Yale University
Department of Computer Science

Spanning Graphs for Optimum Broadcasting and Personalized Communication in Hypercubes

S. Lennart Johnsson and Ching-Tien Ho

YALEU/DCS/TR-500
November 1986

A short version of this paper was awarded the "Outstanding Paper Award" at the 1986 International Conference on Parallel Processing. This work was supported by the Office of Naval Research under Contract No. N00014-84-K-0043. Approved for public release: distribution is unlimited.

Table of Contents

Spanning Graphs for Optimum Broadcasting and Personalized Communication in Hypercubes

S. Lennart Johnsson and Ching-Tien Ho

Departments of Computer Science and
Electrical Engineering
Yale University
New Haven, CT 06520

Abstract

High communication bandwidth in standard technologies is more expensive to realize than a high rate of arithmetic or logic operations. The effective utilization of communication resources is crucial for good overall performance in highly concurrent systems. In this paper we address four different communication problems in Boolean n-cube configured multiprocessors: 1) broadcasting, i.e., distribution of common data from a single source to all other nodes, 2) sending personalized data from a single source to all other nodes, 3) distribution of common data from all nodes to all other nodes, 4) sending personalized data from all nodes to all other nodes. Routing according to the well-known spanning tree obtained by bit-wise complementation of leading zeroes (referred to as SBT routing for *Spanning Binomial Tree*) is compared with routing based on multiple spanning binomial trees obtained through rotation, nRSBT, n *Rotated Spanning Binomial Trees*, and rotation and translation, nESBT, n *Edge-disjoint Spanning Binomial Trees*. The nESBT routing with appropriate scheduling offers a speed-up over the SBT routing by a factor of up to n for broadcasting. The nESBT algorithm fully utilizes the bandwidth of the Boolean cube. It can also be used to accomplish graceful degradation under faulty conditions.

For personalized communication and all-to-all broadcasting, i.e., cases 2, 3 and 4, routing according to a *Spanning Balanced n-Tree* (SBnT) offers a lower complexity than SBT routing. The potential improvement is by a factor of n. Our analysis takes into account the size of the data sets, the communication bandwidth, and the overhead in communication. We also provide some experimental data for the Intel iPSC/d7.

1. Introduction

In this paper we investigate two generic communication problems in Boolean n-cube configured ensemble architectures. The first problem is ***broadcasting***, i.e., the distribution of the same data set from one node to all other nodes, or a subset thereof. The second problem is that of ***personalized communication*** in which case a node sends a unique data set to all other nodes, or a subset thereof. This is the familiar situation in which each recipient receives a personalized message from the same agency. We consider broadcasting from a single source to all other nodes, ***one-to-all broadcasting***, as well as concurrent broadcasting from all nodes to all other nodes, or ***all-to-all broadcasting*** in the case all nodes are performing one-to-all broadcasting. Broadcasting, in the sense of one node distributing the same data set to all nodes in a cube, is used in a variety of linear algebra algorithms [11, 20, 19], such as matrix-vector multiplication, matrix-matrix multiplication, LU-factorization, and Householder transformations. It is also used in data base queries and transitive

closure algorithms [4]. For the broadcasting operation a spanning tree can be used for the routing. Data is replicated according to the fan-out of the nodes in the tree as it passes from the root towards the leaves. The reverse of the broadcasting operation is reduction, in which the data set is reduced according to the fan-in as data is moving from the leaves towards the root. The reduction operation may be addition/subtraction as used in the computation of global sums for inner products or histogramming, or a max/min operator as used in searching/sorting, or a composition of reduction and other operators as in the solution of tridiagonal systems by cyclic reduction [5], recursive doubling [22], and parallel prefix computations [23].

In personalized communication a node delivers (or collects) personalized information to (from) all other nodes, or a subset thereof. We restrict ourselves to two cases: *one-to-all personalized communication* and *all-to-all personalized communication.* The fundamental difference between broadcasting and personalized communication is that in the latter no replication/reduction of data takes place. The bandwidth requirement is highest at the root of the spanning tree and is reduced monotonically towards the leaves. Personalized communication is used, for instance, in transposing a matrix, and the conversion between different data structures [20, 16]. Matrix transposition is useful in the solution of tridiagonal systems on Boolean cubes for certain combinations of machine characteristics [21], and for matrix-vector multiplication.

The routing problem investigated here is that of routing according to some form of a *spanning graph* SG for a graph $G = (V, E)$, where V is the node set and E is the set of edges. A spanning graph is a connected, directed, graph with node set V and edge set being a subset of the edge set E; $SG = (V, SE), \quad SE \subseteq E$. One class of spanning graphs is that of *spanning trees.* A Boolean cube has many spanning trees. A Hamiltonian path is one example. In real architectures a time is associated with each edge traversed, as well as each operation performed in the nodes of the paths. In order to minimize the maximum time for a data element to reach any destination it is of interest to minimize the maximum path length, i.e., the *diameter*, or *height*, of the spanning graph. Minimizing the height minimizes the *propagation time*, the time for the first data element to reach the furthest destination(s). The data unit may be atomic (a bit, byte, word, record), or a collection of atomic units.

In broadcasting several atomic units minimizing the time to completion may be a more interesting criterion of effectiveness than minimizing the propagation time. A spanning tree only uses $N-1$ of the nN directed edges of an n-cube. The bandwidth of the Boolean cube is poorly used. In a spanning graph the bandwidth of the Boolean cube may be used more effectively by incorporating additional edges in the routing algorithm. The spanning graphs we consider are either spanning trees, or compositions of n spanning trees. The composition is made through rotation, or rotation and translation, and a direct sum. In the composition, each tree is given an equal weight $\frac{1}{n}$. Hence, the weight of all nodes is 1, but the weight of the edges falls in the range $[\frac{1}{n}, 1]$. The smallest value applies for graph edges that are used only in one of the trees used in the composition. A graph edge occurring in every spanning tree, if there is such an edge, has the weight 1.

In a spanning graph there is a set of outgoing edges associated with every incoming edge. The sets are not necessarily disjoint. The task of the *routing* algorithm is to retransmit the appropriate parts of the incoming messages on the outgoing edges, or *ports.* In broadcasting using a spanning tree the incoming message is retransmitted on every outgoing port in the set of ports associated with the input port. Broadcasting using other spanning graphs may use message combination such that

a message needs to be disassembled upon receipt, and different pieces retransmitted on different ports, potentially combined with messages from other input ports. In personalized communication the incoming message is broken up into pieces, one for each port in the output set associated with the input port, and one for the node itself. In a distributed computing environment it is clearly of interest that the routing algorithm be distributed. All routing algorithms discussed in this paper only use local information, and are synchronized by message arrivals.

The routing algorithm establishes the paths of the messages. The *scheduling* algorithm determines the order in which messages are sent on different ports as well as the message order for each port. The goal of the scheduling algorithm is to avoid unnecessary delays and allow messages to propagate at the maximum speed. We investigate two extreme cases with respect to communication constraints: communication on one-port-at-a-time for every node, ***one-port*** communication, and concurrent communication on all n ports of every node, ***n-port*** communication. In broadcasting, the data order for a given port is irrelevant, but in personalized communication it effects the communication complexity. We investigate two orderings: *postorder* starting with the largest subtree recursively, and *reverse-breadth-first* ordering. The first ordering is of interest in ***one-port*** communication, the second for ***n-port*** communication. In the *reverse-breadth-first* ordering data for the most remote nodes are ordered first and data for the nodes adjacent to the source node last. With the restriction of *one-port* communication we interleave the communication on the different ports such that the minimum time between successive communications on the same port is maximized. For the spanning binomial tree this port scheduling results in communications on ports in the same order as the order in which dimensions are encountered in the ***binary-reflected*** Gray code.

For all-to-all communication the interleaving of the spanning graphs rooted at different nodes is of interest. In particular, it is important to find the maximum number of elements that traverse any cube edge during any routing cycle (assuming a synchronous algorithm). For all-to-all broadcasting it is sufficient to know the number of edges of the spanning graph in a given dimension and at a given distance from the source in order to determine the number of elements contending for an edge during a routing cycle. In personalized communication and *postorder* scheduling it suffices to know, for every level of the spanning graph, the sizes of the subtrees connected through a given dimension to the nodes at a preceding level. In *reverse-breadth-first* scheduling the number of nodes at successively decreasing distances in subtrees needs to be considered.

In real architectures communications overhead and buffer sizes are determined as a trade-off between implementation cost and performance. The communications overhead is a fixed delay, start-up τ, for each communication. Such a delay often gives rise to an optimization problem in which delay is traded for increased utilization. This point is easily illustrated for broadcasting M elements along a path of length N. The minimum number of start-ups is clearly N, and the total time $N(Mt_c + \tau)$, where t_c is the transmission time for an element. By dividing the set M into packets of size B and pipelining the communications the total time becomes $(\lceil \frac{M}{B} \rceil + N - 1)\tau + (M + (N-1)B)t_c$, which has a minimum of $T_{min} = (\sqrt{Mt_c} + \sqrt{\tau(N-1)})^2$ for $B_{opt} = \sqrt{\frac{M\tau}{(N-1)t_c}}$.

Pipelining, as described above for broadcasting, is part of the scheduling discipline. It is also applicable to personalized communication. In the *reverse-breadth-first* scheduling, successive levels are pipelined. In the *postorder* scheduling, pipelining is also used, in that several packets may be progressing away from the source in the same subtree, but successive communications are typically

made on different ports. In broadcasting pipelining implies dividing the set M into pieces. In personalized communication it never pays to subdivide M, but it may be beneficial to divide the data set for a given port into several messages. For instance, for personalized communication based on a Hamiltonian path, the communication time is minimized by communicating data for each node separately, but in the case of a spanning binomial tree and *one-port* communication the optimum is an entire subtree at a time, as we will show later. With *n-port* communication and *reverse-breadth-first* scheduling it is never beneficial with respect to communication time to divide the data for a given level into more than one packet.

For the all-to-all communication, pipelining is not an issue for a lower bound algorithm. With *one-port* communication all the links in a dimension are evenly used during each routing cycle, and with *n-port* communication all links in any dimension are evenly used. The data transfer times are already optimal in both cases. Pipelining can only increase the number of start-ups.

Data communication in Boolean cubes has received significant interest recently due to the success of the Caltech Cosmic Cube project [29] and commercially available Boolean cube configured concurrent processors (from Intel, NCUBE[13], and Ametek, and cube-like architectures from Floating-Point Systems[12] and Thinking Machines Corp. [14]). The embedding of complete binary trees is treated in [31, 20, 27, 7, 3]. Wu also discusses the embedding of k-ary trees. Embedding of arbitrary binary trees is discussed in [3] and improved in [2]. Efficient routing using randomization for arbitrary permutations has been suggested by Valiant [30]. Broadcasting of data from a single source to all other nodes, and from all nodes to all other nodes is also studied in [9, 28]. For one-to-all broadcasting we propose an algorithm that offers a speed-up of a factor of $\log_2 N$ over the algorithm in [9, 28]. The communication complexity of our algorithm is the same as the lower bound. We also present lower bound algorithms for personalized communication, lower bound algorithms for all-to-all broadcasting similar to the one in [28], and lower bound algorithms for all-to-all personalized communication. We consider both communication restricted to a single port as in [9] and concurrent communication on multiple ports. We give both routing and scheduling algorithms, and analyze the complexity in detail. The analysis is compared with experimental data. The one-to-all communication algorithms are discussed briefly in [15].

In this paper we focus on spanning graphs in the form of *Hamiltonian paths* (HP), *two-rooted complete binary trees* (TCBT), *spanning binomial trees* (SBT), and *spanning balanced n-trees* (SBnT), and combinations thereof in the form of n *rotated SBTs*, nRSBT, and n *edge-disjoint SBTs*, nESBT. A balanced n-tree has a fan-out of n at the root, and each of the subtrees of the root has approximately the same number of nodes. Hence, a complete binary tree is a balanced 2-tree. We first state some of the essential characteristics of the Boolean cube, then describe some of the topological properties of each of the spanning graphs as well as distributed routing algorithms for generating the graphs. Then we consider one-to-all and all-to-all communication. For each we consider broadcasting and personalized communication, and present routing algorithms and derive optimum scheduling strategies and packet sizes.

2. Notations and Definitions.

A Boolean n-cube has $N = 2^n$ nodes, diameter n, $\binom{n}{i}$ nodes at distance i from a given node, and n disjoint paths between any pair of nodes. The paths are either of the same length as the Hamming distance between the end points of the paths, or the Hamming distance plus two [27].

The fan-out/fan-in of every node is n, and the total number of communication links is $\frac{1}{2}nN$, or nN if the links are bidirectional (concurrent communication in both directions).

In the following, N denotes the number of nodes in the Boolean cube and $n = \log_2 N$ the dimension of the cube. Nodes in the cube are assigned binary addresses such that adjacent nodes differ in precisely one bit. Address bits are numbered from 0 trough $n-1$ with the lowest order bit being the 0^{th} bit. Node i is the node that has a binary address equal to i, i.e., $i = (a_{n-1}a_{n-2}\ldots a_0)$. Let $\oplus$ be the bit-wise exclusive-or operation. The j^{th} port of a node i connects to the node k that differs from i in the j^{th} bit, i.e., $i \oplus k = (0_{n-1}0_{n-2}...0_{j+1}1_j0_{j-1}...0_0)$. There is a *port*, or *dimension* d, for each address bit, and ports (dimensions) are numbered from 0 through $n-1$. Let $|i|$ denote the number of bits with value one in the binary number i, hence $|i \oplus j|$ denote the *Hamming* distance between the binary numbers i and j.

A *spanning tree* of the Boolean cube is a connected subgraph of the Boolean cube graph which contains all the nodes of the Boolean cube and is a tree. In this paper, we consider the spanning tree as directed. A *source node*, or a *root node*, is a node with in-degree 0. A *sink node*, or a *leaf node*, has out-degree 0. Nodes that are neither source (root) nor sink (leaf) nodes are *internal nodes*. A *spanning graph*, as used in this paper, is either a spanning tree, or a sum of n spanning trees. A spanning graph is a connected, directed, graph with weighted edges. The weight of an edge in each of the spanning trees forming the spanning graph is $\frac{1}{n}$, except in the trivial case where the spanning graph is a spanning tree, in which case the weight of every edge is 1. The weight of an edge in the spaning graph is equal to the number of times the edge occurs in any of the trees making up the spanning graph, divided by n. The weight represents the fraction of the data each edge is transmitting.

A *greedy* spanning tree is a spanning tree for which the path length from the root to any node is equal to the shortest distance between the root and the node in the graph G, i.e., the Hamming distance in the case of a Boolean cube. A *greedy* spanning graph is a spanning graph such that the composed spanning trees are all greedy.

The root of a spanning graph is at *level 0* and the level, l, increases by one for each directed edge traversal away from the root. The *height*, h, of a spanning graph is equal to the maximum level. The nRSBT, nESBT and SBnT spanning graphs are constructed out of n spanning trees (or subtrees) labeled 0 through $n-1$ (from left to right in the figures of this paper).

R denotes the right rotation function defined by $R(i) = (a_0a_{n-1}a_{n-2}\ldots a_1)$, where $i = (a_{n-1}a_{n-2}\ldots a_0)$, and $R^j = R^{j-1} \circ R$ means a right rotation of j steps. The *rotation* of a graph with binary node addresses is accomplished by applying the same rotation function to all its addresses. Similarly for the *translation* of a graph. Clearly, adjacency is preserved under rotation and translation. Translation also preserves the relative order of dimensions. The period of a binary number i, P_i, is the least $j > 0$ such that $i = R^j(i)$. For example, the period of (011011) is 3. A binary number is *cyclic* if its period is less than its length and it is *non-cyclic* otherwise. A *relative address* of a node i in a spanning tree rooted at node s is $i \oplus s$. A *cyclic node* is a node with cyclic relative address. Note that a cyclic node is defined only when the source node is given.

The communication is assumed to be packet switched. M denotes the number of elements to be transmitted from the source node to any other node, t_c the transfer time for an element, and τ the start-up time for the communication of a packet of maximum B_m elements. We assume concurrent bi-directional communication, i.e., that a pair of adjacent nodes can exchange a pair

of messages during any routing cycle. With *one-port* communication we mean that each node can *exchange* data with one other node. In *n-port* communication a node can concurrently exchange data with all of its neighbors.

3. Spanning Graphs

In this section we define and characterize a few spanning graphs. For the all-to-all communication we assume that the spanning graphs for the different source nodes are distinct translations of each other. The interleaving properties of the different graphs determine the complexity of the communication. The translation of a graph through a bit-wise exclusive-or operation preserves the order of the dimensions, and hence the number of edges in each dimension. Let E_d be the number of spanning graph edges in dimension d, and Ew_d be the corresponding sum of weights of edges. For a spanning tree $Ew_d = E_d$. Moreover, let EL_d^N denote the number of spanning graph edges mapped to every cube edge in dimension d for N distinctly translated spanning graphs. To determine the *total load* on an edge, EwL_d^N, during all-to-all broadcasting based on N distinctly translated spanning graphs, the sum of weights of spanning graph edges mapped to every cube edge in dimension d is needed. To determine the number of elements, ewL_{ld}^N, contending for any cube edge in dimension d during a given routing cycle during all-to-all broadcasting the total weight of edges between levels l and $l+1$ in dimension d, ew_{ld}, of the spanning graph is needed. To determine ew_{ld} we first consider e_{ld}, the number of edges between levels l and $l+1$ in dimension d. Clearly, $\sum_{l=0}^{h-1} ew_{ld} = Ew_d$. For a spanning tree $ew_{ld} = e_{ld}$ and $Ew_d = E_d$. Moreover,

Lemma 3.1. *For a spanning graph composed of n edge-disjoint spanning trees of equal weight, $Ew_d = \frac{1}{n}E_d$ and $ew_{ld} = \frac{1}{n}e_{ld}$.*

Lemma 3.2. *For a spanning graph SG composed of n distinctly rotated spanning trees ST of equal weight $Ew_d^{SG} = \frac{1}{n}\sum_{d=0}^{n-1} E_d^{ST} = \frac{N-1}{n}$ $0 \le d \le n-1$, and $ew_{ld}^{SG} = \frac{1}{n}\sum_{d=0}^{n-1} e_{ld}^{ST} = \frac{1}{n}$ (the number of edges between levels l and $l+1$ in (any) one spanning tree), $0 \le d \le n-1$.*

Let E_d^N denote the number of edges in dimension d for N distinctly translated spanning graphs, and e_{ld}^N the total number of edges in dimension d extending from nodes at distance l from their respective sources. Then $E_d^N = N \times E_d$, $e_{ld}^N = N \times e_{ld}$, and $e^N w_{ld} = N \times ew_{ld}$.

Lemma 3.3. *For N distinctly translated spanning graphs, the number of spanning graph edges mapped to every cube edge in dimension d, EL_d^N, is E_d, and the number of spanning graph edges extending from nodes at distance l from their sources mapped to every cube edge in dimension d, eL_{ld}^N, is e_{ld}. Correspondingly, $EwL_d^N = Ew_d$, and $ewL_{ld}^N = ew_{ld}$.*

Proof. Any spanning graph edge is mapped to a distinct cube edge in the same dimension through N distinct exclusive-or operations. Hence, E_d spanning graph edges are mapped to every cube edge in dimension d, and $EL_d^N = E_d$. The bitwise exclusive-or operation also preserves the topology of a spanning graph, hence the number of edges at a given distance from the source node. It follows that $eL_{ld}^N = e_{ld}$. The argument for the weighted quantities is identical.

∎

The value of $EwL_d^N = Ew_d$ determines the total data volume that needs to be communicated across a cube edge during broadcasting, which is a lower bound for all-to-all broadcasting based

on the N translated spanning graphs, and concurrent communication. With synchronous routing and the same scheduling discipline in each node, $ewL_{ld}^{N} = ew_{ld}$ determines the number of elements contending for the same cube edge for *n-port* communication. This gives an upper bound.

For personalized communication we consider two scheduling disciplines:

- *Postorder* for each port and maximizing the minimum time between successive communications on the same port for *one-port* communication. With *n-port* all-to-all personalized communication, we also consider *postorder* for each port and concurrent communication for all the n ports.
- *Reverse-breadth-first* order for *n-port* communication.

In the following we refer to the first scheduling discipline simply as *postorder* scheduling. With *n-port* all-to-all personalized communication, a determining factor for the complexity of *postorder* scheduling is the maximum size of any subtree rooted at successive levels of the spanning graph. For the *reverse-breadth-first* discipline, the maximum number of nodes at successively decreasing distances from the root of any subtree of height at least equal to the given distance is a determining factor. Figure 1 shows the node sets considered for three routing cycles.

