Efficient Routing Algorithms for Folded-Cube Networks

Emmanouel (Manos) Varvarigos†

University of California - Santa Barbara
Electrical and Computer Engineering Dept.
Santa Barbara, CA 93106
Email: manos@ece.ucsb.edu

Abstract

We consider the partial multinode broadcast, the total exchange, and several other prototype communication tasks in a folded-cube network of processors. In the partial multinode broadcast in a N-processor network, each one of M arbitrary nodes ($M \leq N$) broadcasts a packet to all the remaining $N-1$ nodes. In the total exchange task each processor sends a separate (personalized) packet to every other processor. We propose algorithms for the folded-cube topology that execute these tasks in optimal or near-optimal time. We also present an efficient scheme for the dynamic version of the broadcasting problem, where broadcast requests are generated at each node of the folded-cube at random times. The dynamic broadcasting scheme has asymptotically optimal stability and average delay properties.

1 Introduction.

There are some communication tasks that arise frequently in parallel computing applications. As a result, it is desirable to find algorithms that execute these prototype tasks in the minimum number of steps. Such algorithms can be called as communication primitives by the programmer or the compiler of a multiprocessor computer, in the same way that subroutines implementing standard functions are called from a library of functions in a conventional computer. The time required to execute the prototype tasks is a good performance measure in comparing topologies for multiprocessor networks (see [BeT89]).

Algorithms for routing messages between different processors have been studied by several authors under a variety of assumptions on the communication network connecting the processors. In this paper we focus on routing algorithms for prototype communication tasks in the folded-cube network topology. The simplest communication task to be considered is the *single node broadcast* (SNB for brevity) where the same packet is sent (copied) from a given node to all other nodes. A multinode version of the single node broadcast task is the *multinode broadcast* (or MNB), where each node broadcasts a packet to all the other nodes. A task more general than the MNB is the *partial multinode broadcast* (or PMNB) where an arbitrary subset of the nodes want to broadcast a packet to all the nodes. The PMNB task, along with being important on its own merit, it is also a critical component of the dynamic broadcasting schemes to be discussed later. A dual to the partial multinode broadcast problem is the *partial multinode accumulation* (PMNA), where every node of the network sends a packet to all nodes in a given set of nodes; here, we assume that packets with the same destination can be combined on a link, with the operator used in combining the packets being any associative operator. The PMNA implements a concurrent write from all nodes to each one of a given set of memory locations (a different number is written at each location). The PMNB and the PMNA are dual tasks to each other: origin (destination) nodes in the PMNB correspond to destination (origin) nodes in the PMNA, and the copying operation in the PMNB corresponds to the combining operation in a PMNA. The first to consider the PMNB and PMNA tasks were Stamoulis and Tsitsiklis [StT93b] for the hypercube topology, and Varvarigos and Bertsekas [VaB94] for d-dimensional meshes.

Another prototype communication task is the *single node scatter* (SNS), which involves sending a separate packet from a given node to every other node (see [BeT89]). A multinode version of the single node scatter is the *total exchange* (TE), where each node sends a separate (personalized) packet to every other node. A total exchange arises, for example, during the transposition of a $N \times N$ matrix stored by columns in

† Supported by NSF under Grant NSF-RIA-08930554.

0-7803-2492-7/95 $4.00 © 1995 IEEE

an N-processor computer. Johnsson and Ho [JoH89], Bertsekas et al [BOS91], Edelman [Ede91], Fraigniaud [Fra92], and Varvarigos and Bertsekas [VaB92] have developed minimum and nearly minimum completion time algorithms for the total exchange task in hypercubes and other topologies under a variety of communication models. Several other works deal with prototype communication tasks related to those of this research; for an overview of such problems see [HHL88], [SaS89], and [BeT89].

In this paper we present optimal and near-optimal algorithms to execute the PMNB, the MNB, the TE, and other communication tasks in a folded-cube network of processors. The algorithms presented, appropriately modified, can also be used to execute the duals of the previous tasks optimally or near-optimally, providing a rather complete set of communication primitives for the folded-cube topology.

We will first present an algorithm that executes the PMNB task in a folded-cube in near-optimal time, independently of the position of the nodes that have a packet to broadcast. The PMNB algorithm is on-line, distributed, and does not require prior knowledge of the position of the participating nodes. It is the first time that the PMNB task is considered for the folded-cube topology.

We also consider the TE communication task, and present an algorithm that executes this task in optimal time. The first to study the TE task for the folded-cube topology was C.-T. Ho in [Ho90], who also gave an algorithm of optimal order (but not optimal). The TE algorithm given in [Ho90] assumes that the messages can be split into small packets that are routed independently and are recombined at the destination. In the TE algorithm that we propose, the segmentation of messages is not allowed, and messages are always transmitted as one packet. This seems to be a more appropriate model for multiprocessor computers, where the overhead due to splitting and recombining of packets is usually significant. Our algorithm has a smaller execution time than the algorithm in [Ho90], even if the overhead associated with the splitting and the recombining of packets is ignored.

