Yale University Department of Computer Science

Spanning Balanced Trees in Boolean Cubes

Ching-Tien Ho and S. Lennart Johnsson

YALEU/DCS/TR-611
February 1988

This work has in part been supported by the Office of Naval Research under contracts N00014-84-K-0043 and N00014-86-K-0564. Approved for public release: distribution is unlimited.

† A revised edition of TR-508. To appear in *SIAM Journal on Algebraic and Discrete Methods*.

Spanning Balanced Trees in Boolean Cubes

Ching-Tien Ho and S. Lennart Johnsson[1]
Department of Computer Science
Yale University
New Haven, CT 06520

Abstract. A *Spanning Balanced n-tree* (SBnT) in a Boolean n-cube is a spanning tree in which the root has fanout n, and all the subtrees of the root have $O(\frac{2^n}{n})$ nodes. The number of tree edges in each dimension of the n-cube is of order $O(\frac{2^n}{n})$. The spanning balanced n-tree allows for scheduling disciplines that realize lower bound (within a factor of two) one-to-all personalized communication, all-to-all broadcasting, and all-to-all personalized communication on a Boolean n-cube [1,5]. The improvement in data transfer time over the familiar binomial tree routing is a factor of $\frac{n}{2}$ for concurrent communication on all ports and one-to-all personalized communication and all-to-all broadcasting. For all-to-all personalized communication on all ports concurrently the improvement is of order $O(\sqrt{n})$. We give distributed routing algorithms defining the spanning balanced n-tree. The balanced n-tree is not unique, and we provide a few definitions of n-trees that are effectively edge-disjoint. Some implementation issues are also discussed.

Binary numbers obtained from each other through rotation forms necklaces that are full if the period is equal to the length of the number, otherwise they are degenerate. As an intermediary result we show that the ratio between the number of degenerate necklaces and the total number of necklaces with l bits equal to one is at most $\frac{4}{4+n}$ for $1 \leq l < n$.

Key words. Boolean cubes, balanced trees, spanning trees, personalized communication, routing, shuffles, necklaces, periodicity

AMS(MOS) subject classification. 68P05, 68Q25

1 Introduction

High performance computer architectures require a high processing and communication bandwidth, which can be achieved in standard technologies by using a large number of processing elements, and interconnection networks that allow many independent communications concurrently. One such interconnection network is the Boolean cube. The number of dimensions for currently available systems range from 5 to 12. For highly concurrent computations, also known as data parallel computations in that the degree of parallelism is determined by the size of the

[1]Also Department of Electrical Engineering, Yale University, and Thinking Machine Corp., 245 First Street, Cambridge, MA 02142.

data set rather than control constructs, the number of arithmetic/logic operations per communication is often low, and the efficient use of the communication facilities is important for total performance. Some frequently occurring operations are broadcasting of data from one node to a subset of other nodes, or all other nodes, or the reverse operation: reduction. Common reduction operators are sum, min, max, and logical operators. Other global communications are the sending of a unique piece of data from one node to all other nodes, or the reverse operation. We refer to this type of communication as *personalized communication* [1,5]. Many linear algebra algorithms require broadcasting [4]. Certain matrix transpose algorithms and other permutations [2] effectively use *one-to-all* or *all-to-all personalized communication* [1].

For global communication a spanning tree, or a composition of many spanning trees, is needed. In this paper we describe and analyze a particular spanning tree, a *Spanning Balanced n-tree* (SBnT). It makes possible scheduling disciplines that realize lower bound communication within a factor of two [1]. A balanced n-tree is a tree with fanout n at the root, and approximately the same number of nodes in each subtree of the root. A commonly used spanning tree for global communication in Boolean cubes is a binomial tree. Such a tree has half of the number of nodes in one subtree, a quarter in another, etc. Another important property which makes lower bound algorithms for all-to-all broadcasting and all-to-all personalized communication possible is that for each level of the spanning balanced n-tree, the edges are evenly distributed among the n dimensions of an n-cube, while for the binomial tree, half of the edges are in dimension $n-1$, a quarter in dimension $n-2$, etc. A binomial tree is easily constructed by complementing leading or trailing zeroes. The construction algorithm is a distributed algorithm. Each node only needs to know the address of the source node and its own address (and perform an exclusive-or operation on the two before computing the address of children nodes through bit-complementation).

The construction of a balanced n-tree is somewhat more complex, but has a resemblance with the construction of the binomial tree. One key observation is that if one subtree of the root is known, then if all the addresses of the nodes in that subtree are rotated by the same amount, another subtree is obtained. That subtree has the same topology as the original subtree. Since n distinct rotations are possible on the node addresses in a Boolean n-cube, the balanced tree can be generated by this strategy, provided that all addresses have a period of n. Nodes with a period less than n will appear in several subtrees of the root. For the outlined construction to work, approximately $\frac{2^n}{n}$ node addresses must have a period of n. Address rotation defines a *shuffle* operation, and the addresses so obtained form a *necklace*. A necklace with n nodes is *full*. Other necklaces are *degenerate*. It follows from a result by Leighton [6] that there exist $O(\frac{2^n}{n})$ node addresses that cannot be obtained from each other through rotation. We prove a stronger result and show that the ratio between the number of degenerate necklaces and the total number of necklaces for l bits equal to one is at most $\frac{4}{4+n}$ for $1 \leq l < n$. For the construction of the balanced tree at most one node from every necklace is selected for the generating subtree, and an interconnection scheme need to be found that makes use of the Boolean cube topology. We describe one such construction that, viewed in the appropriate way, implies complementing some of the leading zeroes (but not all) of the node address in order to obtain the address(es) of the children node(s). The definition can be used as a distributed routing algorithm.

In this paper we define the balanced n-tree more precisely, analyze some of its topological properties, and give distributed algorithms for constructing balanced n-trees. We also present a spanning balanced graph, SBG, that minimizes the maximum edge load for *one-to-all personal-*

ized communication, *all-to-all broadcasting* and *all-to-all personalized communication.* The SBG is derived from the balanced n-tree. In personalized communication the root sends a unique piece of information to every node. We also present a few alternative definitions of the SBnT to demonstrate that the SBnT is not unique, and that the technique used to define it can easily be modified to generate other spanning balanced n-trees. These trees may have the same distribution of nodes among subtrees, but the subtrees have different topologies, and use different sets of edges in the cube. Fault tolerance for SBnT communication is equivalent to finding an SBnT that does not include the faulty edges. There exist different SBnT's that only share the edges emanating from the root, i.e., that are edge-disjoint below level 1 with the root at level 0. Similarly, the different definitions of the SBnT have different sets of leaf nodes, giving some flexibility for reducing the consequences of node failures. Note that all the spanning balanced n-trees defined here use all edges from the root. For personalized communication the communication complexity cannot be reduced by using several SBnT's concurrently. The root is the bottleneck, as shown later.

In section 2 the notations and definitions used throughout this paper are introduced. Section 3 contains a definition of the SBnT, and an analysis of its properties. Section 4 gives some alternative definitions of the SBnT and compares the characteristics of these alternative SBnT's with those of the SBnT in section 3. In section 5 we prove and give some complexity estimates for personalized communication based on the SBnT. Implementation issues are discussed in section 6 followed by a summary in section 7.

2 Preliminaries

A Boolean n-cube, is an n-dimensional cube with two nodes in each dimension. The total number of nodes is $N = 2^n$, and the total number of bidirectional links is $n\frac{N}{2}$. Nodes in the Boolean cube can be given n bit addresses such that adjacent nodes differ in precisely one bit. The distance between nodes $i = (i_{n-1}i_{n-2}\ldots i_0)$ and $i' = (i'_{n-1}i'_{n-2}\ldots i'_0)$ is defined as the *Hamming* distance between the nodes, i.e., $distance(i, i') = Hamming(i, i') = \sum_{m=0}^{n-1}(i_m \oplus i'_m)$. The number of 1-bits in the binary representation of i is denoted $||i|| = \sum_{m=0}^{n-1} i_m = Hamming(i, 0)$. The number of nodes at distance l from a node is $\binom{n}{l}$.

For the definition of the spanning balanced n-tree we make use of *rotations* and *translations.* The *right-rotation of a node* i is $R(i) = (i_0 i_{n-1} i_{n-2} \ldots i_1)$, and $R^m = R^{m-1} \circ R$ is a right rotation of m steps. The inverse operation $R^{-1} = L$ is a left rotation, i.e., $R^{-1}(i) = L(i) = (i_{n-2}i_{n-3}\ldots i_1 i_0 i_{n-1})$. The *right-rotation of a graph* $G = (\mathcal{V}, \mathcal{E})$ is $R(G) = (R(\mathcal{V}), R(\mathcal{E}))$, where $R(\mathcal{V}) = \{R(i) | \forall i \in \mathcal{V}\}$ and $R(\mathcal{E}) = \{(R(i), R(j)) | \forall (i, j) \in \mathcal{E}\}$. The *translation of a node* i by s is $T(i, s) = i \oplus s = c$, where c is the *relative address* of i with respect to s. The *translation of a graph* G by s is $T(G, s) = (T(\mathcal{V}, s), T(\mathcal{E}, s))$ with $T(\mathcal{V}, s) = \{T(i, s) | \forall i \in \mathcal{V}\}$ and $T(\mathcal{E}, s) = \{(T(i, s), T(j, s)) | \forall (i, j) \in \mathcal{E}\}$. The *bit-reversal* of a node i is $B(i) = (i_0 i_1 \ldots i_{n-1})$. Adjacency is preserved under rotation, translation, and bit-reversal [5]. Translation preserves the order of dimensions, rotation the relative order of dimensions, cyclically. The bit-reversal operation is its own inverse, and the following relationship between bit-reversal and rotation holds: $RBR = B$, $LBL = B$, $R = BLB$, and $L = BRB$.

A *necklace* [6] is a set of nodes with addresses that can be obtained from each other through rotation (shuffle operation). For example, (001001), (010010) and (100100) are in the same necklace. The numbers (110000), (011000), (001100), (000110), (000011) and (100001) are also in the same necklace (but not the same as the preceding ones). The *period* of a binary number i, P_i, is the least $u > 0$ such that $i = R^u(i)$. 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. Note that complementation of a binary number preserves the period. A *cyclic node* i is a node with *cyclic relative address* $c = i \oplus s$ with respect to a given node s. We define cyclic nodes with respect to a given node, since we are only interested in this property for trees with a defined root, and not for the node address itself. We also refer to the root node as the *source* node. Throughout the paper, we use c to represent the *relative* address of node i. Non-cyclic nodes belong to *full necklaces* and cyclic nodes to *degenerate necklaces* [6]. The number of nodes in the necklace to which c belongs is P_c. If there were no cyclic nodes then all necklaces are full, and a balanced n-tree can be generated through rotation of a *generating subtree* of the root. We label the subtrees of the root 0 - $n-1$, with the generating subtree being subtree 0. The generating subtree is assembled with one node from every necklace. These nodes are called *distinguished nodes*. The spanning balanced n-trees we define differ in the selection of distinguished nodes. The first SBnT we define use the node with minimum value of the nodes in a necklace as the distinguished node. This definition of distinguished node is similar, but not identical, to the one used by Leighton [6]. He defines the distinguished node as the node having the longest block of leading zeroes, which is also minimum, if there is a unique longest block of zeroes. If n is a prime number then there are only two degenerate necklaces, namely those formed by the nodes with relative addresses $(00\ldots0)$ and $(11\ldots1)$. The construction and analysis of spanning balanced trees for arbitrary n are complicated by the treatment of cyclic nodes.