Lemma 3.4. *Given a spanning tree, let $\phi(i,j)$ be the number of nodes at distance j from node i in the subtree rooted at node i. If $\phi(i,j) \geq \phi(k,j)$ for any child node k of node i, then the data transfer time of n-port one-to-all personalized communication based on the given tree and reverse-breadth-first scheduling is dominated by the edges from the root.*

Proof. It follows from the fact that the propagation time for the internal nodes is at most the same as the transmission time for the root during each routing cycle. ∎

The property in lemma 3.4 simplifies the analysis of the communication complexity. Only the largest subtree need to be considered if the largest subtree is also of maximum height.

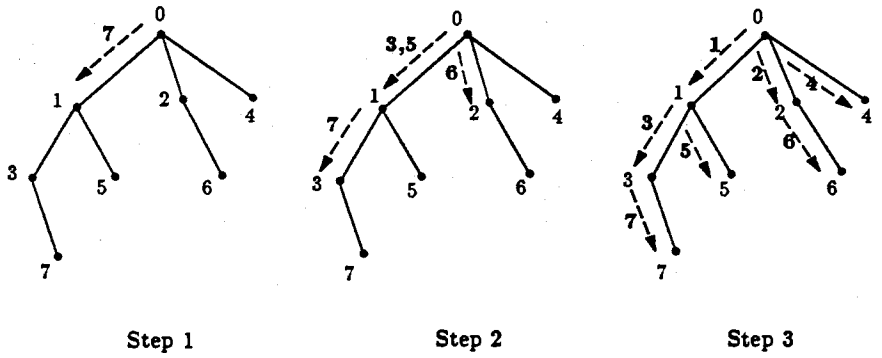

Figure 1: Node sets for *reverse-breadth-first* scheduling.

The total load on a cube edge for all-to-all personalized communication based on N translated spanning graphs is determined by the total number of subtree nodes connected through a given dimension. For a spanning graph s_{ld} denotes the total number of nodes in the subtrees rooted at level $l+1$ having subtree roots connected to level l through an edge in dimension d. sw_{ld} denotes

the corresponding weighted quantity. For N distinctly translated spanning graphs sL_{ld}^N denotes the total number of nodes, which receive data through a link in dimension d and have a predecessor node at level l. swL_{ld}^N denotes the weighted quantity.

For *n-port* communication and the *reverse-breadth-first* scheduling discipline, the data for nodes at level l is sent during routing cycle $h-l$, assuming commencement of the routing at cycle 0, and a height h of the spanning graph. Consequently, nodes with children at a maximum distance of $h-1-k$ will receive data from its parent during cycle k and become active sending out data from cycle $k+1$ until the last cycle, i.e., cycle $h-1$. Sender nodes are between levels 0 and k. All the data for nodes between levels $h-k$ and h are on their way during the k^{th} cycle. All the data elements will arrive at their destinations during the last routing cycle. We define r_{ld} to be the sum of nodes at distance l from ancestor nodes, with ancestor nodes connected by edges in dimension d to their parents, and rw_{ld} to be the corresponding weighted quantity. The subscript l, defined as the relative height, is the complement of the routing cycle such that $l+k=h-1$, Figure 1. For N distinctly translated graphs the number of edges being mapped to every cube edge in dimension d is equal to the number of edges in dimension d of the spanning graph being translated. Hence, the number of elements transmitted across any edge in dimension d during the k^{th} routing cycle, $rwL_{(h-1-k)d}^N$, is equal to the number of elements transmitted through dimension d by all nodes at levels 0 through k, $rw_{(h-1-k)d}$.

Lemma 3.5. *For N translated spanning graphs the number of subtree nodes in subtrees rooted at nodes of level $l+1$ and connected to a node at level l through dimension d is $sL_{ld}^N = s_{ld}$. Similarly, $swL_{ld}^N = sw_{ld}$. The number of nodes that are at a distance of $l+1$ from the originating node of every cube edge in dimension d is $rL_{ld}^N = r_{ld}$. Similarly, $rwL_{ld}^N = rw_{ld}$.*

Proof. Same argument as in the previous proof.

∎

Lemma 3.6. *For a spanning graph composed of n edge-disjoint spanning trees of equal weight, $sw_{ld} = \frac{1}{n}s_{ld}$ and $rw_{ld} = \frac{1}{n}r_{ld}$.*

Lemma 3.7. *For a spanning graph composed of n distinctly rotated spanning trees of equal weight and with height h, $sw_{ld} = rw_{ld} = \sum_{i=l}^{h-1} ew_{id}$. $0 \le l \le h-1, 0 \le d \le n-1$.*

Proof. From the property of n distinct rotations it follows that s_{ld} is the sum of the number of nodes at levels i, $l+1 \le i \le h$, for one spanning tree. Similarly, r_{ld} is the sum of the number of nodes at levels i, $l+1 \le i \le h$, for one spanning tree. Hence, $s_{ld} = r_{ld}$. By lemma 3.2, e_{ld} = the number of edges between levels l and $l+1$ in one spanning tree, which is also the number of nodes at level $l+1$ in one spanning tree. Lemmas 3.1 and 3.6 completes the proof.

∎

Lemma 3.8. *A spanning tree of an n-cube is greedy iff the number of nodes at level l is $\binom{n}{l}$.*

Proof. There are $\binom{n}{l}$ nodes at distance l from any cube node.

∎

Corollary 3.1. *Any greedy spanning graph has minimum height.*

Note that a spanning graph of minimum height is not necessarily a greedy spanning graph. For instance, a two-rooted complete binary tree has minimum height, but is not a greedy spanning

graph. In all-to-all personalized communication only greedy spanning graphs can serve as a basis for scheduling disciplines that accomplishes lower bound communication.

Lemma 3.9. *A greedy spanning graph of an n-cube is acyclic.*

Proof. A greedy spanning graph, by definition, is the sum of greedy spanning trees. Since all the edges of a greedy spanning tree rooted at node s are pointing from node i to node j with $|j \oplus s| = |i \oplus s| + 1$, it is acyclic.

∎

3.1. The Hamiltonian Path

There are many ways of generating Hamiltonian paths in a Boolean cube. One such way is to use a ***binary-reflected*** Gray code [26]. We refer to such a path as a BRG path.

Lemma 3.10. *The number of edges E_d in dimension d of a BRG path is 2^{n-d-1} and the number of edges e_{ld} in dimension d between levels l and $l+1$ of the BRG path is*

$$e_{ld} = \begin{cases} 0, & d \neq t; \\ 1, & d = t, \end{cases}$$

where t is the dimension of transition of the binary reflected Gray code in proceeding from l to $l+1$, which is also the dimension (bit position) of the rightmost 0 of the binary encoding of l.

Lemma 3.11. *The number of nodes in subtrees rooted at level $l+1$ and having subtree roots connected to level l through edges in dimension d is*

$$s_{ld} = \begin{cases} 0, & d \neq t; \\ N-l-1, & d = t. \end{cases}$$

The number of nodes r_{ld} at distance l from their ancestor nodes with ancestor nodes connected by edges in dimension d to their parents is

$$r_{ld} = \frac{N}{2^{d+1}} - \lceil \frac{l - 2^d + 1}{2^{d+1}} \rceil.$$

By rotation of the dimensions of the BRG path, n different paths are generated.

Lemma 3.12. *The paths of an nRBRG graph obtained through n distinct rotations of a BRG path are not edge-disjoint, for $n > 2$.*

Proof. By direct evaluation. For instance, the edge $2 \rightarrow 6$ is used in two paths of the 3RBRG, and those two paths are part of every nRBRG graph for $n > 3$.

∎

In fact, it can be shown that path i and path j share $2^{n-\alpha-1} + 2^{n-\beta-1} - 2$ edges where $\alpha = (i-j) \bmod n$ and $\beta = (j-i) \bmod n$. Also, n edges are shared by $n-1$ paths, i.e., the edges from node $(00...01_j0..0)$ to node $(00..01_{j+1}1_j0..0)$, $0 \leq j \leq n-1$. Figure 2 shows three rotated BRG paths in a 3-cube. The fact that the paths are not edge-disjoint limits the potential for pipelining.

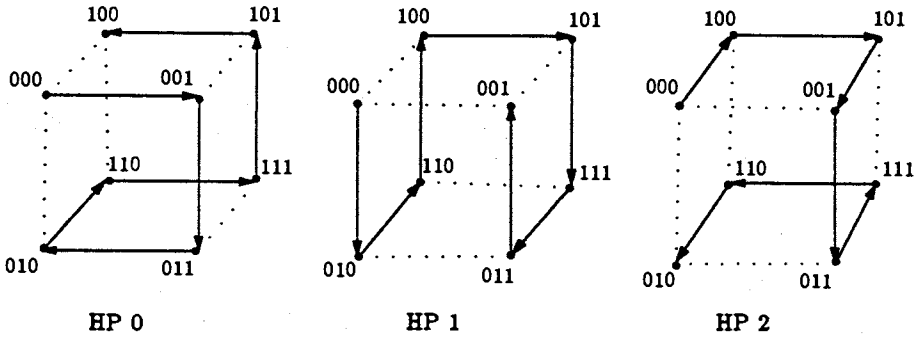

Figure 2: Three rotated binary-reflected Gray code paths in a 3-cube.

Lemma 3.13. *In an nRBRG graph, the edges between nodes at distance l and distance $l+1$ from the source node are edge-disjoint for $0 \leq l < N-1$.*

Proof. By construction the edges are in different dimensions. ∎

Lemma 3.14. *In an nRBRG graph, $E_d = \frac{3}{4}N$, $e_{ld} = 1$, $Ew_d = \frac{1}{n}(N-1)$ and $ew_{ld} = \frac{1}{n}$, $d \in [0, n-1]$, $l \in [0, N-1]$.*

Proof. From the definition of the nRBRG graph, it can be shown that path j and ($\bigcup$ path $i, i \in \{0, 1, ..., j-1\}$) have $\frac{N}{4} - 1 + \lfloor 2^{j-2} \rfloor$ edges in common, $1 \leq j \leq n-1$. Hence, the total number of cube edges in the nRBRG graph is $\frac{3}{4}nN$ and $E_d = \frac{3}{4}N$. e_{ld} is derived from e_{ld} of the BRG. Ew_d and ew_{ld} are derived by lemma 3.2.

Lemma 3.15. *In an nRBRG graph, $sw_{ld} = rw_{ld} = \frac{1}{n}(N-l-1)$.*

Proof. By lemmas 3.7 and 3.14. ∎

3.2. The Spanning Binomial Tree

A 0-level binomial tree has 1 node. An n-level binomial tree is constructed out of two $(n-1)$-level binomial trees by adding one edge between the roots of the two trees, and by making either root the new root, [1, 8]. It follows from this recursive construction that:

1. An n-level binomial tree has $\binom{n}{i}$ nodes at level i.
2. The n-level binomial tree is composed of n subtrees each of which is a binomial tree of $0, 1, \ldots, n-1$ levels respectively. The k level subtree has 2^k nodes. Hence, one subtree has $\frac{1}{2}N$ nodes, another $\frac{1}{4}N$ nodes etc.
3. An n-level binomial tree can be obtained from a k-level binomial tree, $k < n$, by replacing each node of the k-level binomial tree by an $(n-k)$-level binomial tree. The children nodes of a node in the k-level tree become children of the root of the replacing $(n-k)$-level binomial tree.

Since an n-level binomial tree can be embedded in an n-cube as a spanning tree, we called it a spanning binomial tree (SBT). It is a greedy spanning tree.

The familiar spanning tree rooted in node 0 of a Boolean n-cube generated by complementing leading zeroes of the binary encoding of a processor address i [10, 20, 25, 27, 29] is indeed a spanning binomial tree. For an arbitrary source node s the spanning tree is simply translated by a bit-wise exclusive or operation on all addresses with the address of the source node, i.e., $c = i \oplus s$ is formed. Complementation of those bits of i that correspond to the leading zeroes of c defines the edges of the translated spanning tree. More precisely, let $s = (s_{n-1}s_{n-2}...s_0)$, $i = (a_{n-1}a_{n-2}...a_0)$, and $c = (c_{n-1}c_{n-2}...c_0)$, where $c_m = s_m \oplus a_m$. Let $c_k = 1$ and $c_m = 0, \forall m > k$ with $k = -1$ for $c = 0$, i.e., k is the highest order bit of c that is 1. Let $children_{SBT}(i,s)$ be the set of children nodes of node i in the SBT rooted at node s and $M_{SBT}(i \oplus s) = \{k+1, \ldots, n-1\}$. Then,

$$children_{SBT}(i,s) = \{(a_{n-1}a_{n-2}\ldots\bar{a}_m\ldots a_0)\}, \quad \forall m \in M_{SBT}(i \oplus s).$$

The children function defines the SBT. This definition can be used as a distributed algorithm for generating the SBT. Each node only needs information of which node is the source node, and its own address. The parent function, $parent_{SBT}(i,s)$ is the inverse of the children function.

$$parent_{SBT}(i,s) = \begin{cases} \phi. & i = s; \\ (a_{n-1}a_{n-2}....\bar{a}_k...a_0), & i \neq s. \end{cases}$$

It is easy to verify that the parent and children functions are consistent, i.e., that node j is a child of node i iff node i is the parent of node j. Figure 3 shows a spanning tree generated by the children (or parent) function for the root located at node 0 in a 4-cube.

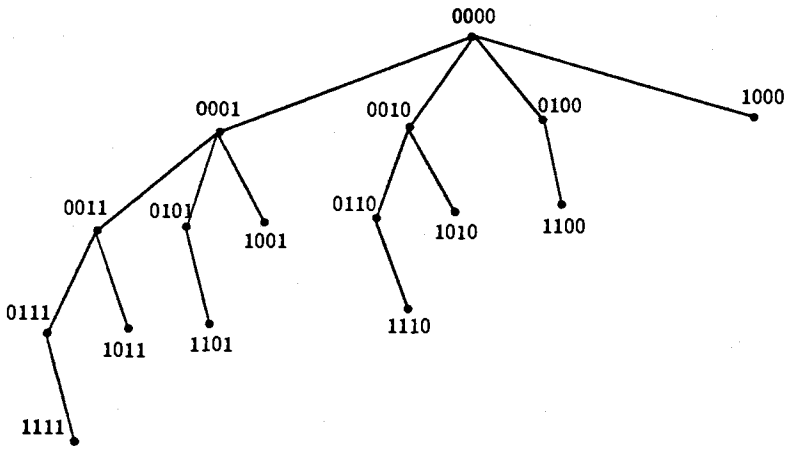

Figure 3: A spanning binomial tree in a 4-cube.

Lemma 3.16. *Let $\phi(i,j)$ be the number of nodes at distance j from node i in the subtree rooted at node i. Then, $\phi(i,j) \geq \phi(k,j)$ where node k is a child of node i.*

Proof. From the definition of the SBT, the subtree rooted at node k is a connected subgraph of the subtree rooted at node i.

∎

Lemma 3.17. *The number of edges E_d in dimension d is 2^d, $0 \leq d \leq n-1$ and the number of edges e_{ld} between level l and $l+1$ in dimension d is $\binom{d}{l}$. where the root is at level 0.*

Proof. By induction on the cube size. ∎

Lemma 3.18. *The number of nodes s_{ld} in subtrees rooted at level $l+1$ with subtree roots connected to level l through dimension d is $\binom{d}{l}2^{n-d-1}$. (Note that $\binom{x}{y} = 0$ if $x < y$.) Moreover,* $\max\ s_{ld}$, $0 \leq d \leq n-1$ *is attained at* $d = \min(2l, n-1)$.

$$\max\ s_{ld} = \begin{cases} \binom{2l}{l}2^{n-2l-1}, & \text{if } 0 \leq l \leq \lfloor \frac{n-1}{2} \rfloor; \\ \binom{n-1}{l}, & \text{if } \lfloor \frac{n-1}{2} \rfloor < l \leq n-1. \end{cases}$$

$$r_{ld} = \binom{n-d-1}{l}2^d = s_{l(n-d-1)}, 0 \leq l, d \leq n-1.$$

3.3. The Complete Binary Tree

A complete binary tree of $2^n - 1$ nodes cannot be embedded in an n-cube with edge dilation 1 [28, 3, 7], but a *Two-rooted complete binary tree* (TCBT) can [3, 7]. A two-rooted complete binary tree is a complete binary tree with the root split into two nodes. Hence, the TCBT has 2^n nodes. Figure 4 shows a 16 nodes two-rooted complete binary tree. The distance between the root and the furthest node is n for an N nodes TCBT. The number of nodes at level i of an N nodes TCBT is

$$\begin{cases} 1, & \text{if } i = 0; \\ 2, & \text{if } i = 1; \\ 3 \times 2^{i-2}, & \text{if } 2 \leq i < n; \\ N/4, & \text{if } i = n. \end{cases}$$

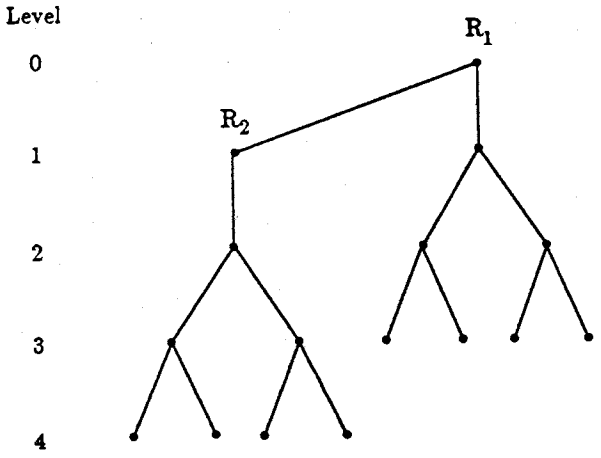

Figure 4: A 16 nodes two-rooted complete binary tree.

3.4. The nRSBT Spanning Graph

The nRSBT graph is the union of n spanning binomial trees each of which is a distinct rotation of the SBT described previously. Figure 5 shows the nRSBT graph of a 3-cube with the appropriate weights on the edges. The nRSBT graph contains all cube edges directed in the $0 \rightarrow 1$ direction, i.e., half of the total number of cube edges. The nRSBT graph is a greedy spanning graph, and hence acyclic by lemma 3.9.

Lemma 3.19. *If the source is node 0, then for any node the weight of a $0 \rightarrow 1$ incoming edge in dimension d is $\frac{k+1}{n}$, where k is the number of consecutive 0-bits immediately to the left of bit d.*

Proof. From the definition of the SBT and the rotation property, the given cube edge in dimension d occurs in SBT $d+1, d+2, \ldots, d+k+1$, i.e., in these $k+1$ SBTs, bit d is the last bit complemented in reaching the node being considered.

∎

For instance, node (011001) has an incoming edge in dimension 0 weighted 1/2, an incoming edge in dimension 3 weighted 1/6, and an incoming edge in dimension 4 weighted 1/3. As a consequence of this lemma, the weight of a $0 \rightarrow 1$ outgoing edge in dimension d is $\frac{k+1}{n}$ with the same k defined above. Nodes at level l have l parents and $(n-l)$ children in the nRSBT graph.

Corollary 3.2. *The sum of the weights of incoming edges is 1 for every node, except for the source.*

Proof. From lemma 3.19 it is clear that the weight of a $0 \rightarrow 1$ edge is equal to $\frac{1}{n}$ times the number of dimensions encountered before the next 1 in the address is found to the left of the dimension considered. Hence, the total weight is equal to $\frac{1}{n}$ times the number of address bits.

∎

Corollary 3.2 guarantees that every node receives the entire data set.

Lemma 3.20. *For the nRSBT graph $E_d = \frac{1}{2}N$ and $Ew_d = \frac{1}{n}(N-1)$. The edge weight in dimension d between levels l and $l+1$ is $ew_{ld} = \frac{1}{n}\binom{n}{l+1}$, $d \in [0, n-1]$, $\quad l \in [0, n-1]$.*

Proof. Since $E_{n-1}^{SBT} = \frac{1}{2}N$ it follows that $E_d^{nRSBT} = \frac{1}{2}N$ for every d, since rotation does not change the direction of an edge. The rest of the lemma follows from lemma 3.2.

∎

From this lemma and the greedy property, we conclude that all the $\frac{1}{2}nN$ cube edges pointing away from the root node belong to the nRSBT graph. Also, none of the $\frac{1}{2}nN$ cube edges pointing toward the root node belong to the nRSBT graph.

Lemma 3.21. *The values of sw_{ld} and rw_{ld} are $sw_{ld} = rw_{ld} = \frac{1}{n}\sum_{i=l+1}^{n}\binom{n}{i}$.*

Proof. By lemma 3.7.

