JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING **11,** 263–275 (1991)

Optimal Communication Algorithms for Hypercubes*

D. P. BERTSEKAS, C. ÖZVEREN, G. D. STAMOULIS, P. TSENG, AND J. N. TSITSIKLIS[†]

Laboratory for Information and Decision Systems, Room 35-210, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

We consider the following basic communication problems in a hypercube network of processors: the problem of a single processor sending a different packet to each of the other processors, the problem of simultaneous broadcast of the same packet from every processor to all other processors, and the problem of simultaneous exchange of different packets between every pair of processors. The algorithms proposed for these problems are optimal in terms of execution time and communication resource requirements; that is, they require the minimum possible number of time steps and packet transmissions. In contrast, algorithms in the literature are optimal only within an additive or multiplicative factor. © 1991 Academic Press, Inc.

1. INTRODUCTION AND PROBLEM FORMULATION

When algorithms are executed in a network of processors, it is necessary to exchange some intermediate information between the processors. The interprocessor communication time may be substantial relative to the time needed exclusively for computations, so it is important to carry out the information exchange as efficiently as possible. There are a number of generic communication problems that arise frequently in numerical and other algorithms. In this paper, we describe new algorithms for solving some of these problems on a hypercube. Algorithms for solving such problems have been studied in such works as [15, 7], among others. In this paper, we present some new algorithms for the hypercube that are *optimal*, in the sense that they execute the required communication tasks in the minimum possible number of time steps and link transmissions.

To define a hypercube network (or d-cube), we consider the set of points in d-dimensional space with each coordinate equal to 0 or 1. We let these points correspond to processors, and we consider a communication link for every two points differing in a single coordinate. We thus obtain an undirected graph with the processors as nodes and the communication links as arcs. The binary string of length d that corresponds to the coordinates of a node of the d-cube is referred to as the *identity number* of the node. We recall that a hypercube of any dimension can be constructed by connecting lower-dimensional cubes, starting with a 1-cube. In particular, we can start with two $(d-1)$-dimensional cubes and introduce a link connecting each pair of nodes with the same identity number (see, e.g., [1, Sect. 1.3]). This constructs a d-cube with the identity number of each node obtained by adding a leading 0 or a leading 1 to its previous identity, depending on whether the node belongs to the first $(d-1)$-dimensional cube or the second (see Fig. 1). When confusion cannot arise, we refer to a d-cube node interchangeably in terms of its identity number (a binary string of length d) and in terms of the decimal representation of its identity number. Thus, for example, the nodes $(00\cdots0)$, $(00\cdots1)$, and $(11\cdots1)$ are also referred to as nodes 0, 1, and 2^d-1, respectively.

The *Hamming distance* between two nodes is the number of bits in which their identity numbers differ. Two nodes are directly connected with a communication link if and only if their Hamming distance is unity, that is, if and only if their identity numbers differ in exactly one bit. The number of links on any path connecting two nodes cannot be less than the Hamming distance of the nodes. Furthermore, there is a path with a number of links that is equal to the Hamming distance, obtained, for example, by switching in sequence the bits in which the identity numbers of the two nodes differ (equivalently, by traversing the corresponding links of the hypercube). Such a path is referred to as a *shortest path* in this paper and a tree consisting of shortest paths from some node to all other nodes is referred to as a *shortest path tree*.

Information is transmitted along the hypercube links in groups of bits called *packets*. In our algorithms we assume that the time required to cross any link is the same for all packets and is taken to be one unit. Thus, our analysis applies to communication problems where all packets have roughly equal length. We assume that packets can be simultaneously transmitted along a link in both directions and that their transmission is error free. Only one packet can travel along a link in each direction at any one time; thus, if more than one packet is available at a node and is scheduled to be transmitted on the same incident link of the node, then only one of these packets can be transmitted at the next time period, while the remaining packets must be stored at the node while waiting in queue.

* Research supported by the NSF under Grants ECS-8519058 and ECS-8552419, with matching funds from Bellcore Inc., the ARO under Grant DAAL03-86-K-0171, and the AFOSR under Grant AFOSR-88-0032.

† Laboratory for Information and Decision Systems, M.I.T., Cambridge, MA 02139.

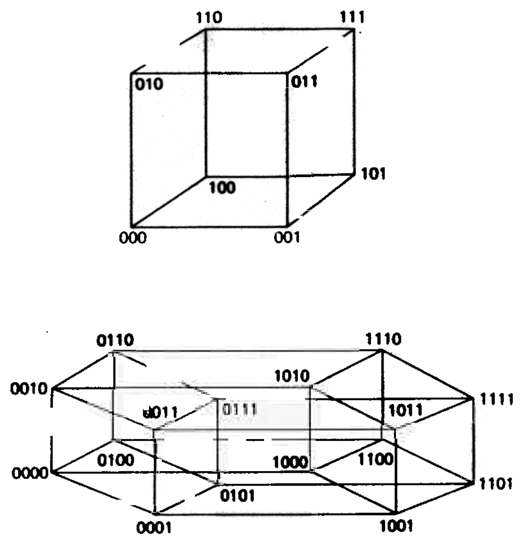

FIG. 1. Construction of a 3-cube and a 4-cube by connecting the corresponding nodes of two identical lower-dimensional cubes. A node belongs to the first lower-dimensional cube or the second depending on whether its identity has a leading 0 or a leading 1.

Each node is assumed to have infinite storage space. Moreover, we assume that all incident links of a node can be used simultaneously for packet transmission and reception; this is called the *Multiple Link Availability* (*or MLA*) *assumption*. Another possibility is the *Single Link Availability* (*or SLA*) *assumption*, where it is assumed that, at any time, a node can transmit a packet along at most one incident link and can simultaneously receive a packet along at most one incident link. Optimal algorithms under this assumption are considerably simpler than the ones for the MLA case and are not considered here (see [15, 1, 7]). Finally, we assume that each of the algorithms proposed in this paper is simultaneously initiated at all processors. This is a somewhat restrictive assumption, essentially implying that all processors can be synchronized with a global clock. For recent research on communication problems where packets are generated at random times, see [5, 20, 24].

We now describe the communication problems that we are concerned with.

The Communication Problems

One of the simplest problems is the *single node broadcast*, where we want to send the same packet from a given node, called the *root*, to every other node. Clearly, to solve this problem, it is sufficient to transmit the packet along a directed spanning tree emanating from the root node, that is, a spanning tree of the network together with a direction on each link of the tree such that there is a unique directed path on the tree from the root to every other node (all links of the path must be oriented away from the root). This problem has been discussed extensively under various assumptions in [15, 12, 7], so we do not discuss it further.

In a generalized version of the single node broadcast, we want to do a single node broadcast simultaneously from all nodes (we call this a *multinode broadcast*). To solve the multinode broadcast problem, we need to specify one spanning tree per root node. The difficulty here is that some links may belong to several spanning trees; this complicates the timing analysis, because several packets can arrive simultaneously at a node and require transmission on the same link with a queueing delay resulting.

Another interesting communication problem is sending a packet from every node to every other node (here a node sends different packets to different nodes, in contrast with the multinode broadcast problem, where a node sends the same packet to every other node). We call this the *total exchange problem*. A related problem, called the *single node scatter* problem, involves sending a separate packet from a single node, called the root, to every other node.

Table 1 gives the main results for the preceding three communication problems. One of the columns gives the number of time units required to solve the problems. We show that each of these numbers is a lower bound on the number of time units taken by any algorithm that solves the corresponding problem, and we describe an algorithm that attains the lower bound. The other column gives the number of packet transmissions required to solve the corresponding communication problems. These numbers are lower bounds on the number of packet transmissions taken by any algorithms that solve the corresponding problems, and the lower bounds are attained by the same algorithms that attain the corresponding lower bounds for the execution time. Thus, there are algorithms that are simultaneously optimal in terms of execution time and number of packet transmissions for our communication problems.