All the tasks described above are *static* in the sense that there is some work to be performed once and for all. The packets that each node has to send are available at time $t = 0$, and all the nodes are synchronized to start at the same time; the only objective is to finish the job as fast as possible. Except for the static communication tasks, where conditions are rather favorable, one can envision situations where communication requests are not deterministic, but they are generated at random time instants. We call such an environment *dynamic*. The execution of asynchronous computation algorithms is one such situation, but it is reasonable to expect that in many systems a dynamic, largely unpredictable environment may be the rule and not the exception.

In this paper we also consider the dynamic version of the broadcasting problem. In our model, broadcast requests are generated at random times at each node of a folded-cube. Based on the PMNB algorithm, we propose a dynamic decentralized scheme to execute the broadcasts in this stochastic environment. The dynamic scheme is provably stable for load asymptotically equal to the maximum possible, and its average delay is of the order of the diameter of the folded-cube for any load in the stability region.

The remainder of the paper is organized as follows. In Section 2 we describe the communication model used, and present some properties of the folded-cube topology. In Section 3 we focus on broadcasting tasks. We first present algorithms to execute the PMNB and the MNB communication tasks in near-optimal time. We then use the PMNB algorithm to get a routing scheme for the dynamic broadcasting problem. In Section 4 we give an optimal TE algorithm, and discuss the performance of greedy routing schemes. Finally, in Section 5 we conclude the paper.

2 Model and Properties.

The communication model that we will use is the following. A message is always transmitted as a single packet, and the splitting and recombining of packets is not allowed. The time required to cross any link is the same for all packets, and is taken to be one unit (or one slot). Packets can be simultaneously transmitted along a link in both directions, but only one packet can travel along a link in each direction at any one time. Finally, we assume that all incident links of a node can be used simultaneously for packet transmission and reception.

We will say that a communication algorithm is *near-optimal* if the potential loss of optimality with respect to completion time is of strictly smaller order of magnitude than the optimal completion time itself. We generally derive the optimal completion time by deriving a lower bound to the completion time of any algorithm and by constructing an algorithm that attains the lower bound;

this latter algorithm is said to be *optimal*. We will say that an algorithm is of *optimal order* if its worst case time complexity is asymptotically within a constant factor from the optimal value.

The d-dimensional folded-cube network, first defined in [AdS82], has $N = 2^d$ nodes and $(d+1)2^{d-1}$ links. Each node is represented by an *identity number* in the set $\{0, 1, \ldots, N-1\}$. A node can alternatively be represented by the (column) vector that corresponds to its binary representation. For example, the nodes 0,1, and $2^d - 1$ will also be referred to as nodes $(00\cdots00)^{\mathrm{T}}$, $(00\cdots01)^{\mathrm{T}}$, and $(11\cdots11)^{\mathrm{T}}$, respectively, where $^{\mathrm{T}}$ denotes the transpose operation. Each node has $d+1 = \log n + 1$ incident links. In particular, there are links between nodes whose representation vectors differ either in precisely one entry, or in all entries. We let e^i, $i = 0, 1, \ldots, d-1$, be the binary vector whose entries are equal to zero, except for the i^{th} entry, which is is equal to one. For completeness, we also let e^d the binary vector that has all entries equal to one. The *j-dimensional* link, $j = 0, 1, \ldots, d-1$, of node $s = (s_{d-1} \ldots s_j \ldots s_0)^{\mathrm{T}}$ is the link connecting that node to node $s \oplus e^j = (s_{d-1} \ldots \overline{s}_j \ldots s_0)^{\mathrm{T}}$, where $\overline{x}$ denotes the complement of x, and $\oplus$ denotes the componentwise exclusive OR operation between binary vectors. We also define the *d-dimensional* link of a node $s = (s_{d-1} \ldots s_j \ldots s_0)^{\mathrm{T}}$ as the link connecting node $(s_{d-1} \ldots s_j \ldots s_0)^{\mathrm{T}}$ with its componentwise complement $s \oplus e^d = (\overline{s}_{d-1} \ldots \overline{s}_j \ldots \overline{s}_0)^{\mathrm{T}}$. A d-dimensional link will also be referred to as a *complementary* link.

A folded-cube can be obtained from a hypercube by adding one complementary link per node. The following lemma shows that if we remove all links of a particular dimension from a d-dimensional folded-cube, we obtain a d-dimensional hypercube.

Lemma 1: If we remove from a folded-cube F all the links of a particular dimension j, $j \in \{0, 1, \ldots, d\}$, the resulting graph F_j is isomorphic to a d-dimensional hypercube.