∎

3.5. The nESBT Spanning Graph

The nESBT (n Edge-disjoint Spanning Binomial Trees) graph is composed of n SBTs with one tree rooted at each of the nodes adjacent to the source node. The SBTs are rotated such that the source node of the nESBT graph is in the smallest subtree of each SBT. The nESBT graph is then obtained by reversing the edges from the roots of the SBTs to the source node. After the edge reversal each SBT becomes an ERSBT (for Edge Reversed Spanning Binomial Tree). Figure 6 shows an nESBT graph formed by three SBTs in a 3-cube. The height of the nESBT graph is $n+1$, since the source node is adjacent to all the roots of the SBTs used in the definition of the nESBT graph. The total number of edges in the n SBTs is $n(N-1)$.

Before proving that the SBTs are edge disjoint we specialize the children function to the particular situation in the nESBT graph. Assume first that the source node is node 0. The SBTs used for

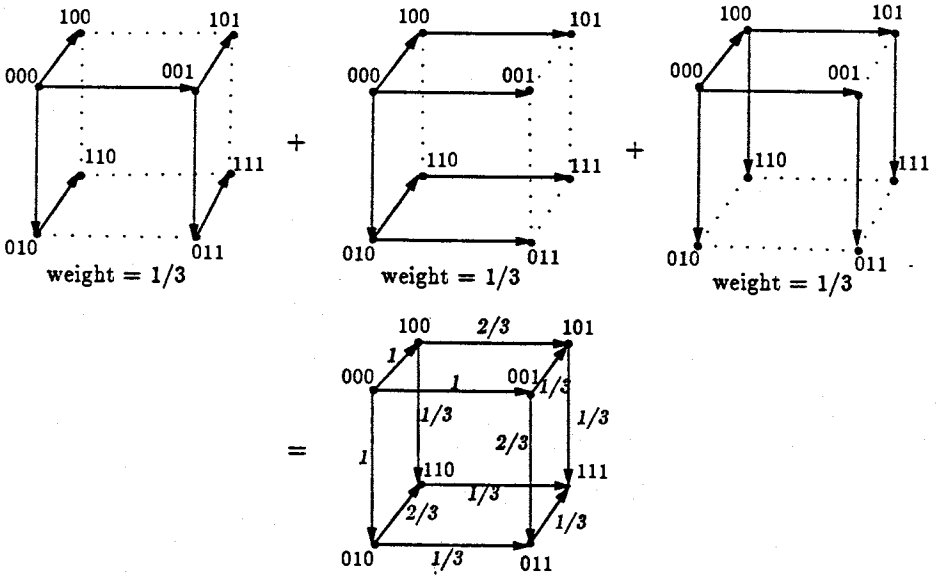

Figure 5: Three rotated spanning binomial trees as a spanning graph in a 3-cube.

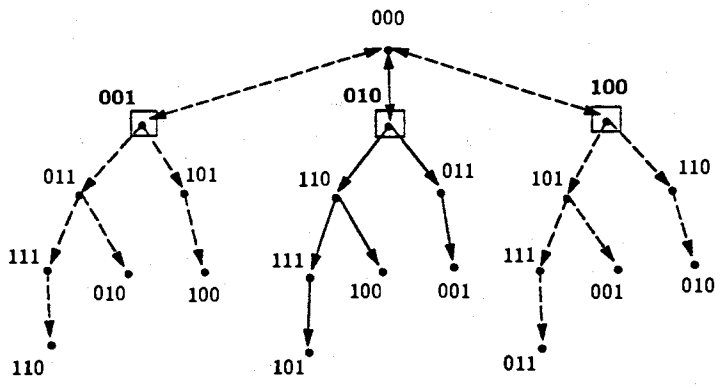

Figure 6: Subtrees of an nESBT viewed as SBTs.

the construction of the nESBT graph rooted at node 0 are rooted at nodes $(00\ldots01_j0\ldots0), \forall j \in \{0,1,\ldots,n-1\}$. We refer to the SBT rooted at node $(00\ldots01_j0\ldots0)$ as the j^{th} SBT of the nESBT graph. The j^{th} ERSBT is obtained from the j^{th} SBT by reversing the edge directed to node 0 (the source). The j^{th} SBT is obtained by rotating left $j+1$ steps the SBT rooted at node 0, then translating its root to location $(00\ldots01_j0\ldots0)$. The children of a node in a given subtree, say the j^{th} , can in principle be determined by a translation to move the root of the j^{th} subtree to node 0, perform a $j+1$ step right rotation of its address, execute the children function in the definition of the SBT, perform a $j+1$ step left rotation, and finally a translation to move the root back to node $(00\ldots01_j0\ldots0)$. This cumbersome procedure is readily accommodated in a modified *children* function.

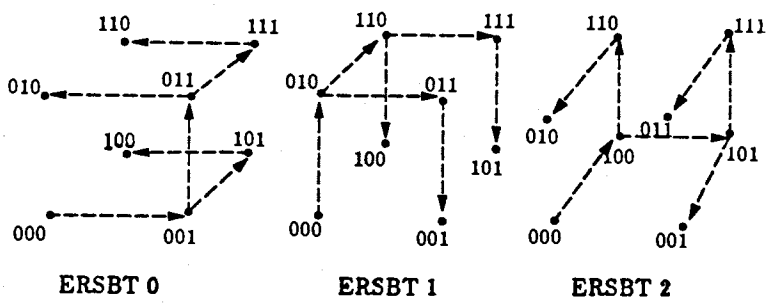

Figure 7: Three edge-disjoint directed spanning trees in a 3-cube.

Let $i = (a_{n-1}a_{n-2}\ldots a_0)$ and k be such that $a_k = 1$, and $a_m = 0, \forall m \in \mathcal{M}_{nESBT}(i,j)$, where $\mathcal{M}_{nESBT}(i,j) = \{(k+1) \bmod n, (k+2) \bmod n, \ldots, (j-1) \bmod n\}$. Hence, k is the first bit to the right of bit j, cyclically, which is equal to one, if $k \neq j$. For $i = 0$ $k = -1$. For the j^{th} ERSBT of the nESBT graph with source node 0 the set of children nodes of node i is defined by

$$children_{nESBT}(i,j,0) = \begin{cases} (a_{n-1}a_{n-2}\ldots\overline{a}_j\ldots a_0), \forall j \in \{0,1,\ldots,n-1\}, & \text{if } k = -1; \\ \{(a_{n-1}a_{n-2}\ldots\overline{a}_m\ldots a_0)\}, \forall m \in \mathcal{M}_{nESBT}(i,j) \bigcup \{j\}, & \text{if } a_j = 1,\ k \neq j; \\ \{(a_{n-1}a_{n-2}\ldots\overline{a}_m\ldots a_0)\}, \forall m \in \mathcal{M}_{nESBT}(i,j), & \text{if } a_j = 1,\ k = j; \\ \phi, & \text{if } a_j = 0, k \neq -1. \end{cases}$$

All nodes with bit j equal to zero are leaf nodes of the j^{th} ERSBT, except node 0. Conversly, all nodes with $a_j = 1$ are internal nodes of the j^{th} ERSBT. The exceptional connection to node 0 is handled by the conditions on k. The first case defines the children for the root of the j^{th} ERSBT, the second the set of children nodes of internal nodes of the j^{th} ERSBT, except the node at level 1. The third case handles the node at level 1, and the last case handles the leaf nodes of the ERSBT. Figure 7 shows that the three ERSBTs of a 3-cube are edge-disjoint. Figures 8 shows nESBT graphs with source node 0 in a 4-cube.

The $parent_{nESBT}$ function is

$$parent_{nESBT}(i,j,0) = \begin{cases} \phi, & \text{if } k = -1; \\ (a_{n-1}a_{n-2}\ldots\overline{a}_j\ldots a_0), & \text{if } a_j = 0, k \neq -1; \\ (a_{n-1}a_{n-2}\ldots\overline{a}_k\ldots a_0), & \text{if } a_j = 1. \end{cases}$$

Theorem 3.1. *The n directed ERSBTs are edge disjoint.*

Proof. We only need to prove that for an arbitrary node the address of its parent node in each of the n ERSBTs is obtained by complementing a distinct bit.

From the definition of the ERSBTs it is clear that a node is a leaf node of the j^{th} ERSBT iff the j^{th} bit is 0, with the exception of node 0. If a node is a leaf node of an ERSBT, then its parents address in that ERSBT is obtained by complementing the corresponding bit (bit j for the j^{th} ERSBT). If a node is an internal node of an ERSBT, say the j^{th} ERSBT, then the corresponding bit is 1, and the parents address is obtained by complementing the first bit cyclically to the right of the j^{th} bit that is equal to 1. Hence, the addresses of the parent nodes for all the ERSBTs of

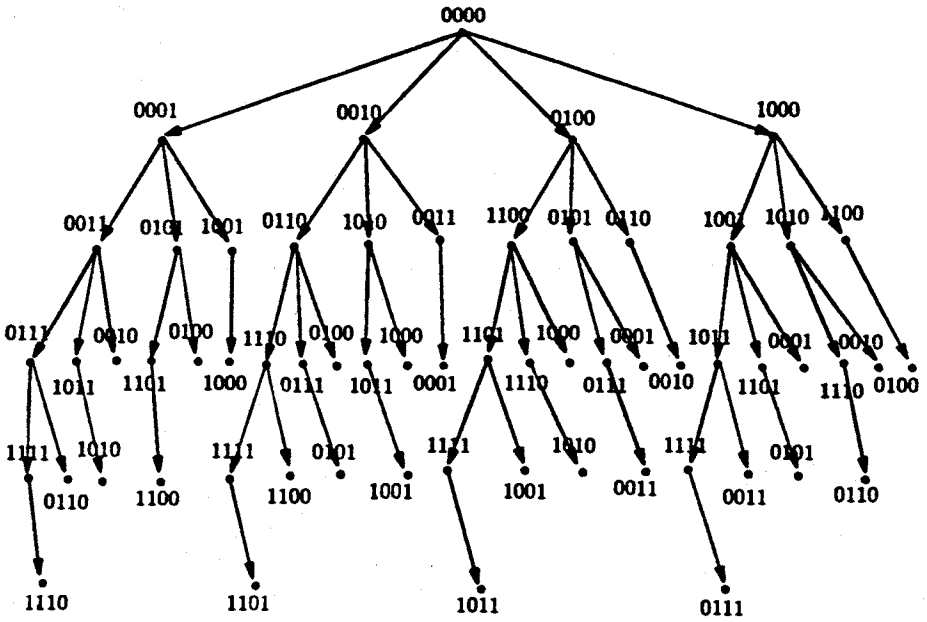

Figure 8: Four edge-disjoint directed spanning trees in a 4-cube.

which the considered node is an internal node are also obtained by complementing distinct bits. Figure 9 shows the parents and children of one node in a 6-cube. The numbers on the edges are the dimensions through which the node connects to its parents or children nodes in different ERSBTs. The labels on the nodes denotes the ERSBT to which the parent and children nodes belong.

∎

For an arbitrary source node s an nESBT graph is defined by translating the nESBT graph rooted at node 0. The only difference in the definition of the $parent_{nESBT}$ and $children_{nESBT}$ functions is that k is determined from $c = i \oplus s$. Hence, for a source node s, k is such that $c_k = 1$, and $c_m = 0, \forall m \in \mathcal{M}_{nESBT}(i \oplus s, j)$. For the special case of $c = 0$, $k = -1$.

$$children_{nESBT}(i,j,s) = \begin{cases} (a_{n-1}a_{n-2}...\overline{a}_j...a_0), \forall j \in \{0,1,\ldots,n-1\}, & \text{if } k = -1; \\ \{(a_{n-1}a_{n-2}...\overline{a}_m...a_0)\}, & \\ \quad \forall m \in \mathcal{M}_{nESBT}(i \oplus s, j) \bigcup \{j\}, & \text{if } c_j = 1,\ k \neq j; \\ \{(a_{n-1}a_{n-2}...\overline{a}_m...a_0)\}, & \\ \quad \forall m \in \mathcal{M}_{nESBT}(i \oplus s, j), & \text{if } c_j = 1,\ k = j; \\ \phi, & \text{if } c_j = 0, k \neq -1. \end{cases}$$

$$parent_{nESBT}(i,j,s) = \begin{cases} \phi, & \text{if } k = -1; \\ (a_{n-1}a_{n-2}...\overline{a}_j...a_0), & \text{if } c_j = 0, k \neq -1; \\ (a_{n-1}a_{n-2}...\overline{a}_k...a_0), & \text{if } c_j = 1. \end{cases}$$

Lemma 3.22. *The j^{th} ERSBT can be derived by $(j - i)$ mod n steps left rotations of each node of the i^{th} ERSBT.*

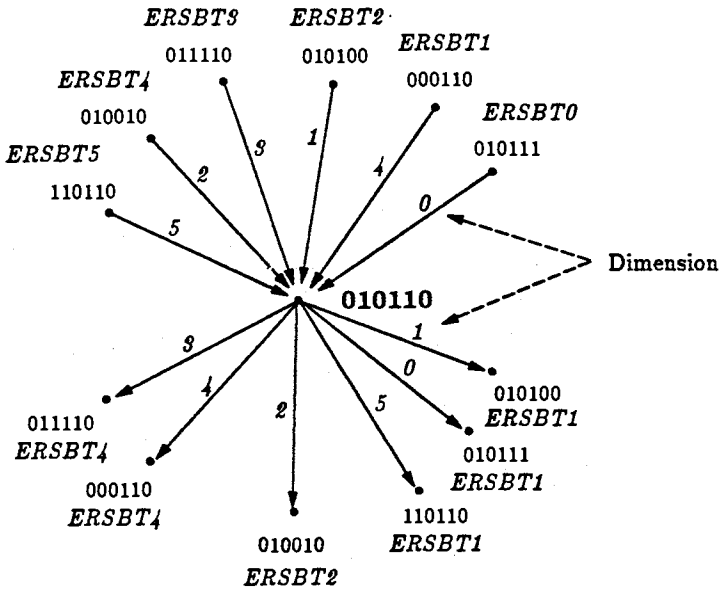

Figure 9: Parents and children of a node in a 6-cube.

Proof. From the definition of the parent or children functions. ∎

Corollary 3.3. *There exist an edge-disjoint embedding of n Spanning Binomial Trees in an n-cube.*

Corollary 3.4. *The nESBT graph for a Boolean n-cube is a directed graph, such that all directed cube edges, except those incident on the source node, appears precisely once in the nESBT graph.*

Proof. Follows from theorem 3.1. ∎

Corollary 3.5. *The in-degree and the out-degree of any node in an nESBT graph is n, with the exception that the source has in-degree 0 and all the neighbors of the source have out-degree $n-1$.*

Theorem 3.2. *The height of the nESBT graph is minimal among all possible configurations of n edge disjoint spanning trees.*

Proof. To prove that with n edge disjoint spanning trees, the height $n+1$ is minimal, we prove that n disjoint spanning trees with height n is impossible. Without loss of generality, we assume that the address of the root node is 0. The total number of directed edges in an n-cube is nN, but only the edges directed out from the source may be used (assuming dilation 1). Each spanning tree has $N-1$ edges. Hence, every eligible edge is used by the n edge disjoint spanning trees. It follows that the edges directed out from the node with address $N-1$ also must be used, and since this node is at distance n from the source node, the theorem follows. ∎

Lemma 3.23. *Labeling the levels of the nESBT graph from 0 to $n+1$ with 0 at the root, the number*

of nodes at level i of a single ERSBT is

$$\begin{cases} \binom{n-1}{i-1} + \binom{n-1}{i-2} = \binom{n}{i-1}, & \text{for } n+1 \geq i \geq 1, i \neq 2; \\ 1, & \text{for } i = 0; \\ n-1, & \text{for } i = 2. \end{cases}$$

Proof. From the definition. ∎

Lemma 3.24. *For the nESBT $E_d = N - 1$ and $e_{ld} = \binom{n}{l}$, $l \in \{2, n\}$. $e_{0d} = 1$ and $e_{1d} = n - 1$. $Ew_d = \frac{1}{n}(N-1)$ and $ew_{ld} = \frac{1}{n}e_{ld}$.*

Proof. By lemmas 3.1, 3.2 and 3.23. ∎

Lemma 3.25. *The values of sw_{ld} and rw_{ld} of an nESBT are*

$$sw_{ld} = rw_{ld} = \begin{cases} \frac{1}{n}(N-1), & \text{for } l = 0; \\ \frac{1}{n}(N-2), & \text{for } l = 1; \\ \frac{1}{n}\sum_{i=l}^{n}\binom{n}{i}, & \text{for } 2 \leq l \leq n. \end{cases}$$

Proof. By lemmas 3.7 and 3.24. ∎

3.6. The Spanning Balanced n-Tree

In the *Spanning Balanced n-Tree* [17] the node set is divided into n sets of nodes with approximately an equal number of nodes. Each such set forms a subtree of the source node. The maximum load on the edges directed away from the source node is minimized for personalized communication.

We define the SBnT by pruning the nESBT graph. This SBnT is very well balanced, but can be perfectly balanced. We show how this balancing can be accomplished. The two-stage definition of the SBnT is made for ease of exposition. In the SBT a node i belongs to the j^{th} subtree iff $a_j = 1$, $a_k = 0$, $k < j$. In the nESBT graph a node is an internal node of the j^{th} ERSBT if $a_j = 1$. Bit j can be considered as a *base* for the j^{th} subtree. For the SBnT we define the base as follows: Let $J_i = \{j_1, j_2, \ldots, j_m\}$, where $j_1 < j_2 < \ldots j_m$, $R^u(i) = R^v(i)$, $u, v \in J_i$, and $R^u(i) < R^l(i)$, $u \in J_i$, $l \notin J_i$. $|J_i| = n/P_i$ where P_i is the period of i. Define $base(i) = j_1$, i.e., the value of the base equals the minimum number of right rotations which minimizes the value of i. For non-cyclic nodes $|J_i| = 1$, but for a non-cyclic node i, $P_i < n$, hence, $|J_i| > 1$. The notion of *base* is similar to the idea of distinguished node used in [24] in that $base = 0$ distinguishes a node from a generator set (necklace). For ease of notation we omit the subscript on j in the following. For the definition of the $parent_{SBnT}$ and $children_{SBnT}$ functions we first find the position k of the first bit cyclically to the right of bit j that is equal to 1, i.e., $a_k = 1$, and $a_m = 0, \forall m \in \mathcal{M}_{nESBT}(i,j)$, ($k = j$, if every bit but bit j is 0). For $i = 0$ $k = -1$. Then

$$children_{SBnT}(i,0) = \begin{cases} \{(a_{n-1}a_{n-2}\ldots\overline{a}_m\ldots a_0)\}, \forall m \in \{0,1,\ldots,n-1\}, & \text{if } i = 0; \\ \{q_m = (a_{n-1}a_{n-2}\ldots\overline{a}_m\ldots a_0)\}, \\ \quad \forall m \in \mathcal{M}_{nESBT}(i,j) \text{ and } base(q_m) = base(i), & \text{if } i \neq 0. \end{cases}$$

$$parent_{SBnT}(i,0) = \begin{cases} \phi, & \text{if } i = 0; \\ (a_{n-1}a_{n-2}\ldots\overline{a}_k\ldots a_0), & \text{otherwise.} \end{cases}$$

The $parent_{SBnT}$ function preserves the base, since for any node i with base j, a_k is the highest order bit of $R^j(i)$. Complementing this bit cannot change the base. It is also readily seen that the $parent_{SBnT}$ and $children_{SBnT}$ functions are consistent.

Theorem 3.3. *The $parent_{SBnT}$ (or the $children_{SBnT}$) function defines a spanning tree rooted at node 0.*

Proof. For every node i the $parent_{SBnT}$ function generates a path to node 0. Hence, the graph is connected. Moreover, the parent node of a node at distance d from node 0 is at distance $d-1$ from node 0, and each node only has one parent node. Hence, the graph is a spanning tree.

∎

Figure 10 shows that a SBnT is a connected subgraph of the nESBT graph. Figure 11 shows a spanning balanced 5-tree.

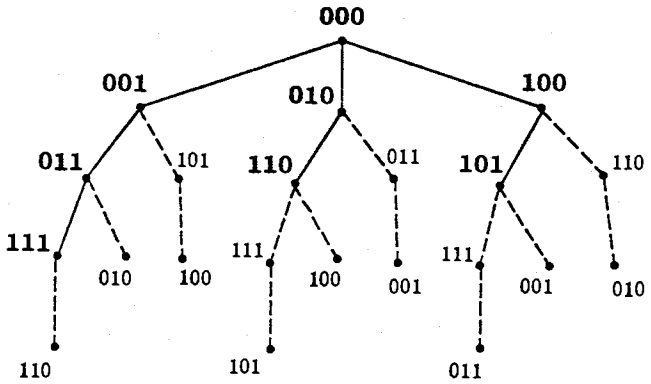

Figure 10: A SBnT as a subgraph of an undirected nESBT in a 3-cube.