Related Research

Algorithms for the communication problems of this paper were first considered in [15], which also discusses the effects

TABLE I

Optimal Times and Optimal Numbers of Packet Transmissions for Solving the Three Basic Communication Problems on a *d*-Cube for the Case Where Simultaneous Transmission Along All Incident Links of a Node Is Allowed (the MLA Assumption)

Problem	Time	Number of transmissions
Single node scatter	$\left\lceil \frac{2^d - 1}{d} \right\rceil$	$d2^{d-}$
Multinode broadcast	$\left\lceil \frac{2^d - 1}{d} \right\rceil$	$2^d(2^d - 1)$
Total exchange	2^{d-1}	$d2^{2d-1}$

Note. We assume that each packet requires unit time for transmission on any link.

of the packet overhead and the data rate (denoted by β and τ, respectively, in [15]) on the transmission times. The problems are named somewhat differently in [15] than here. We essentially follow the communication model of [15], except that the hypercube links are assumed to be unidirectional in that work; this increases the algorithm execution times by a factor of 2 for the multinode broadcast and the total exchange problems. To compare the results of [15] with those of the present paper, the times of [15] should be used with $\beta = 0$, $m = 1$, and $\tau = 1$ and, for the aforementioned problems, should be divided by 2. A multinode broadcast algorithm (under the MLA assumption) which is slightly suboptimal (by no more than d time units) is given in [15]. This algorithm is constructed by specifying a packet transmission schedule at a single node and then properly replicating that schedule at each node, exploiting the symmetry of the d-cube. In contrast, we obtain optimal multinode broadcast algorithms starting from a suitable single node broadcast algorithm and replicating that algorithm at each node. This approach was used for meshes in general in [14], where a slightly suboptimal (by no more than $d - 3$ time units) multinode broadcast algorithm was given. The same approach was also used in [7]. The total exchange algorithm, given in Ref. [15] under the MLA assumption, assumes that each node has m packets to send to every other node. This algorithm is optimal only if m is a multiple of d; for $m = 1$, it is suboptimal by a factor of d. An algorithm similar to the total exchange algorithm of [15] is also given in [12]. An alternative approach to optimal algorithms for the total exchange problem was recently presented in [24, 23]. Optimal algorithms for single node scatter and total exchange were also given in [15] under the SLA assumption (even though the SLA assumption does not explicitly appear in [15]).

The problems of this paper have also been considered in [7], where optimal and nearly optimal algorithms are given on the basis of a different model of communication. This model differs from ours in that it quantifies the effects of setup time (or overhead) per packet, while it allows packets to have variable length and to be split and be recombined prior to transmission on any link in order to save on setup time. In the model of [7], each packet may consist of data originating at different nodes and/or destined for different nodes. The extra overhead for splitting and combining packets is considered negligible in the model of [7]. Our model may be viewed as the special case of the model of [7] in which packets have a fixed length and splitting and combining of packets is not allowed. Under the assumptions of our model, the algorithms given in [7] for single node scatter, multinode broadcast, and total exchange are not exactly optimal, although some of them are optimal up to a small additive term, and are exactly optimal when d is a prime number (they are also optimal if each node has a multiple of d packets to send to each destination node). In contrast, our corresponding algorithms are exactly optimal for all d and are unimprovable as far as time and communication requirements are concerned.

Even if there is no incentive to combine packets into larger packets to save on setup time, there is sometimes an incentive for splitting packets into smaller packets that can travel independently through the network. The idea here is to parallelize communication through pipelining the smaller packets over paths with multiple links and is inherent in proposals for virtual cut-through and wormhole routing [11, 3]. A little thought shows that (as long as the associated extra overhead is not excessive) the single node broadcast time can be reduced by dividing packets into smaller packets (see also [7]). On the other hand this is essentially impossible for the three basic communication problems considered in this paper (under the MLA assumption); from Table 1 it is seen that in an optimal algorithm, there is almost 100% utilization of some critical communication resource (the d links outgoing from the root in single node scatter and all of the $d2^d$ directed network links in multinode broadcast and total exchange). Any communication algorithm for these problems that divides packets into smaller packets cannot reduce the total usage of the corresponding critical resource and therefore cannot enjoy any pipelining advantage.

We also note that in addition to [15, 12, 7], there are several other works dealing with various communication problems and network architectures related to those discussed in the present paper (see [2, 4, 6, 9, 10, 16–19, 21, 22]).

To summarize, the new results of the present paper are the optimal algorithms for single node scatter, multinode broadcast, and total exchange (under the MLA assumption). In all of our algorithms, all packets originating from the same node are routed through a spanning tree rooted at this node. Our single node scatter algorithm uses a new perfectly balanced spanning tree construction, that is, a spanning tree with d subtrees that are as close to being equal in size as possible. To the best of our knowledge, this is the first spanning tree that is provably perfectly balanced. (A spanning tree proposed in [7] attains this property only when d is a prime number.) Also, our multinode broadcast algorithm uses another new and interesting spanning tree construction. These spanning trees could prove useful in other algorithms; in general, our algorithms can be used for the optimal solution of various other communication problems, by introducing appropriate modifications (see [1, Sect. 1.3]). Regarding the appropriateness of our assumptions, our view is that it is important to consider several types of communication models, given the broad variety of present and future communication hardware. Our fixed packet length model has the advantages of simplicity and flexibility; we believe that algorithms based on our model are likely to be adaptable with small variations to many types of communication contexts. (For such an adaptation and corresponding analysis of our multinode broadcast algorithm in the case where the packet lengths are random, see [24].) It is also worth noting that the emerging standard for high-speed communications, the Asynchronous Transfer Mode (see, e.g., [13]), is based on fixed packet lengths as well as minimal packet processing at the nodes, which favors neither splitting nor combining packets.

2. MULTINODE BROADCAST UNDER THE MLA ASSUMPTION

We first note that in a multinode broadcast each node must receive a total of $2^d - 1$ packets over its d incident links, so $\lceil (2^d - 1)/d \rceil$ is a lower bound for the time required by any multinode broadcast algorithm under the MLA assumption. We obtain an algorithm that attains this lower bound.

As a first step toward constructing such an algorithm, we represent any single node broadcast algorithm from node $(00\cdots0)$ to all other nodes that takes q time units by a sequence of sets of directed links $A_1, A_2, \ldots, A_q$. Each A_i is the set of links on which transmission of the packet begins at time $i - 1$ and ends at time i. Naturally, the sets A_i must satisfy certain consistency requirements for accomplishing the single node broadcast. In particular, if S_i and E_i are the sets of identity numbers of the start nodes and end nodes, respectively, of the links in A_i, we must have $S_1 = \{(00\cdots0)\}$ and $S_i \subset \{(00\cdots0)\} \cup (\cup_{k=1}^{i-1} E_k)$. Furthermore, every nonzero node identity must belong to some E_i. The set of all nodes together with the set of links $(\cup_{i=1}^{q} A_i)$ must form a subgraph that contains a spanning tree (see Fig. 2a); in fact, to minimize the number of packet transmissions, the sets of links $A_1, A_2, \ldots, A_q$ should be disjoint and should form a spanning tree.

Consider now a d-bit string t representing the identity number of some node on the d-cube. For any node identity z, we denote by $t \oplus z$ the d-bit string obtained by performing modulo 2 addition of the jth bit of t and z for each $j = 1, 2, \ldots, d$. It can be seen that an algorithm for broadcasting a packet from the node with identity t can be specified by the sets

$$A_i(t) = \{(t \oplus x, t \oplus y) \mid (x, y) \in A_i\}, \quad i = \ ,2, \quad ,q,$$

where $A_i(t)$ denotes the set of links on which transmission of the packet begins at time $i - 1$ and ends at time i. The proof of this is based on the fact that $t \oplus x$ and $t \oplus y$ differ in a particular bit if and only if x and y differ in the same bit, so $(t \oplus x, t \oplus y)$ is a link if and only if (x, y) is a link. Figure 2 illustrates the sets $A_i(t)$ corresponding to all possible t for the case where $d = 3$.