Proof: It is clear that F_d is a hypercube (F_d contains all links of F except for the complementary links). Therefore, it is enough to show that F_j is isomorphic to F_d, for every j.

We represent the graph F_j, $j = 0, 1, \ldots, d$, by the pair (V_j, L_j), where V_j is the set of nodes and $L_j \subset V_j \times V_j$ is the set of links of F_j. We let E be the $d \times (d+1)$ binary matrix whose columns are the vectors e^i, $i = 0, 1, \ldots, d-1, d$:

$$E = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ e^0 & e^1 & \vdots & e^d \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}.$$

We also let C^0 be the $(d+1) \times d$ matrix whose first row has all entries equal to zero, and whose last d rows form an identity matrix:

$$C^0 = \begin{pmatrix} 0 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{pmatrix}.$$

We denote by C^j, $j = 0, 1, \ldots, d$, the matrix obtained by cyclically shifting each column of C^0 by j positions downwards. We define a function $\mathcal{V}_j : V_j \rightarrow V_d$ that maps each node s of F_j to the node

$$\mathcal{V}_j(s) = E \cdot C^j \cdot s$$

of F_d (all products involving binary matrices and/or vectors correspond to modulo 2 arithmetic). Since the matrix $E \cdot C^j$ has dimension $d \times d$ and linearly independent columns, $\mathcal{V}_j$ is a 1-1 function. We denote by $\hat{+}$ the modulo $d+1$ addition [that is, $i \hat{+} j = i + j \bmod (d+1)$], and by $\hat{-}$ the modulo $d+1$ substraction. The neighbor $s \oplus e^i$ of s in F_j, $i \neq j$ (recall that links of dimension j are not present in F_j), is mapped to the neighbor $\mathcal{V}_j(s \oplus e^i) = \mathcal{V}_j(s) \oplus E \cdot C^j \cdot e^i = \mathcal{V}_j(s) \oplus e^{d \hat{+} j \hat{-} i}$ of $\mathcal{V}_j(s)$ in F_d [to see that $\mathcal{V}_j(s) \oplus e^{d \hat{+} j \hat{-} i}$ is a neighbor of $\mathcal{V}_j(s)$, note that for $i \neq j$, we have $d \hat{+} j \hat{-} i \neq d$ and that only links of dimension d are missing from F_d]. Therefore, the function $\mathcal{L}_j : L_j \rightarrow L_d$ that maps link $(s, s \oplus e^i) \in L_j$ to link $(\mathcal{V}_j(s), \mathcal{V}_j(s) \oplus e^{d \hat{+} j \hat{-} i}) \in L_d$ is well defined. The pair $(\mathcal{V}_j, \mathcal{L}_j)$ represents an isomorphism between the graphs F_j and F_d. **Q.E.D.**

The isomorphism $(\mathcal{V}_j, \mathcal{L}_j)$ maps the links of dimension i of hypercube F_j to links of a *different* dimension $d \hat{+} j \hat{-} i$ of hypercube F_d. Since any d of the binary vectors $e^0, e^1, \ldots e^{d-1}, e^d$ form a linearly independent basis of the space $\{0,1\}^d$, a packet can be routed (not necessarily over a shortest path) from any node to any other node by using only d of the $d+1$ dimensions of the folded-cube. We define the *Hamming distance* $H(s, w)$ between nodes s and w as the number of bits in which their binary vectors differ. For any pair of nodes, there

is a connecting path with length equal to their Hamming distance, obtained, for example by the switching in sequence of bits in which the bit representations of the nodes differ (equivalently, by traversing the corresponding links of the folded-cube). There is also a path between node s and w with length $d+1-h(s,w)$ obtained by first traversing a link of dimension d, and then switching in sequence the bits at which $s \oplus e^d$ and w differ. Therefore, for any two nodes s and w there is a connecting path of length

$$P(s,w) = \min\left(h(s,w), d+1-h(s,w)\right).$$

It can be seen that any path between nodes s and w has length greater than or equal to $P(s,w)$. The diameter D of the network is equal to

$$D = \max_{s,w} P(s,w) = \left\lceil \frac{d}{2} \right\rceil.$$

The *mean internodal distance* MID of the folded-cube is defined as the average shortest distance between two nodes:

$$MID = \frac{1}{N^2} \sum_{s,w} P(s,w).$$

We will show in Section 4 that

$$MID = \frac{d+1}{2} - \frac{d+1}{2N} \binom{d}{\lceil d/2 \rceil}. \qquad (1)$$

3 Static and Dynamic Broadcasting Tasks.

A single node broadcast can be accomplished by transmitting the packet along a spanning tree rooted at the source. Since for every nodethere exists a spanning tree rooted at that node with depth equal to the diameter, the single node broadcast (or the single node accumulation) in a folded-cube requires $\lceil d/2 \rceil$ time units to execute. In this section we focus on two more complicated static broadcasting tasks, the PMNB and the MNB, and present algorithms to execute them in near-optimal time. We also consider the dynamic broadcasting problem, and present a dynamic scheme that has asymptotically optimal stability and average delay properties.