For an arbitrary source node s we translate the SBnT rooted at node 0 to node s by performing for each node the bit-wise exclusive-or function of its address and the address of the source node. The base of a node is determined from $c = i \oplus s$, and the children and parent functions are readily modified.

Let $J_{i,s} = \{j_1, j_2, \ldots, j_m\}$, where $j_1 < j_2 < \ldots j_m$, $R^u(c) = R^v(c), \quad u, v \in J_{i,s}$, and $R^u(c) < R^l(c), \quad u \in J_{i,s}, \quad l \notin J_{i,s}$. Then $base(c) = j_1 = j$ and k is defined by $c_k = 1$ and $c_m = 0, \forall m \in \mathcal{M}_{nESBT}(i \oplus s, j)$ with $k = -1$ if $c = 0$.

$$children_{SBnT}(i,s) = \begin{cases} \{(a_{n-1}a_{n-2}\ldots\overline{a}_m\ldots a_0)\}, \forall m \in \{0,1,\ldots,n-1\}, & \text{if } c = 0; \\ \{q_m = (a_{n-1}a_{n-2}\ldots\overline{a}_m\ldots a_0)\}, \\ \quad \forall m \in \mathcal{M}_{nESBT}(i \oplus s, j) \text{ and } base(q_m \oplus s) = base(i \oplus s), & \text{if } c \neq 0. \end{cases}$$

$$parent_{SBnT}(i,s) = \begin{cases} \phi, & \text{if } c = 0; \\ (a_{n-1}a_{n-2}\ldots\overline{a}_k\ldots a_0). & \text{otherwise.} \end{cases}$$

Lemma 3.26. *The number of distinct nodes at level l is $\binom{n}{l}$.*

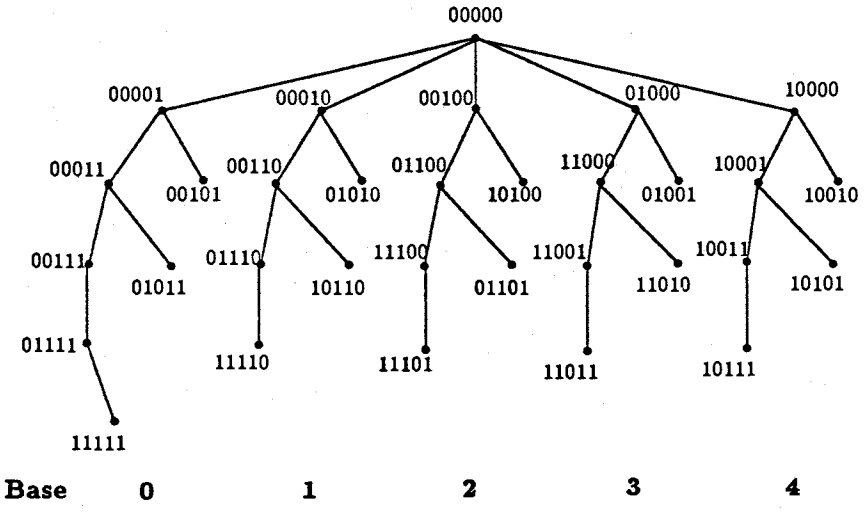

Figure 11: A spanning balanced 5-tree in a 5-cube.

Proof. From the definition of the parent (or children) function it follows that the distance from any node to the source node is $|c|$, i.e., the SBnT graph is greedy, and the lemma follows by lemma 3.8. ∎

Lemma 3.27. *Let $\phi(i,j)$ be the number of distinct nodes at distance j from node i in the subtree rooted at node i. Then, $\phi(i,j) \geq \phi(k,j)$. where node k is a child of node i.*

Theorem 3.4. *Excluding node $i \oplus s = (11\ldots1)$, all the subtrees of the SBnT are isomorphic if n is a prime number. Furthermore, the j^{th} subtree can be derived by $(j-i)$ mod n left rotation steps of each node of the i^{th} subtree.*

Proof. Since n is a prime number there are no cyclic nodes except nodes $(00\ldots0)$ and $(11\ldots1)$. The latter part can be shown from the definitions of parent or children function. With the root at node 0 and excluding node $N-1$ all nodes are non-cyclic, and since the different subtrees are obtained through rotations of the addresses they are isomorphic. The proof is completed by noticing that a translation does not alter the topology. ∎

The load imbalance in the SBnT graph is caused by the cyclic nodes. By allowing multiple paths to cyclic nodes the load can be perfectly balanced among the subtrees of the root. The number of paths to a node i is n/P_i. To allow for multiple paths to cyclic nodes the definition of base becomes the set $J_{i,s}$ instead of the integer j_1. The modified SBnT graph, SBnTc is no longer a tree. However, the SBnTc can be viewed as composed of n SBnT, called SBnT 0, 1, ..., $n-1$, such that the base of SBnT r is defined as j_1 where $J_{i,s} = \{j_1, j_2, \ldots, j_m\}$ and $(j_1 + r) \bmod n < (j_2 + r) \bmod n < \ldots (j_m + r) \bmod n$, $R^u(c) = R^v(c)$, $u, v \in J_{i,s}$. and $R^u(c) < R^l(c)$, $u \in J_{i,s}$, $l \notin J_{i,s}$.

The SBnTc graph is perfectly balanced for $M \bmod n = 0$. In the complexity analysis we assume that the personalized data is partitioned into n/P_i sets for every node i. Then, the load on each edge directed away from the source node is $\frac{(N-1)M}{n}$.

	Ew_d	ew_{ld}	sw_{ld}	rw_{ld}
BRG	2^{n-d-1}	0 or 1	0 or $N-l-1$	$\frac{N}{2^{d+1}} - \lceil \frac{l-2^d+1}{2^{d+1}} \rceil$
nRBRG	$\frac{N-1}{n}$	$\frac{1}{n}$	$\frac{1}{n}(N-l-1)$	$\frac{1}{n}(N-l-1)$
SBT	2^d	$\binom{d}{l}$	$\binom{d}{l}2^{n-d-1}$	$\binom{n-d-1}{l}2^d$
nRSBT	$\frac{N-1}{n}$	$\frac{1}{n}\binom{n}{l+1}$	$\frac{1}{n}\sum_{i=l+1}^{n}\binom{n}{i}$	$\frac{1}{n}\sum_{i=l+1}^{n}\binom{n}{i}$
nESBT	$\frac{N-1}{n}$	$\frac{1}{n}\binom{n}{l}, l \geq 2$	$\frac{1}{n}\sum_{i=l}^{n}\binom{n}{i}, l \geq 2$	$\frac{1}{n}\sum_{i=l}^{n}\binom{n}{i}, l \geq 2$
SBnT	$\frac{N-1}{n}$	$\frac{1}{n}\binom{n}{l+1}$	$\frac{1}{n}\sum_{i=l+1}^{n}\binom{n}{i}$	$\frac{1}{n}\sum_{i=l+1}^{n}\binom{n}{i}$

Table 1: Summary of Ew_d, ew_{ld}, sw_{ld} and rw_{ld}.

Lemma 3.28. *The values of $Ew_d, ew_{ld}, sw_{ld}, rw_{ld}$ for a SBnTc are*

$$Ew_d = \frac{1}{n}(N-1), \qquad ew_{ld} = \frac{1}{n}\binom{n}{l+1}, \qquad sw_{ld} = rw_{ld} = \frac{1}{n}\sum_{i=l+1}^{n}\binom{n}{i}.$$

Proof. By lemmas 3.2 and 3.7. ∎

In the following, SBnT always means the SBnTc graph. Table 1 lists the values of Ew_d, ew_{ld}, sw_{ld} and rw_{ld} for various spanning graphs.

4. One-to-All Broadcasting

In this section we derive complexity estimates for broadcasting based on the various spanning graphs. A lower bound for one-to-all broadcasting with *one-port* communication is $(M+n-1)t_c+n\tau$, since the height of any spanning graph is at least n, and a node must receive M elements. This is not a strict lower bound, in general. With *n-port* communication a lower bound for the data transfer time is $(\lceil \frac{M}{n} \rceil + n - 1)t_c$, since the fan-out of the source is n. With $M = 1$ the strict bound is $(t_c + \tau)n$ both for *one-port* and *n-port* communication. This bound is realized by SBT routing and appropriate scheduling. Indeed, with *n-port* communication any spanning graph of height n can realize the lower bound communication for $M = 1$. With $1 < M \leq n$ and *n-port* communication, the nRSBT routing attains the lower bound, $n(t_c + \tau)$. The nRSBT routing does not utilize pipelining. In general, it needs a time of $\lceil \frac{M}{n} \rceil n(t_c + \tau)$ with *n-port* communication.

The nESBT routing yields the lowest communication complexity of the spanning graphs we consider, both for *one-port* and *n-port* communication, except if only one routing cycle is required by the root, i.e., $M = 1$ for *one-port* communication and $1 < M \leq n$ for *n-port* communication. For the first case the SBT routing is of lower complexity than the nESBT routing, and for the second the nRSBT routing is optimum. However, the nESBT routing is only inferior by 1 routing cycle.

The SBnT routing is not competitive for one-to-all broadcasting, but it is a good graph for all-to-all broadcasting and personalized communication. Hence, we do not analyze the complexity of one-to-all broadcasting based on the SBnT graph.

4.1. The Complexity of One-to-All Broadcasting

4.1.1. The Hamiltonian Path

With *one-port* communication the broadcasting of M elements requires a time of $(Mt_c + \lceil \frac{M}{B} \rceil \tau)(N-1)$ without pipelining, i.e., $(Mt_c + \tau)(N-1)$, if $B \geq M$. When pipelining is applied

$B_{opt} = \sqrt{\frac{2M\tau}{(N-3)t_c}}$ and $T_{min} = (\sqrt{2Mt_c} + \sqrt{(N-3)\tau})^2$. For one send operation concurrently with one receive operation on different ports, $B_{opt} = \sqrt{\frac{M\tau}{(N-2)t_c}}$, and $T_{min} = (\sqrt{Mt_c} + \sqrt{(N-2)\tau})^2$. With *n-port* communication and paths generated by rotated binary-reflected Gray code sequences the time is $\frac{M(N-1)}{n}t_c + \lceil \frac{M}{nB} \rceil (N-1)\tau$ without pipelining, which is $(\frac{M}{n}t_c + \tau)(N-1)$, if $B_m \geq \frac{M}{n}$. The n rotated binary-reflected Gray code (nRBRG) paths are not edge-disjoint, lemma 3.12, and the advantage of pipelining is limited.

4.1.2. The Spanning Binomial Tree

With *one-port* communication we choose a scheduling discipline servicing subtrees in order of decreasing height. Since the binomial tree is composed of two $n-1$ level binomial trees the broadcasting operation after the source has communicated all its data to the largest subtree is reduced to the broadcasting of data in two same size, disjoint, subtrees. The process is repeated n times and the complexity is $T = (Mt_c + \lceil \frac{M}{B} \rceil \tau)n$. Clearly $B_{opt} = M$ and $T_{min} = (Mt_c + \tau)n$. The data transfer time is independent of the packet size, but the number of start-ups decreases.

With *n-port* communication pipelining can be employed extensively. The propagation time to the node furthest away from the source is at least $(Bt_c + \tau)n$. When this node has received all packets the broadcasting is terminated. Hence, $T = (M + (n-1)B)t_c + (n + \lceil \frac{M}{B} \rceil - 1)\tau$, and $B_{opt} = \sqrt{\frac{M\tau}{(n-1)t_c}}$, and $T_{min} = (\sqrt{Mt_c} + \sqrt{(n-1)\tau})^2$.

4.1.3. The Two-rooted Complete Binary Tree

With *one-port* communication the TCBT (Two-rooted Complete Binary Tree) routing the root can send one packet every three cycles. The furthest node will receive the first packet $2(n-1)$ cycles after initialization. For the internal nodes, three cycles are required to receive a packet from the parent and propagate it to both children nodes. The total time $T = (3\lceil \frac{M}{B} \rceil + 2n - 5)\tau + (3M + (2n-5)B)t_c$; B_{opt}, is $\sqrt{\frac{3M\tau}{(2n-5)t_c}}$ and $T_{min} = (\sqrt{3Mt_c} + \sqrt{(2n-5)\tau})^2$. If each node can support one send *and* one receive operation concurrently on distinct ports then the propagation time is still $2(n-1)$, but the source node can send out a new packet every two cycles. Hence, $T = (2\lceil \frac{M}{B} \rceil + 2n - 4)\tau + (2M + (2n-4)B)t_c$, $B_{opt} = \sqrt{\frac{M\tau}{(n-2)t_c}}$, and $T_{min} = (\sqrt{2Mt_c} + \sqrt{(2n-4)\tau})^2$.

With *n-port* communication the complexity of broadcasting is the same as that of the SBT.

4.1.4. n Rotated Spanning Binomial Trees

Broadcasting based on nRSBT and nESBT routings is performed by splitting the data into n parts with each part transmitted through one spanning tree. The difference between nRSBT and nESBT is that the rotated SBTs in an nRSBT are not edge-disjoint. The weight of each of the edges from the source is 1. Hence, the minimum transmission time for *one-port* communication is at least nMt_c, the same as for the SBT routing. Except in the case $M = 1$ this time is higher than the lower bound. The minimum number of start-ups is $2n-1$. Label the SBT rotated j steps by j. Then, the scheduling of data for the j^{th} SBT is initialized during cycle j. The data for different SBTs routed over the same edge and scheduled during the same cycle are combined into one packet. The packet size increases linearly from $\frac{1}{n}M$ to M, then decreases again to $\frac{1}{n}M$ for the last routing cycle; $T = nMt_c + (2\sum_{i=1}^{n-1}\lceil \frac{Mi}{nB} \rceil + \lceil \frac{M}{B} \rceil)\tau$. For $B \geq M$ the number of start-ups is minimized.

With *n-port* communication the scheduling of data for each SBT used in the composition of the nRSBT is made as in the case of *one-port* communication for a single SBT. Since the SBTs are

rotated, all ports of the root are used in every routing cycle until the last packet leaves the root. Data for the j^{th} SBT is sent across dimension $(j+i)$ mod n during cycle i, $0 \leq i, j \leq n-1$. There is no edge-conflict for the non-pipelined routing. Figure 12 shows the routings of the three distinctly rotated SBTs in a 3-cube. The labels on the edges represent the routing cycle. The communication complexity is the same as that of a single SBT with data set $\frac{1}{n}M$, i.e., $T = Mt_c + \lceil \frac{M}{nB} \rceil n\tau$, which yields $T_{min} \doteq Mt_c + n\tau$ with $B_m \geq \frac{M}{n}$, assuming M mod $n = 0$.

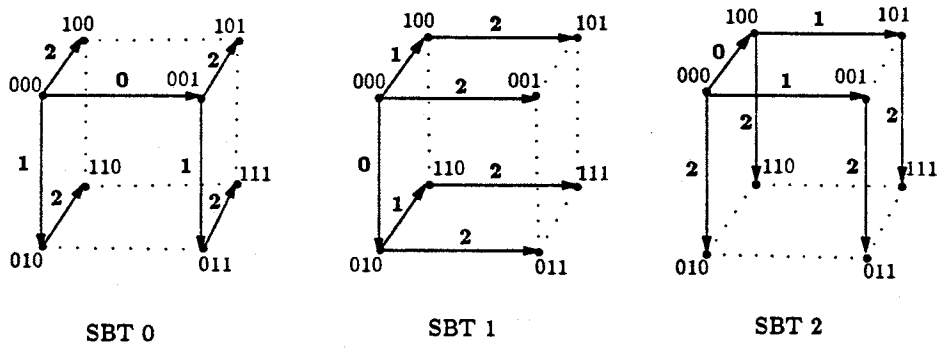

Figure 12: Broadcasting based on 3 rotated SBTs in a 3-cube.

4.1.5. n Edge-disjoint Spanning Binomial Trees

The minimum number of routing cycles to broadcast n packets for any routing algorithm and scheduling discipline is at least $2n-1$ for *one-port* communication and n for *n-port* communication. For the nESBT the bounds are $2n$ and $n+1$ for *one-port* and *n-port* communications respectively. Note that for *n-port* communication if the bandwidth is fully utilized at the source by sending out n packets per cycle, then each node receives n packets each cycle, when the pipe is full. Hence, even the node at distance n from the source must send out packets every cycle. Consequently, any routing/scheduling that fully utilizes the bandwidth at the source node must use a spanning graph with diameter at least $n+1$, unless $\frac{\tau}{t_c} \rightarrow \infty$.

To realize the lower bound for *one-port* communication it is required that a scheduling discipline be found that allows concurrent communication within all subtrees without violating the constraint on communication. We describe such a scheduling discipline in terms of labeling the nESBT graph with the least label being 0. A valid labeling for *one-port* communication, that allows pipelining every n cycles, requires that the following conditions be satisfied:

1. For any node of each subtree the least label on the output edges is greater than the label on the input edge.
2. For any cube node the labels on its input edges are distinct modulo n. (If there is more than one packet per subtree, then the root can send out a new packet to every subtree every n cycles.)
3. For any cube node the labels on the output edges are distinct modulo n.
4. Identical input and output labels (mod$n$) must be on edges between the same pair of processors.

Property 4 guarantees that each node sends and receives (if both exist) through the same port during the same cycle. Let $i = (a_{n-1}a_{n-2}...a_0)$ and $f(i,j)$ be the label of the input edge of node i in the j^{th} subtree for an nESBT graph with source node s. Let $c = i \oplus s$, $c_k = 1$ and $c_m = 0, \forall m \in \mathcal{M}_{nESBT}(i \oplus s, j)$. If $c = 0$ then $k = -1$. Then

Theorem 4.1. *For the nESBT graph the scheduling discipline defined by the following labeling*

$$f(i,j) = \begin{cases} \phi, & \text{if } k = -1; \\ j+n, & \text{if } c_j = 0, k \neq -1; \\ k, & \text{if } c_j = 1, k \geq j; \\ k+n, & \text{if } c_j = 1, k < j \end{cases}$$

allows conflict free communication for one-port communication.

Proof. Property 1 can be proved by deriving the labels of output edges from the definitions of the function $children_{nESBT}$ and $f(i,j)$. Consider a node with $c_j = 1$ and $k \geq j$. The labels of the output edges are $\{k+1, k+2, ..., n-1, n, ..., n+j\}$ (excluding $n+j$ if $k = j$). For $c_j = 1$ and $k < j$ the labels of the output edges are $\{k+1+n, k+2+n, ..., j-1+n\}$. For $c_j = 0, k \neq -1$, there is no child. The uniqueness of the input labels to a node is proved by the same arguments as in the proof of theorem 3.1. From the definition of the children function it is readily seen that for a given subtree, say the j^{th} subtree, the outgoing edges are labeled $\{(k+1) \bmod n, (k+2) \bmod n, \ldots, (j-1) \bmod n, j\}$, (excluding j if $k = j$). Property 3 is established by noticing that each bit equal to 1 in the binary encoding of i partitions the outgoing edges into distinct sets with each set corresponding to a subtree, and for each subtree the edges are labeled corresponding to the bit position. From the labeling scheme, the largest label of all the input edges is $2n-1$, i.e., broadcasting the first n packets (one packet per subtree) can be done in $2n$ cycles. For pipelining additional packets, we notice that for each node the labels of the input edges are distinct modulo n. So are the labels of the output edges. Property 4 follows from the fact that edges in dimension j are labeled $j (\bmod n)$.

∎

For *n-port* communication it is easy to determine the time of arrival of messages. Let $c = i \oplus s$ for an arbitrary node i and source node s, $i \neq s$. Then the path length between nodes s and i in the j^{th} spanning tree is equal to

$$\begin{cases} |c|, & \text{if } c_j = 1; \\ |c|+2, & \text{if } c_j = 0. \end{cases}$$

The input ports of node i that correspond to bits that are equal to 1 in the binary encoding of c receives the first element during the $|c|^{th}$ routing cycle. The other input ports receive the first element during cycle $(|c|+2)^{th}$.

Theorem 4.2. *The complexity of nESBT broadcasting of M elements and one-port communication is at most $(M+nB)t_c + (\lceil \frac{M}{B} \rceil + n)\tau$; $T_{min} = (\sqrt{Mt_c} + \sqrt{n\tau})^2$.*

Proof. The number of cycles follows from the definition and proof of the scheduling discipline.

∎

Figure 13 shows an nESBT graph for a 3-cube labeled by the algorithm above.