The addresses of cyclic nodes are made up of repeating blocks. We use the notation $a|b$ to denote that a divides b. For cyclic nodes there exists at least one rotation of less than n steps of the address that yields the original value. Let J_c be the set of distinct rotations that minimize the value of the rotated address, i.e., $J_c = \{j_1, j_2, \ldots, j_m\}$, where $0 \le j_1 < j_2 < \cdots < j_m < n$, $R^u(c) = R^v(c)$, $u, v \in J_c$, and $R^u(c) < R^w(c)$, $u \in J_c$, $w \in \{0, 1, \ldots, n-1\} - J_c$. Clearly $|J_c| = \frac{n}{P_c}$. Graphs constructed through rotation of a generating subtree, SBG, are balanced, but are not trees, since there exist $|J_c|$ paths from the source node s to node i ($c = i \oplus s$). For a balanced n-tree we select a particular path of the J_c paths from the source node to each cyclic node. For the graph with J_c paths from source to destination the load on the edges from the source can be made even by dividing the data set for a node into $|J_c|$ pieces, one for each path to node c. The *index* of a node c defines the subtree(s) of the root to which it belongs; $index(c) = j_1$ for a single path to every node, and $index_m(c) = J_c$ for multiple paths to cyclic nodes. The distinguished nodes are defined by the set $\{c | index(c) = 0, \forall i \in \mathcal{V} - \{s\}\}$.

In a binomial tree defined by complementing leading zeroes all nodes in a given subtree of the root has the same number of trailing zeroes. The lowest order bit that is one can be viewed as the index of the subtree. For the first definition of a generating subtree we also consider the block of leading zeroes for defining connections to *children* nodes. However, complementing bits in this block may yield a node i' with relative address c' such that $index(c') \neq index(c)$ for the SBnT, or $index(c') \notin index_m(c)$ for the SBG graph. For instance, $index(00\ldots01) = 0$, but $index(10\ldots01) = n-1$. We define the SBnT and SBG graphs directly from the node addresses without explicit rotation of the generating subtree. It is convenient to introduce the following definitions. Let k be defined by $c_k = 1$, $c_j = 0, j \in \{(k+1) \bmod n, (k+2) \bmod n, \ldots, (index(c)-$

$1) \bmod n\} = \mathcal{M}(c)$, and $k = -1$ if $c = 0$. The k^{th} bit is the last 1-bit to the left of bit $index(c)$, cyclically. The set $\mathcal{M}(c)$ is the maximum set of consecutive indices of 0-bit positions immediately to the right of bit $index(c)$, cyclically. For non-root nodes, bit $index(c)$ is always a 1-bit. $|\mathcal{M}(c)| = \alpha_c$ is the number of leading zeroes of $R^{index(c)}(c)$. For example, for $c = (010110)$, $index(c) = 1$, $k = 4$, $\mathcal{M}(c) = \{5, 0\}$ and $\alpha_c = 2$.

3 A Spanning Balanced n-Tree

Our definition of the balanced n-tree can serve as a distributed routing algorithm. For the complexity estimates of various communication operations, it is of interest to characterize the distribution of nodes among the subtrees of the root as well as the fanout of the nodes. We give a lower bound $\frac{N}{n+2}$ on the number of nodes in a subtree of the root with the fewest number of nodes, which is low by at most a term of approximately $\frac{2N}{n^2}$. We also derive an upper bound of $\frac{2N}{n+2}$ on the number of nodes in a subtree. We present a table for the actual number of nodes in the maximum and minimum subtrees generated by the SBnT algorithm for up to 20-dimensional cubes, and show that the relative difference approaches 0 as the number of nodes grow, Table 1.

We define the balanced n-tree by two alternative functions: the $children_{SBnT}(i, s)$ and $parent_{SBnT}(i, s)$ functions.

$$children_{SBnT}(i,s) = \begin{cases} \{(i_{n-1}i_{n-2}...\bar{i}_m...i_0)\}, \forall m \in \mathcal{M}(c), & \text{if } c = 0; \\ \{q_m = (i_{n-1}i_{n-2}...\bar{i}_m...i_0)\}, & \\ \quad \forall m \in \mathcal{M}(c) \text{ and } index(q_m \oplus s) = index(c), & \text{if } c \neq 0. \end{cases}$$

$$parent_{SBnT}(i,s) = \begin{cases} \phi, & \text{if } c = 0; \\ (i_{n-1}i_{n-2}...\bar{i}_k...i_0), & \text{otherwise.} \end{cases}$$

The $index(parent_{SBnT}(i, s))$ is the same as $index(c)$, since for any node with relative address c, k is the highest-order bit of $R^{index(c)}(c)$ that is 1. Complementing this bit cannot change the index. It is also readily seen that the $parent_{SBnT}$ and $children_{SBnT}$ functions are consistent, i.e., for any node i it is a parent of a node j iff j is a child of i.

Lemma 1 *The $parent_{SBnT}$ ($children_{SBnT}$) function defines a spanning tree rooted at node s.*

Proof: The parent node of a node at distance d from node s is at distance $d - 1$ from node s, and each node only has one parent node. Traversing the edges defined by successive applications of the $parent_{SBnT}$ function of any node generates a path to node s for any node. Hence, the graph is a spanning tree rooted at node s. ▮

Figures 1, 2 and 3 show spanning trees generated by the algorithm above for the root located at node 0 in 3-, 4- and 5-cubes. Figure 4 shows subtree 0 of an SBnT in a 6-cube. Nodes in square boxes are cyclic nodes.

Lemma 2 *The number of nodes at level l from the source is $\binom{n}{l}$.*

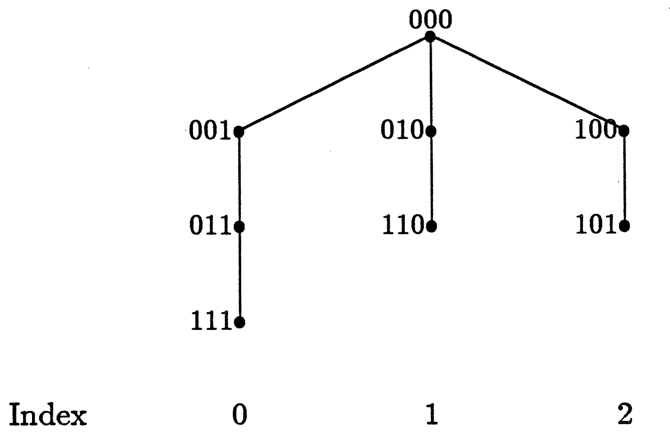


Figure 1: A spanning balanced 3-tree in a 3-cube.

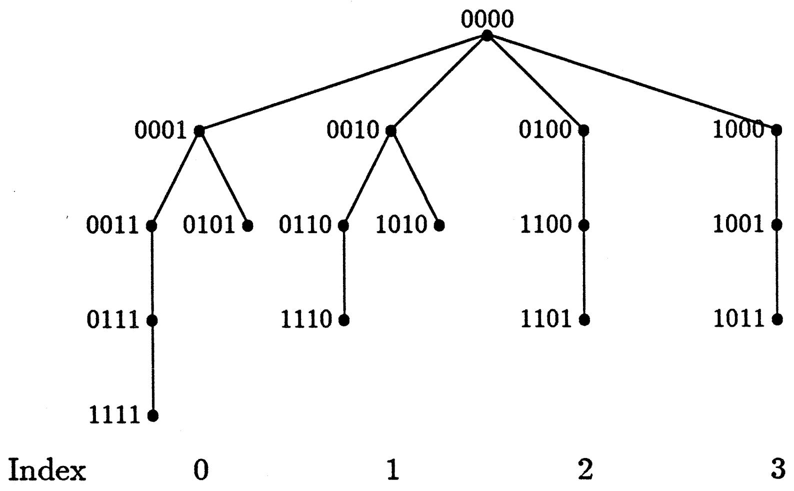


Figure 2: A spanning balanced 4-tree in a 4-cube.

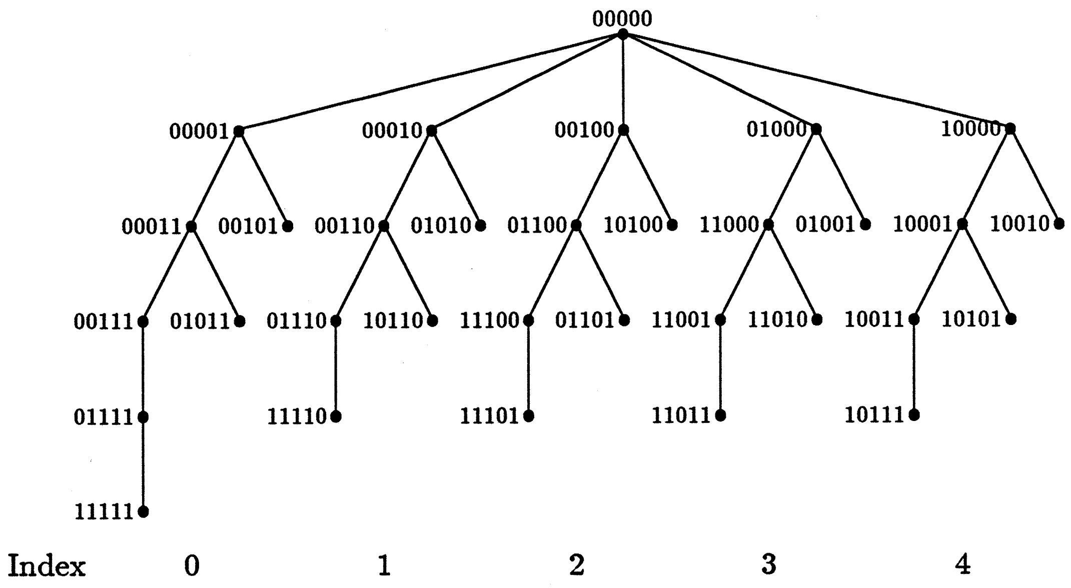


Figure 3: A spanning balanced 5-tree in a 5-cube.

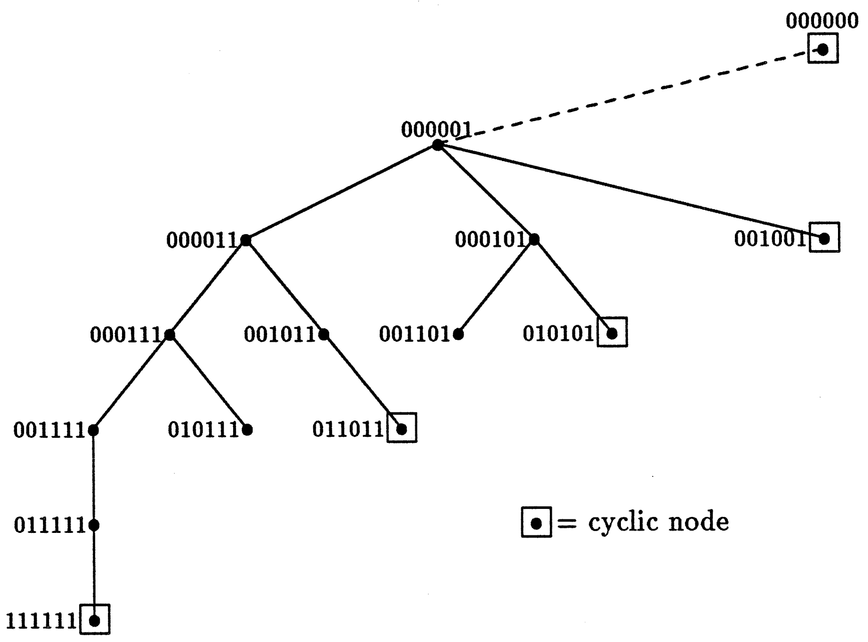


Figure 4: Subtree 0 of a spanning balanced 6-tree in a 6-cube.