We start with the PMNB task, where M arbitrary nodes (called *active nodes*) of an N-processor folded-cube have to broadcast a packet to all the other nodes. Let T_{PMNB} be the optimal time required to execute a PMNB in a folded-cube. T_{PMNB} may actually depend on the position of the nodes that have a packet to broadcast. A lower bound, however, is always

$$T_{PMNB} \geq \max\left(\lceil d/2 \rceil, \left\lceil \frac{M-1}{d+1} \right\rceil \right), \qquad (2)$$

where $\lceil x \rceil$ denotes the smallest integer which is greater than or equal to x. Equation (2) can be seen by arguing that each node has to receive $M-1$ or M packets, and has only $d+1$ input ports. Futhermore, since packets are transmitted in a store-and-forward way (and splitting and recombining of packets is not allowed), the diameter of the network is a lower bound on the broadcast delay.

The PMNB algorithm that we propose is based on Lemma 1 of Section 2, and uses as a component the hypercube PMNB algorithm proposed by Varvarigos and Bertsekas (Algorithm $\mathcal{A}_0$ in [VaB95]). The idea is to simultaneously execute copies of algorithm $\mathcal{A}_0$ in each of the d-dimensional hypercubes $F_0, F_1, \ldots, F_{d-1}, F_d$ that are embedded in a folded-cube in the way described in Section 2. The algorithm $\mathcal{A}_0$ consists of a parallel prefix, a packing, and a broadcast phase. As proved in Lemma 2 of [VaB95], the algorithm $\mathcal{A}_0$ executes the PMNB in time

$$T_0 \leq M + 2dt_p + 2d - 1, \qquad (3)$$

where M is the number of active nodes, and t_p is the time required for a single parallel prefix step. At each step of a parallel prefix operation only one byte has to be transmitted between neighbor nodes. Therefore, $t_p < 1$, where one unit of time is the time required to transmit a packet over a link; in fact one expects $t_p << 1$ since many parallel computers have very efficient implementations of the parallel prefix operation.

As is evident from Eqs. (2) and (3), algorithm $\mathcal{A}_0$ is suboptimal for the folded-cube topology by a factor of roughly $d+1$. However, algorithm $\mathcal{A}_0$ has the useful property that it uses at each step only links of a particular dimension. We will use this property to design copies of the algorithm that can execute simultaneously on a folded-cube without interacting with each other. We let the algorithm $\mathcal{A}_0$ execute in the hypercube F_0 that is embedded in the folded-cube F; since F and F_0 have the same set of nodes, algorithm $\mathcal{A}_0$ executes a PMNB in F in time less than or equal to T_0, independently of the location of the active nodes. For any $c \in \{0, 1, \ldots, d-1, d\}$, we define another PMNB algorithm to be referred to as algorithm $\mathcal{A}_c$. Algorithm $\mathcal{A}_c$ is similar to algorithm $\mathcal{A}_0$, but it executes in hypercube

F_c. In particular, $\mathcal{A}_c$ is obtained from $\mathcal{A}_0$ by renaming the nodes and the dimensions in the following way: node s of F_c is mapped to node $\mathcal{V}_c(\mathcal{V}_0^{-1}(s))$ of F_0, where the functions $\mathcal{V}_j$, $j = 0, 1, \ldots, d-1, d$ are defined as in the proof of Lemma 1 ($\mathcal{V}_c$ is a 1-1 mapping of the nodes of F_c to the nodes of F_d, and $\mathcal{V}_0^{-1}$ is a 1-1 mapping of the nodes of F_d to the nodes of F_0), and a link of dimension i in F_c ($i \neq c$), is mapped to a link of dimension $i \hat{-} c$. Since algorithm $\mathcal{A}_0$ performs the PMNB in F independently of the location of the active nodes, algorithm $\mathcal{A}_c$ also executes the PMNB task in F, and in the same amount of time. For $i \neq j$, algorithms $\mathcal{A}_i$ and $\mathcal{A}_j$ use links of different dimensions at each step, and they can be executed independently of each other.

We are now in a position to describe the PMNB algorithm for folded-cubes. Let $s_1, s_2, \ldots, s_M$, $M \leq N$, be the active nodes. We define the *rank* r_s of an active node s as

$$r_s = \sum_{t<s} x_t - 1,$$

where x_t is equal to one if processor t has a packet to broadcast, and zero otherwise. The PMNB algorithm consists of two parts.