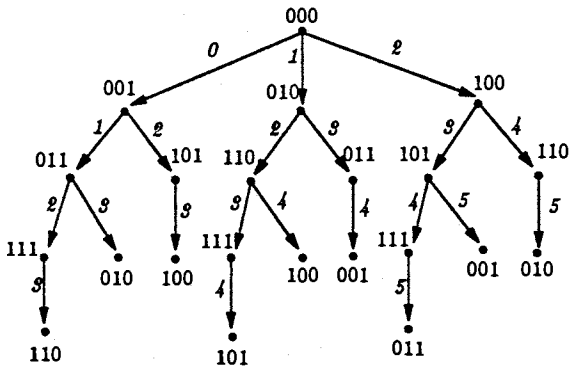

Figure 13: Scheduling in an nESBT graph with ***one-port*** communication.

Algorithm	*one-port*	*n-port*
BRG	$N-1$	$N-1$
SBT	n	n
nRSBT	n	n
TCBT	$2n-2$	n
nESBT	$n+1$	$n+1$

Table 2: Propagation delays.

For communication restricted to one send *or* one receive operation per node, each cycle defined previously is made into two cycles. Notice that in the previous routing algorithm, all the communication links are only used in one direction during the first n routing cycles, and the last routing cycle.

Theorem 4.3. *The nESBT graph allows M elements to be broadcast in $2\lceil\frac{M}{B}\rceil+n-1$ routing cycles under the constraint of at most one receive or one send operation during each cycle.*

Proof. The number of routing cycles can be derived by subtracting $n+1$ from twice the number of routing cycles given in theorem 4.2. ∎

With ***one-port*** communication $T=(\lceil\frac{M}{B}\rceil+n)\tau+(M+nB)t_c$; $B_{opt}=\sqrt{\frac{M\tau}{nt_c}}$. $T_{min}=(\sqrt{Mt_c}+\sqrt{n\tau})^2$. Restricting the communication to one send *or* one receive operation at a time, $T=(2\lceil\frac{M}{B}\rceil+n-1)\tau+(2M+(n-1)B)t_c$ and $T_{min}=(\sqrt{2Mt_c}+\sqrt{(n-1)\tau})^2$. For ***n-port*** communication $T=(\lceil\frac{M}{Bn}\rceil+n)\tau+(\frac{M}{n}+nB)t_c$; $T_{min}=(\sqrt{\frac{Mt_c}{n}}+\sqrt{n\tau})^2$ and $B_{opt}=\frac{1}{n}\sqrt{\frac{M\tau}{t_c}}$.

4.2. Comparison and Conclusion

There are two factors that determine the communication complexity of broadcasting with pipelining. One factor is how often the source node can send out a new packet (containing new data). The other is the minimum time for a message to propagate to every node. The algorithms based on the SBT, the nRSBT, the nESBT, and the TCBT routings have propagation times in the range of n to $2n$, Table 2.

Algorithm	*one-port*	*n-port*
BRG	2	1
SBT	n	1
nRSBT	n	1
TCBT	3	1
nESBT	1	$\frac{1}{n}$

Table 3: Number of cycles per distinct packet.

For ***one-port*** communication the root can only send out a packet with new data every n cycles in the SBT routing. In the TCBT routing the root can send out one new packet every three cycles, while the nESBT routing allows one new packet every cycle. In the nRSBT routing the source needs n cycles for each packet, but can start a new packet every cycle if data routed through different SBTs are combined into a larger packet. For a fixed packet size, one distinct packet is sent out every n cycles, on the average. This assumption is made for Table 3, which concerns the root. Note that broadcasting through a Hamiltonian path on an n-cube may be faster than broadcasting based on the SBT or even the TCBT routing depending on the values of M, t_c, τ and N. In a Hamiltonian path a new packet can be sent every other cycle. The number of start-ups is $N-1$.

With ***n-port*** communication the source can send out n distinct packets every cycle in the nESBT routing, while the SBT and TCBT routing only allows the sending of one distinct packet every cycle. In the case of n distinctly rotated Gray code sequences the source can send out n distinct packets per cycle, but pipelining is limited. The situation is the same for the nRSBT routing, in which n packets can be sent out during cycle 0, n , $2n$,..., etc. The average number of cycles per distinct packet is 1. Table 3 compares the number of cycles per distinct packet for various routings. Some variations exist like using two Hamiltonian paths with opposite directions sending distinct data, or using one Hamiltonian path with the source node at the center of the path. However, these variations only affect (either increase or decrease) the delays and the number of cycles per packet by at most a factor of two. The complexity estimates are summarized in Tables 4 and 5.

For ***n-port*** communication the number of sequential start-ups and the bandwidth requirement is reduced by a factor of approximately n for an arbitrary packet size in nRBRG, SBT, nRSBT, and nESBT routing. TCBT routing does not fully utilize the bandwidth of a cube. The reduction in communication complexity for *n-port* communication is a factor of 3. Optimizing the packet size for each situation brings the number of start-ups to $O(n)$ (or $O(N)$ for BRG) both for ***one-port*** or ***n-port*** communication. With *n-port* communication the bandwidth requirement is reduced by a factor of n for the nRBRG, SBT, nRSBT, and the nESBT routings, also when packet sizes are optimized. However, the bandwidth requirement of the TCBT routing is reduced by a factor of 3.

The nESBT routing always offers a reduction in bandwidth requirement for individual communication links by a factor of approximately n over SBT and nRSBT routings. With *n-port* communication the nESBT routing has a communication complexity that is lower than that of TCBT routing by a factor of n. Even with *one-port* communication the nESBT routing still is faster than TCBT routing by a factor of 3. For an arbitrary packet size the nESBT routing also offers a reduction in the number of start-ups. The reduction may be of order $O(n)$. The nESBT routing offers a speed-up of up to n over SBT and nRSBT routings for sufficiently large values of

Algorithm	T	B_{opt}	T_{min}
BRG	$(2\lceil\frac{M}{B}\rceil + N - 3)\tau + (2M + (N-3)B)t_C$	$\sqrt{\frac{2M\tau}{(N-3)t_c}}$	$(\sqrt{2Mt_c} + \sqrt{(N-3)\tau})^2$
SBT	$\lceil\frac{M}{B}\rceil n\tau + Mnt_c$	M	$n(Mt_c + \tau)$
nRSBT	$nMt_c + (2\sum_{i=1}^{n-1}\lceil\frac{Mi}{nB}\rceil + \lceil\frac{M}{B}\rceil)\tau$	M	$nMt_c + (2n-1)\tau$
TCBT	$(3\lceil\frac{M}{B}\rceil + 2n - 5)\tau + (3M + (2n-5)B)t_c$	$\sqrt{\frac{3M\tau}{(2n-5)t_c}}$	$(\sqrt{3Mt_c} + \sqrt{(2n-5)\tau})^2$
nESBT	$(\lceil\frac{M}{B}\rceil + n)\tau + (M + nB)t_c$	$\sqrt{\frac{M\tau}{nt_c}}$	$(\sqrt{Mt_c} + \sqrt{n\tau})^2$

Table 4: The complexity of *one-port* one-to-all broadcasting.

Algorithm	T	B_{opt}	T_{min}
nRBRG	$\lceil\frac{M}{nB}\rceil(N-1)\tau + \frac{M(N-1)}{n}t_c$	$\frac{M}{n}$	$(\frac{M}{n}t_c + \tau)(N-1)$
SBT	$(\lceil\frac{M}{B}\rceil + n - 1)\tau + (M + (n-1)B)t_c$	$\sqrt{\frac{M\tau}{(n-1)t_c}}$	$(\sqrt{Mt_c} + \sqrt{(n-1)\tau})^2$
nRSBT	$Mt_c + \lceil\frac{M}{nB}\rceil n\tau$	$\frac{M}{n}$	$Mt_c + n\tau$
TCBT	$(\lceil\frac{M}{B}\rceil + n - 1)\tau + (M + (n-1)B)t_c$	$\sqrt{\frac{M\tau}{(n-1)t_c}}$	$(\sqrt{Mt_c} + \sqrt{(n-1)\tau})^2$
nESBT	$(\lceil\frac{M}{Bn}\rceil + n)\tau + (\frac{M}{n} + nB)t_c$	$\frac{1}{n}\sqrt{\frac{M\tau}{t_c}}$	$(\sqrt{\frac{Mt_c}{n}} + \sqrt{n\tau})^2$

Table 5: The complexity of *n-port* one-to-all broadcasting.

Communication Assumption	Algorithm	one packet	$\frac{M}{B} \gg n$	$B = B_{opt}$, $\tau n \gg Mt_c$	$B = B_{opt}$, $\tau n \ll Mt_c$
one-port	SBT,nRSBT/nESBT	$\frac{n}{n+1}$	n	1	n
one-port	TCBT/nESBT	$\frac{2n-2}{n+1}$	3	2	3
n-port	SBT,nRSBT,TCBT/nESBT	$\frac{n}{n+1}$	n	1	n

Table 6: The relative complexity of routings for one-to-all broadcasting compared to the nESBT routing.

M, both for *one-port* and *n-port* communication. The communication complexities of broadcasting based on the SBT, nRSBT, and the TCBT routings are compared with that based on the nESBT in Table 6. Notice that the entry for the last column and the last row in the table is based on the assumption that $B = B_{opt}, \tau n^2 \ll Mt_c$.

5. One-to-All Personalized Communication

In personalized communication no replication of information takes place during distribution, nor is there any reduction during the reverse operation. In broadcasting the bandwidth requirement of a cut at a distance i from the source node grows precisely as the number of nodes grows. In personalized communication the source node has to distribute $(N-1)M$ elements. The bandwidth requirement for a cut at distance i from the source decreases in proportion to the number of nodes at and within distance i from the source. The total bandwidth requirement for the cube is at least $\sum_{i=1}^{n} M \times i \times$ (number of nodes at distance i from the root), which is $\frac{n}{2}NM$ compared to $(N-1)M$ for one-to-all broadcasting. The root is the "bottleneck". A lower bound for one-to-all personalized communication is $(N-1)Mt_c + n\tau$ for *one-port* communication, and $\frac{(N-1)M}{n}t_c + n\tau$ for *n-port* communication. The minimum of the maximum number of elements transmitted across

an edge, the *edge load*, is $\frac{(N-1)M}{n}$. Clearly, a necessary condition for lower bound personalized communication is that the maximum edge load is minimized. Two strategies can be applied to reach this goal:

- Partition the node set into n equal sets.
- Provide n paths from the source to every node, and partition the data set M into n equal subsets.

In SBT and TCBT routing for personalized communication the maximum edge load is $\frac{1}{2}NM$. The bandwidth of the cube is not used effectively, when *n-port* communication is possible. In a SBnT all subtrees of the root have the same edge load, i.e., $\frac{(N-1)M}{n}$, compared to $\frac{NM}{2}$ for a complete binary tree. The SBnT graph achieves a minimax edge load through partitioning of the node set, and, for cyclic nodes, also the data set.

The nRSBT graph and the nESBT graphs both provides n paths from the source to every other node. The maximum edge load is minimized for both graphs, but the height of the nESBT graph is $n+1$. We do not analyze the complexity of personalized communication based on the nESBT graph.

5.1. Complexity of Personalized Communication

For *one-port*, personalized communication, we consider a scheduling discipline using postorder traversal in order of decreasing subtrees, and maximizing the minimum time between communications on the same port. In the particular case of a packet size $B = M$ and a spanning binomial tree, this scheduling discipline leads to communication on ports in a *binary-reflected* Gray code order [15, 9]. We assume the same scheduling discipline for every node. For *n-port* communication we consider *reverse-breadth-first* scheduling.

With *one-port* communication the number of start-ups for the *reverse-breadth-first* scheduling discipline is equal to the sum of the number of levels in the subtrees of the source node, if the fan-out of any node is at most equal to the fan-out of its parent node. The number of start-ups is at least of order $O(n^2)$ for the spanning graphs considered here. In the postorder scheduling discipline the number of start-ups may be as low as n, but not higher than $2n$ for spanning graphs of height n. The data transmission time may in either scheduling discipline be limited by the root.

5.1.1. The Hamiltonian Path

One-port personalized communication along a BRG path requires a time $T = (2N-3)Mt_c + \max(2\lceil\frac{(N-1)M}{B}\rceil - \lfloor\frac{M}{B}\rfloor, 2N-3)\tau$. With one send operation concurrent with a receive operation on a different port the complexity is reduced to $T = (N-1)Mt_c + \max(\lceil\frac{(N-1)M}{B}\rceil, N-1)\tau$.

Both routings assume pipelining. With *n-port* communication and nRBRG routing, pipelining is limited due to edge-conflicts, lemma 3.12. With no pipelining $T = \frac{N(N-1)M}{2n}t_c + \sum_{i=1}^{N-1}\lceil\frac{iM}{nB}\rceil\tau$, which is greater than routing according to one BRG path.

5.1.2. The Spanning Binomial Tree

One-port personalized communication based on SBT routing and the *postorder* scheduling discipline with a packet size of B yields a complexity $T \simeq (N-1)Mt_c + \tau(\frac{NM}{B} + \log\lceil\frac{B}{M}\rceil - 1)$, $M \leq B \leq \frac{NM}{2}$, which is minimized for $B_{opt} = \frac{NM}{2}$ with $T_{min} = (N-1)Mt_c + n\tau$. This complexity is the same as that of the lower bound for *one-port* personalized communication. For $B \leq M$, $T \simeq (N-1)Mt_c + \lceil\frac{(N-1)M}{B}\rceil\tau$. This complexity is the same as that of communication in

a Hamiltonian path. We conclude that for personalized communication, SBT routing and *one-port* communication the complexity is equal to the lower bound if the maximum packet size is sufficiently large, but may be equal to that of a Hamiltonian path if $B \leq M$.

With *n-port* communication a reduction in the transfer time by a factor of 2 is possible compared to *one-port* communication by use of pipelining. From the minimax edge load it is clear that the reduction in the data transfer time can be at most 2.

Lemma 5.1. *With reverse-breadth-first scheduling and n-port communication the time for personalized communication is limited by the root. The complexity is* $T = \frac{1}{2}NMt_c + \sum_{i=0}^{n-1}\lceil\binom{n-1}{i}\frac{M}{B}\rceil\tau$ *with a minimum of* $T = \frac{1}{2}NMt_c + n\tau$ *for a packet size* $B \geq max\binom{n-1}{i} \approx \frac{NM}{\sqrt{2\pi(n-1)}}$.

Proof. Follows immediately from lemmas 3.4 and 3.16. ∎

We conclude that for SBT routing the packet size is of greater importance than concurrent communication on all ports.

5.1.3. The Two-rooted Complete Binary Tree

With *one-port* communication and TCBT routing for personalized communication the minimum number of start-ups is $2n-2$. The minimum transfer time is $(2N-2n-1)Mt_c$ with the minimum number of start-ups. With additional start-ups, the transfer time can be reduced. However, the transfer time is at least of $O(N)$. For *n-port* communication the minimum number of start-ups is n, and the minimum transfer time with the minimum number of start-ups is $(\frac{3}{4}N-1)Mt_c$, which can be achieved by the *reverse-breadth-first* scheduling. The total time $T = (\frac{3}{4}N-1)Mt_c + n\tau$. In comparison with the SBT routing, the TCBT routing complexity is always higher. The slow-down is approximately 2.

5.1.4. n Rotated Spanning Binomial Trees

The subtrees of the source node are all of equal height in the nRSBT graph. With *one-port* communication the number of start-ups is $2n-1$ and the data transmission time is $M(N-1)t_c$. This data transmission time is the same as that of the SBT routing. The data transfer time is minimized for the nRSBT routing with the scheduling for the j^{th} SBT starting during cycle j. The data for different SBTs routed over the sane edge and scheduled during the same cycle are combined into one packet. The packet size during cycle i is $\frac{NM}{n}(1-\frac{1}{2^{i+1}})$, for $0 \leq i \leq n-1$, and $\frac{NM}{n}(\frac{1}{2^{i+1-n}}-\frac{1}{N})$, for $n \leq i \leq 2n-2$. The data set that needs to be transmitted to each subtree is $\frac{M}{n}(N-1)$. $T = M(N-1)t_c + (\sum_{i=1}^{n-1}\lceil\frac{M(N-2^i)}{nB}\rceil + \sum_{i=1}^{n}\lceil\frac{M(2^i-1)}{nB}\rceil)\tau$. For *n-port* communication the time for a *reverse-breadth-first* scheduling discipline is $T = \frac{(N-1)M}{n}t_c + \sum_{i=0}^{n-1}\lceil\binom{n}{i}\frac{M}{nB}\rceil\tau$. For $B = \frac{M}{n}$, $T = (\frac{M}{n}t_c+\tau)(N-1)$, a reduction by a factor of n compared to *one-port* communication. For $B \geq \frac{M}{n}\binom{n}{\frac{n}{2}} \simeq \sqrt{\frac{2}{\pi}}\frac{NM}{n^{3/2}}$, $T = \frac{M}{n}(N-1)t_c + n\tau$, a reduction in the data transmission time by a factor of n. This complexity is indeed the same as the lower bound.

The above results are strictly only true for the case when n divides M. With $(M \bmod n) = k \neq 0$ a combination of *reflections* and *rotations* minimizes the maximum edge load. For k even $k/2$ distinct rotations should be used, and for every rotated graph a reflected graph is also used. For k even and optimally rotated SBTs the maximum edge load is $(N-1)\frac{2^{\frac{n}{k}}-1}{2^{\frac{n}{k}}-1}$ and for optimally

reflected and rotated SBTs the maximum edge load is $(N-1)\frac{2^{\frac{2n}{k}-1}+1}{2^{\frac{2n}{k}}-1}$.

5.1.5. The Spanning Balanced n-Tree

With *one-port* personalized communication and SBnT routing the root can send data to the subtrees cyclically. With a maximum packet size $B > M$, data for several nodes can be merged into one packet. The receiving node has sufficient time to retransmit pieces on all its ports, should that be required, since a new packet only arrives every n cycles. The root requires a time of $T \simeq \frac{(N-1)M}{nB}(Bt_c+\tau)n$. For $B = M, T = (N-1)(Mt_c+\tau)$, i.e., the same as in the SBT and Hamiltonian path routing. For $B \geq \frac{(N-1)M}{n}$ the root of the SBnT need only perform one communication per subtree, and it completes the communication in a time of $T = (N-1)Mt_c + n\tau$. But, unlike in the SBT routing the communication is not terminated when the root is done. The message to the last visited subtree needs to traverse $n-1$ communication links. The bandwidth requirement of each subtree can be shown to be $\approx \frac{2N\log n}{n}$ [17]. An upper bound on the time for personalized communication based on the SBnT with unbounded packet size is $T = N(1+\frac{2\log n}{n})Mt_c+(2n-1)\tau$. The number of start-ups is almost twice that of the SBT personalized communication, and the total transfer time is higher by a lower order term. The time for personalized communication based on the SBnT routing is minimized for $B \geq \frac{(N-1)M}{n}$. With a maximum packet size of $\frac{(N-1)M}{n}$, the number of start-ups for the SBT routing is approximately also $2n$. We conclude that for *one-port* personalized communication the SBT routing yields a lower complexity than the SBnT routing. However, in the case of *n-port* communication the SBnT routing does offer a lower complexity.

With *n-port* communication the time for personalized communication with SBnT routing is $T = \frac{(N-1)M}{n}t_c+\lceil\frac{(N-1)M}{nB}\rceil\tau$. If $B = M$ then $T = \frac{(N-1)M}{n}t_c+\lceil\frac{N-1}{n}\rceil\tau$, a reduction in the number of start-ups and the transfer time by a factor of $\frac{1}{2}n$ over the SBT routing for personalized communication. The communication time is minimized for $B \geq \sqrt{\frac{2}{\pi}}\frac{N}{n^{3/2}}M$ and $T_{min} = \frac{N-1}{n}Mt_c+n\tau$, the minimum possible. This complexity estimate is true by lemmas 3.4 and 3.27, which guarantees that the data transfer time is dominated by the root. By merging data for different levels of the SBnT the requirement on the maximum packet size can be reduced. We conclude that in the case of *n-port* communication the SBnT routing is always of a lower complexity than the SBT routing.

The nRSBT and SBnT graphs yield the same communication complexity and the same maximum packet size, $\sqrt{\frac{2}{\pi}}\frac{NM}{n^{3/2}}$. In the case $M \bmod n \neq 0$, the SBnT graph is always superior.

5.2. Comparison and Conclusion

Tables 7, 8 and 9 show the communication complexities of personalized communication based on various spanning graphs. With *one-port* communication and a maximum packet size $B \leq M$ the complexity of SBT, TCBT, nRSBT, and SBnT and Hamiltonian path routing is the same. For $B > M$, the SBT routing yields a lower complexity than the Hamiltonian path, the nRSBT, the SBnT, and the TCBT routing. For sufficiently large maximum packet size the SBT routing has n start-ups compared to $2n-1$ for the nRSBT and SBnT routings, and $2n-2$ start-ups for the TCBT routing. The transmission times are comparable, though the transmission time for the SBnT and TCBT routings are higher. Note that as in broadcasting the minimum number of start-ups can be accomplished for sufficiently large packet size.