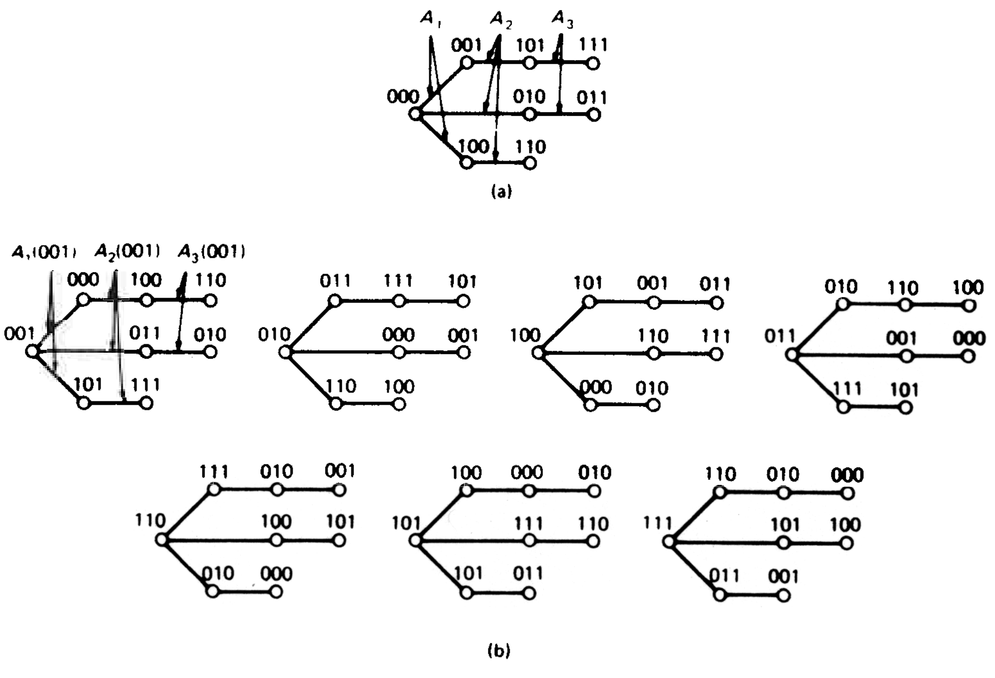

FIG. 2. Generation of a multinode broadcast algorithm for the d-cube, starting from a single node broadcast algorithm. (a) The algorithm that broadcasts a packet from the node with identity $(00\cdots0)$ to all other nodes is specified by a sequence of sets of directed links $A_1, A_2, \ldots, A_q$. Each A_i is the set of links on which transmission begins at time $i - 1$ and ends at time i. (b) A corresponding broadcast algorithm for each root node identity t is specified by the sets of links $A_i(t) = \{(t \oplus x, t \oplus y) \mid (x, y) \in A_i\}$, where we denote by $t \oplus z$ the d-bit string obtained by performing modulo 2 addition of the jth bit of t and z for $j = 1, 2, \ldots, d$. The multinode broadcast algorithm is specified by the requirement that transmission of the packet of node t starts on each link in $A_i(t)$ at time $i - 1$. The figure shows the construction for an example where $d = 3$. Here the set A_2 has two links of the same type and the multinode broadcast cannot be executed in three time units. However, if the link (000, 010) belonged to A_1 instead of A_2, the required time would be the optimal three time units.

We now describe a procedure for generating a multinode broadcast algorithm specified by the sets $A_i(t)$ for all possible values of i and t, starting from a single node broadcast algorithm specified by the sets $A_1, A_2, \ldots, A_q$. Let us say that *a link* (x, y) *is of type* j if x and y differ in the jth bit. We make the following key observation: consider a single node broadcast algorithm specified by the link sets $A_1, \ldots, A_q$. If, for each i, the links in A_i are of different types, then, for each i, the sets $A_i(t)$, where t ranges over all possible identities, are disjoint. [If, for $t \neq t'$, two links $(t \oplus x, t \oplus y) \in A_i(t)$ and $(t' \oplus x', t' \oplus y') \in A_i(t')$ were the same, then the links (x, y) and (x', y') would be different (since $t \neq t'$), and they would be of the same type because (x, y) and (x', y') are of the same type as $(t \oplus x, t \oplus y)$ and $(t' \oplus x', t' \oplus y')$, respectively, which contradicts the fact that (x, y) and (x', y') belong to A_i.] This implies that the single node broadcasts of all nodes t can be executed simultaneously, without any further delay. In particular, we have a multinode broadcast algorithm that takes q time units. We proceed to give a method for selecting the sets A_i with the links in each A_i being of different types. Furthermore, we ensure that each one of the sets $A_1, \ldots, A_{q-1}$ has exactly d elements, which is the maximum possible (since there exist only d link types), thereby resulting in the minimum possible execution time of $q = \lceil (2^d - 1)/d \rceil$ units.

Let N_k, $k = 0, 1, \ldots, d$, be the set of node identities having k unity bits and $d - k$ zero bits. The number of elements in N_k is $\binom{d}{k} = d!/(k!(d-k)!)$. In particular, N_0 and N_d contain one element, the strings $(00\cdots0)$ and $(11\cdots1)$, respectively; the sets N_1 and N_{d-1} contain d elements; and for $2 \leq k \leq d-2$ and $d \geq 5$, N_k contains at least $2d$ elements (when $d = 4$, the number of elements of N_2 is six, as shown in Fig. 3). We partition each set N_k, $k = 1, \ldots, d-1$, into disjoint subsets $R_{k1}, \ldots, R_{kn_k}$, which are equivalence classes under a single bit rotation to the left. We impose the restriction that R_{k1} is the equivalence class of the element whose k rightmost bits are unity. We associate each node identity t with a distinct number $n(t) \in \{0, 1, 2, \ldots, 2^d - 1\}$ in the order

$$(00\cdots0)R_{11}R_{21} \cdots R_{2n_2} \cdots R_{k1} \cdots R_{kn_k} \cdot R_{(d-2)1} \cdots R_{(d-2)n_{d-2}} R_{(d-1)1} (\quad \cdots 1)$$

[i.e., $n(00\cdots0) = 0$, $n(11\cdots1) = 2^d - 1$, and the other node identities are numbered consecutively in the above order between 1 and $2^d - 2$]. Let

$$m(t) = \quad + [(n(t) - 1)(\text{mod } d)].$$

Thus, the sequence of numbers $m(t)$ corresponding to the sequence of node identities

$$R_{11}R_{21}R_{22} \cdots R_{(d-1)1}$$

is $1, 2, \ldots, d, 1, 2, \ldots, d, 1, 2, \ldots$ (cf. Fig. 3 for the case $d = 4$). We specify the order of node identities within each set R_{kn} as follows: the first element t in each set R_{kn} is chosen so that the relation

$$\text{the bit in position } m(t) \text{ from the right is a} \qquad 1)$$

is satisfied, and the subsequent elements in R_{kn} are chosen so that each element is obtained by a single bit rotation to the left of the preceding element. Also, for the elements t of R_{k1}, we require that the bit in position $m(t) - 1$ [if $m(t) > 1$] or d [if $m(t) = 1$] from the right be a 0. For $i = 1, 2, \ldots, \lceil (2^d - 1)/d \rceil - 1$, define

$$E_i = \{t \mid (i-1)d + \quad \leq n(t) \leq id\},$$

and for $i = 0$ and $i = q = \lceil (2^d - 1)/d \rceil$, define

$$E_0 = \{(00\cdots0)\},$$
$$E_q = \{t \mid (q-1)d + \quad \leq n(t) \leq 2^d -$$

We define the set of links A_i as follows:

For $i = 1, 2, \ldots, q$, each set A_i consists of the links that connect the node identities $t \in E_i$ with the corresponding node identities of $\bigcup_{k=0}^{i-1} E_k$ obtained from t by reversing the bit in position $m(t)$ [which is always a 1 by property (1)]. In particular, the node identities in each set in R_{k1} are connected with corresponding node identities in $R_{(k-1)1}$, because, by construction, the bit in position $m(t)$ lies next to a 0 for each node identity t in the set R_{k1}.

To show that this definition of the sets A_i is legitimate, we need to verify that by reversing the specified bit of a node identity $t \in E_i$, we indeed obtain a node identity t' that belongs to $\bigcup_{k=1}^{i-1} E_i$, as opposed to E_i. [It cannot belong to E_k for $k > i$, because $n(t') < n(t)$.]

To see this in the exceptional case where $t = (11\cdots1)$, note that by the preceding rule, t' is the element of $R_{(d-1)1}$ with a zero bit in position $m(t)$ from the right. The elements of $R_{(d-1)1}$ are ordered so that the bit of t' in position $m(t') - 1$ (if $m(t') > 1$) or in position d (if $m(t') = 1$) is a 0. Since $2^d - 1$ is not divisible by d (see the appendix), we have $m(t) \neq d$. Thus, the zero bit of t' cannot be in position d, so it must be in position $m(t') - 1$, implying that $m(t) = m(t') - 1$. The set $R_{(d-1)1}$ has d elements, and as a result, its first element t'' satisfies $m(t'') = m(t)$, so t' must be the second element of $R_{(d-1)1}$. Since $m(t) \neq d$, E_q has at most $d - 1$ elements, and thus, we obtain the desired conclusion $t' \notin E_q$.