Class Computation Part:

The rank r_s, $0 \leq r_s \leq M-1$, $s \in \{s_1, s_2, \ldots, s_M\}$, of each active node is computed through a parallel prefix operation on hypercube F_d. This can be done in $2d$ steps by performing a *parallel prefix operation* (see [Lei92]) on a tree P, called *parallel prefix tree*, embedded in the hypercube F_d. The i^{th} leaf of the tree from the left is the i^{th} node of the folded-cube. The operation is described in [Lei92], pp. 37-44. The class computation part requires $2dt_p$ time units, where t_p is the time for a single parallel prefix step. The packet of node s is assigned a *class number* $c = r_s \bmod (d+1)$.

Main Part:

The packets of class c are routed according to algorithm $\mathcal{A}_c$. It can be seen that each class has at most $\lceil M/(d+1) \rceil$ packets. Using Eq. (3) with $\lceil M/(d+1) \rceil$ instead of M, we conclude that the main part requires time less than $\frac{M}{d+1} + 2d + 2dt_p - 1$.

Adding the durations of the two parts, we find that the total duration T_{PMNB} of the algorithm satisfies

$$T_{PMNB} \leq \left\lceil \frac{M}{d+1} \right\rceil + 2d + 4dt_p - 1, \qquad (4)$$

independently of the value of M and the location of the active nodes. Comparing Eq. (4) with the lower bound of Eq. (2), we see that the terms that depend on M at the right hand sides have the same coefficient; therefore the algorithm is near-optimal.

The PMNB algorithm presented above gives rise to an efficient algorithm for the MNB task. Indeed, a MNB can be treated as a PMNB with $M = N$. The parallel prefix operations of the class computation part and of the main part (recall that the algorithms $\mathcal{A}_c$ used in the main part also include a parallel prefix operation) are not necessary any more, since the information obtained in them can be considered as known in advance. Therefore, the MNB can be executed in a folded-cube in time

$$T_{MNB} \leq \left\lceil \frac{N}{d+1} \right\rceil + 2d - 1,$$

which is optimal within $2d-1$ time units.

In the preceding MNB algorithm, messages are always transmitted as one packet. C.-T. Ho [Ho90] has proposed an alternative MNB algorithm for the folded cube topology, which has optimal completion time $(N-1)/(d+1)$, and assumes that messages can be split into smaller packets that can be routed independently and can be recombined at the destination without any overhead. The splitting of messages may be undesirable when the messages are small (as they usually are), in which case the overhead introduced is significant. For large messages, where the overhead associated with the splitting of messages is proportionally small, Ho's MNB algorithm may perform better.

In the MNB task each node receives $N-1$ different packets, one from every other node of the network. Therefore, a MNB algorithm can also be used to execute the single node gather task, and by reversing the transmissions, it can also execute its dual task, the single node scatter. It is easy to see that a lower bound on the time required to execute a SNS or a SNG is $(N-1)/(d+1)$. The MNB algorithm described above, appropriately modified, gives rise to near-optimal algorithms for the SNS and the SNG tasks. Note that MNB algorithms that require the segmentation of packets cannot be used to execute the SNG task. This is because in the SNG, whenever two packets are combined into one packet using some associative operator, the whole information contained in each of the packets is needed.

The PMNB algorithm presented above also gives rise to an efficient scheme for the dynamic broadcasting problem. In the dynamic broadcasting problem packets

arrive at each node of a folded-cube, according to a random process with rate λ, over an infinite time horizon, and each of them has to be broadcast to all other nodes. The dynamic broadcasting problem has been studied in [StT93a] and [VaB94], [VaB94] for hypercubes and d-dimensional meshes, respectively. It has been proven (see [VaB95]) that if we have an efficient algorithm to execute a PMNB in a given network, we can also obtain an efficient dynamic broadcasting scheme for that network. In particular, the dynamic broadcasting scheme can be taken to consist of the repeated execution of PMNB algorithms, each starting when the previous one has finished. The following is a corrolary to the Dynamic Broadcasting Theorem of [VaB95].

Corrolary 1: Let the arrival process of broadcast requests at a node of an N-processor folded-cube be Poisson with rate λ, and the arrival processes at different nodes be independent. We call

$$\rho = \frac{\lambda N}{d+1}$$

the *utilization factor* of the folded-cube. Then there exists a dynamic broadcasting scheme (namely the one that consists of the repeated execution of the PMNB algorithm), that is stable for

$$\rho < \frac{1}{1+\frac{(d+1)(4dt_p+2d)}{2^d}}. \tag{5}$$

and has average delay T that satisfies

$$T \leq (1+\rho)\cdot\frac{\frac{\rho}{d+1}+(3-\rho)(2d+4dt_p)}{2\left(1-\rho-\rho\frac{(d+1)(2d+4dt_p)}{2^d}\right)}+\frac{1}{d+1}. \tag{6}$$

By considering the stability conditions for a particular node, it is easy to prove that the maximum value of ρ that can be accomodated by *any* broadcasting scheme in a folded-cube is equal to one. Since the right hand side of Eq. (5) tends to one as the dimension d tends to infinity, our dynamic scheme is stable for load asymptotically equal to the maximum possible. Also, for any fixed ρ in the stability region, we have from Eq. (6) that $T = O(d)$. Therefore, for any fixed load the average delay of the broadcasting scheme is of the order of the diameter of the folded-cube; this is the best we can hope for since the diameter is a lower bound for any broadcasting task.