Proof: From the $parent_{SBnT}$ function it follows that node i with $||c|| = l$ is at level l. Furthermore, there exist $\binom{n}{l}$ distinct permutations of l 1-bits out of a string of n bits. ▮

The SBnT is a *greedy* tree [5] in the sense that the distance for any node in the SBnT to the root is minimal. Personalized communication using a greedy spanning tree guarantees that the minimum communication bandwidth is used. All data is sent via a shortest path.

Corollary 1 *The height of one subtree is n, and the height of all other subtrees is $n-1$.*

Proof: There is only one node at distance n from the root, and there are n nodes at distance $n-1$ from the root, each of which has a different index. ▮

The fan-out of nodes is of particular importance for estimating the complexity of scheduling algorithms for personalized communication. If the fanout is non-increasing as a function of the distance from the root, then the analysis is considerably simplified.

Lemma 3 *The maximum fanout of a node at level l is $\lceil \frac{n-l}{2} \rceil$, for $1 \leq l \leq n$.*

Proof: Let the relative address $c = 2^l - 1$. Then $||c|| = l$ and complementing bits $\{l, l+1, \ldots, l + \lceil \frac{n-l}{2} \rceil - 1\}$ does not change the index, but complementing the higher-order bits does. Hence, the maximum fanout is at least $\lceil \frac{n-l}{2} \rceil$. But, for any other c such that $||c|| = l$, the maximum size of any block of consecutive zeroes, cyclically, is also $n-l$; hence $|\mathcal{M}(c)| \leq n-l$, and the argument can be applied to the leading block of $R^{index(c)}(c)$, and the proof is complete. ▮

For one-to-all personalized communication in Boolean cubes that allow concurrent communication on all *n-ports* of a node one effective scheduling discipline is a *reverse-breadth-first* order [5], i.e., scheduling data for all nodes at a given distance during the same routing cycle, and in order of decreasing distance from the source. The complexity analysis of this scheduling discipline is simplified, if the communication bandwidth required by nodes forwarding data from the root to the final destination is guaranteed to be at most the same as that required by the root. If the following property holds, then it suffices to consider the root alone.

Lemma 4 *Let $\phi(c,l)$ be the number of nodes at distance l from node c in the subtree rooted at node c. Then, $\phi(c,l) \geq \phi(c',l)$, where node c' is a child of node c.*

Proof: Let c' be any non-root node and node c its parent node. We prove the lemma first for subtree 0 by showing that for any node at distance l below node c', there is a unique corresponding node at distance l below node c. Recall that $\alpha_{c'}$ is the number of leading 0's of $R^{index(c')}(c') = c'$. Clearly, the number of leading 0's of c must be $\alpha_{c'} + \beta + 1$, where $\beta \geq 0$ is the number of consecutive 0's in c' between the two leftmost 1-bits. Any node at distance l below c', say c'_l, has an address that can be derived by complementing l out of the $\alpha_{c'}$ leading 0's of c', and with the index unchanged. There exists a corresponding node c_l at distance l below node c. The address of node c_l can be constructed by leaving $\beta + 1$ leading 0's of the address of c

unchanged, and making the following $\alpha_{c'}$ bits equal to the first $\alpha_{c'}$ bits of node c'_l. This process preserves the index for any given node c'_l. The same argument applies to any other subtree j by considering $R^j(c)$ and $R^j(c')$. ▮

Lemma 5 *Excluding node $c = (11\ldots1)$, all the subtrees of the SBnT are isomorphic if n is a prime number.*

Proof: For n a prime number there are no cyclic nodes, except the nodes with relative addresses $(00\ldots0)$ and $(11\ldots1)$. Since different subtrees are obtained through rotations of the addresses, the proof is complete. ▮

Corollary 2 *The subtrees of the root of the SBnT are isomorphic, if cyclic nodes are excluded.*

For communication from every node to every other node, *all-to-all communication*, using 2^n distinctly translated balanced n-trees a number of tree edges are mapped to the same cube edge. Since translation preserves the dimension of an edge (but not necessarily the direction) the number of tree edges mapped to any cube edge in a given dimension, say d, is equal to the number of SBnT edges in that dimension. For the complexity analysis of scheduling algorithms it is also necessary to know at what distance from the source node these edges are [1].

Corollary 3 *If n is prime, the number of edges in dimension d between levels $l-1$ and l is equal to $\frac{1}{n}\binom{n}{l}$ for $l = \{1, 2, \ldots, n-1\}$. For $l = n$ there is only one edge, and it is in dimension $n-1$. The total number of edges in dimension d is equal to $\frac{N-2}{n} + 1$ for dimension $n-1$ and $\frac{N-2}{n}$ for the other dimensions.*

Definition 1 A *treetop* of a tree T is a tree which is a connected subgraph of T containing the root of T.

Lemma 6 *Subtree i of the root of an SBnT is isomorphic to a treetop of subtree j of the root of the SBnT, if $i > j$.*

Proof: Subtree i is derived from subtree j by pruning away cyclic nodes that have periods p with $j < p \le i$ and corresponding incident edges, and performing a left cyclic shift of $i - j$ bits for all nodes in the pruned subtree. ▮

Corollary 4 *Subtree i contains all the nodes with periods $p > i$.*

Corollary 5 *There are $\frac{1}{2}n$ subtrees without cyclic nodes if n is even, and at least $\frac{2}{3}n$ subtrees without cyclic nodes if n is odd.*

Lemma 7 *Subtree 0 of an SBnT is isomorphic to a treetop of a Spanning Binomial Tree of an $(n-1)$-cube. Subtrees 1 through $n-1$ of an SBnT are isomorphic to a treetop of a Spanning Binomial Tree of an $(n-2)$-cube.*

Proof: Consider the relative addresses of nodes in subtree 0 of an SBnT of an n-cube. They all have the form $(i_{n-1}i_{n-2}\cdots i_1 1)$, where $i_m = 0$ or 1. From the definition of the children function of an SBnT, if node j is a child of node i then node j can be derived by complementing one of the leading 0-bits of node i. Recall that the children function of the spanning binomial tree(SBT) is defined by complementing *any* one of the leading 0-bits. Hence, subtree 0 of the SBnT is a treetop of an SBT of an $(n-1)$-cube. For the nodes in subtree 1 of the SBnT, they all have a relative address of the form $(i_{n-1}i_{n-2}\cdots i_2 10)$. Again, the addresses of the children can be derived by complementing one of the leading 0-bits. So, subtree 1 is a treetop of a SBT of an $(n-2)$-cube. By lemma 6, it follows that subtree 2 to subtree $n-1$ are treetops of a SBT of an $(n-2)$-cube. ∎

Lemma 8 *Any cyclic node is a leaf node of the SBnT.*

Proof: From the definition of the $index(c)$ and $children_{SBnT}(i,s)$ functions for a node i in a tree rooted at node s, the connections to the children nodes are defined by complementing the subset of the leading zeroes of $R^{index(c)}(c)$ that preserves $index(c)$. But, if c is periodic, then the index is changed since the leading repetitive pattern of c has a larger value than the following patterns. ∎

The imbalance between the subtrees of the root is caused by the cyclic nodes. We will now study the distribution of cyclic nodes in some detail. First we give a bound on the total number of cyclic nodes in a subtree.

Theorem 1 *The number of nodes in a subtree is of order $O(\frac{N}{n})$.*

Proof: With A cyclic nodes there are at least $\frac{N-A}{n}$ nodes in a subtree. Denote the number of degenerate necklaces by B. Since the length of each necklace is at least 2, except for the necklaces formed by nodes $(00\ldots0)$ and $(11\ldots1)$, $B \le \frac{A-2}{2}+2$. It follows that the maximum number of nodes in a subtree is $\frac{N-A}{n}+B-1 \le \frac{2N+(n-2)A}{2n}$. To derive bounds on A we use the complex-plane diagram used by Hoey and Leiserson [3] in studying the shuffle-exchange network. Leighton[6] shows that $B = O(\sqrt{N})$.

Full necklaces are mapped to circles. Degenerate necklaces are mapped to the origin. In the context of the shuffle-exchange graph each node that is mapped to the origin of the complex plane is adjacent (via an exchange edge) to a node at position $(1,0)$ or $(-1,0)$. Hence, for every full necklace of n nodes there are at most 2 cyclic nodes. It follows that an upper bound on A is $\frac{2N}{n+2}$ and the number of nodes in a subtree is at least $\frac{N}{n+2}$ and at most $\frac{2N}{n+2}$. The relative difference in the number of nodes in the maximum and minimum subtrees is $\frac{B}{O(\frac{N}{n})} = \frac{O(\sqrt{N})}{O(\frac{N}{n})}$, which approaches 0 for $N \to \infty$. ∎

n	A	B	SBT(max)	SBnT(max)	SBnT(min)	$(N-1)/n$	factor
2	2	2	2	2	1	1.50	1.33
3	2	2	4	3	2	2.33	1.29
4	4	3	8	5	3	3.75	1.33
5	2	2	16	7	6	6.20	1.13
6	10	5	32	13	9	10.50	1.24
7	2	2	64	19	18	18.14	1.05
8	16	6	128	35	30	31.88	1.10
9	8	4	256	59	56	56.78	1.04
10	34	9	512	107	99	102.30	1.05
11	2	2	1024	187	186	186.09	1.00
12	76	17	2048	351	335	341.25	1.03
13	2	2	4096	631	630	630.08	1.00
14	130	21	8192	1181	1161	1170.21	1.01
15	38	10	16384	2191	2182	2184.47	1.00
16	256	36	32768	4115	4080	4095.94	1.00
17	2	2	65536	7711	7710	7710.06	1.00
18	568	70	131072	14601	14532	14563.50	1.00
19	2	2	262144	27595	27594	27594.05	1.00
20	1036	111	524288	52487	52377	52428.75	1.00

Table 1: A comparison of subtree sizes of spanning binomial trees and spanning balanced n-trees.

Table 1 gives the sizes of the minimum (SBnT(min)) and maximum (SBnT(max)) subtrees generated according to the definition of the SBnT for up to 20-dimensional cubes. The relative difference approaches 0 rapidly. For comparison we have included the number of nodes in the largest subtree of the corresponding Spanning Binomial Tree. The last column contains the ratio of SBnT(max) to $\frac{N-1}{n}$. Figure 5 contains the same information as the table. The curves for the maximum and minimum SBnT become indistinguishable as the cube dimension increases.

In theorem 1 we showed that the total number of cyclic nodes is at most $\frac{2N}{n+2}$. We now first show that the ratio of the number of cyclic nodes at level l to the total number of nodes at level l is at most $\frac{2}{n}$. Then show that for any level of any subtree of the root, the ratio of the number of cyclic nodes to the total number of nodes at the same level of the same subtree is at most $\frac{4}{n+4}$ with the exception of the last level of subtree 0. We do that by defining a function described in lemma 12 and by showing that this function maps each cyclic node to a disjoint set of non-cyclic nodes at the same level of the same subtree (except the root node, s, and the node at the last level of subtree 0, $\bar{s}$). Some properties of the period of a cyclic number are needed.