In the case where $t \neq (11\cdots1)$, it is sufficient to show that $n(t) - n(t') \geq d$. We consider two cases: (a) If $t \in R_{kn}$ for some $n > 1$, then all of the d elements of R_{k1} are between

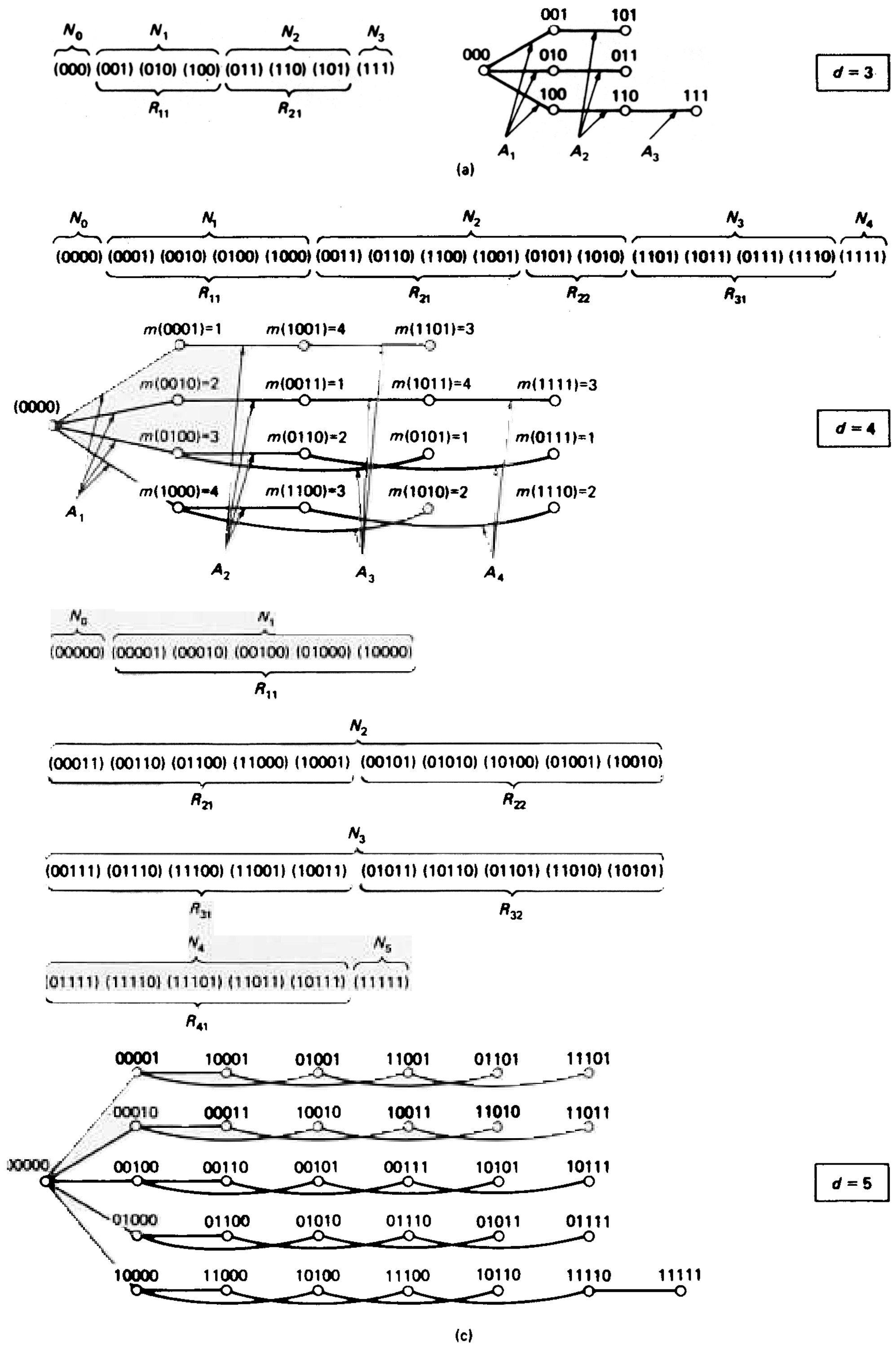

FIG. 3. Construction of a multinode broadcast algorithm for a d-cube that takes $\lceil (2^d - 1)/d \rceil$ time.

t' and t, and the inequality $n(t) - n(t') \geqslant d$ follows. (b) If $t \in R_{k1}$, then $t' \in R_{(k-1)1}$, and all the elements of the sets $R_{(k-1)2}, \ldots, R_{(k-1)n_{k-1}}$ are between t' and t. There are $\binom{d}{k-1} - d$ such elements. If $2 < k < d$ and $d \geqslant 5$, it can be verified that $\binom{d}{k-1} - d \geqslant d$, and we are done. The cases $d = 3$ and $d = 4$ can be handled individually (see Fig. 3). The cases $k = 1, 2$ create no difficulties because $R_{11} = E_1$, $R_{21} = E_2$.

We have thus shown that the sets A_i are properly defined, and we also note that any two links in each set A_i are of different types, implying that the corresponding multinode broadcast algorithm takes $q = \lceil (2^d - 1)/d \rceil$ time units. Thus, the algorithm attains the lower bound of execution time over all multinode broadcast algorithms under the MLA assumption and is optimal.

The preceding algorithm requires $2^d(2^d - 1)$ packet transmissions. This is also a lower bound on the number required by any multinode broadcast algorithm, since each of the 2^d nodes must receive a total of $2^d - 1$ different packets (one from each of the other nodes). Therefore the algorithm is also optimal in terms of total required communication resource.

3. SINGLE NODE SCATTER UNDER THE MLA ASSUMPTION

Consider the d-cube and the problem of single node scatter with root node s. Since $2^d - 1$ different packets must be transmitted by the root node over its d incident links, any algorithm solving these problems requires at least $\lceil (2^d - 1)/d \rceil$ time units under the MLA assumption. This time can be achieved by modifying the corresponding optimal multinode broadcast algorithm of the previous section, thereby justifying the entries of Table 1 for single node scatter.

The modified multinode broadcast algorithm is not, however, optimal for the scatter problem with respect to the number of packet transmissions. To see this, note that a packet destined for some node must travel a number of links at least equal to the Hamming distance between that node and the root. Therefore, a lower bound for the optimal number of packet transmissions is the sum of the Hamming distances of all nodes to the root. There are $\binom{d}{k} = d!/(k!(d-k)!)$ nodes that are at distance k from the root, so this bound is

$$\sum_{k=1}^{d} k\binom{d}{k} = d2^{d-1} \tag{2}$$

The lower bound of Eq. (2) is much smaller than the $2^d(2^d - 1)$ packet transmissions required by a multinode broadcast. While it is possible to extract from the optimal multinode broadcast algorithm a single node scatter algorithm which attains the lower bound of Eq. (2), such an algorithm is quite complex to visualize and to implement. The following alternative algorithm is much simpler.

For any spanning tree T of the d-cube, let r be the number of neighbor nodes of the root node s in T, and let T_i be the subtree of T rooted at the ith neighbor of s. Consider the following rule for s to send packets to each subtree T_i:

Continuously send packets to distinct nodes in the subtree (using only links in T), giving priority to nodes furthest away from s (break ties arbitrarily).

With this rule, s starts transmitting its last packet to the subtree T_i no later than time $N_i - 1$, where N_i denotes the number of nodes in T_i, and all nodes in T_i receive their packet no later than time N_i. (To see the latter, note that all packets destined for the nodes in T_i that are k links away from s are sent no later than time $N_i - k$, and each of these packets completes its journey in exactly k time units.) Therefore, all packets are received at their respective destinations in $\max\{N_1, N_2, \ldots, N_r\}$ time units. Hence, the above algorithm attains the optimal time if and only if T has the property that s has d neighbors in T and that each subtree T_i, $i = 1, \ldots, d$, contains at most $\lceil (2^d - 1)/d \rceil$ nodes. If T is in addition a shortest path tree from s, then each packet travels along the shortest path to its destination and this algorithm also attains the optimal number of packet transmissions.