With *n-port* communication the number of start-ups and the transmission time of SBnT and nRSBT routing is lower than that of the SBT routing by a factor of $\frac{1}{2}n$, and is lower than that for

Algorithm	T
BRG, *one-port*	$(2N-3)Mt_c + \max(2\lceil\frac{(N-1)M}{B}\rceil - \lfloor\frac{M}{B}\rfloor, 2N-3)\tau$
SBT, *one-port*	$(N-1)Mt_c + \sum_{i=0}^{n-1}\lceil\frac{2^iM}{B}\rceil\tau$
nRSBT, *one-port*	$M(N-1)t_c + (\sum_{i=1}^{n-1}\lceil\frac{M(N-2^i)}{nB}\rceil + \sum_{i=1}^{n}\lceil\frac{M(2^i-1)}{nB}\rceil)\tau$
TCBT, *one-port*	$\approx (2N-2n-1)Mt_c + 2\sum_{i=1}^{n-1}\lceil\frac{(2^i-1)M}{B}\rceil\tau$
SBnT, *one-port*	$\approx N(1+\frac{2\log n}{n})Mt_c + (n\lceil\frac{NM}{nB}\rceil + \lceil\frac{2N\log nM}{nB}\rceil)\tau$
nRBRG, *n-port*	$\frac{N(N-1)M}{2n}t_c + \sum_{i=1}^{N-1}\lceil\frac{iM}{nB}\rceil\tau$
SBT, *n-port*	$\frac{NM}{2}t_c + \sum_{i=0}^{n-1}\lceil\binom{n-1}{i}\frac{M}{B}\rceil\tau$
nRSBT, *n-port*	$\frac{(N-1)M}{n}t_c + \sum_{i=0}^{n-1}\lceil\binom{n}{i}\frac{M}{nB}\rceil\tau$
TCBT, *n-port*	$\approx (\frac{3N}{4}-1)Mt_c + (\lceil\frac{NM}{4B}\rceil + \sum_{i=2}^{n}\lceil\frac{NM}{2^iB}\rceil)\tau$
SBnT, *n-port*	$\frac{(N-1)M}{n}t_c + \sum_{i=0}^{n-1}\lceil\binom{n}{i}\frac{M}{nB}\rceil\tau$

Table 7: The complexity of one-to-all personalized communication.

Algorithm	B_{opt}	T_{min}
BRG	M	$(2N-3)Mt_c + (2N-3)\tau$
SBT	$\frac{NM}{2}$	$(N-1)Mt_c + n\tau$
TCBT	$\frac{NM}{2}$	$\leq (2N-2n-1)Mt_c + (2n-2)\tau$
nRSBT	$\frac{(N-1)M}{n}$	$(N-1)Mt_c + (2n-1)\tau$
SBnT	$\frac{(N-1)M}{n}$	$\leq N(1+\frac{2\log n}{n})Mt_c + (2n-2)\tau$

Table 8: The optimum complexity of *one-port* one-to-all personalized communication.

Algorithm	B_{opt}	T_{min}
nRBRG	$\frac{(N-1)M}{n}$	$\frac{N(N-1)M}{2n}t_c + (N-1)\tau$
SBT	$\frac{NM}{\sqrt{2\pi(n-1)}}$	$\frac{NM}{2}t_c + n\tau$
TCBT	$\frac{NM}{4}$	$(\frac{3}{4}N-1)Mt_c + n\tau$
nRSBT	$\sqrt{\frac{2}{\pi}\frac{NM}{n^{3/2}}}$	$\frac{(N-1)M}{n}t_c + n\tau$
SBnT	$\sqrt{\frac{2}{\pi}\frac{NM}{n^{3/2}}}$	$\frac{(N-1)M}{n}t_c + n\tau$

Table 9: The optimum complexity of *n-port* one-to-all personalized communication.

the TCBT routing by a factor of $\frac{3}{4}n$ for a maximum packet size $B \leq M$. With a sufficiently large packet size all routings yield a minimum of n start-ups. The SBnT and nRSBT routings have a total transmission time that is lower than that of SBT routing by a factor of $\frac{1}{2}n$, and lower than that of TCBT routing by a factor of $\frac{3}{4}n$. Moreover, it is achieved at a maximum packet size of $\sqrt{\frac{2}{\pi}\frac{NM}{n^{3/2}}}$, compared to a maximum packet size of $\frac{NM}{\sqrt{2\pi(n-1)}}$ for the SBT routing. We conclude that for *n-port* communication, the communication complexity of the SBnT and nRSBT routings may be lower by a factor of $\frac{1}{2}n$ compared to the SBT routing and by a factor of $\frac{3}{4}n$ compared to the TCBT routing. The nRSBT routing is never of a lower complexity than the SBnT routing, and of a higher complexity if n does not divide M.

6. All-to-All Broadcasting

Theorem 6.1. *The lower bound for all-to-all broadcasting is* $(N-1)Mt_c + n\tau$ *for one-port communication and* $\frac{(N-1)M}{n}t_c + n\tau$ *for n-port communication.*

Proof. Each node receives M elements from every other node, i.e., each node receives $(N-1)M$ elements. Hence, for *one-port* communication a lower bound for the data transfer time is $(N-1)Mt_c$, and with *n-port* communication the time is bounded by $\frac{(N-1)M}{n}t_c$.

∎

Theorem 6.2. *The communication time for all-to-all broadcasting based on N translated spanning graphs of height h and n-port communication requires a time of at most*

$$T = \sum_{l=0}^{h-1}(\lceil \frac{M}{B} \times \max_{0\le d\le n-1} ew_{ld} \rceil \tau + Mt_c \times \max_{0\le d\le n-1} ew_{ld}).$$

If $B \ge \max_{0\le l\le h-1,\ 0\le d\le n-1}(M \times ew_{ld})$ *then*

$$T = (\sum_{l=0}^{h-1} \max_{0\le d\le n-1} ew_{ld})Mt_c + h\tau.$$

Proof. Each node broadcasts its data set M according to its own spanning graph. During the l^{th} routing cycle all nodes at level $l+1$ of each spanning graph receives messages sent out from the roots during the 0^{th} routing cycle. The number of messages contending for a communications link in dimension d between levels l and $l+1$ during the l^{th} cycle is ewL_{ld}^N, i.e., the sum of weights of spanning graph edges mapped to a cube edge. By lemma 3.3 the number of elements contending for an edge is ew_{ld}.

∎

Theorem 6.1 gives a lower bound for communication time based on N spanning graphs. Theorem 6.2 gives an upper bound. In the following we give complexity estimates for some spanning graphs, and show that all-to-all broadcasting based on the SBT graph yields a lower bound algorithm for *one-port* communication, and that the SBnT and nRSBT graphs yield a lower bound algorithm for *n-port* communication.

Corollary 6.1. *For a given spanning graph satisfying* $ew_{li} = ew_{lj}$, $0 \le i,j < n$, *and* $0 \le l < h$, *the communication time for all-to-all broadcasting based on N translated spanning graphs of height h and n-port communication is*

$$T = \frac{(N-1)M}{n}t_c + h\tau, \text{ if } B_m \ge \max_{0\le l<h}(M \times ew_{ld}).$$

Proof. It follows from theorem 6.2.

∎

From Table 1 and this corollary, the nRBRG, nRSBT, nESBT and the SBnT routings all yield the lower bound for the data transfer time with *n-port* communication. The nRSBT and

the SBnT both with minimum height also attain the lower bound time, theorem 6.1. In fact, any spanning graph composed of n distinctly rotated spanning trees of minimum height attains the lower bound. The greedy spanning graph, while it is a necessary condition for the all-to-all personalized communication, is not necessary for the all-to-all broadcasting to attain the lower bound.

Corollary 6.2. *For N translated spanning graphs each composed of n distinctly rotated greedy spanning trees the time for all-to-all broadcasting with n-port communication attains the lower bound.*

$$T_{min} = \frac{(N-1)M}{n} t_c + n\tau, \text{ if } B_m \geq \sqrt{\frac{2}{\pi}} \frac{NM}{n^{3/2}} \text{ approximately.}$$

Proof. It follows from corollary 6.1 and lemma 3.8.

∎

Lemma 6.1. *The data transfer time for one-port communication is minimized if one dimension is routed per cycle, and all nodes use the same scheduling discipline.*

Proof. The load on every edge in the routed dimension is the same, since the spanning graph for the different sources are translations of each other.

∎

For an optimum *one-port* all-to-all broadcasting it remains to minimize the number of start-ups, preserving the data transfer time. This can be accomplished through the labeling of one spanning graph. The rules are:

1. For each composed spanning tree, labels of the outgoing edges of any node are greater than the label of the incoming edge.
2. All the edges with the same label in the spanning graph are in the same dimension.

The label on the edge corresponds to the cycle during which the data is transferred. Rule 1 is obvious. Rule 2 is due to the *one-port* communication constraint. The number of start-ups required for the broadcasting is equal to the maximum label plus 1, if the least label is 0. For a spanning tree of height h, the minimax label is at least $h-1$ by rule 1. For the BRG path, we label the i^{th} edge in the path by $i-1$. The maximum label is then $N-2$, which is a minimax label because the BRG path is of height $N-1$. For the SBT labeling edges in dimension i by i satisfies both rules. The maximum label is $n-1$, which is also a minimax label. For the nESBT, nRSBT and SBnT, we first define the edge label in the 0^{th} spanning tree (or subtree for the SBnT). Edges in dimension i are labeled i, except the edges to the leaf nodes which are labeled $i+n$. The 0^{th} subtree of the nESBT is equal to an n level SBT, deleting the smallest subtree. Hence, the minimax label is n for ERSBT 0. The minimax label of the 0^{th} subtree of the SBnT can be shown to be $n-1$ from the definition of the SBnT. The j^{th} spanning tree (or subtree) is the left rotation of the i^{th} spanning tree (or subtree) by $(j-i)$ mod n steps, lemma 3.22 and theorem 3.4. The labels of the edges in subtree j are defined by adding j to the label of the corresponding edges in subtree 0. The minimax label for the entire graph is equal to the minimax label of subtree 0

BRG	SBT	nESBT	nRSBT	SBnT	TCBT
$N-1$	n	$2n$	$2n-1$	$2n-1$	$\geq 2n-2$

Table 10: The minimum number of start-ups in *one-port* all-to-all broadcasting.

Algorithm	T
BRG	$(N-1)Mt_c + \lceil \frac{M}{B} \rceil (N-1)\tau$
SBT	$(N-1)Mt_c + \sum_{i=0}^{n-1} \lceil \frac{2^i M}{B} \rceil \tau$
nRSBT	$(N-1)Mt_c + (\sum_{i=0}^{n-1} \lceil \frac{(2^{i+1}-1)M}{nB} \rceil + \sum_{i=n}^{2n-2} \lceil \frac{(N-2^{i-n+1})M}{nB} \rceil)\tau$
nESBT	$(N-1)Mt_c + (\sum_{i=0}^{n-1} \lceil \frac{2^i M}{nB} \rceil + \sum_{i=n}^{2n-1} \lceil \frac{(N-1-2^{i-n})M}{nB} \rceil)\tau$
SBnT	$\approx (N-1)Mt_c + \max(2n-1, \frac{(N-1)M}{B})\tau$

Table 11: The complexity of *one-port* all-to-all broadcasting.

Algorithm	B_{opt}	T_{min}
BRG	M	$(N-1)Mt_c + (N-1)\tau$
SBT	$\frac{NM}{2}$	$(N-1)Mt_c + n\tau$
nRSBT	$\frac{(N-1)M}{n}$	$(N-1)Mt_c + (2n-1)\tau$
nESBT	$\frac{(N-2)M}{n}$	$(N-1)Mt_c + 2n\tau$
SBnT	$\frac{(N-1)M}{n}$	$(N-1)Mt_c + (2n-1)\tau$

Table 12: The optimum complexity of *one-port* all-to-all broadcasting.

Algorithm	T
nRBRG	$\frac{(N-1)M}{n} t_c + \lceil \frac{M}{nB} \rceil (N-1)\tau$
SBT	$\frac{NM}{2} t_c + \sum_{i=0}^{n-1} \lceil \binom{n-1}{i} \frac{M}{B} \rceil \tau$
nRSBT	$\frac{(N-1)M}{n} t_c + \sum_{i=1}^{n} \lceil \binom{n}{i} \frac{M}{nB} \rceil \tau$
nESBT	$\frac{(N-1)M}{n} t_c + (\sum_{i=2}^{n} \lceil \binom{n}{i} \frac{M}{nB} \rceil + \lceil \frac{M}{nB} \rceil + \lceil \frac{(n-1)M}{nB} \rceil)\tau$
SBnT	$\frac{(N-1)M}{n} t_c + \sum_{i=1}^{n} \lceil \binom{n}{i} \frac{M}{nB} \rceil \tau$

Table 13: The complexity of *n-port* all-to-all broadcasting.

plus $n-1$. The minimax label of the nESBT, nRSBT and SBnT is $2n-1$, $2n-2$ and $2n-2$ respectively. The labeling of the nESBT graph is the same as the labeling for *one-port* one-to-all broadcasting, Figure 13. Table 10 lists the minimum number of start-ups for various spanning graphs. The amount of data transfer during cycle i is equal to $M\times$ (sum of ***weighted edges*** labeled i). The maximum packet size and the number of start-ups in terms of packet size can be derived easily for various spanning graphs given the labels of the edges in the spanning graph.

6.1. The Complexity of All-to-All Broadcasting

Tables 11, 12, 13 and 14 summarize the complexity of all-to-all broadcasting.

The *one-port* N BRG path routing is employed in the matrix multiplication algorithm by Dekel [6]. Messages with different source nodes are routed through different BRG paths. There are no communication conflicts because pipelining is not used, lemma 3.13. Messages are exchanged along a sequence of dimensions as 0, 1, 0, 2, 0, 1, 0, 3, ... etc. Sharing one Hamiltonian circuit yields the

Algorithm	B_{opt}	T_{min}
nRBRG	$\frac{M}{n}$	$\frac{(N-1)M}{n}t_c + (N-1)\tau$
SBT	$\frac{NM}{\sqrt{2\pi(n-1)}}$	$\frac{NM}{2}t_c + n\tau$
nRSBT	$\sqrt{\frac{2}{\pi}}\frac{NM}{n^{3/2}}$	$\frac{(N-1)M}{n}t_c + n\tau$
nESBT	$\sqrt{\frac{2}{\pi}}\frac{NM}{n^{3/2}}$	$\frac{(N-1)M}{n}t_c + (n+1)\tau$
SBnT	$\sqrt{\frac{2}{\pi}}\frac{NM}{n^{3/2}}$	$\frac{(N-1)M}{n}t_c + n\tau$

Table 14: The optimum complexity of ***n-port*** all-to-all broadcasting.

same complexity. With *n-port* communication and N nRBRG paths the complexity in Table 11 is derived from Table 1 and theorem 6.2.

With *one-port* communication for the SBT routing it can be completed in n routing cycles, as in the single source case. In the all-to-all case this routing scheme amounts to exchanges in the different dimensions [27]. Note that if the maximum packet size is smaller than the data set to be broadcast, then the SBT routing is of the same complexity as the BRG routing. But, if $B \geq \frac{1}{2}MN$ then $T_{min} = (N-1)Mt_c + n\tau$, which is the lower bound. The start-up time is reduced by a factor of $\frac{(N-1)}{n}$ compared to the BRG routing. With *n-port* communication the SBT algorithm is no longer optimal. The data transmission time is at most reduced by a factor of 2, and the maximum number of elements contending for communication by a factor of about $\sqrt{n}$.

The following pseudo code implements the N SBT routing for *n-port* communication.

```
for port 0 to port n − 1 do concurrently
    send out resident data (length M);
    receive the incoming data (length M);
enddo
for i:= 1 to n − 1 do
    for port j:= i to n − 1 do concurrently
        send out the data accumulated in buffers of inports
            {i − 1, i − 2, ..., j − 1} (length (j choose i));
        receive and append to each inport buffer;
    enddo
enddo
```

With *one-port* communication and TCBT routing, the number of start-ups is at least $2n-2$, the optimum data transfer time can be attained by increasing the number of start-ups. With *n-port* communication, the data transfer time is not optimal due to the fact that the bandwidth is not fully utilized during the first $\log n$ cycles. Also, edges at the same level are not evenly distributed in different dimensions.

With *one-port* communication and nESBT routing, the data transfer during cycle i and the number of start-ups are

$$\begin{cases} \frac{2^iM}{n}, & \text{for } 0 \leq i \leq n-1; \\ \frac{(N-1-2^{i-n})M}{n}, & \text{for } n \leq i \leq 2n-1. \end{cases} \quad \text{and} \quad \sum_{i=0}^{n-1}\lceil\frac{2^iM}{nB}\rceil + \sum_{i=n}^{2n-1}\lceil\frac{(N-1-2^{i-n})M}{nB}\rceil \text{ respectively.}$$

The total data transfer time is the same as the lower bound, lemma 6.1. For $B_m \geq \frac{M(N-2)}{n}$, the number of start-ups is reduced to $2n$. The maximum packet size occurs during cycle n. With *n-port* communication the data transmission term is $\frac{(N-1)M}{n}t_c$, i.e., equal to the lower bound. For $B \geq \sqrt{\frac{2}{\pi}}\frac{NM}{n\sqrt{n}}$ the number of start-ups is equal to $n+1$.

With *one-port* communication and nRSBT routing, the data transfer during cycle i and the number of start-ups are

$$\begin{cases} \frac{(2^{i+1}-1)M}{n}, & \text{for } 0 \leq i \leq n-1; \\ \frac{(N-2^{i-n+1})M}{n}, & \text{for } n \leq i \leq 2n-2, \end{cases} \text{ and } \sum_{i=0}^{n-1}\lceil\frac{(2^{i+1}-1)M}{nB}\rceil + \sum_{i=n}^{2n-2}\lceil\frac{(N-2^{i-n+1})M}{nB}\rceil \text{ respectively.}$$

The total data transfer time is the same as the lower bound, lemma 6.1. For $B \geq \frac{M(N-1)}{n}$, the number of start-ups is reduced to $2n-1$. The maximum packet size occurs during cycle $n-1$. For *n-port* communication the nRSBT routing attains the lower bound, corollary 6.2. The maximum packet size is $\sqrt{\frac{2}{\pi}}\frac{NM}{n^{3/2}}$.

With *one-port* communication and SBnT routing the number of start-ups is $2n-1$ (if $B \geq \frac{M(N-1)}{n}$) and the data transfer time is $M(N-1)t_c$. The packet size increases from cycle 0 to cycle $n-1$ and decreases from cycle $n-1$ to cycle $2n-2$. The sum of packet sizes for cycle i and cycle $i+n$ is equal to $B = \frac{(N-1)M}{n}$, $0 \leq i \leq n-2$. With *n-port* communication the transmission time is $\frac{(N-1)M}{n}t_c$, but the minimum number of start-ups is n. The maximum number of elements that contend for the same communications link is $\approx \sqrt{\frac{2}{\pi}}\frac{NM}{n\sqrt{n}}$.

If the source node address is included in the message, then a node upon receipt of a message makes use of the *children* function to determine on what ports the received message should be retransmitted. The following shows the routing algorithm in pseudo code.

```
msg := (mynode, data);
% Let M be the length of msg;
for i:= 0 to n − 1 do
    out_buf [i] := msg;
enddo
for step:= 1 to n do
    send (out_buf [i]) through port i, for all 0 ≤ i ≤ n − 1 concurrently;
    for i:= 0 to n − 1 do concurrently;
        receive (in_buf [i]) from port i;
        for each data of length M in in_buf [i] do
            extract source;
            {c1, c2, ...ck} := children(mynode, source);
            % At the last step, children will be φ;
            Let pj be the output port to cj;
            for j:= 1 to k do
                append this msg of length M to out_buf [pj];
            enddo
        enddo
    enddo
```

enddo

6.2. Comparison and Conclusion

With *one-port* communication SBT routing for all-to-all broadcasting attains the lower bound, $T_{min} = M(N-1)t_c + n\tau$ for $B_m \geq \frac{MN}{2}$. The BRG, nESBT, nRSBT and SBnT routings all have the minimum data transfer time. The number of start-ups is $(N-1)$, $2n$, $2n-1$ and $2n-1$, respectively, with the maximum packet size $\frac{M(N-2)}{n}$, $\frac{M(N-1)}{n}$ and $\frac{M(N-1)}{n}$, respectively. With the maximum packet size B_m of order $\frac{MN}{n}$, the number of start-ups for SBT, nESBT, nRSBT and SBnT routings are all compatible.