Note that

1. if $a|c$, and $b|c$ then $\text{lcm}(a,b)|c$, where a, b, c are integers,

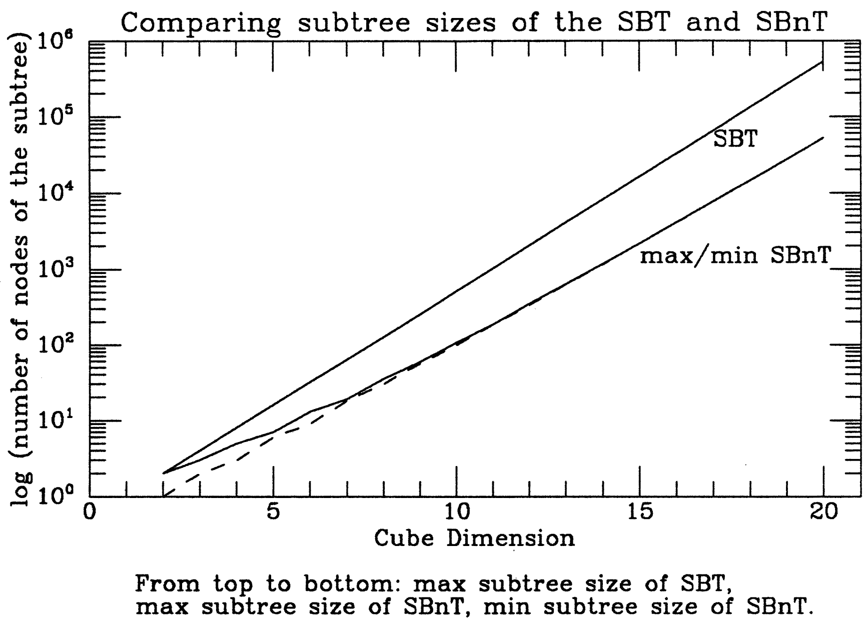


Figure 5: Comparing subtree sizes of spanning binomial trees and SBnT.

2. if c is an *n-bit* cyclic number with period p, then $\frac{n}{p} | \ ||c||$.

Lemma 9 *Let c_1 and c_2 be two distinct n-bit cyclic numbers with periods p_1 and p_2 respectively and $||c_1|| = ||c_2||$. Then* $\gcd(p_1, p_2) > 1$.

Proof: Let $||c_1|| = q$. Assume $\gcd(p_1, p_2) = 1$. Then $p_1|n, p_2|n$ and $\gcd(p_1, p_2) = 1$ imply $n = \gamma p_1 p_2$ for some positive integer γ. By property 2, we have $\frac{n}{p_1}|q, \frac{n}{p_2}|q$. But $n = \gamma p_1 p_2 \Rightarrow \gamma p_2|q$ and $\gamma p_1|q$, which imply $\gamma p_1 p_2|q$, i.e., $n|q$, by property 1. But $0 < q < n$ since $c_1 \neq c_2$ and we have a contradiction. ▮

Lemma 10 *Let c_1 and c_2 be two distinct n-bit cyclic numbers with periods p_1 and p_2 respectively, $p_1|p_2$ and $||c_1|| = ||c_2||$. Then Hamming$(c_1, c_2) \geq \frac{2n}{p_2}$.*

Proof: We derive a lower bound for the Hamming distance between c_1 and c_2, by finding the minimum number of bits of c_1 that have to be complemented to yield c_2, for all possible c_1 and c_2. c_2 consists of $\frac{n}{p_2}$ blocks of length p_2 each. In order to change the period from p_1 to p_2 (or change from c_1 to c_2 if $p_1 = p_2$), at least one bit in each block of p_2 bits of c_1 should be complemented. So, at least $\frac{n}{p_2}$ bits of c_1 should be complemented. However, either all bits are changed from 1 to 0 or vice versa to maintain periodicity. Hence, the number of 1-bits either decreases or increases by $\frac{n}{p_2}$ and thus $\frac{n}{p_2}$ bits should be changed in the opposite direction to satisfy $||c_1|| = ||c_2||$. It follows that the Hamming distance between c_1 and c_2 is at least $\frac{2n}{p_2}$. ▮

Lemma 11 *Let c_1 and c_2 be two distinct n-bit cyclic numbers with periods p_1 and p_2 respectively and $||c_1|| = ||c_2||$. Then $Hamming(c_1, c_2) \geq \frac{2n}{p_1 p_2}(p_1 + p_2 - 2\gcd(p_1, p_2))$.*

Proof: Let $g = \gcd(p_1, p_2)$ and c be an *n-bit* cyclic number with period g and $||c|| = ||c_1||$. By lemma 10, the Hamming distance between c and c_1 is at least $\frac{2n}{p_1}$. Similarly, the Hamming distance between c and c_2 is at least $\frac{2n}{p_2}$. To obtain c_1 from c we change 1-bits to 0-bits (and 0-bits to 1-bits) for every p_1 bits of c to produce c_1, and change 1-bits to 0-bits (and 0-bits to 1-bits) for every p_2 of c bits to produce c_2. The number of common bit positions of c that has been changed to generate c_1 and c_2 is $\frac{2n}{\text{lcm}(p_1, p_2)}$, if we changed $\frac{2n}{p_1}$ and $\frac{2n}{p_2}$ bits of c to convert it to c_1 and c_2 respectively. In general, $\frac{g}{p_2}$ of the bits we changed to generate c_1 and $\frac{g}{p_1}$ of the bits we changed to generate c_2 correspond to the common bit positions. So, the Hamming distance between c_1 and c_2 is at least $\frac{2n}{p_1} + \frac{2n}{p_2} - \frac{4n}{\text{lcm}(p_1, p_2)}$, i.e., $\frac{2n}{p_1 p_2}(p_1 + p_2 - 2g)$. ▮

Corollary 6 *Let c_1 and c_2 be two distinct n-bit cyclic numbers with periods p_1 and p_2 respectively, $||c_1|| = ||c_2||$ and $\gcd(p_1, p_2) \neq p_1$ and $\neq p_2$. Then $Hamming(c_1, c_2) \geq 6$.*

Proof: By lemma 11, $Hamming(c_1, c_2) \geq \frac{2n}{p_1 p_2}(p_1 + p_2 - 2\gcd(p_1, p_2))$, i.e., $Hamming(c_1, c_2) \geq \frac{2n}{\text{lcm}(p_1, p_2)}(\frac{p_1 + p_2}{\gcd(p_1, p_2)} - 2)$. The maximum value of $\text{lcm}(p_1, p_2)$ is n and the minimum value of $\frac{p_1 + p_2}{\gcd(p_1, p_2)}$ is 5 (for $\gcd(p_1, p_2) \neq p_1$ and $\neq p_2$). So, $Hamming(c_1, c_2) \geq 6$ follows. ▮

Corollary 7 *Let c_1 and c_2 be two distinct n-bit cyclic numbers with periods p_1 and p_2 respectively, $||c_1|| = ||c_2||$. Then $Hamming(c_1, c_2) \geq 4$.*

Proof: If $\gcd(p_1, p_2) = p_1$ or p_2, then by lemma 10, $Hamming(c_1, c_2) \geq \frac{2n}{\max(p_1, p_2)}$. Since $\max(p_1, p_2) \leq \frac{n}{2}$, $Hamming(c_1, c_2) \geq 4$.

If $\gcd(p_1, p_2) \neq p_1$ and $\neq p_2$, then by corollary 6, $Hamming(c_1, c_2) \geq 6$. ▮

Corollary 8 *Any node has at most one cyclic node as a child.*

Proof: It follows from corollary 7. ▮

The following theorem gives a bound on the ratio of the number of cyclic nodes at each level of the whole SBnT.

Theorem 2 *In an SBnT, the ratio of the number of cyclic nodes at level l, $0 < l < n$, to the number of nodes at the same level is at most $\frac{2}{n}$.*

Proof: To prove the theorem we show that for each cyclic number i such that $||i|| = l$, we can find a set NC_i of non-cyclic numbers such that $|NC_i| = \frac{n}{2} - 1$ and for $j \in NC_i$, $||j|| = l$.

Moreover, the sets for different cyclic numbers are disjoint. A binary number consists of a *string* of bits. Let f be a function that maps a string s of length q to a set of strings of the same length S_f. We define f to be the function that exchanges any 0-bit with the rightmost 1-bit, or any 1-bit with the rightmost 0-bit. The number of strings in the set S_f is 0, if s contains all 0-bits, or all 1-bits, and $q-1$ otherwise. That $|S_f| = q-1$, if $||s|| \neq 0$ follows from the fact that each 1-bit and 0-bit determine a unique string, except that the rightmost 1-bit and the rightmost 0-bit determine the same string. For each cyclic string s of length n and period p, we first find the largest p' satisfying $p|p'$, $p'|n$ and $p' < n$. Note that $p' \leq \frac{n}{2}$. We now want to generate $n - p' - 1$ non-cyclic strings from the given string s. The first p' bits of these strings are the same as the first p' bits of the string s. The last $n - p'$ bits of the strings are derived by applying the function f to the last $n - p'$ bits of the string s. Each generated string is non-cyclic because the Hamming distance between string s and each generated string is 2, and any two cyclic strings of the same length and containing the same number of 1-bits have a Hamming distance of at least 4, corollary 7. Since $p' \leq \frac{n}{2}$, the generated number of non-cyclic strings is at least $\frac{n}{2} - 1$. We now show that the sets S_f generated by two distinct cyclic strings are disjoint. Let c_1 and c_2 be two distinct cyclic strings with periods p_1 and p_2 respectively, and $||c_1|| = ||c_2||$. Consider the following three cases:

- $p_1 = p_2$: Clearly, the two generated sets are disjoint because $c_1 \neq c_2$.
- $\gcd(p_1, p_2) \neq p_1$ and $\neq p_2$: Since the Hamming distance between the generated strings and the given string is 2, and by corollary 6 the Hamming distance between c_1 and c_2 is at least 6, the two sets are disjoint.
- $\gcd(p_1, p_2) = p_1$ or p_2: Let $\gcd(p_1, p_2) = p_1$ without loss of generality, i.e., $p_1|p_2$. Since the first p_2 bits of the generated strings of c_1 and c_2 are distinct, the two generated sets are disjoint. ▮

Let c be a cyclic node at level l, $1 \leq l < n$, of subtree 0, and with period p. Let $f(c)$ be the set of nodes obtained by complementing the leftmost 1-bit of c and complementing one of the $\lceil \frac{n-l}{2} \rceil$ closest 0-bits to the right of it. For example, for $c = (0010100101)$, $f(c) = \{(0001100101), (0000110101), (0000101101)\}$. Let $C_l = \{c_1, c_2, \ldots, c_{\delta_l}\}$ be the set of all cyclic nodes at level l of subtree 0.

Lemma 12 *Every node in the set $\cup_{c_i \in C_l} f(c_i)$ is a non-cyclic node at level l of subtree 0. Moreover, if $c_i \neq c_j$, $c_i, c_j \in C_l$, then $f(c_i) \cap f(c_j) = \phi$.*

Proof: By corollary 7, the mapping function $c \rightarrow f(c)$ maps a cyclic node to a set of non-cyclic nodes. Since the leftmost 1-bit of c and some 0-bit to the right of it are exchanged, the index is preserved, i.e., the node obtained through the exchange operation is in the same subtree as c, and at the same level since the number of 1-bits is preserved. Let $c_1 = (r_1 r_2 \ldots r_{\frac{n}{p_1}})$ and $c_2 = (s_1 s_2 \ldots s_{\frac{n}{p_2}})$ be two arbitrary nodes in C_l with periods p_1 and p_2 respectively, and r_i (s_i) is the bit string of length p_1 (p_2). Without loss of generality we assume $p_1 \leq p_2$ and consider the following cases:

- $p_1 < p_2$ and $\gcd(p_1, p_2) \neq p_1$: From lemma 6, $\mathit{Hamming}(c_1, c_2) \geq 6$. Since the Hamming distance between c_1 and any node in $f(c_1)$ is 2, and the Hamming distance between c_2 and any node in $f(c_2)$ is 2, $\mathit{Hamming}(c'_1, c'_2) \geq 2, \forall c'_1 \in f(c_1), c'_2 \in f(c_2)$, i.e., $f(c_1) \cap f(c_2) = \phi$.