We assume without loss of generality that $s = (00 \cdots 0)$ in what follows. To construct a spanning tree T with the above two properties, let us consider the equivalence classes R_{kn} introduced in Section 2 in connection with the multinode broadcast problem. As in Section 2, we order the classes as

$$(00 \quad 0) R_{11} R_{21} \cdot \quad R_{2n_2} \cdots R_{k1} \cdots R_{kn_k}$$
$$\cdots R_{(d-2)} \quad \cdot R_{(d-2)n_{d-2}} R_{(d-1)1} (11 \cdots 1$$

and we consider the numbers $n(t)$ and $m(t)$ for each identity t, but for the moment, we leave the choice of the first element in each class R_{kn} unspecified. We denote by m_{kn} the number $m(t)$ of the first element t of R_{kn} and we note that this number depends only on R_{kn} and not on the choice of the first element within R_{kn}.

We say that class $R_{(k-1)n'}$ is *compatible* with class R_{kn} if $R_{(k-1)n'}$ has d elements (node identities) and there exist identities $t' \in R_{(k-1)n'}$ and $t \in R_{kn}$ such that t' is obtained from t by changing some unity bit of t to a 0. Since the elements of $R_{(k-1)n'}$ and R_{kn} are obtained by left shifting the bits of t' and t, respectively, it is seen that for every element x' of $R_{(k-1)n'}$ there is an element x of R_{kn} such that x' is obtained from x by changing one of its unity bits to a 0. The reverse is also true, namely that for every element x of R_{kn} there is an element x' of $R_{(k-1)n'}$ such that x is obtained from x' by changing one of its zero bits to unity.

An important fact for the subsequent spanning tree construction is that for every class R_{kn} with $2 \leqslant k \leqslant d - 1$, there exists a compatible class $R_{(k-1)n'}$. Such a class can be obtained as follows: Take any identity $t \in R_{kn}$ whose rightmost bit is a 1 and leftmost bit is a 0. Let σ be a string of consecutive 0s with maximal number of bits and let t' be the identity obtained from t by changing to 0 the unity bit immediately to the right of σ. [For example, if $t = (0010011)$, then $t' = (0010001)$ or $t' = (0000011)$, and if $t = (0010001)$, then $t' = (0010000)$.] Then the equivalence class of t' is compatible with R_{kn}, because it has d elements [$t' \neq (00 \cdots 0)$ and t' contains a unique substring of consecutive 0s with maximal number of bits, so it cannot be replicated by left rotation of less than d bits].

The spanning tree T with the desired properties is constructed sequentially by adding links incident to elements of the classes R_{kn} as follows (see Fig. 4):

> Initially T contains no links. We choose arbitrarily the first element of class R_{11} and we add to T the links connecting $(00\cdots0)$ with all the elements of R_{11}. We then consider the classes R_{kn} $(2 \leqslant k \leqslant d-1)$ one-by-one in the order indicated above, and for each R_{kn}, we find a compatible class $R_{(k-1)n'}$ and the element t' in $R_{(k-1)n'}$ such that $m(t') = m_{kn}$ (this is possible because $R_{(k-1)n'}$ has d elements). We then choose as the first element of R_{kn} an element t such that t' is obtained from t by changing one of its unity bits to a 0. Since $R_{(k-1)n'}$ has d elements and R_{kn} has at most d elements, it can be seen that, for any x in R_{kn}, we have $m(x') = m(x)$, where x' is the element of $R_{(k-1)n'}$ obtained by shifting t' to the left by the same amount as that needed to obtain x by shifting t to the left. Moreover, x' can be obtained from x by changing some unity bit of x to a 0. We add to T the links (x', x), for all $x \in R_{kn}$ (with x' defined as above for each x). After exhausting the classes R_{kn}, $2 \leqslant k \leqslant d-1$, we finally add to T the link $(x, (11\cdots1))$, where x is the element of $R_{(d-1)1}$ with $m(x) = m(11\cdots1)$.

The construction of T is such that each node $x \neq (00\cdots0)$ is in the subtree $T_{m(x)}$. Since there are at most $\lceil(2^d-1)/d\rceil$ nodes x having the same value of $m(x)$, each subtree contains at most $\lceil(2^d-1)/d\rceil$ nodes. Furthermore, the number of links on the path of T connecting any node and $(00\cdots0)$ is the corresponding Hamming distance. Hence, T is also a shortest path tree from $(00\cdots0)$, as desired.

Note that for the spanning tree constructed above, each of the subtrees T_i, contains either $\lfloor(2^d-1)/d\rfloor$ or $\lceil(2^d-1)/d\rceil$ nodes. (This follows from the discussion above and from the fact that there are at least $\lfloor(2^d-1)/d\rfloor$ nodes x having the same value of $m(x)$.) Hence, T is perfectly balanced in the sense that the T_i's are as close to being equal in size as possible.

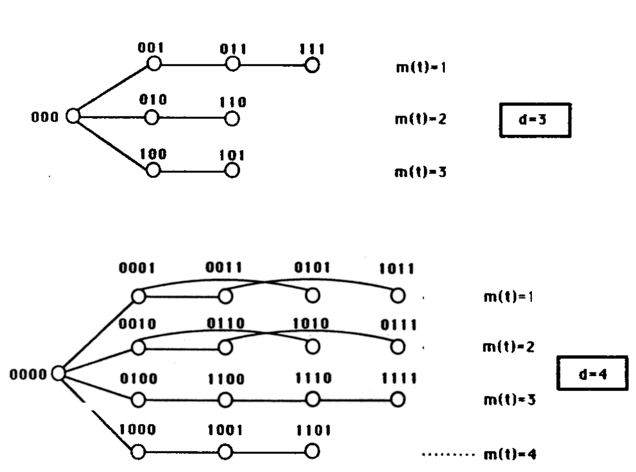

FIG. 4. Spanning tree construction for optimal single node scatter under the MLA assumption for $d = 3$ and $d = 4$.

4. TOTAL EXCHANGE UNDER THE MLA ASSUMPTION

Consider the total exchange problem under the MLA assumption. We showed in the preceding section that for any single node scatter algorithm in the d-cube, the number of packet transmissions is bounded below by $d2^{d-1}$, and it is equal to $d2^{d-1}$ if and only if packets follow shortest paths from the root to all other nodes. Since a total exchange can be viewed as 2^d separate versions of single node scatter, a lower bound for the total number of transmissions is

$$d2^d2^{d-1}. \tag{3}$$

Since each node has d incident links, at most $d2^d$ transmissions may take place at each time unit. Therefore, if T_d is the execution time of a total exchange algorithm in the d-cube, we have

$$T_d \geqslant 2^{d-1}. \tag{4}$$

For an algorithm to achieve this lower bound, it is necessary that packets follow shortest paths and that all links are busy (in both directions) during all of the 2^{d-1} time units. In what follows, we present an algorithm for which $T_d = 2^{d-1}$. In light of the above, this algorithm is optimal with respect to both the time and the number of packet transmissions criteria and achieves 100% link utilization.