4 Total Exchange in Folded-Cubes.

C.-T. Ho [Ho90] has given an algorithm of optimal order to execute the TE task in a folded-cube network of processors in time

$$T_{TE} \leq \frac{dN}{2(d+1)}. \tag{8}$$

The same author has also shown that the completion time of any TE algorithm in a folded-cube network satisfies

$$T_{TE} \geq 2^{d-1} - \frac{1}{2}\binom{d}{\lceil d/2 \rceil}. \tag{7}$$

The algorithm given in [Ho90] requires the splitting of each packet into $d+1$ parts that are routed independently and are recombined at the destination. Although the lower and upper bounds of Eqs. (7) and (8) are of the same order of magnitude, the gap between them is quite significant in practice. For example, in the case $d = 16$ the lower bound is 26333 time units, while the algorithm in [Ho90] requires 30840 time units (for comparison, the TE task in a 16-dimensional hypercube requires 32768 time units).

In this section we present an alternative algorithm to execute the TE task in the folded-cube topology. Our algorithm has completion time equal to the lower bound of Eq. (7), and is therefore optimal. During the execution of the TE algorithm packets travel over shortest paths, while at the same time 100% utilization of the links is achieved. The algorithm does not require the splitting and the recombining of packets, which eliminates the associated overhead, and makes the algorithm easy to implement.

We first consider the case where d is even. In the algorithm that we propose, the packets carry with them a $d+1$-bit binary vector called *routing tag*. A packet originated at node s and destined for a node t at Hamming distance less than or equal to $d/2$, will not use any d-dimensional links, and its routing tag is defined as

$$(0, s_{d-1} \oplus t_{d-1}, s_{d-2} \oplus t_{d-2}, \ldots, s_0 \oplus t_0),$$

where $\oplus$ denotes the exclusive OR operation. A packet originated at node s and destined for a node t at Hamming distance greater than or equal to $d/2+1$ will use exactly one d-dimensional link, and its routing tag is defined as

$$(1, \overline{s_{d-1} \oplus t_{d-1}}, \overline{s_{d-2} \oplus t_{d-2}}, \ldots, \overline{s_0 \oplus t_0}),$$

where $\overline{x} = 1 - x$ denotes the complement of x.

An important data structure that will be useful in describing the TE algorithm is that of the *task matrix*. The task matrix $\mathcal{T}(s)$ of node s is a binary matrix whose rows are the routing tags of all the packets with origin s. The routing tags appear as rows of the task matrices $\mathcal{T}(s)$ in some arbitrarily chosen order. The task matrices that correspond to the TE task are, by symmetry, identical for all nodes s, and will be denoted by $\mathcal{T}_0$. Note that $\mathcal{T}_0$ has $d+1$ columns (each corresponding to a link of a node) and $N-1$ rows (each corresponding to a packet involved in the task). We define the *critical sum* h of a matrix as the maximum of the column sums and the row sums of the matrix. Then the following lemma holds.

Lemma 2: There exists a distributed (based only on local information) routing algorithm that executes the TE task in time equal to the critical sum of the task matrix.

Proof: **(outline)** A permutation matrix is a matrix with entries equal to 0 or 1 with the property that each row or column of the matrix has at most one nonzero entry. Using Hall's Theorem (see [Rys65]) we can write a task matrix $\mathcal{T}_0$ as the sum $\sum_{k=1}^{h} S_k$ of permutation matrices $S_1, S_2, \ldots, S_h$. If $\mathcal{T}_k$ is the task matrix before the k^{th} slot, and $S_k \leq \mathcal{T}_k$, is a permutation matrix, then S_k can be viewed as specifying the packet (if any) that will be transmitted on each link at the time slot beginning at time k. In particular, a unit at entry (i,j) of S_k corresponds to the packet with routing tag equal to the i^{th} row of $\mathcal{T}_k$ being transmitted on the link of dimension j during time slot k. If during any time slot t all the nodes use the same permutation matrix S_k for their switching assignments, then the switching scheme is called symmetric. The key fact is that if at some time t, the task matrices $\mathcal{T}_t(s)$ are the same for all nodes s, and a symmetric switching scheme is used, then the next task matrices $\mathcal{T}_{t+1}(s)$ will be the same for all nodes. In particular, the task matrix at times t with $1 \leq t < h$ will be equal to $\mathcal{T}_t = \mathcal{T}_0 - \sum_{k=1}^{t} S_k$. At time $t = h$ the task matrix will have all entries equal to zero; this corresponds to the TE task having been completed. Hence the TE communication task can be completed after h steps. **Q.E.D.**