- $p_1 = p_2 = p$: Pick any node $c'_1 \in f(c_1)$ and any node $c'_2 \in f(c_2)$. Let x_1 (x_2) be the number of bits in the string r_i (s_i) which is to the right of the leftmost 1-bit including this 1-bit. From the definition of f, the last x_1 bits of c'_1 are the same as the last x_1 bits of c_1. Similarly, the last x_2 bits of c'_2 are the same as the last x_2 bits of c_2. If $x_1 = x_2$, then the string formed by the last x_1 bits of c_1 and the string formed by the last x_1 bits of c_2 are distinct since $c_1 \neq c_2$. Hence, the last x_1 bits of c'_1 and c'_2 are distinct, i.e., $c'_1 \neq c'_2$ and $f(c_1) \cap f(c_2) = \phi$, since c'_1 and c'_2 are arbitrarily chosen.

  If $x_1 < x_2$, then the last x_1 bits of c'_1 has $\frac{nl}{p}$ 1-bits and the last x_2 bits of c'_2 has also $\frac{nl}{p}$ 1-bits. The leading bit of the last x_2 bits of c'_2 is a 1-bit by definition. Hence, the last x_1 bits of c'_2 contain at most $\frac{nl}{p} - 1$ 1-bits. So, $c'_1 \neq c'_2$ or $f(c_1) \cap f(c_2) = \phi$. The proof for $x_1 > x_2$ is similar.

- $p_1 < p_2$ and $\gcd(p_1, p_2) = p_1$: Let $p_2 = \gamma p_1$, $\gamma > 1$. Partition each bit string s_i of c_2, each of length p_2, into γ sub-strings of length p_1 each. Denote these substrings $s_i^1, s_i^2, \ldots, s_i^\gamma$. So,

$$c_2 = (s_1^1 s_1^2 \ldots s_1^\gamma s_2^1 s_2^2 \ldots s_2^\gamma \ldots s_{\frac{n}{p_2}}^1 s_{\frac{n}{p_2}}^2 \ldots s_{\frac{n}{p_2}}^\gamma).$$

  For convenience, we also relabel

$$c_1 = (r_1^1 r_1^2 \ldots r_1^\gamma r_2^1 r_2^2 \ldots r_2^\gamma \ldots r_{\frac{n}{p_2}}^1 r_{\frac{n}{p_2}}^2 \ldots r_{\frac{n}{p_2}}^\gamma).$$

  Note that $r_{i_1}^{j_1} = r_{i_2}^{j_2}, \forall 1 \leq i_1, i_2 \leq \frac{n}{p_2}, 1 \leq j_1, j_2 \leq \gamma$; also $s_{i_1}^j = s_{i_2}^j, \forall 1 \leq i_1, i_2 \leq \frac{n}{p_2}$, $1 \leq j \leq \gamma$. Let the leftmost 1-bit of c_2 be in s_1^j. If $j > 1$, then the last $(\gamma - j + 1)p_1$ bits of c'_2 contain $\frac{nl}{p_2}$ 1-bits, while the last $(\gamma - j + 1)p_1$ bits of c'_1 contain $\frac{(\gamma - j + 1)nl}{\gamma p_2}$ 1-bits, which is less than $\frac{nl}{p_2}$. So, $c'_1 \neq c'_2$. If $j = 1$, then consider the two cases:

  - $r_1^y \neq s_1^y$ for some $y, 1 < y \leq \gamma$: By definition, the last $(\gamma - 1)p_1$ bits of c'_1 are identical to the string $(r_1^2 r_1^3 \ldots r_1^\gamma)$, and the last $(\gamma - 1)p_1$ bits of c'_2 are identical to the string $(s_1^2 s_1^3 \ldots s_1^\gamma)$. So, $c'_1 \neq c'_2$.
  - $r_1^1 \neq s_1^1$: Let x_1 (x_2) be the number of bits of r_1^1 (s_1^1) to the right of the leftmost 1-bit of r_1^1 (s_1^1) including this 1-bit. Consider the last x_1 (x_2) bits of $r_{\frac{n}{p_2}}^1$ ($s_{\frac{n}{p_2}}^1$). The proof of $c'_1 \neq c'_2$ follows that of the case $p_1 = p_2$. ▌

Theorem 3 *For an SBnT, the ratio of the number of cyclic nodes at any level l, $1 \leq l < n$, of any subtree to the number of nodes at the same level of the same subtree is at most* $\frac{1}{1+\max(\lceil\frac{n-l}{2}\rceil, \lceil\frac{l}{2}\rceil)} \leq \frac{4}{4+n}$.

Proof: By lemma 12, the function $c \rightarrow f(c)$ maps any cyclic node $c \in C_l$ at level l of subtree 0 to a set of $\lceil\frac{n-l}{2}\rceil$ non-cyclic nodes at the same level of the same subtree. Moreover, the sets $f(c)$

for different cyclic nodes are disjoint. Since complementing all bits of a binary number preserves the period, the number of degenerate necklaces with l 1-bits for the nodes in the necklace is the same as the number of degenerate necklaces with $n-l$ 1-bits for the nodes in the necklace. So, the ratio of the number of cyclic nodes to the total number of nodes at level l of subtree 0 is the same as the ratio at level $n-l$ of subtree 0. Since subtree 0 has the maximum number of cyclic nodes at each level, and non-cyclic nodes at each level are evenly distributed among the n subtrees, the ratio for subtree 0 is an upper bound for all subtrees. ▮

Corollary 9 *With the exception of the last level, the number of cyclic nodes at any level of any subtree of an SBnT is at most the same as the number of non-cyclic nodes at the same level of the same subtree of the SBnT.*

Proof: By theorem 3, $\frac{4}{4+n} \leq \frac{1}{2}$ for $n \geq 4$. But since for $n = 2$ and $n = 3$, n is a prime number and there are no cyclic nodes, except nodes $c = (00\ldots00)$ and $(11\ldots11)$ the proof is complete. ▮

Theorem 3 gives a bound on the ratio of the number of cyclic nodes at each level in each subtree. Figure 6 shows the ratio of the actual number of cyclic nodes to the total number of nodes for each level of up to 16-dimensional cubes for subtree 0. The bound given by the theorem is pessimistic, and we conjecture that the ratio is $\frac{2}{n}$ (which is true for up to 16-dimensional cubes). In Figure 6 the letters encode cube dimensions greater than 9, and digits cube dimensions less than 10. For the number of cube dimensions being a prime number there are no cyclic nodes, except at levels 0 and n. Note that the ratio neither decreases monotonically with the cube size for a given level, nor decreases monotonically with increasing level for a given cube size.

Corollary 10 *If n is even, then the ratio of the number of cyclic nodes at level 2 (or $l-2$) of subtree 0 to the number of nodes at the same level of the same subtree is $\frac{2}{n}$.*

Proof: There is only one cyclic node and $\frac{n}{2} - 1$ non-cyclic nodes at level 2 $(l-2)$. ▮

If the conjecture is true, then for all even n the maximum ratio occurs at levels 2 and $n-2$ (excluding levels 0 and n).

The definition of the SBnT given above has a unique path from the source to every other node. For personalized communication the number of data elements that need to be transferred to subtree 0 exceeds the number of elements that need to be transferred to subtrees $\frac{n}{2}$ - $n-1$ by the number of degenerate necklaces times the data set transferred to each node (assuming the same amount of data to each node). By allowing multiple parent nodes for every cyclic node c, and by splitting the data set for each cyclic node of period P_c into $\frac{n}{P_c}$ parts, the load becomes the same for each subtree of the root. The bandwidth requirement for each link from the root is $\frac{(N-1)M}{n}$, where M is the size of the data set for each node.

To carry out the load balancing operation, the definition of the set $\mathcal{M}(c)$ is modified by using the set $index_m(c)$ for the definition instead of $index(c)$. Let k_q be such that $c_{k_q} = 1$,

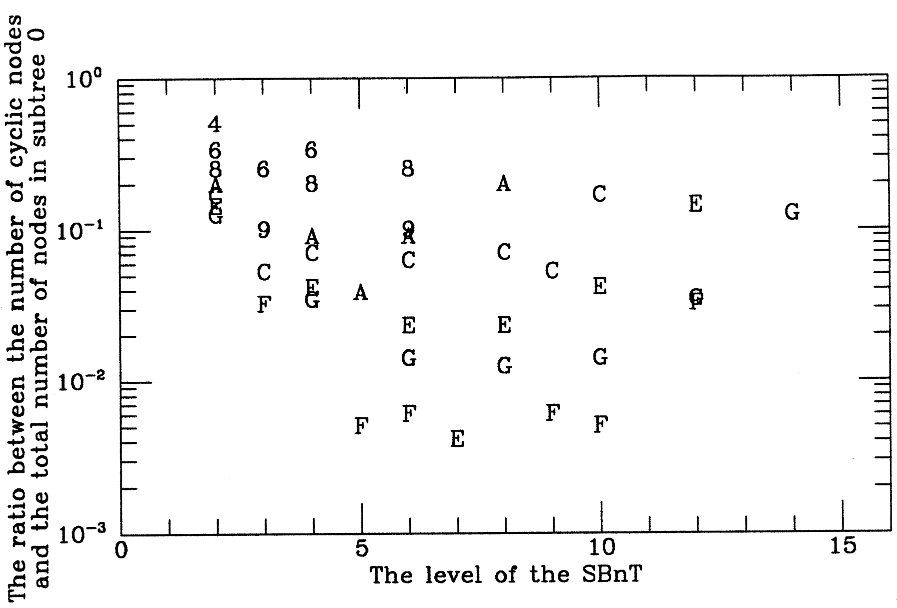


Figure 6: The ratio of the number of cyclic nodes to the total number of nodes at level l, $1 \leq l < n$, of subtree 0.

$c_r = 0, r \in \{(k_q + 1) \bmod n, (k_q + 2) \bmod n, \ldots, (j_q - 1) \bmod n\}$ and $k = -1$ if $c = 0$. Then,

$$\mathcal{M}_m(c) = \cup_{j_q \in J_c}\{(k_q + 1) \bmod n, (k_q + 2) \bmod n, \ldots, (j_q - 2) \bmod n, (j_q - 1) \bmod n\}.$$

The $children_{SBnT}$ and $parent_{SBnT}$ functions are modified as follows:

$$children_{SBG}(i, s) = \begin{cases} \{(i_{n-1}i_{n-2}...\bar{i}_m...i_0)\}, \forall m \in \mathcal{M}_m(c), & \text{if } c = 0; \\ \{q_m = (i_{n-1}i_{n-2}...\bar{i}_m...i_0)\}, & \\ \quad \forall m \in \mathcal{M}_m(c) \text{ and } index_m(q_m \oplus s) = index_m(c), & \text{if } c \neq 0. \end{cases}$$

$$parent_{SBG}(i, s) = \begin{cases} \phi, & \text{if } c = 0; \\ (i_{n-1}i_{n-2}...\bar{i}_{k_q}...i_0), q = \{1, 2, \ldots, m\}, & \text{otherwise.} \end{cases}$$

For example, node (011011011) appears in subtrees 0, 3 and 6 with parents (001011011), (011011001) and (011001011), respectively. The modified SBnT, SBG, is a spanning graph [5], which can be viewed as composed of n rotated SBnT's with a weight of $\frac{1}{n}$ each. The parent of a cyclic node c in the j^{th} SBnT is derived by choosing the dimension from the set $index_m(c)$ that is of the lowest order greater than j, cyclically.