We construct the algorithm recursively. We assume that we have an optimal algorithm for total exchange in the d-cube with certain properties to be stated shortly, and we use this algorithm to perform an optimal total exchange in the $(d+1)$-cube. The construction is as follows: we decompose the $(d+1)$-cube into two d-cubes, denoted C_1 and C_2 (cf. the construction of Fig. 1). Without loss of generality we assume that C_1 contains nodes $0, \ldots, 2^d-1$, and that their counterparts in C_2 are nodes $2^d, \ldots, 2^{d+1}-1$, respectively. The total exchange algorithm for the $(d+1)$-cube consists of three phases. In the first phase, there is a total exchange (using the optimal algorithm for the d-cube) within each of the cubes C_1 and C_2 (each node in C_1 and C_2 exchanges its packets with the other nodes in C_1 and C_2, respectively). In the second phase, each node transmits to its counterpart node in the opposite d-cube all of the 2^d packets that are destined for the nodes of the opposite d-cube. In the third phase, there is an optimal total exchange in each of the two d-cubes of the packets received in phase 2 (see Fig. 5). Phase 3 must

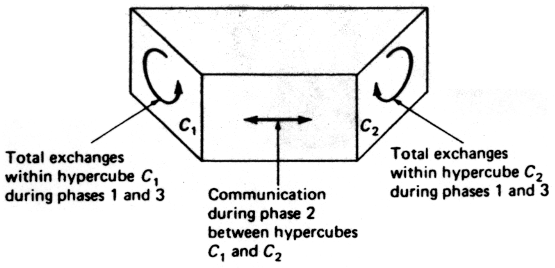
C_1
C_2
Total exchanges within hypercube C_1 during phases 1 and 3
Communication during phase 2 between hypercubes C_1 and C_2
Total exchanges within hypercube C_2 during phases 1 and 3

FIG. 5. Recursive construction of a total exchange algorithm for the d-cube. Let the $(d+1)$-cube be decomposed into two d-cubes denoted C_1 and C_2. The algorithm consists of three phases. In the first phase, there is a total exchange within each of the cubes C_1 and C_2. In the second phase, each node transmits to its counterpart node in the opposite d-cube all of the 2^d packets that are destined for the nodes of the opposite d-cube. In the third phase, there is a total exchange in each of the two d-cubes of the packets received in phase 2.

be carried out after phase 1 because during phase 1 all the links of the cubes C_1 and C_2 are continuously busy (since the d-cube total exchange algorithm is assumed optimal). On the other hand, phase 2 may take place simultaneously with both phase 1 and phase 3. In an algorithm presented in [1], phase 3 starts after the end of phase 2, resulting in an execution time of $2^d - 1$ units. Here, we improve on this time by allowing phase 3 to start before phase 2 ends. To illustrate how this is possible, consider the packet originating at some node $i \in C_1$ and destined for its counterpart node in C_2, namely $i + 2^d$, and the packet originating at $i + 2^d$ and destined for i. These packets are not transmitted at all during phase 3. Therefore, if they are transmitted last in phase 2 then phase 3 can start one time unit before the end of phase 2. This idea can be generalized as follows: clearly, if it were guaranteed that packets going from C_1 to C_2 and from C_2 to C_1 arrive sufficiently early at C_2 and at C_1, respectively, then phase 3 may be carried out just after phase 1, without completing phase 2. In such a case, the first half of phase 2 would be carried out simultaneously with phase 1, while the second half would be carried out simultaneously with phase 3, and we would have $T_{d+1} = 2T_d$. Since, by assumption, T_d is equal to the lower bound 2^{d-1} of Eq. (4) for a total exchange in the d-cube, we would have $T_{d+1} = 2^d$, implying that such an algorithm would achieve the lower bound of Eq. (4) for the $(d+1)$-cube. We prove that this is indeed feasible.

Suppose that an optimal total exchange algorithm has already been devised for the d-cube. Let $N_d(i, n)$ denote the number of its own packets that node i has transmitted up to and including time n, for $n = 1, \ldots, 2^{d-1}$ [$N_d(i, n)$ ranges from 1 to $2^d - 1$]. We can use $N_d(i, n)$ to express the requirement that phase 3 packets originating at nodes of C_1 are available in time at the appropriate nodes of C_2, so that phase 3 begins right after phase 1 and continues without delay. In particular, it is necessary that

$$N_d(i, n) \leq 2^{d-1} + n - 1, \quad \forall n = 1, \ldots, 2^{d-1}, \; i = 0, \ldots, 2^d - 1 \tag{5}$$

To see this, note that the left-hand quantity in Eq. (5) is the number of packets of node $i \in C_1$ that must be transmitted by node $i + 2^d$ during the first n time units of phase 3, while the right-hand quantity in Eq. (5) is the number of available time units within phase 2 for transferring these packets from node i to node $i + 2^d$. There is also a requirement analogous to Eq. (5) for the nodes i of C_2.

We proceed by induction, using the requirement of Eq. (5) as part of the inductive hypothesis. In particular, we prove that for every d, there exists a total exchange algorithm for the d-cube satisfying

$$T_d = 2^{d-1} \text{ and } N_d(i, n) \leq 2^{d-1} + n - 1, \quad \forall n = 1, \ldots, 2^{d-1}, \; i = 0, \quad , 2^d - 1. \tag{6}$$

We have $T_1 = 1$ and $N_1(i, 1) = 1$, for $i = 0, 1$, which proves the inductive hypothesis for $d = 1$. Assume that for some d, we have a total exchange algorithm for the d-cube that satisfies the inductive hypothesis (6), and let $s(i, j, d)$ denote the time unit in this algorithm during which node i transmits its own packet that is destined for node j. We construct a three-phase total exchange algorithm for the $(d+1)$-cube of the type described above that satisfies the inductive hypothesis. Suppose that packets are transmitted in phase 2 according to the following rules (in view of the symmetry of the transmissions of nodes of the d-cubes C_1 and C_2, we describe the rules for phase 2 packet transmissions for only the nodes of C_1):

(a) Each node $i \in C_1$ transmits its packets to node $i + 2^d$ in the order in which the latter node forwards them in phase 3 (ties are broken arbitrarily); i.e., the packet destined for $j \in C_2, j \neq i + 2^d$, is transmitted before the packet destined for $j' \in C_2, j' \neq i + 2^d$, if

$$s(i, j - 2^d, d) < s(i, j' - 2^d, d).$$

(b) Each node $i \in C_1$ transmits its packet destined for node $i + 2^d$ last.

We claim that, under the above rules, phase 3 can proceed uninterrupted after phase 1. To show this, consider any $i \in C_1$ (the case of $i \in C_2$ can be treated analogously). At the end of phase 1 node i has received exactly 2^{d-1} packets from node $i + 2^d$ (since phase 1 lasts 2^{d-1} time units by induction). Hence, n time units after the end of phase 1, node i has received exactly $2^{d-1} + n$ packets from node $i + 2^d$. On the other hand, the total number of packets of node $i + 2^d$ that node i forwards after $n + 1$ time units of phase 3 is exactly $N_d(i, n + 1)$. Since [cf. Eq. (6)] $N_d(i, n+1) \leq 2^{d-1} + n$ for all $n = 0, 1, \ldots, 2^{d-1} - 1$ and node $i + 2^d$ transmits its packets to node i according to the above rules, node i always has enough packets from node $i + 2^d$ for transmission if phase 3 begins immediately after phase 1. Since $i \in C_1$ was chosen arbitrarily, this holds for all $i \in C_1$.

Consider the total exchange algorithm for the $(d + 1)$-cube whereby phase 3 proceeds uninterrupted immediately following phase 1 as described above. Since according to the inductive hypothesis, each of phases 1 and 3 takes time $T_d = 2^{d-1}$, this algorithm takes time $2T_d = 2^d$. There remains to show that the second part of the inductive hypothesis is satisfied for $d + 1$. For any node i, let $N_{d+1}(i, n)$ denote the number of node i's own packets that i has transmitted up to and including time n in this algorithm. Since, in the first 2^{d-1} time units of this algorithm, phases 1 and 2 execute simultaneously, we obtain

$$N_{d+1}(i, n) = N_d(i, n) + n, \quad \forall n = 1, \ldots, 2^{d-1}.$$

By combining this equation with the inequality $N_d(i, n) \leqslant 2^d - 1$, which holds for all n, we obtain

$$N_{d+1}(i, n) \leqslant 2^d + n - 1, \quad \forall n = 1, \ldots, 2^{d-1}.$$

Since, in the next 2^{d-1} time units of this algorithm, phases 3 and 2 execute simultaneously (and i does not transmit any packet of its own in phase 3), we have

$$N_{d+1}(i, n) = 2^d - \quad + n, \quad \forall n = 2^{d-1} + \qquad , 2^d.$$

By combining the last two relations, it follows that $N_{d+1}(i, n)$ satisfies

$$N_{d+1}(i, n) \leqslant 2^d + n - \qquad \forall n = \qquad , 2^d.$$

Since the choice of i was arbitrary, this implies that the inductive hypothesis (6) holds for the $(d + 1)$-cube.

Implementation of the Optimal Algorithm

In what follows, we present the rules used by the nodes of the d-cube for transmitting their own packets and forwarding the packets they receive from other nodes, whenever they require further transmission. We write the identity of node i as $(i_d, \ldots, i_1)$, where each i_k, $k = 1, \ldots, d$, is a 0 or a 1. Moreover, we denote by e_k the identity of node 2^{k-1}, for $k = 1, \ldots, d$. The link between i and $i \oplus e_k$ is called the *kth link* incident to i. Finally $\bar{i}_k$ denotes the reverse of bit i_k; namely $\bar{i}_k = (i_k + 1) \bmod 2$.

