-1$
5   Do in parallel for all B_i, $i = 1, 2, \ldots, n$
6   Perform total exchange in B_i: node (v_i, u_j) sends
$\{m_{(v_i,u_j)}(v_{i \oplus_n ((r-1) \oplus_{n-1} \ell)}, u_{j \oplus_n \ell}) \mid \ell = 1, 2, \ldots, n-1\}$;
7   Do in parallel for all B_i, $i = 1, 2, \ldots, n$
8   Perform total exchange in B_i: node (v_i, u_j) sends
$m_{(v_i,u_j)}(v_i, *)$;

TEA()
9   Do in parallel for all A_j, $j = 1, 2, \ldots, n$
10   Perform total exchange in A_j: node (v_i, u_j) sends
$m_{(v_i,u_j)}(*, u_j)$;
11   For $r = 2, \ldots, n-1$
12   Do in parallel for all A_j, $j = 1, 2, \ldots, n$
13   Perform total exchange in A_j: node (v_i, u_j) sends
the messages received from the second dimension (B_i);

Figure 5: Algorithm A2 for multiport homogeneous networks: $G = A \times B$ and $A = B = H$.