4 Other Choices of Spanning Balanced n-Trees

For the definition of the balanced n-tree and the balanced graph SBG we defined the generating subtree based on distinguished nodes selected as the node of a necklace with minimum value. Another obvious choice is to select the node with maximum value. Generating subtrees defined on distinguished nodes obtained from these two definitions are not rotations of each other. For instance, the nodes with relative addresses (0001), (0011), (0111), and (1111) are all minimal in their respective necklaces, and belong to the generating subtree for the SBnT we defined previously. Nodes (1000), (1100), (1110), and (1111) are all maximal in their respective necklaces, and belong to the generating subtree for an SBnT based on selecting maximum values in necklaces. The number of right rotations to match a node from one selection criteria with a node from the other selection criteria is node dependent. We will now define and compare spanning balanced trees based on maximum values, and minimum and maximum bit reversed values. Different definitions of the distinguished nodes result in balanced trees, or graphs, that use different edges of the cube, with the exception that they all use all the edges directed away from the root.

We refer to a balanced n-tree defined through maximizing left rotations by SBnT^{maxL}, and the previous tree as SBnT^{minR}. The set of indices that maximizes the left rotations is denoted $J_c^{maxL} = \{j_1, j_2, \ldots, j_m\}$ and it is defined by $L^u(c) = L^v(c)$, if $u, v \in J_c^{maxL}$, and $L^u(c) > L^w(c)$, if $u \in J_c^{maxL}$ and $w \in \{0, 1, \ldots, n-1\} - J_c^{maxL}$. Moreover $0 \leq j_1 < j_2 < \cdots < j_m < n$ and $index^{maxL}(c) = j_1$ and $index_m^{maxL}(c) = J_c^{maxL}$. The formal definitions of the $parent_{SBnT^{maxL}}$ and $children_{SBnT^{maxL}}$ functions are based on a set of dimensions $\mathcal{M}^{maxL}(c)$ similar to the set $\mathcal{M}(c)$ for right rotations. Let k be such that $c_k = 1$ and $c_j = 0, j \in \{(k-1) \bmod n, (k-2) \bmod n, \ldots, (n - index^{maxL}(c)) \bmod n\} = \mathcal{M}^{maxL}(c)$ and $k = n$ if $c = 0$. The SBnT^{maxL} is defined by

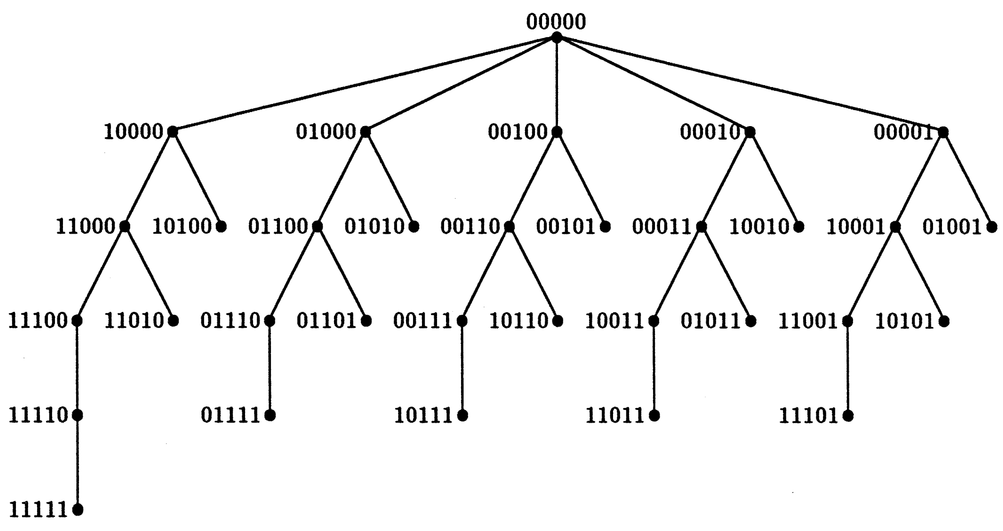
$$children_{SBnT^{maxL}}(i,s) = \begin{cases} \{(i_{n-1}i_{n-2}...\bar{i}_m...i_0)\}, \forall m \in \mathcal{M}^{maxL}(c), & \text{if } c = 0; \\ \{q_m = (i_{n-1}i_{n-2}...\bar{i}_m...i_0)\}, \quad \forall m \in \mathcal{M}^{maxL}(c) & \\ \qquad \text{and } index^{maxL}(q_m \oplus s) = index^{maxL}(c), & \text{if } c \neq 0. \end{cases}$$

$$parent_{SBnT^{maxL}}(i,s) = \begin{cases} \phi, & \text{if } c = 0; \\ (i_{n-1}i_{n-2}...\bar{i}_k...i_0), & \text{otherwise.} \end{cases}$$

Figure 7 shows the SBnT^{maxL} in a 5-cube. A balanced SBG^{maxL} graph can be defined similarly to the balanced SBG^{minR} graph. Note that the choice of *minimizing* the *right* rotation in the SBnT^{minR} is made such that the root of subtree j is $(00\ldots01_j0\ldots0)$. If the number of left rotations that identifies a node with a distinguished node were used instead of the number of right rotations for the assignment of a node to a subtree, then subtrees $1, 2, \ldots, n-1$ in such an SBnT^{minL} are relabelings of subtrees $n-1, n-2, \ldots, 1$ of the SBnT^{minR}, respectively. A SBnT^{minL} and an SBnT^{minR} are identical and only differ in the labeling of the subtrees. A similar argument applies to the SBnT^{maxL} and the SBnT^{maxR}.

Lemma 13 *The number of nodes of each subtree of the tree SBnT^{maxL} is equal to the number of nodes in the same subtree of the tree SBnT^{minR}.*

Proof: For a non-cyclic node, all the n nodes in the same necklace belong to n different subtrees. For a cyclic node with period p, all the p nodes in the same necklace belong to subtree 0 to $p-1$. ▌



Index 0 1 2 3 4

Figure 7: A SBnT^{maxL} in a 5-cube.

Theorem 4 *The SBnT^{maxL} and SBnT^{minR} are not topologically equivalent, in general.*

Proof: We prove the theorem by a direct evaluation for a 6-cube. In a 6-cube, the maximum fanout of any node at level 2 of an SBnT^{minR} is 2, Figure 4, and the maximum fanout of any node at level 2 of an SBnT^{maxL} is 3, Figure 9. ∎

Two definitions of balanced n-trees dual to the SBnT^{minR} and the SBnT^{maxL}, and that use mostly a different set of edges can be derived from the bit-reversed representation of the addresses. The dual of the SBnT^{minR} is denoted SBnT^{minBL} and the dual of the SBnT^{maxL} is denoted SBnT^{maxBR}. The SBnT^{minBL} is defined as follows:

Let $J_c^{minBL} = \{j_1, j_2, \ldots, j_m\}$, where $0 \le j_1 < j_2 < \cdots < j_m < n$, $L^u(c) = L^v(c)$, $u, v \in J_c^{minBL}$, and $B \circ L^u(c) < B \circ L^w(c)$, $u \in J_c^{minBL}$, $w \in \{0, 1, \ldots, n-1\} - J_c^{minBL}$. Then $index^{minBL}(c) = j_1$ and k is defined by $c_k = 1$ and $c_j = 0$, $j \in \{(k-1) \bmod n, (k-2) \bmod n, \ldots, (n - index^{minBL}(c)) \bmod n\} = \mathcal{M}^{minBL}(c)$ and $k = n$ if $c = 0$.

$$children_{SBnT^{minBL}}(i,s) = \begin{cases} \{(i_{n-1}i_{n-2}\ldots\bar{i}_m\ldots i_0)\}, \forall m \in \mathcal{M}^{minBL}(c), & \text{if } c = 0; \\ \{q_m = (i_{n-1}i_{n-2}\ldots\bar{i}_m\ldots i_0)\}, \quad \forall m \in \mathcal{M}^{minBL}(c) & \\ \qquad \text{and } index^{minBL}(q_m \oplus s) = index^{minBL}(c), & \text{if } c \neq 0. \end{cases}$$

$$parent_{SBnT^{minBL}}(i,s) = \begin{cases} \phi, & \text{if } c = 0; \\ (i_{n-1}i_{n-2}\ldots\bar{i}_k\ldots i_0), & \text{otherwise.} \end{cases}$$

The SBnT^{maxBR} is defined as follows:

Let $J_c^{maxBR} = \{j_1, j_2, \ldots, j_m\}$, where $0 \le j_1 < j_2 < \cdots < j_m < n$, $R^u(c) = R^v(c)$, $u, v \in J_c^{maxBR}$, and $B \circ R^u(c) > B \circ R^w(c)$, $u \in J_c^{maxBR}$, $w \in \{0, 1, \ldots, n-1\} - J_c^{maxBR}$.

Then $index^{maxBR}(c) = j_1$ and k is defined by $c_k = 1$ and $c_j = 0$, $j \in \{(k+1) \bmod n, (k+2) \bmod n, \ldots, (index^{maxBR}(c) - 1) \bmod n\} = \mathcal{M}^{maxBR}(c)$ and $k = -1$ if $c = 0$.

$$children_{SBnT^{maxBR}}(i,s) = \begin{cases} \{(i_{n-1}i_{n-2}...\bar{i}_m...i_0)\}, \forall m \in \mathcal{M}^{maxBR}(c), & \text{if } c = 0; \\ \{q_m = (i_{n-1}i_{n-2}...\bar{i}_m...i_0)\}, \quad \forall m \in \mathcal{M}^{maxBR}(c) & \\ \qquad \text{and } index^{maxBR}(q_m \oplus s) = index^{maxBR}(c), & \text{if } c \neq 0. \end{cases}$$

$$parent_{SBnT^{maxBR}}(i,s) = \begin{cases} \phi, & \text{if } c = 0; \\ (i_{n-1}i_{n-2}...\bar{i}_k...i_0), & \text{otherwise.} \end{cases}$$

$index^{minR}(c)$ is the number of right rotations yielding the longest block of leading 0-bits, and $index^{maxL}(c)$ the number of left rotations yielding the longest block of leading 1-bits. Similarly, $index^{minBL}(c)$ is the number of left rotations yielding the longest block of trailing 0-bits, and $index^{maxBR}(c)$ the number of right rotations yielding the longest block of trailing 1-bits. For example, $index^{minR}$, $index^{maxL}$, $index^{minBL}$ and $index^{maxBR}$ of node (1110100010) are 5, 0, 8 and 7, respectively.

Note that the rotation and bit-reversal operations do not commute, i.e., $R \circ B \neq B \circ R$ and $L \circ B \neq B \circ L$.

Lemma 14 *The bit-reversed value of a bit string with a minimum value among all its rotations is not necessarily the maximum value among the bit-reversals of its rotations, i.e.,* $B \min_x R^x(i) \neq \max_x BR^x(i)$ *for some* i.