We first describe the order in which an arbitrary node i transmits its own packets. It can be seen that during time units $1, \ldots, 2^{k-1}$, node i transmits all its packets destined for nodes

$$(i_d, \ldots, i_{k+1}, \bar{i}_k, x_{k-1}, \ldots, x_1),$$
$$\text{where } x_m = 0 \text{ or } 1, \text{ for } m = \qquad , k -$$

through its kth incident link, for $k = 1, \ldots, d$. The last packet to be transmitted in this group is the one destined for node $i \oplus e_k$. For $i = 0$, the exact order in which i transmits its packets on each of its incident links may be derived by using a sequence of d tables, which may be constructed iteratively. The kth table consists of k columns, the mth of which contains the destinations of the packets transmitted through link m. The first table contains only e_1. For $k = 2, \ldots, d$, the first $k - 1$ columns of the kth table are identical to those of the $(k - 1)$st table, whereas its last column consists of e_k and the entries of the $(k - 1)$st table with their kth bit being set to 1. In the last column, entries corresponding to the same row of the $(k - 1)$st table appear one after the other, ordered (arbitrarily) from left to right; entries corresponding to different rows of the $(k - 1)$st table are ordered from top to bottom. The last element of the last column is e_k. This scheme follows from the recursive construction of the algorithm. As an example, we present the scheme for $d = 4$ in Fig. 6.

For any other node i, the corresponding order of destinations may be obtained by forming the $\oplus$ operation of each entry of the dth table of 0 with i.

Time	Link 1
1	0001

Time	Link 1	Link 2
1	0001	0011
		0010

Time	Link 1	Link 2	Link 3
1	0001	0011	0101
2		0010	0111
3			0110
4			0100

Time	Link 1	Link 2	Link 3	Link 4
1	0001	0011	0101	1001
2		0010	0111	1011
3			0110	1101
4			0100	1010
5				1111
6				1110
7				1100
8				1000

FIG. 6. Implementation of the total exchange algorithm for $d = 4$.

We now consider the packets arriving at some node i and present the rules under which these packets are forwarded by i (whenever necessary). Packets arrive in i, through the kth link, in groups of 2^{k-1}, for $k = 1, \ldots, d$. Each group contains all the packets originating from the same node $(y_d, \ldots, y_{k+1}, \bar{i}_k, i_{k-1}, \ldots, i_1)$ (where $y_m = 0$ or 1, for $m = k + 1, \ldots, d$) and destined for all nodes of the form $(i_d, \ldots, i_k, x_{k-1}, \ldots, x_1)$ (where $x_m = 0$ or 1, for $m = 1, \ldots, k-1$). The order of group arrivals is lexicographic on $(y_d \oplus i_d, \ldots, y_{k+1} \oplus i_{k+1})$. Routing is accomplished as follows:

A packet destined for node $(i_d, \ldots, i_k, x_{k-1}, \ldots, x_1)$ is placed in the queue which contains packets to be transmitted by i through the k_0th link, where

$$k_0 = \max_{1 \leq m \leq k-1} \{m \mid x_m = \bar{i}_m\}.$$

Packets originating from different nodes j, j' and placed in the same queue are ordered according to the lexicographic order between $j \oplus i$ and $j' \oplus i$. Packets originating from the same node and placed in the same queue preserve their order of arrival. Forwarding packets in the kth link starts at time $2^{k-1} + 1$, for $k = 1, \ldots, d-1$; no forwarding takes place in the dth link.

The rules presented above follow from the recursive construction of the algorithm. Our earlier analysis guarantees that packets are always in time at the intermediate nodes (if any) of the paths they have to traverse. Note that the traveling schedule of each packet may be locally determined at the intermediate nodes of its path by examining the packet's origin and destination, so packets do not have to carry timing information.

5. CONCLUSIONS

Excessive communication time is widely recognized as the principal obstacle for achieving large speedup in many problems using massively parallel computing systems. This emphasizes the importance of optimal communication algorithms. In this paper, we have shown that a very strong form of optimality can be achieved for some basic communication problems in the hypercube architecture.

Our methodological ideas may find application in other related contexts. In particular, variations of our algorithms can be investigated under different and possibly less restrictive assumptions. Furthermore, some of our algorithmic constructions can be applied in other architectures to obtain optimal or nearly optimal algorithms for the communication problems of this paper. Finally, it is worth considering the potential existence of optimal algorithms for specialized communication tasks, arising in the context of specific numerical and other methods.

APPENDIX: INDIVISIBILITY OF $2^d - 1$ BY d

The proof of the following result is due to David Gillman and Arthur Mattuck of the M.I.T. Department of Mathematics and is given here with their kind permission.

PROPOSITION. *For any integer $d \geq 2$, $2^d - 1$ is not divisible by d.*

Proof. For any positive integers a, b, and d we use the notation $a \equiv b \pmod{d}$ to indicate that a and b give the same remainder when divided by d; this remainder is denoted $a \bmod d$ or $b \bmod d$. We note that for all positive integers a, b, d, and t we have

$$a \equiv b \pmod{d} \Rightarrow ta \equiv tb \pmod{d}.$$

(Write $a = p_a d + w$, $b = p_b d + w$, $tw = pd + r$, where $w = a \bmod d = b \bmod d$ and $r = (tw) \bmod d$, and note that $r = (ta) \bmod d = (tb) \bmod d$.)

It suffices to consider odd $d \geq 2$. We argue by contradiction. If the claim does not hold, let d be the smallest odd integer which is larger than 1 and is such that

$$2^d \equiv 1 \pmod{d}.$$

Let m be the smallest positive integer for which

$$2^m \equiv \quad \pmod{d}. \tag{A3}$$

We claim that $m < d$. To see this, note that the numbers

$$2 \bmod d,\ 2^2 \bmod d,\ \cdots,\ 2^d \bmod d$$

belong to $\{1, \ldots, d-1\}$. Since there are d such numbers, some of them must repeat; i.e., for some integers r and s with $1 \leq r < s \leq d$,

$$2^r \equiv 2^s \pmod{d}.$$

Using Eq. (A1) with $a = 2^r$, $b = 2^s$, and $t = 2^{d-s}$, and using also Eq. (A2), we obtain

$$2^{d-s+r} \equiv 2^d \equiv \quad \pmod{d}.$$

Since m is the smallest positive integer for which Eq. (A3) holds, we obtain $m \leq d - s + r$, so finally $m < d$.

Let us now express d as $d = pm + r$, where $r = d \bmod m$. By multiplying by 2^m in Eq. (A3) and using Eq. (A1), we obtain

$$2^{2m} \equiv 2^m \equiv \quad \pmod{d}.$$

By multiplying again by 2^m and using Eq. (A3) and (A1), we have

$$2^{3m} \equiv 2^m \equiv \quad \pmod{d},$$

and by continuing similarly,

$$2^{pm} \equiv \quad \pmod{d}.$$

By multiplying by 2^r in this equation and again using Eq. (A1) together with Eq. (A2), we obtain

$$2^r \equiv 2^{pm+r} \equiv 2^d \equiv \quad \pmod{d}.$$

Since $r < m$, our hypothesis on m implies that $r = 0$. Thus, d is divisible by m. This implies that m is odd (since d is odd) and that $2^m \equiv 1 \pmod{m}$ (since Eq. (A3) holds). In view of the definition of d and the fact $d > m$, we must have $m = 1$, which contradicts Eq. (A3). Q.E.D.