With *n-port* communication the nRBRG, nESBT, nRSBT, and SBnT routings achieves the lower bound transmission time for a sufficiently large packet size. Both the nRSBT and SBnT routings also attain n start-ups, i.e., the lower bound complexity, with the same maximum packet size $B \geq \sqrt{\frac{2}{\pi}}\frac{NM}{n^{3/2}}$. If $M \bmod n \neq 0$, then the nRSBT routing does not attain the lower bound complexity. The nESBT routing attains a nearly optimal bound with the exception of one additional start-up. The SBT routing yields a slow-down of approximately $\frac{n}{2}$ for the data transfer time compared to the lower bound routing.

7. All-to-All Personalized Communication

In all-to-all personalized communication every node sends M distinct elements to every other node.

Theorem 7.1. *The lower bound for one-port all-to-all personalized communication is* $n(\frac{NM}{2}t_c + \tau)$. *The maximum packet size* B_m *must be at least* $\frac{NM}{2}$ *to attain this lower bound.*

Proof. The bandwidth requirement for distributing personalized data from one node is

$$\sum_{i=0}^{n} i\binom{n}{i}M = \frac{nNM}{2}.$$

The total bandwidth requirement is $\frac{nN^2M}{2}$. During each cycle only N edges of the n-cube can communicate in the case of *one-port* communication, so $\frac{nNM}{2}$ is the minimum number of element transfers in sequence. The number of start-ups is at least n. The maximum packet size can be derived by dividing the total bandwidth requirement $\frac{nN^2M}{2}$ by the number of cycles n, and the number of directed edges that can be used in each routing cycle N.

∎

Theorem 7.2. *The lower bound for n-port all-to-all personalized communication is* $\frac{NM}{2}t_c + n\tau$. *The maximum packet size* B_m *must be at least* $\frac{NM}{2n}$ *to attain this lower bound.*

Proof. From theorem 7.1 the total bandwidth requirement is $\frac{nN^2M}{2}$. During each routing cycle nN directed edges can communicate concurrently. The maximum packet size is derived by dividing the total bandwidth requirement by the number of cycles n and the total number of links nN.

∎

Theorem 7.3. *The time for n-port all-to-all personalized communication based on N translated spanning graphs of height h and postorder scheduling concurrently for all subtrees of the source is*

$$T = \sum_{l=0}^{h-1}(\lceil \frac{M}{B} \times \max_{0\leq d\leq n-1} sw_{ld} \rceil \tau + Mt_c \times \max_{0\leq d\leq n-1} sw_{ld}).$$

If $B \geq \max_{0\leq l\leq h-1,\ 0\leq d\leq n-1}(M \times sw_{ld})$ *then*

$$T = (\sum_{l=0}^{h-1} \max_{0\leq d\leq n-1} sw_{ld})Mt_c + h\tau.$$

Proof. For each spanning graph, the amount of data transmitted across all edges in dimension d during routing cycle l is $sw_{ld} \times M$, $0 \leq l \leq h-1$. For the N translated spanning graphs, the amount of data transmitted across each cube edge in dimension d during routing cycle l is $sw_{ld} \times M$.

∎

Theorem 7.4. *The time for n-port all-to-all personalized communication based on N translated spanning graphs of height h and reverse-breadth-first scheduling is*

$$T = \sum_{l=0}^{h-1}(\lceil \frac{M}{B} \times \max_{0\leq d\leq n-1} rw_{ld} \rceil \tau + Mt_c \times \max_{0\leq d\leq n-1} rw_{ld}).$$

If $B \geq \max_{0\leq l\leq h-1,\ 0\leq d\leq n-1}(M \times rw_{ld})$ *then*

$$T = (\sum_{l=0}^{h-1} \max_{0\leq d\leq n-1} rw_{ld})Mt_c + h\tau.$$

Proof. Similar to theorem 7.3.

∎

Theorem 7.5. *The all-to-all personalized communication based on N translated spanning graphs will not attain the lower bound for data transfer time if the spanning graph is not greedy; and will not attain the lower bound for the start-up time if the height of the spanning graph is not minimum.*

Proof. The bandwidth requirement for each node is

$$\sum_{l=0}^{n-1}\sum_{d=0}^{n-1} sw_{ld} = \sum_{l=0}^{n-1}\sum_{d=0}^{n-1} rw_{ld} = \sum_{l=1}^{h} l \times \text{ (the number of nodes at level } l).$$

Hence, non-greedy spanning graphs require more data transfer than greedy spanning graphs. The latter follows from theorems 7.1, 7.2 and its non-optimal height.

∎

Theorem 7.6. *All-to-all personalized n-port communication based on N translated greedy spanning graphs, each composed of n distinctly rotated greedy spanning trees, can attain the lower bound communication time* $T_{min} = \frac{NM}{2}t_c + n\tau$ *for* $B \geq \frac{(N-1)M}{n}$ *both for postorder scheduling concurrently for each subtree, and reverse-breadth-first scheduling.*

Proof. A spanning graph derived by combining n distinctly rotated greedy spanning trees has the properties that

Algorithm	T
BRG	$\frac{(N-1)NM}{2}t_c + \sum_{i=1}^{N-1}\lceil\frac{iM}{B}\rceil\tau$
SBT	$\frac{nNM}{2}t_c + \lceil\frac{NM}{2B}\rceil n\tau$
nRSBT	$\frac{nNM}{2}t_c + (\sum_{i=0}^{n-1}\lceil\frac{(i+1)NM}{2nB}\rceil + \sum_{i=n}^{i=2n-2}\lceil\frac{(2n-i-1)NM}{2nB}\rceil)\tau$
nESBT	$(\frac{N}{2n}+N-2)Mt_c + (\sum_{i=0}^{n-1}\lceil(\frac{(i+2)N}{2}-1)\frac{M}{nB}\rceil + \sum_{i=1}^{n}\lceil(\frac{iN}{2}-1)\frac{M}{nB}\rceil)\tau$
SBnT	$\approx \frac{nNM}{2}t_c + \max(2n-1, \frac{nNM}{2B})\tau$

Table 15: The complexity of *one-port* all-to-all personalized communication.

Algorithm	B_{opt}	T_{min}
BRG	$(N-1)M$	$\frac{(N-1)NM}{2}t_c + (N-1)\tau$
SBT	$\frac{NM}{2}$	$\frac{nNM}{2}t_c + n\tau$
nRSBT	$\frac{NM}{2}$	$\frac{nNM}{2}t_c + (2n-1)\tau$
nESBT	$\frac{M}{n}(\frac{N(n-1)}{2}-1)$	$(\frac{nN}{2}+N-2)Mt_c + 2n\tau$
SBnT	$\frac{NM}{2}$	$\frac{nNM}{2}t_c + (2n-1)\tau$

Table 16: The optimum complexity of *one-port* all-to-all personalized communication.

1. $ew_{ld} = \frac{\binom{n}{l}}{n}$, $0 \le d \le n-1$, $0 \le l \le n-1$.
2. $sw_{ld} = \sum_{i=l+1}^{n}\frac{\binom{n}{i}}{n}$, $0 \le d \le n-1$, $0 \le l \le n-1$,
3. $rw_{ld} = \sum_{i=l+1}^{n}\frac{\binom{n}{i}}{n}$, $0 \le d \le n-1$, $0 \le l \le n-1$.

From theorem7.3 and property 2 above, the data transfer time is

$$Mt_c\sum_{l=0}^{n-1}\sum_{i=l+1}^{n}\frac{\binom{n}{i}}{n} = \frac{NM}{2}t_c.$$

The maximum packet size $B = \frac{(N-1)M}{n}$ occurs during routing cycle 0. Similarly, the *reverse-breadth-first* scheduling discipline can be shown to be optimum, with the same value of the maximum packet size. The maximum packet size occurs during the last routing cycle.

∎

Corollary 7.1. *All-to-all personalized n-port communication based on N translated nRSBT and SBnT graphs can attain the lower bound complexity.*

Notice that in *one-port* communication the data transfer time is always optimal if the routing is based on N translated greedy spanning graphs. But, not all greedy spanning graphs have the same number of start-ups. Only the SBT graph allows a minimum of n start-ups for sufficiently large packet size. The minimum number of start-ups can be decided by the same labeling rules as were used in all-to-all broadcasting. The minimum number of start-ups is the same as for the all-to-all broadcasting, Table 10. The difference is that the amount of data transferred during cycle i is equal to the sum of weighted subtree sizes with the root of each subtree connected through an edge labeled i to its parent.

7.1. The Complexity of All-to-All Personalized Communication

Tables 15, 16, 17 and 18 summarize the complexity estimates.

Algorithm	T
nRBRG	$\frac{(N-1)NM}{2n}t_c + \sum_{i=1}^{N-1}\lceil\frac{iM}{nB}\rceil\tau$
SBT	$(\sum_{l=0}^{\lfloor\frac{n-1}{2}\rfloor}\binom{2l}{l}2^{n-2l-1} + \sum_{l=\lfloor\frac{n+1}{2}\rfloor}^{n-1}\binom{n-1}{l})Mt_c$ $+(\sum_{l=0}^{\lfloor\frac{n-1}{2}\rfloor}\lceil\binom{2l}{l}\frac{2^{n-2l-1}M}{B}\rceil + \sum_{l=\lfloor\frac{n+1}{2}\rfloor}^{n-1}\lceil\binom{n-1}{l}\frac{M}{B}\rceil)\tau$
nRSBT	$\frac{NM}{2}t_c + \lceil\frac{NM}{2nB}\rceil n\tau$
nESBT	$(\frac{N}{2}+\frac{N-2}{n})Mt_c + (\sum_{i=2}^{n}\lceil\sum_{j=i}^{n}\binom{n}{j}\frac{M}{nB}\rceil + \lceil\frac{(N-2)M}{nB}\rceil + \lceil\frac{(N-1)M}{nB}\rceil)\tau$
SBnT	$\frac{NM}{2}t_c + \sum_{i=1}^{n}\lceil\sum_{j=i}^{n}\binom{n}{j}\frac{M}{nB}\rceil\tau$

Table 17: The complexity of ***n-port*** all-to-all personalized communication.

Algorithm	B_{opt}	T_{min}
nRBRG	$\frac{(N-1)M}{n}$	$\frac{(N-1)NM}{2n}t_c + (N-1)\tau$
SBT	$\frac{NM}{2}$	$O(\sqrt{n})NMt_c + n\tau$
nRSBT	$\frac{NM}{2n}$	$\frac{NM}{2}t_c + n\tau$
nESBT	$\frac{(N-1)M}{n}$	$(\frac{N}{2}+\frac{N-2}{n})Mt_c + (n+1)\tau$
SBnT	$\frac{(N-1)M}{n}$	$\frac{NM}{2}t_c + n\tau$

Table 18: The optimum complexity of ***n-port*** all-to-all personalized communication.

With ***one-port*** communication and BRG routing both the data transfer time and the start-up times are off by a factor of $\frac{N}{n}$ compared to the optimum for ***one-port*** communication. The former is due to the non-greedy property, and the latter is due to the non-optimum height. The complexity for ***n-port*** communication and nRBRG routing holds for both the concurrent ***postorder*** scheduling and the ***reverse-breadth-first*** scheduling. The routing fully utilizes the cube bandwidth; however, much of the data transfer is not through the shortest path, i.e., non-greedy.

Figure 14 shows all-to-all personalized communication on a 3-cube based on 8 SBTs. The shaded area represents the portion of the data residing in processor 0 (denoted P0). The task is to exchange the j^{th} block of data of processor i with the i^{th} block of processor j for any two distinct processors i and j. If initially processor i owns the i^{th} block column as in figure 14-(1) then on completion of the all-to-all personalized communication processor i contains the i^{th} block row as in figure 14-(4).

Lemma 7.1. *All-to-all personalized one-port communication based on N translated SBTs can achieve the lower bound* $\frac{nNM}{2B}(Bt_c+\tau)$.

Proof. During the first routing cycle, data is exchanged along the highest dimension, i.e., each node exchanges $\frac{1}{2}NM$ elements with the neighbor in the highest dimension. Then, the procedure is applied recursively with the data set doubling for each cycle of the recursion and the dimension of the cube decreasing by 1. Let $T(i,M)$ be the time required by the stated personalized all-to-all routing algorithm with initially M data per node in an i-cube. Clearly, $T(1,M) = \frac{M}{B}(Bt_c+\tau)$, $T(k,M) = \frac{2^{k-1}M}{B}(Bt_c+\tau)+T(k-1,2M)$. Hence, $T(n,M) = \sum_{i=0}^{n-1}\frac{NM}{2B}(Bt_c+\tau) = \frac{nNM}{2B}(Bt_c+\tau)$. ∎

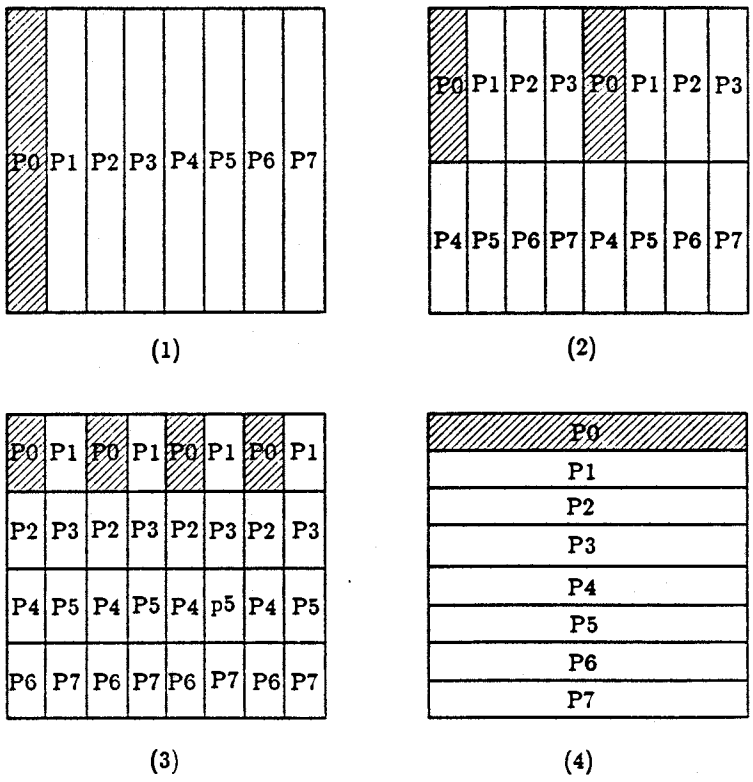

Figure 14: All-to-all personalized communication in a 3-cube based on SBTs.

In the case of *n-port* communication we can find the communication complexity from the previously derived formula. The interesting quantity for *postorder* scheduling concurrently for each subtree is $\sum_{l=0}^{n-1} \max s_{ld}$.

$$\sum_{l=0}^{n-1} \max_d s_{ld} = \sum_{l=0}^{\lfloor \frac{n-1}{2} \rfloor} \binom{2l}{l} 2^{n-2l-1} + \sum_{l=\lfloor \frac{n+1}{2} \rfloor}^{n-1} \binom{n-1}{l}$$

which is of $O(\sqrt{n}N)$. Since $r_{ld} = s_{l(n-d-1)}$ the *reverse-breadth-first* scheduling yields the same result.

The TCBT is not a greedy spanning tree, and the data transfer time will not be optimum. However, the number of start-ups can be optimum for *n-port* communication. The amount of data transfer is $(\frac{(2n-3)N}{2} + 2)M$ for a TCBT, compared to $\frac{nN}{2}M$ for a greedy spanning tree. The data transfer time is at least twice that of the lower bound, approximately. With *one-port* communication the number of start-ups will be greater than $2n-2$. Unlike routing according to the SBT, nESBT, nRSBT, and SBnT graphs, the TCBT graph cannot perfectly interleave the communication on edges in different subtrees. With *n-port* communication, the fact that the first $\log n$ cycles do not utilize the edges of all dimensions, and the uneven distribution of edges over dimensions at the

same level, will make the factor of 2 even larger.

With *one-port* communication and nRSBT routing, the data transfer time during cycle i is

$$\begin{cases} \frac{NM}{2n}(i+1)t_c, & \text{for } 0 \le i \le n-1; \\ \frac{NM}{2n}(2n-i-1)t_c, & \text{for } n \le i \le 2n-2. \end{cases}$$

which yields the lower bound transfer time $\frac{nNM}{2}t_c$. The number of start-ups is $2n-1$ if $B_m \ge \frac{NM}{2}$. For $B_m \le \frac{NM}{2n}$, the number of start-ups is the same as the SBT based routing.

With *n-port* communication, both the *postorder* and the *reverse-breadth-first* scheduling disciplines attain the lower bound, $T_{min} = \frac{NM}{2}t_c + n\tau$ for $B_m \ge \frac{(N-1)M}{n}$.

The maximum packet size can in fact be reduced to $\frac{NM}{2n}$ by using a variation of the nRSBT scheduling discipline. We describe an alternative discipline in terms of one SBT rooted at node 0, and with weight $\frac{1}{n}$. During the first routing cycle, node 0 sends out $\frac{NM}{2n}$ data across the link in dimension 0. During the second routing cycle, node 0, and the node receiving data during the previous cycle, send $\frac{NM}{4n}$ data through links in dimension 1 to node 2 and node 3 respectively. During cycle i, there are 2^i links in dimension $i-1$ communicating $\frac{NM}{2^i n}$ data elements each. Now, consider *n-port* communication on nN rotated/translated SBTs. Then, each link transmits an equal amount of data, $\frac{NM}{2n}$ during every routing cycle. Note that the same scheme, if applied to all-to-all broadcasting, will increase the maximum packet size from $\sqrt{\frac{2}{\pi}}\frac{NM}{n^{5/2}}$ to $\frac{NM}{2n}$.

Since the nESBT is not a greedy graph and also is not of optimal height, both the data transfer time and the start-up time exceed the lower bound. The data transfer exceeds the lower bound by at least $M(N-2)$. The number of start-ups is at least $2n$ for *one-port* communication and $n+1$ for *n-port* communication. In fact, from theorems 7.3 and 7.4, $T_{min} = (\frac{N}{2} + \frac{N-2}{n})Mt_c + (n+1)\tau$ for *n-port* communication and $T_{min} = (\frac{nN}{2} + N - 2)Mt_c + 2n\tau$ for *one-port* communication. The maximum packet sizes are $\frac{(N-1)M}{n}$ and $\frac{M}{n}(\frac{N(n+1)}{2} - 1)$, respectively.

With *one-port* communication the data transfer time is optimal, $\frac{NM}{2}t_c$, and the number of start-ups is $2n-1$ with $B_m \ge \frac{NM}{2}$. With *n-port* communication, the SBnT routing attains the lower bound by corollary 7.1. The maximum packet size is $\frac{(N-1)M}{n}$, a factor of 2 larger than the minimum maximum packet size, $\frac{NM}{2n}$, for any lower bound algorithm.

7.2. Comparison and Conclusion

With *one-port* communication, the SBT routing for all-to-all personalized communication attains the lower bound. Only greedy spanning graphs attain the optimum data transfer time. In fact, an optimum data transfer time can always be attained for the greedy spanning graphs, if the N spanning graphs are translations of each other. The number of start-ups is equal to the maximum edge label for a graph labeling such that edges in the same dimension carry the same label, and outgoing edges always carry a higher label than incoming edges for the spanning trees making up the spanning graph. Both the nRSBT and the SBnT routings attain the optimum data transfer. The nESBT, the TCBT, and the BRG routings are about a factor of $\frac{n+2}{n}$, 2, and $\frac{N}{n}$ higher than the lower bound data transfer time. The number of start-ups is in the range n to $2n$, except for the BRG routing which has N start-ups.

With *n-port* communication, both the nRSBT and the SBnT routings attain the lower bound for *postorder* and *reverse-breadth-first* scheduling. The nESBT and the nRBRG routings do not attain the lower bound due to their non-greedy properties. The SBT routing, though greedy, does

not evenly utilize edges in the same dimension, and hence is non-optimum. The data transfer time for the nESBT, the SBT, and the nRBRG routings are a factor of $\frac{n+2}{n}$, $\sqrt{n}$, and $\frac{N}{n}$ higher than the optimum time.