Proof: (001101) is the minimum value among all its rotations. But, (101100) is not the maximum bit-reversed value among all rotations of (001101). ▮

Note that for $n \leq 5$, the minimum value among all rotations of an address also yields the maximum bit-reversed value of its rotations. This means that the SBnT^{maxBR} (SBnT^{maxL}) and the SBnT^{minR} are topologically equivalent for up to 5-dimensional cubes.

Theorem 5 *The SBnT^{minR} and SBnT^{minBL} are topologically equivalent for all n, and so are the SBnT^{maxL} and SBnT^{maxBR}.*

Proof: Consider any node i in subtree 0 of the SBnT^{minR} and the node $B(i)$ in SBnT^{minBL}. This mapping is one-to-one and onto. We first show that every node of subtree 0 of the SBnT^{minR} has a corresponding node in subtree 0 of the SBnT^{minBL}. To show this property we need to show that $BL^xB(i)$ is minimized for $x = 0$ since i is in subtree 0 of the SBnT^{minR}. But, $BL^xB(i) = BL^{x-1}LB(i) = BL^{x-1}BR(i)$ and it follows that $BL^xB(i) = R^x(i)$, which is minimized for $x = 0$. Correspondingly, every node of subtree 0 of SBnT^{minBL} has a counterpart in subtree 0 of the SBnT^{minR}. The same argument applies for any other subtree and it follows that the number of nodes in every subtree of the SBnT^{minR} is the same as the number of nodes in the same subtree of the SBnT^{minBL}. To complete the proof we notice that the bit-reversal operation preserves adjacency.

For the SBnT^{maxL} and SBnT^{maxBR} case we instead use the property that $BR^xB(i) = L^x(i)$. ▮

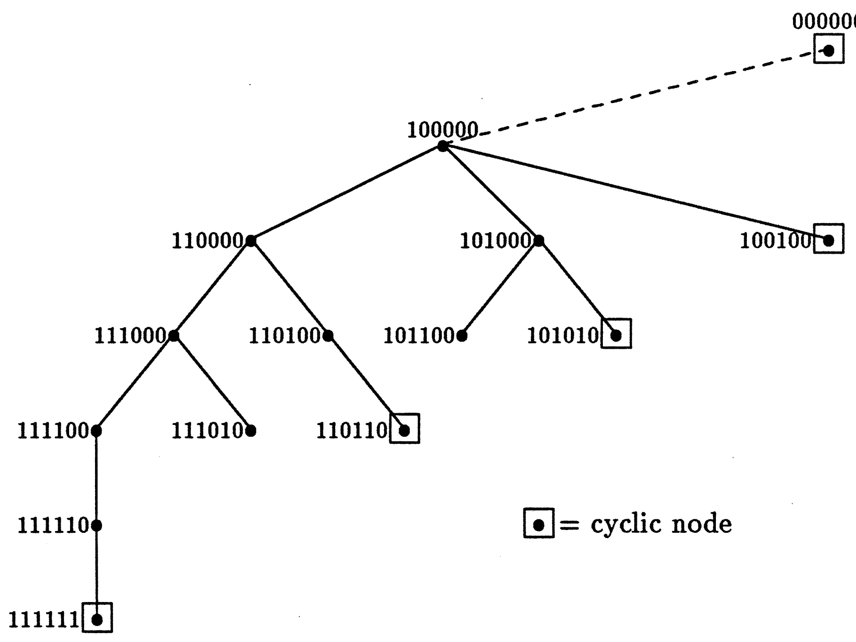


Figure 8: Subtree 0 of an SBnT^{minBL} in a 6-cube.

Figures 8, 9 and 10 show subtree 0 of an SBnT^{minBL}, SBnT^{maxL} and SBnT^{maxBR} in a 6-cube. The nodes in square boxes are cyclic.

For *n-port one-to-all personalized communication*, the SBnT^{minR} routing has an advantage over the SBnT^{maxL} routing in that the maximum fanout is for most levels lower than for the SBnT^{maxL} routing. The fanout decreases monotonically for the SBnT^{minR} by lemma 4, but this is only true for the SBnT^{maxL} for levels $l \geq 2$. Any spanning tree satisfying lemma 4 guarantees that the complexity of personalized communication with concurrent communication on all ports is determined by the root. The maximum fanout of nodes at level l of the SBnT^{maxL} is

$$\begin{cases} \lceil \frac{n-l}{2} \rceil, & \text{if } l = 1; \\ n-l-1, & \text{if } 2 \leq l \leq n-2; \\ 1, & \text{if } l = n-1. \end{cases}$$

For the SBnT^{minR} the fanout at level l is $\lceil \frac{n-l}{2} \rceil$, $1 \leq i < n$, by lemma 3. The preference of the SBnT^{minR} over the SBnT^{minBL} is due to the simpler computation of the index.

Lemma 15 *For any node below level 1 of an SBnT, the $parent_{SBnT^{minR}}$ and $parent_{SBnT^{minBL}}$ functions define two distinct nodes, if the relative address has a unique longest consecutive block of zeroes, cyclically.*

Proof: By definition, $index^{minR}$ is the dimension of the 1-bit immediately to the left of the longest block of zeroes. The parent address can be derived by complementing the 1-bit, which is immediately to the right of the longest block of zeroes. Similarly, $index^{minBL}$ is the dimension

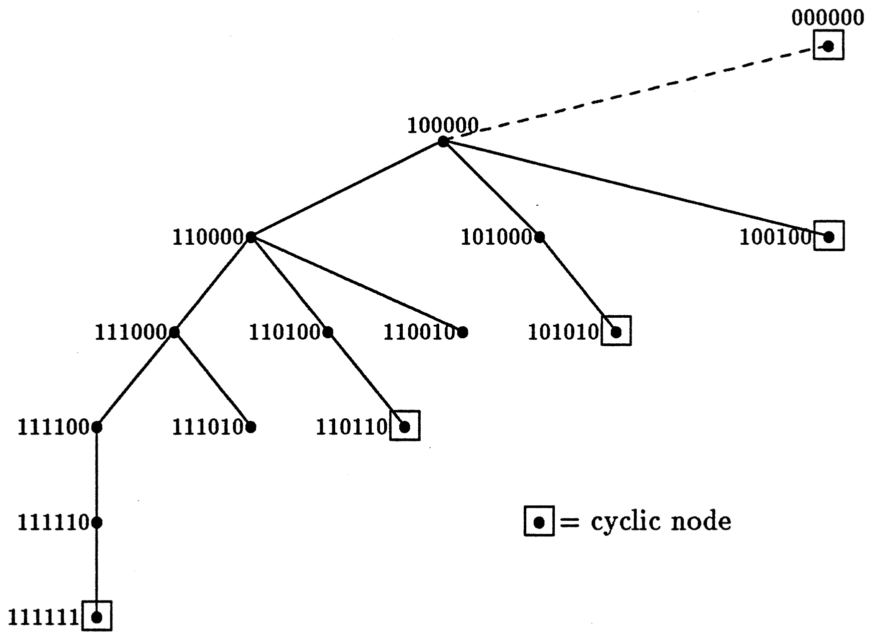


Figure 9: Subtree 0 of an SBnT^{maxL} in a 6-cube.

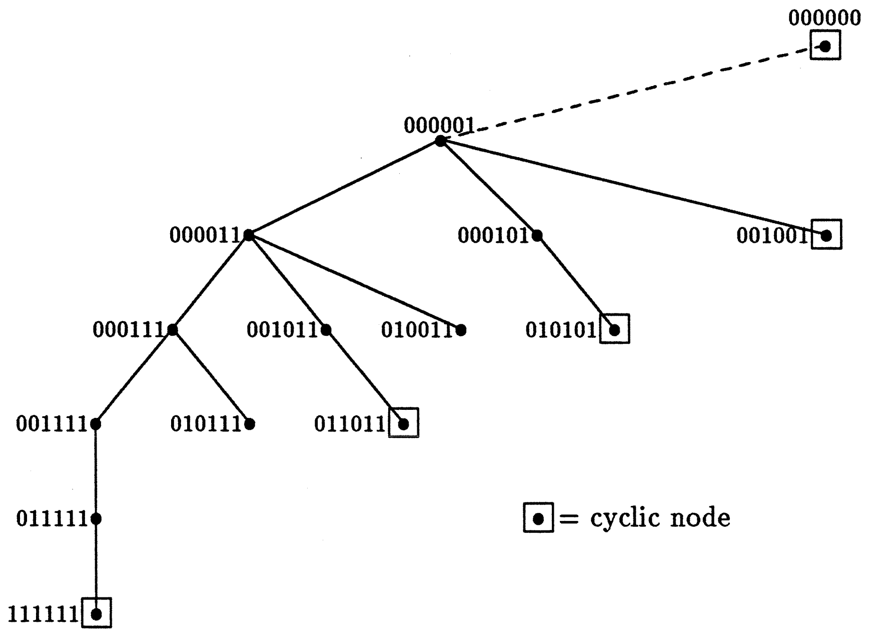


Figure 10: Subtree 0 of an SBnT^{maxBR} in a 6-cube.

of the 1-bit immediately to the right of the longest block of zeroes. The parent address can be derived by complementing the 1-bit, which is immediately to the left of the longest block of zeroes. ▮

The $parent_{SBnT^{minR}}$ and the $parent_{SBnT^{minBL}}$ functions are distinct. It can be shown that the SBnT^{minR} and the SBnT^{minBL} are edge-disjoint below level 1 for up to 4-dimensional cubes. For 5- and 6-cubes, there are 5 and 6 common edges. For 7- and 8-cubes, there are 14 and 16 common edges. The incoming edges of nodes (01011), (010111) and (0010011) are examples of shared edges. Modifications to the $parent_{SBnT}$ function, such as permutation of the dimensions, can be made to insure that the modified SBnT, and, for instance, SBnT^{minR}, are edge-disjoint below level 1. The existence of SBnT's that are edge-disjoint below level 1 is important for fault-tolerance.

5 Personalized Communication Based on an SBnT Graph

As an example of the use of the SBnT we give some complexity results for personalized communication in a Boolean n-cube. We first consider the case of *one-to-all personalized communication* with the communication restricted to *one-port* at a time. With this restriction we assume that the entire data set for the subtree rooted at the sending node is communicated in one routing cycle. For each node we employ a *scheduling discipline* for which data is sent to subtrees in order of increasing dimension of the connecting edge. For the root there is a minor difference in the data volume to the different subtrees for SBnT routing, but no difference for SBG routing.

With *one-port* communication the root requires a time of $M(N-1)t_c + n\tau$ for packet switched communication with an unbounded buffer size, a data set M communicated to each node, a transfer time of t_c for each element of the data set, and a start-up time τ for each routing cycle. After the last routing cycle of the root, the last subtree to receive the data has to distribute it to its nodes. We will now prove that a node c in subtree $index(c)$ receives its data during routing cycle $index(c) + n - 1 - \alpha_c$, and that the data transfer time for each subtree is bounded from above by $O(M\frac{N \log n}{n} t_c)$. To prove these results we need the following lemmas.