REFERENCES

1. Bertsekas, D. P., and Tsitsiklis, J. N. *Parallel and Distributed Computation: Numerical Methods.* Prentice–Hall, Englewood Cliffs, NJ, 1989.
2. Bhatt, S. N., and Ipsen, I. C. F. How to embed trees in hypercubes. Res. Rep. YALEU/DCS/RR-443, Department of Computer Science, Yale University, 1985.
3. Dally, W. J., and Seitz, C. L. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Comput.* **C-36** (1987), 547–553.
4. Dekel, E., Nassimi, D., and Sahni, S. Parallel matrix and graph algorithms. *SIAM J. Comput.* **10** (1981), 657–673.
5. Greenberg, A. G., and Hajek, B. Deflection routing in hypercube networks. Unpublished report, 1989.
6. Hedetniemi, S. M., Hedetniemi, S. T., and Liestman, A. L. A survey of gossiping and broadcasting in communication networks. *Networks* **18** (1988), 319–349.
7. Johnsson, S. L., and Ho, C. T. Optimum broadcasting and personalized communication in hypercubes. *IEEE Trans. Comput.* **38** (1989), 1249–1268.
8. Johnsson, S. L. Cyclic reduction on a binary tree. *Comput. Phys. Comm.* **37** (1985), 195–203.
9. Johnsson, S. L. Communication efficient basic linear algebra computations on hypercube architectures. *J. Parallel Distrib. Comput.* **4** (1987), 133–172.
10. Krumme, D. W., Venkataraman, K. N., and Cybenko, G. The token exchange problem. Tech. Rep. 88-2, Tufts University, 1988.
11. Kermani, P., and Kleinrock, L. Virtual cut-through: A new computer communicating switching technique. *Comput. Networks* **3** (1979), 267–286.
12. McBryan, O. A., and Van de Velde, E. F. Hypercube algorithms and their implementations. *SIAM J. Sci. Stat. Comput.* **8** (1987), 227–287.
13. Minzer, S. E. Broadband ISDN and asynchronous transfer mode (ATM). *IEEE Comm. Mag.* (Sept. 1989), 17–57.
14. Ozveren, C. Communication aspects of parallel processing. Rep. LIDS-P-1721, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, 1987.
15. Saad, Y., and Schultz, M. H. Data communication in hypercubes. Res. Rep. YALEU/DCS/RR-428, Yale University, Oct. 1985 (revised Aug. 1987).
16. Saad Y., and Schultz, M. H. Data communication in parallel architectures. Yale University Report, Mar. 1986.
17. Saad Y., and Schultz, M. H. Parallel direct methods for solving banded linear systems. *Linear Algebra Appl.* **88/89** (1987), 623–650.
18. Saad Y., and Schultz, M. H. Topological properties of hypercubes. *IEEE Trans. Comput.* **37** (1988), 867–872.
19. Saad, Y. Communication complexity of the Gaussian elimination algorithm on multiprocessors. *Linear Algebra Appl.* **77** (1986), 315–340.
20. Stamoulis, G. D., and Tsitsiklis, J. N. Efficient routing schemes for multiple broadcasts in hypercubes. *Proc. 29th IEEE Conf. Decision and Control.* Honolulu, Hawaii, 1990, pp. 1349–1354.
21. Stout, Q. F., and Wagar, B. Passing messages in link-bound hypercubes. *Proc. 1986 Hypercube Conference.* SIAM, Philadelphia, 1987, pp. 251–257.
22. Topkis, D. M. Concurrent broadcast for information dissemination. *IEEE Trans. Software Engrg.* **13** (1983), 207–231.
23. Varvarigos, E. A., and Bertsekas, D. P. Communication algorithms for isotropic tasks in hypercubes and wraparound meshes. Rep. LIDS-P-1972, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, Mar. 1990; submitted for publication.
24. Varvarigos, E. A. Optimal communication algorithms for multiprocessor computers. M.S. thesis, Department of Electrical Engineering, MIT, Cambridge, MA; Rep. CICS-TH-192, Center for Intelligent Control Systems, MIT, Cambridge, MA, Jan. 1990.

DIMITRI P. BERTSEKAS was born in Athens, Greece, in 1942. He received a combined B.S.E.E. and B.S.M.E. from the National Technical University of Athens, Greece, in 1965, the M.S.E.E. degree from George Washington University in 1969, and the Ph.D. degree in system science from the Massachusetts Institute of Technology in 1971. Dr. Bertsekas has held faculty positions with the Engineering–Economic systems Department of Stanford University (1971–1974) and the Electrical Engineering Department of the University of Illinois, Urbana (1974–1979). He is currently a professor of electrical engineering and computer science at the Massachusetts Institute of Technology. He consults regularly with private industry and has held editorial positions in several journals. He was elected Fellow of the IEEE in 1983. Professor Bertsekas has done research in the areas of estimation and control of stochastic systems, linear, nonlinear, and dynamic programming, data communication networks, and parallel and distributed computation and has written numerous papers in each of these areas. He is the author of *Dynamic Programming and Stochastic Control,* Academic Press, 1976, *Constrained Optimization and Lagrange Multiplier Methods,* Academic Press, 1982, and *Dynamic Programming: Deterministic and Stochastic Models,* Prentice–Hall, 1987 and coauthor of *Stochastic Optimal Control: The Discrete-Time Case,* Academic Press, 1978, *Data Networks,* 1987, and *Parallel and Distributed Computation: Numerical Methods,* Prentice–Hall, 1989.

CÜNEYT ÖZVEREN was born in Istanbul, Turkey, on July 20, 1962. He received the B.S. and M.S. degrees in electrical engineering and computer science, the Electrical Engineer degree, the M.S. degree from the Sloan School of Management, and the Ph.D. degree in electrical engineering, all from the Massachusetts Institute of Technology, Cambridge, in 1984, 1987, 1987, 1989, and 1989, respectively. He is currently a senior engineer at Digital Equipment Corp., working on the design and the implementation of a high-speed communications switch. From January to August 1988 he conducted research at the Institut de Recherche en Informatique et Systèmes Aléatoires, France, and from September to December 1989 he was a postdoctoral research associate at the Laboratory for Information and Decision Systems at M.I.T. His interests are in the analysis and control of large-scale dynamic systems, including applications to communications systems, manufacturing systems, and economics. Dr. Özveren is a member of Sigma Chi, Tau Beta

Pi, Eta Kappa Nu, and IEEE. In 1989 he was a finalist for the 28th IEEE Conference on Decision and Control Best Student Paper Award. He is also the 1989 recipient of the Pugh-Roberts Associates Prize in Computer Simulation Applied to Corporate Strategy.

GEORGE D. STAMOULIS was born in Athens, Greece, in 1964. He received a diploma in electrical engineering (with highest honors) from the National Technical University of Athens, Greece, in 1987, and the M.S. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, Massachusetts, in 1988. He is currently completing his Ph.D. degree in electrical engineering at M.I.T. His research interests are in the areas of routing and performance evaluation of multiprocessing systems, communication networks, and queueing theory. Mr. Stamoulis was among the winners of the Greek Mathematic Olympiad in both 1981 and 1982. He also participated in the 23rd International Mathematic Olympiad, in Budapest, in July 1982. He is a member of the Technical Chamber of Greece and Sigma Xi.

PAUL Y. TSENG received the engineering degree in mathematics from Queen's University, Kingston, Canada, in 1981 and the Ph.D. degree in operations research from the Massachusetts Institute of Technology in 1986. From 1986 to 1987, he served on the faculty of the University of British Columbia, and from 1987 to 1990, he was a Postdoctoral Associate at M.I.T. He is currently an assistant professor in the Department of Mathematics, University of Washington, Seattle. His research interests are in optimization algorithms.

JOHN N. TSITSIKLIS was born in Thessaloniki, Greece, in 1958. He received the B.S. degree in mathematics (1980) and the B.S. (1980), M.S. (1981) and Ph.D. (1984) degrees in electrical engineering, all from the Massachusetts Institute of Technology, Cambridge, Massachusetts. During the academic year 1983-1984 he was an acting assistant professor of electrical engineering at Stanford University, Stanford, California. Since 1984 he has been with the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology, where he is currently associate professor. His research interests are in the areas of parallel and distributed computation, systems and control theory, and applied probability. Dr. Tsitsiklis is the coauthor, with Dimitri Bertsekas, of *Parallel and Distributed Computation: Numerical Methods* (1989). He has been a recipient of an IBM Faculty Development Award (1983), an NSF Presidential Young Investigator Award (1986), an Outstanding Paper Award from the IEEE Control Systems Society (for a paper coauthored with M. Athans, 1986), and the Edgerton Faculty Achievement Award from M.I.T. (1989). He is an Associate Editor of the *Applied Mathematical Letters* and the *IEEE Transactions on Automatic Control.*

Received February 21, 1989; revised August 23, 1990; accepted September 12, 1990