The critical sum of the task matrix is equal to

$$\max(\tau_c, \tau_e),$$

where

$$\tau_c = \sum_{i=d/2+1}^{d} \binom{d}{i} = \frac{N}{2} - \frac{1}{2}\binom{d}{d/2}$$

is the column sum that corresponds to dimension d, and

$$\tau_e = \frac{1}{d}\sum_{i=0}^{d/2} \binom{d}{i} i + \frac{1}{d}\sum_{i=0}^{d/2-1} \binom{d}{i} i = \sum_{i=1}^{d/2} \binom{d-1}{i-1} + \sum_{i=1}^{d/2-1} \binom{d-1}{i-1} = \sum_{j=0}^{d/2-1} \binom{d-1}{j} + \sum_{j=0}^{d/2-2} \binom{d-1}{j} = \frac{N}{4} + \frac{N}{4} - \binom{d-1}{d/2-1} = 2^{d-1} - \frac{1}{2}\binom{d}{\lceil d/2 \rceil},$$

is the column sum that corresponds to each of the dimensions $0, 1, \ldots, d-1$. Using Lemma 2 we conclude that for d even the TE task can be executed in time

$$T_{TE} = \tau_c = \tau_e = 2^{d-1} - \frac{1}{2}\binom{d}{\lceil d/2 \rceil},$$

which equals the lower bound.

We next consider the case where d is odd. A packet with origin node s and destination node t at Hamming distance less than or equal to $\lceil d/2 \rceil - 1$ will not use any d-dimensional links, and its routing tag is

$$(0, s_{d-1} \oplus t_{d-1}, s_{d-2} \oplus t_{d-2}, \ldots, s_0 \oplus t_0).$$

A packet with origin node s and destination node t at Hamming distance greater than or equal to $\lceil d/2 \rceil + 1$ will use a d-dimensional link and has routing tag equal to

$$(1, \overline{s_{d-1} \oplus t_{d-1}}, \overline{s_{d-2} \oplus t_{d-2}}, \ldots, \overline{s_0 \oplus t_0}).$$

The packets of s whose destinations are at Hamming distance $\lceil d/2 \rceil$ from s are partitioned into two disjoint sets of equal cardinality, in the same way for all origins s; packets in the one set will use d-dimensional links and packets in the other set will not. The task matrices will be the same for all nodes and have critical sum equal to

$$\max(\tau_c, \tau_e),$$

where

$$\tau_c = \sum_{i=\lceil d/2 \rceil+1}^{d} \binom{d}{i} + \frac{1}{2}\binom{d}{\lceil d/2 \rceil} = \frac{N}{2} - \frac{1}{2}\binom{d}{\lceil d/2 \rceil}$$

is the column sum corresponding to d-dimensional links, and

$$\tau_e = \frac{2}{d}\sum_{i=0}^{\lceil d/2\rceil-1}\binom{d}{i}i + \frac{1}{2d}\binom{d}{\lceil d/2\rceil} = 2\sum_{i=1}^{\lceil d/2\rceil-1}\binom{d-1}{i-1}$$
$$+\frac{1}{2d}\binom{d}{\lceil d/2\rceil} = 2\sum_{j=1}^{\frac{d-1}{2}-1}\binom{d-1}{j} + \frac{1}{2d}\binom{d}{\lceil d/2\rceil}$$
$$= \frac{N}{2} - \binom{d-1}{\lceil d/2\rceil - 1} + \frac{1}{2d}\binom{d}{\lceil d/2\rceil}$$
$$= 2^{d-1} - \frac{1}{2}\binom{d}{\lceil d/2\rceil},$$

is the column sum that corresponds to each of the other dimensions. Thus, for d odd, the TE task can be executed in time

$$T_{TE} = \tau_c = \tau_e = 2^{d-1} - \frac{1}{2}\binom{d}{\lceil d/2\rceil},$$

which again equals the lower bound.

Note that in the algorithm we propose all packets follow shortest paths to their destination and 100% utilization of the links is achieved. Let MID be the mean internodal distance of the folded-cube. Since there are $N(d+1)$ unidirectional links, and N^2 source-destination pairs (we permit packets whose source is the same with their destination), we have $N(d+1)T_{TE} = N^2(MID)$; this proves Eq. (1) of Section 2.

We define a "greedy routing scheme" as a scheme where all nodes take symmetric routing decisions, and no link is left idle at any step if there is a packet that is not transmitted at that step and wants to use it. It can be shown that any greedy scheme executes the TE in a folded-cube in time less than or equal to

$$2^{d-1} - \frac{1}{2}\binom{d}{\lceil d/2\rceil} + d - 1.$$

This is only $d-1$ steps (at most) more than the optimal execution time.