8. Experimental Results

8.1. The Intel iPSC Boolean Cube

The Yale version of the Intel iPSC [18] is a 128-node multiprocessor connected as a 7-dimensional Boolean cube. It has a message passing programming model. Up to $16k$ bytes can be passed in each communication, but the operating system subdivides messages of a size greater than $1k$ byte into $1k$ byte packets. We refer to the user defined packets as ***external packets*** and the operating system defined packets as ***internal packets***. There is a communications overhead (start-up time) associated with each packet. For an external packet we have recorded a start-up time that averages $8ms$. For internal packets the start-up time is approximately $6ms$ (at the time our programs were tested). The interprocessor communication channels are $10M$-bit ethernet channels. Although there are 7 ports per processor in the 7-cube, the storage bandwidth can only support $2-3$ ports concurrently. However, we have effectively been unable to realize this potential with the available operating system. The concurrency in communication on different ports of the same processor amounts to an overlap of about 20%.

8.2. One-to-All Broadcasting Based on the SBT and nESBT Spanning Graphs

For the SBT-based broadcasting a processor retransmits the message it receives to the adjacent nodes that correspond to leading zeroes in its address. The control is indeed very simple. For the nESBT-based broadcasting the control is also simple. From the definition of the $children_{nESBT}$ function it follows that if a node receives a message on a port corresponding to a 0 address bit, then it is the final recipient. If the message was received on a port corresponding to an address bit that is 1, say port k, and the first bit to the left of k, cyclically, that is 1 is bit j, then it retransmits the message to all nodes corresponding to bits $(k+1) \bmod n$, $(k+2) \bmod n$, ..., $(j-1) \bmod n$, $j \bmod n$ (excluding the last term if $k = j$).

Figure 15 shows the measured time to completion of one-to-all broadcasting using SBT routing for cubes of various dimensions and a number of different external packet sizes. As expected, the communication time increases almost linearly for external packet sizes below $1k$ bytes. Figures 16 shows the measured time of SBT and nESBT routing for an external packet size of $1k$ bytes and for cube dimensions ranging from 2 to 6. Figure 17 shows the speed-up of nESBT routing over the SBT routing. The measured speed-up is approximately n, as predicted.

8.3. One-to-All Personalized Communication

For one-to-all personalized communication on the Intel iPSC and SBT routing we use the ***postorder*** scheduling with communication on ports in a ***binary-reflected*** Gray code order, which maximizes the minimum time between successive communications on any port.. This scheduling means that data is communicated over ports in an order corresponding to the transition sequence in a binary-reflected Gray code. Advantage is taken of the approximately 20% overlap in communication on different ports.

For the implementation of the SBnT routing we let the root determine which node belongs in the different subtrees. In the case where n is a prime number the subtrees are isomorphic (excluding

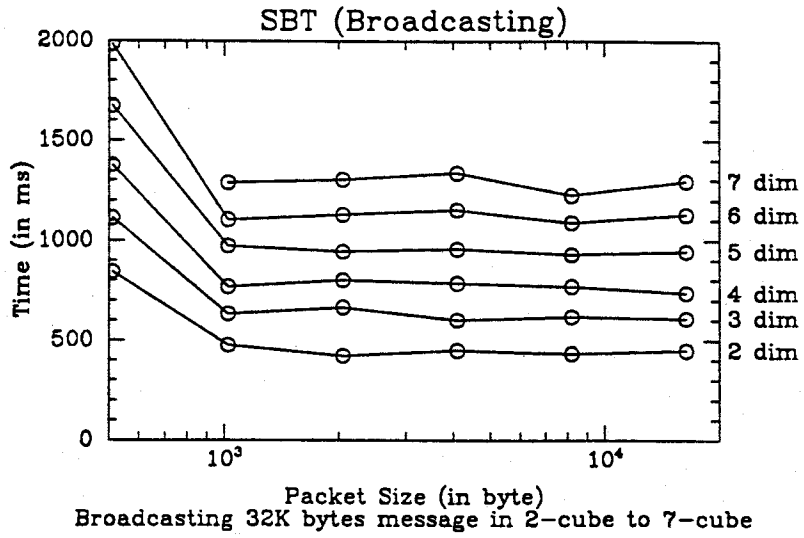

Figure 15: Broadcasting using SBT.

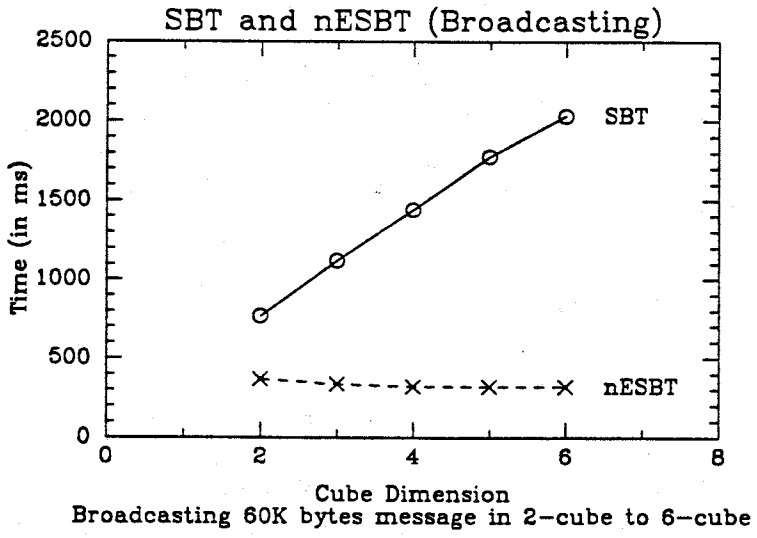

Figure 16: Broadcasting using SBT and nESBT.

node $i \oplus s = N - 1$). For this case the root only needs to keep one table of length $\approx \frac{N}{n}$ with each entry of size n bits. The order of the entries corresponds to the transmission order for each port, e.g., *postorder* or *reverse-breadth-first* order. The table entries point to the messages transmitted over port 0. The pointers for the other ports are obtained by (right) cyclic shifts of the table entries. A one step cyclic rotation is used for port 1, two steps for port 2, etc. For n not a prime number there are also cyclic nodes. If a single path is used to such nodes, then the subtrees of the root are no longer isomorphic. By finding the period P_i for each table entry, and not transmitting the message corresponding to this table entry for ports with index $j \geq P_i$ it is assured that every node gets its data precisely once. If multiple paths are used, then the message needs to be divided

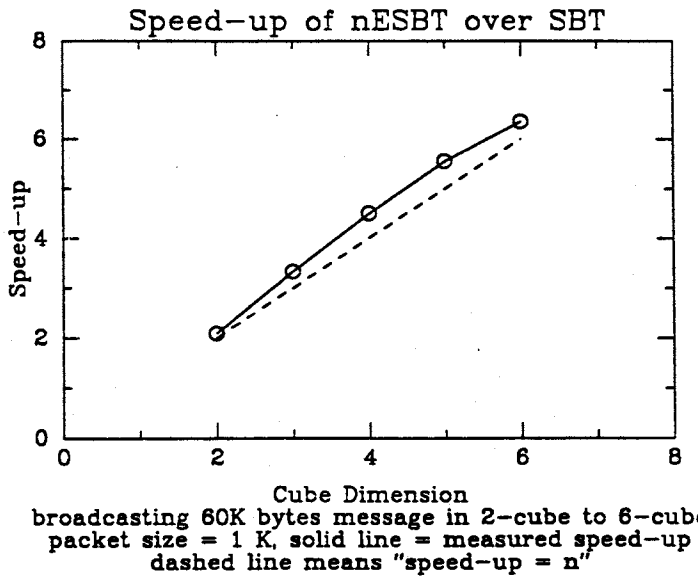

Figure 17: Speed up of nESBT vs SBT.

into P_i pieces, as described previously. Our implementation uses a single path to every node.

Internal nodes can either route according to the destination address if it is included, or use tables. If the destination is included, then a node first checks if it is the destination. Otherwise, the output port is determined by finding the base from $base((myaddress) \oplus (source))$ and then finding the first bit that is equal to 1 in $((myaddress) \oplus (destination))$ to the left (cyclically) of the bit corresponding to the base. If tables are used instead of a destination field, then for *postorder* scheduling it suffices that each internal node keeps a count for each port. Since the number of ports used in each subtree is at most $n-3$ and the number of nodes in the entire subtree is approximately $\frac{N}{n}$, a bound on the table size in each node is n^2 bits. A *reverse-breadth-first* scheduling can be implemented by internal nodes keeping a table of how many nodes there are at a given level in each of its subtrees. The table has at most n^2 entries. An upper bound for the number of nodes in a subtree at any level is $\frac{N}{n^{3/2}}$, and the total table size in a node is at most n^3 bits. Hence, without a more sophisticated encoding the *postorder* scheduling discipline requires less table space. It is used for the measurements presented in Figure 18.

With *one-port* one-to-all personalized communication the expected time for SBT routing and SBnT routing is the same. The observed advantage of the SBnT over the SBT routing is due to the fact that the SBnT can take better advantage of the overlap between communication on different ports. In the SBT case, even though messages were communicated over different ports in a binary-reflected Gray code order, the nodes adjacent to the root are not yet finished with retransmitting the last packet received when a new packet arrives. In the SBnT a subtree receives a packet once every n cycles, and full advantage of the 20% overlap in communications actions is taken.

8.4. All-to-All Broadcasting

We have also coded all-to-all broadcasting using the exchange algorithm, the SBnT, and broadcasting based on a single one- or two-way Hamiltonian circuit. The results are shown in Figure 19. The behavior is to first order the same for all forms of all-to-all broadcasting, as expected for the

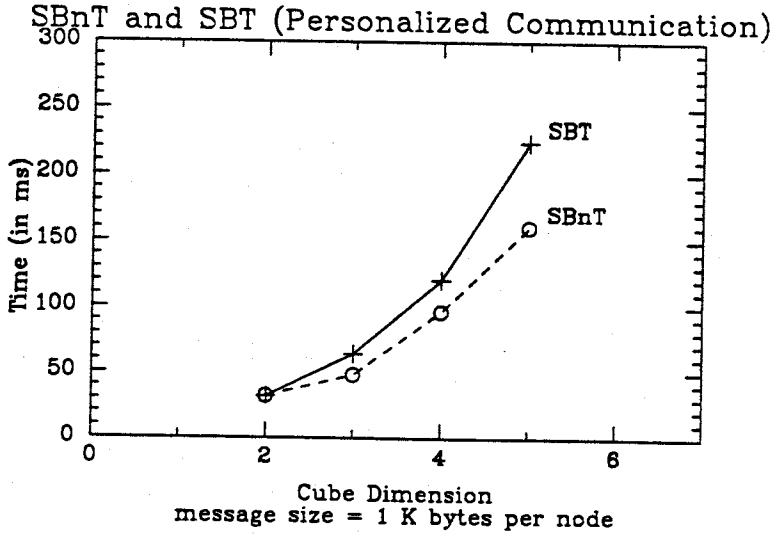

Figure 18: Personalized Communication using SBnT and SBT.

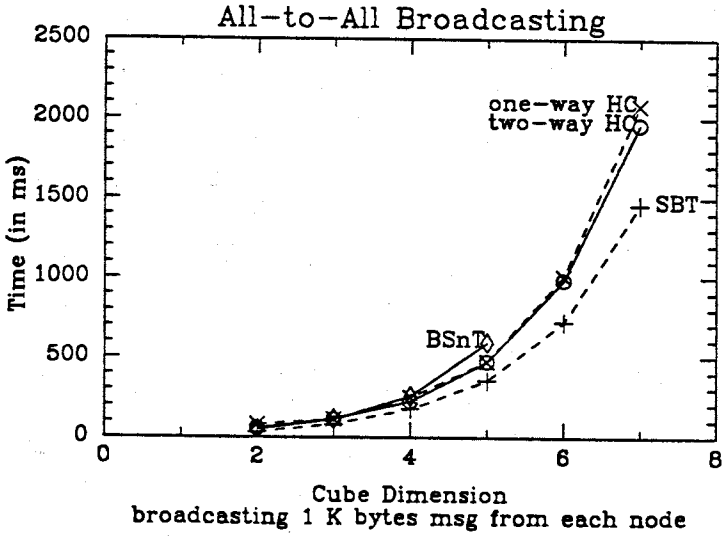

Figure 19: All-to-All Broadcasting.

Intel iPSC.

9. Conclusion

We show that the Boolean n-cube topology allows for the embedding of n edge-disjoint binomial trees, and present routing algorithms and scheduling disciplines for broadcasting that has a complexity equal to the lower bound both for communication restricted to one port at a time, and a potential for concurrent communication on all n ports of a node. We also define a spanning balanced n-tree for personalized communication and all-to-all broadcasting. Each subtree of the

Data Distribution	Assumption	Routings	B_{opt}	T_{min}
One-to-all B.	*one-port* , $M = 1$	SBT	1	$n(t_c + \tau)$
One-to-all B.	*n-port* , $M \leq n$	nRSBT	1	$n(t_c + \tau)$
One-to-all B.	*one-port* , $M > 1$	nESBT	$\sqrt{\frac{M\tau}{nt_c}}$	$(\sqrt{Mt_c} + \sqrt{n\tau})^2$
One-to-all B.	*n-port* , $M > n$	nESBT	$\frac{1}{n}\sqrt{\frac{M\tau}{t_c}}$	$(\sqrt{\frac{Mt_c}{n}} + \sqrt{n\tau})^2$
One-to-all P.C.	*one-port*	SBT	$\frac{NM}{2}$	$(N-1)Mt_c + n\tau$
One-to-all P.C.	*n-port*	SBnT, nRSBT	$\sqrt{\frac{2}{\pi}\frac{NM}{n^{3/2}}}$	$\frac{(N-1)M}{n}t_c + n\tau$
All-to-all B.	*one-port*	SBT	$\frac{NM}{2}$	$(N-1)Mt_c + n\tau$
All-to-all B.	*n-port*	SBnT, nRSBT	$\sqrt{\frac{2}{\pi}\frac{NM}{n^{3/2}}}$	$\frac{(N-1)M}{n}t_c + n\tau$
All-to-all P.C.	*one-port*	SBT	$\frac{NM}{2}$	$\frac{nNM}{2}t_c + n\tau$
All-to-all P.C.	*n-port*	SBnT	$\frac{(N-1)M}{n}$	$\frac{NM}{2}t_c + n\tau$
All-to-all P.C.	*n-port*	nESBT	$\frac{NM}{2n}$	$\frac{NM}{2}t_c + n\tau$

Table 19: Summary of the lower bound algorithms for various data distributions.

root of the spanning balanced n-tree we define has approximately $\frac{N}{n}$ nodes. Personalized communication based on the spanning balanced n-tree is for certain maximum packet sizes of the same complexity as personalized communication based on a binomial tree, and in other cases has at most a factor of 2 higher complexity for communication only on one port at a time. With concurrent communication on all ports, the routing based on the spanning balanced n-tree is superior by a factor of $\frac{n}{2}$ for a variety of combinations of maximum packet sizes, start-up times, transfer rates, and data sizes. We also show that routing based on the spanning balanced n-tree is superior to routing based on binomial trees for all-to-all broadcasting and all-to-all personalized communication when concurrent communication on all n ports of a node is possible. Table 19 lists the communication complexities of lower bound algorithms for the four data distribution problems discussed in this paper.

Fault tolerance is not directly addressed in this paper, but we note that any spanning graph with multiple paths to nodes inherently offers graceful degradation of performance under faulty conditions.

Experimental results on the Intel iPSC/d7 confirm the results of the analysis.

Acknowledgement

Thanks go to Chris Hatchell for his assistance with the manuscript. The generous support of the Office of Naval Research under contract N00014-84-K-0043 is gratefully acknowledged.

References

[1] Aho A.V., Hopcroft J.E., Ullman J.D., *Data Structures and Algorithms,* Addison-Wesley, 1983.

[2] Bhatt S.N.,Chung F.R.K., Leighton F.T., Rosenberg A.L., Optimal Simulations of Tree Machines, *Proc. 27th IEEE Symp. Foundations Comput. Sci.,* IEEE Computer Society, 1986, pp. 274–282.

[3] Bhatt S.N, Ipsen I.C.F., *How to Embed Trees in Hypercubes,* Technical Report YALEU/CSD/RR-443, Yale University, Dept. of Computer Science, December 1985.

[4] Browning S.A., *The Tree Machine: A Highly Concurrent Computing Environment,* Technical Report 1980:TR:3760, Computer Science, California Institute of Technology, January 1980.

[5] Buzbee,B.L., Golub, G.H., Nielson, C.W., *On Direct Methods for Solving Poissons's Equations,* SIAM J. Numer Anal, 7/4 December (1970), pp. 627–656.

[6] Dekel E., Nassimi D., Sahni S., *Parallel Matrix and Graph Algorithms,* SIAM J. Computing, 10 (1981), pp. 657–673.

[7] Deshpande S.R., Jenevin R.M., *Scaleability of a Binary Tree on a Hypercube,* Technical Report TR-86-01, University of Texas at Austin, January 1986.

[8] Fischer M.J., Efficiency of Equivalence Algorithms, *Complexity of Computer Computations,* Plenum Press, 1972, pp. 153–167.

[9] Fox G.C., Furmanski W., *Optimal Communication Algorithms on the Hypercube,* Technical Report, California Institute of Technology, July 1986.

[10] Fox G.C., Jefferson D., *Concurrent Processor Load Balancing as a Statistical Physics Problem,* Technical Report Caltech Concurrent Computation Project Memo 172, California Institute of Technology, dept. of Theoretical Physics, May 1985.

[11] Gannon D., Van Rosendale J., *On the Impact of Communication Complexity in the Design of Parallel Numerical Algorithms,* IEEE Trans. Computers, C-33/12 December (1984), pp. 1180–1194.

[12] Gustafson J.L., Hawkinson S., Scott K., The Architecture of a Homogeneous Vector Supercomputer, *1986 Int. Conf. Parallel Processing,* IEEE Computer Society, 1986, pp. 649–652.

[13] Hayes J.P., Mudge T.N., Stout Q.F., Colley S., Palmer J., Architecture of a Hypercube Supercomputer, *1986 Int. Conf. Parallel Processing,* IEEE Computer Society, 1986, pp. 653–660.

[14] Hillis W.D., *The Connection Machine,* MIT Press, 1985.

[15] Ho C.-T., Johnsson S.L., Distributed Routing Algorithms for Broadcasting and Personalized Communication in Hypercubes., *1986 Int. Conf. Parallel Processing,* IEEE Computer Society, 1986, pp. 640–648.

[16] ———, *Matrix Transposition on Boolean n-cube Configured Ensemble Architectures,* Technical Report YALEU/CSD/RR-494, Yale University, Dept. of Computer Science, September 1986.

[17] ———, *Spanning Balanced Trees in Boolean cubes,* Technical Report YALEU/CSD/RR-, Yale University, Dept. of Computer Science, November 1986.

[18] Intel iPSC System Overview, Intel Corp., January 1986.

[19] Johnsson S.L., *Odd-Even Cyclic Reduction on Ensemble Architectures and the Solution Tridiagonal Systems of Equations,* Technical Report YALE/CSD/RR-339, Department of Computer Science, Yale University, October 1984.

[20] ————, *Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures,* Journal of Parallel and Distributed Computing, (1986). Report YALEU/CSD/RR-361, January 1985, Dept. of Computer Science, Yale University.

[21] ————, *Solving Tridiagonal Systems on Ensemble Architectures,* SIAM J. Sci. Stat. Comp., (1986). Report YALEU/CSD/RR-436, November 1985.

[22] Kogge P.M., Stone H.S., *A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations,* IEEE Trans. Computers, C-22/8 (1973), pp. 786–792.

[23] Ladner R.E., Fischer M.J., *Parallel Prefix Computation,* J. ACM, 27/4 October (1980), pp. 831–838.

[24] Leighton F.T., *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-Exchange Graph and Other Networks,* MIT Press, 1983.

[25] McBryan O.A., Van de Velde E.F., *Hypercube Algorithms and Implementations,* Technical Report , Courant Institute of Mathematical Sciences, New York University, November 1985.

[26] Reingold E.M., Nievergelt J., Deo N., *Combinatorial Algorithms,* Prentice Hall, 1977.

[27] Saad Y., Schultz M.H., *Topological properties of Hypercubes,* Technical Report RR YALEU/DCS/RR-389, Dept. of Computer Science, Yale University, June 1985.

[28] ————, *Data Communication in Hypercubes,* Technical Report RR YALEU/DCS/RR-428, Dept. of Computer Science, Yale University, October 1985.

[29] Seitz C.L., *The Cosmic Cube,* Communications of the ACM, 28/1 (1985), pp. 22–33.

[30] Valiant L., Brebner G.J., Universal Schemes for Parallel Communication, *Proc. of the 13th ACM Symposium on the Theory of Computation,* ACM, 1981, pp. 263–277.

[31] Wu A.Y, *Embedding of Tree Networks in Hypercubes,* Journal of Parallel and Distributed Computing, 2/3 (1985), pp. 238–249.