Basic Communication Tasks for Folded-Cubes

Tasks	Completion times	Lower bounds
SNB/SNA	$\lceil d/2\rceil$	$\lceil d/2\rceil$
SNS/SNG	$\lceil N/(d+1)\rceil + 2d - 1$	$\lceil (N-1)/(d+1)\rceil$
MNB	$\lceil N/(d+1)\rceil + 2d - 1$	$\lceil (N-1)/(d+1)\rceil$
PMNB/PMNA	$\lceil N/(d+1)\rceil + 2d + 4dt_p - 1$	$\max(\lceil (N-1)/(d+1)\rceil, \lceil d/2\rceil)$
TE	$2^{d-1} - \frac{1}{2}\binom{d}{\lceil d/2\rceil}$	$2^{d-1} - \frac{1}{2}\binom{d}{\lceil d/2\rceil}$

Figure 1: The table summarizes the basic static communication tasks in a folded-cube, the corresponding completion times of the algorithms that we propose. We remind the reader that the communication model used does not allow the splitting and the recombining of packets.

5 Conclusions.

We have proposed optimal and near-optimal algorithms to execute several basic communication tasks in folded-cube networks. The tasks considered are the single node broadcast, the single node accumulation, the partial multinode broadcast, the partial multinode accumulation, the single node scatter, the single node gather, and the total exchange. We also considered the dynamic broadcasting problem in folded-cubes networks, and proposed a dynamic scheme that has stability and average delay properties that are asymptotically optimal.

6 References.

[AdS82] Adams, G. B., and Siegel, H. G., "The Extra Stage Cube: a Fault-Tolerant Interconnection Network for Supersystems," IEEE Trans. Computers, 31(5), pp. 443-454, May 1982.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

[BOS91] Bertsekas, D. P., Ozveren, C., Stamoulis, G. D., Tseng, P., and Tsitsiklis, J. N., "Optimal Communication Algorithms for Hypercubes," J. Parallel Distrib. Comput., Vol. 11, pp. 263-275, 1991.

[Ede91] Edelman, A., "Optimal Matrix Transposition and Bit Reversal on Hypercubes: All-to-All Personaliced Communication," J. Parallel Distrib. Comput., Vol. 11, pp. 328-331, 1991.

[Fra92] Fraigniaud, P., "Complexity Analysis of Broadcasting in Hypercubes with restricted Communication Capabilities", J. Parallel Distrib. Comput., Vol. 16, pp. 15-26, 1992.

[HHL88] Hedetniemi, S. M., Hedetniemi, S. T., and Liestman, A. L., "A Survey of Gossiping and Broadcasting in Communication Networks", Networks, Vol. 18, pp. 319-349, 1988.

[Ho90] Ho, C. T., "Full Bandwidth Communications on Folded Hypercubes," Research Report RJ 7434 (69605), IBM Almaden Research Center, April 1990.

[JoH89] Johnsson, S. L., and Ho, C. T., "Optimum Broadcasting and Personalized Communication in Hypercubes," IEEE Trans. Computers, Vol. C-38, pp. 1249-1268, 1989.

[Lei92] Leighton, F. T., *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*, Morgan Kaufmann, San Mateo, CA, 1992.

[Rys65] Ryser, H. J., "Combinatorial Mathematics", The Mathematical Association of America, Rahway, N.J., 1965.

[SaS89] Saad Y., and Schultz, M. H., "Data Communication in Parallel Architectures," Parallel Computing, Vol. 11, pp. 131-150, 1989.

[StT93a] Stamoulis, G. D., and Tsitsiklis, J. N., "Efficient Routing Schemes for Multiple Broadcasts in Hypercubes," IEEE Trans. on Parallel and Distributed Systems, Vol. 4, 1993, pp. 725-739.

[StT93b] Stamoulis, G. D., and Tsitsiklis, J. N., "An Efficient Algorithm for Multiple Simultaneous Broadcasts in the Hypercube," Information Processing Letters, July 1993, pp. 219-224.

[VaB92] Varvarigos, E. A., and Bertsekas, D. P., "Communication Algorithms for Isotropic Tasks in Hypercubes and Wraparound Meshes," Parallel Computing, Vol. 18, pp. 1233-1257, 1992.

[VaB94] Varvarigos, E. A., and Bertsekas, D. P., "Partial Multinode Broadcast and Partial Exchange in D-Dimensional Meshes," J. of Parallel and Distributed Computing, Vol. 23, pp. 177-189, 1994.

[VaB95] Varvarigos, E. A., and Bertsekas, D. P., "Dynamic Broadcasting in Parallel Computing," to appear IEEE Trans. on Parallel and Distributed Systems, Feb. 1995.