Lemma 16 *Let k be the dimension of the edge through which a non-root node c connects to its parent in an SBnT. Let the number of children of c be β. Then, the β children of c are connected through edges in dimensions $(k+j) \bmod n$, $1 \leq j \leq \beta$.*

Proof: Consider nodes in subtree 0 first. From the definitions of the $parent_{SBnT}$ and $children_{SBnT}$ functions, the k^{th} bit is the leftmost 1-bit of c, and the dimension of the edges connecting c to its children can be derived by complementing each of the leading 0-bits for which the index is preserved. Notice that if complementing the $(k+j)^{th}$ bit of c $(k+j < n-1)$ changes the index, then so does complementing each of the bits $\{k+j+1, k+j+2, \ldots, n-1\}$. So, the dimensions of the edges connecting to the children nodes of c form a contiguous set of dimensions (modulo n). For nodes in other subtrees, we consider $R^{index(c)}(c)$ instead, and the proof is similar. ▮

Corollary 11 *The longest path in a subtree of the root is the path corresponding to the path* $(00\ldots01),(00\ldots11),\ldots,(01\ldots11),(11\ldots11)$ *in subtree 0.*

Lemma 17 *With a scheduling of all data for one subtree during a single routing cycle and subtrees in order of increasing dimension of the edge through which they are connected, a node* c *receives the data for the subtree rooted at it during routing cycle* $index(c)+n-1-\alpha_c$ *with the first cycle being numbered 0. The total number of routing cycles,* $2n-2$, *is the minimum possible of the SBnT routing for* one-port *communication.*

Proof: We prove the lemma by labeling the edges of the SBnT. A labeling is valid, if for each node the labels of edges connecting to its children are distinct and greater than the label of the edge to the parent node. The smallest label is greater than or equal to 0. Define a labeling scheme for subtree 0 first. Edges are labeled according to their corresponding dimensions. Hence, the edge connecting to the parent of a node c with α_c leading 0's is labeled $n-1-\alpha_c$. This labeling is valid by lemma 16. For the other subtrees, we simply add $index(c)$ to the corresponding labels of subtree 0.

The maximum label is $n-1$ for subtree 0 by lemma 16 and corollary 11 and $index(c)+n-2$ for subtree $index(c)$. So, the maximum label of the SBnT is $2n-3$. Interpreting the labels on the edges as routing cycles, the proof is complete by noticing that the total number of cycles, $2n-2$, is the minimum possible since the root has n children and each subtree of the root is of height at least $n-2$. ▮

Note that the length of a routing cycle is determined by the data volume that needs to be transmitted over a single edge. The data volume depends on the number of nodes in the subtree connected through that edge. Different edges in a given dimension connects to subtrees of different sizes, unlike in a spanning binomial tree for which all tree edges in a given dimension connects to subtrees of the same size. To estimate the data transfer time we need to find the edge in each dimension that transfers the maximum number of elements, since each dimension is routed only once within a subtree.

Lemma 18 *Each of the edges forming the longest path transfers the maximum number of elements of any edge in that dimension.*

Proof: Consider subtree 0 first. Let T_j be the subtree rooted at the child of node c connected through an edge in dimension $(k+j) \bmod n$, $1 \le j \le \beta$, and S_j be the number of nodes in subtree T_j. Then the proof is complete by proving that $S_1 \ge S_2 \ge \cdots \ge S_\beta$. Every node in subtree T_{j+1} have addresses with bit $k+j$ equal to zero, bit $k+j+1$ equal to one, and the k least significant bits equal to the k least significant bits of c. By moving bit $k+j$ to the most significant bit, we map every node in subtree T_{j+1} to a unique node. Since the mapping preserves the index, the nodes in subtree T_{j+1} are mapped to subtree T_j. So, $S_{j+1} \le S_j$ and the proof for subtree 0 is completed by induction. For nodes in other subtrees of the root, we apply the arguments to the binary number $R^{index(c)}(c)$ instead of c. ▮

For the estimate of the element transfer time we need the following lemma. For proof of this lemma, see [6].

Lemma 19 *The number of n-bit binary strings for which the longest block of consecutive 0-bits (1-bits) has length* $\log n - \log\ln n - 1$, *or length greater than* $2\log n$, *is at most* $O(\frac{N}{n})$.

Theorem 6 *The communication complexity of an SBnT based* one-port one-to-all personalized communication *with a scheduling of all data for a subtree during a single routing cycle, and subtrees in order of increasing dimension of the edge through which they are connected is bounded from above by* $N(1+O(\frac{\log n}{n}))Mt_c + (2n-2)\tau$.

Proof: The root needs n routing cycles and a time of $(N-1)Mt_c + n\tau$ to send the data to its n children. The communication time required after the final routing cycle of the root is at most the same as the time required for communication in subtree 0. By lemmas 11 and 18 the communication time for subtree 0 is dominated by the path $(00\ldots01),(00\ldots11),\ldots,(01\ldots11),(11\ldots11)$. The data transfer time is

$$\begin{aligned}
&= Mt_c \times \sum_{i=2}^{n} (\text{the number of nodes with at least } i \text{ trailing 1's and } index = 0)\\
&= Mt_c \times \sum_{i=2}^{n} (i-1) \times \ (\text{the number of nodes with } i \text{ trailing 1's and } index = 0)\\
&\leq Mt_c \times \sum_{i=2}^{n} (i-1) \times \ (\text{the number of nodes with the length}\\
&\qquad\qquad \text{of the longest consecutive 1-bits being } i \text{ and } index = 0)\\
&\approx Mt_c \times \sum_{i=2}^{n} (i-1) \times \frac{1}{n} \times \ (\text{the number of nodes with the length}\\
&\qquad\qquad \text{of the longest consecutive 1-bits being } i)\\
&\leq Mt_c \times \frac{1}{n}(n \times O(\frac{N}{n}) + 2\log n \times N)\\
&= Mt_c \times O(\frac{N\log n}{n}).
\end{aligned}$$

Note that nodes with i trailing 1's and $index = 0$ may have a length of the longest substring of consecutive 1's greater than i, such as node (0000111101). So, the third equation is an upper bound of the second equation. The order of the fourth equation is the same as that of the third by theorem 3. The fifth equation follows from lemma 19. The first term in the parenthesis is a bound for the nodes below level $2\log n$ and the second term is a bound for nodes above level $2\log n$. The proof is completed by noticing that the last subtree needs only $n-2$ start-ups after the last routing cycle of the root. ▮

A lower bound for *one-port one-to-all personalized communication* is $\max(M(N-1)t_c, n\tau)$. Routing according to a spanning binomial tree and the above scheduling discipline is optimal within a factor of two. For *n-port* communication the lower bound is $\max(\frac{M(N-1)}{n}t_c, n\tau)$. The SBnT routing with a *reverse breadth-first* scheduling [5] is optimal within a factor of two. Routing according to a spanning binomial tree has a complexity which is $O(n)$ worse than the lower bound.

Comm.	one-to-all person. comm.	all-to-all broadcast	all-to-all person. comm.
one-port	$\leq N(1+O(\frac{\log n}{n}))Mt_c+(2n-2)\tau$	$(N-1)Mt_c+(2n-1)\tau$	$\frac{nNM}{2}t_c+(2n-1)\tau$
n-port	$\frac{(N-1)M}{n}t_c+n\tau$	$\frac{(N-1)M}{n}t_c+n\tau$	$\frac{NM}{2}t_c+n\tau$

Table 2: Complexities for some communication algorithms based on the SBnT routing.

The lower bound for *one-port all-to-all personalized communication* is $\max(\frac{nNM}{2}t_c, n\tau)$ [5]. A spanning binomial tree routing with an appropriate scheduling discipline is optimal within a factor of two. The minimum number of start-ups for the SBnT routing is approximately twice that of the binomial tree routing, but the data transfer time is approximately the same. For *n-port* communication the SBnT routing again is a factor of two within the lower bound $\max(\frac{NM}{2}t_c, n\tau)$, but the binomial tree routing is at least $O(\sqrt{n})$ worse than the lower bound [5].

The complexities for some communication algorithms based on the SBnT routing are summarized in Table 2.

6 Implementation Issues

We have implemented the SBnT^{minR} routing algorithm for personalized communication on an Intel iPSC Boolean cube configured multiprocessor with 64 nodes. In the SBnT routing, the root determines which node belongs to which subtree. If n is a prime number the subtrees are isomorphic (excluding node $(\bar{s}_{n-1}\bar{s}_{n-2}\ldots\bar{s}_0)$) and 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. The table entries point to the messages transmitted over port 0. The pointers for the other ports are obtained by (left) 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 other cyclic nodes. The period P_c for each table entry needs to be found, and the message divided into P_c pieces for the SBG routing.

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 $index((myaddress) \oplus (source))$ 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 index. If tables are used instead of a destination field, then for *postorder* scheduling [5] 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 *reversed-breadth-first* scheduling [5] 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 11.

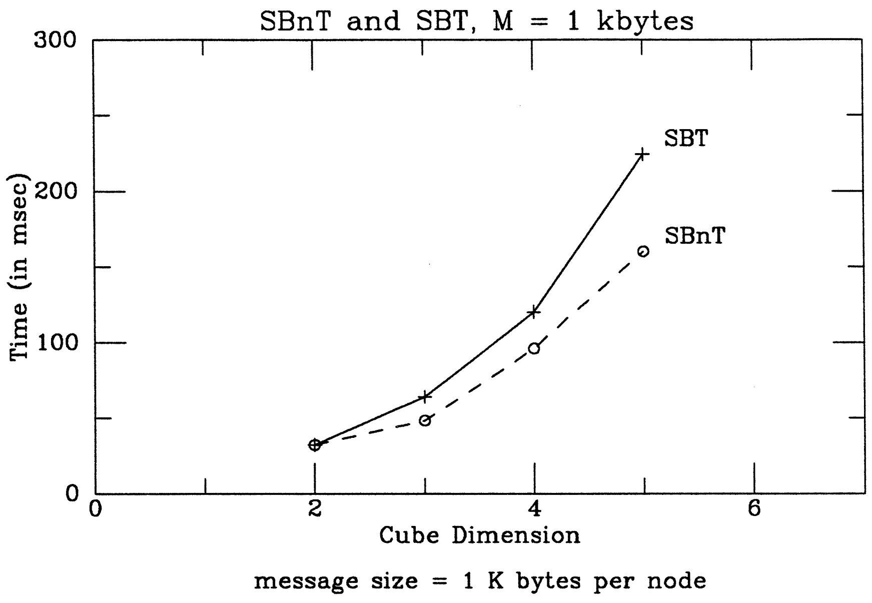


Figure 11: One-to-all personalized communication for the SBnT and SBT routings.

With communication restricted to one port at a time, the expected time of one-to-all personalized communication for the SBT routing and SBnT routing is the same for $B = M$. For *one-to-all personalized communication* based on the SBT routing we schedule port communications in a *binary-reflected* Gray code order to take advantage of the partial overlap in communication on different ports that is possible on the Intel iPSC. 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. 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 may not be finished with retransmitting the last packet received when a new packet arrives. In the SBnT, a subtree receives a packet once every n cycles.

7 Summary

The Spanning Balanced n-Tree (SBnT) allows for scheduling disciplines that realize minimum time (within a factor of two) one-to-all personalized communication, all-to-all broadcasting and all-to-all personalized communication on a Boolean n-cube with *n-port* communication [5]. The number of nodes in each of the n subtrees is $O(\frac{N}{n})$. The SBnT can be modified to a perfectly balanced graph by allowing multiple parents for cyclic nodes, i.e., splitting the data sets for such nodes. A few different definitions of *Spanning Balanced n-trees* are given, and shown to be essentially edge-disjoint. They are of particular interest with respect to fault-tolerant communications. Single edge failure, with the exception of the edges from the root, and several forms of multiple edge failures can be routed around, given that the failure is known and the proper SBnT is chosen.

We also show that for $0 < l < n$ the ratio of the number of degenerate necklaces to the total number of necklaces with precisely l bits equal to one is at most $\frac{1}{1+\max(\lceil \frac{n-l}{2} \rceil, \lceil \frac{l}{2} \rceil)} \leq \frac{4}{4+n}$.

Acknowledgement

This work has in part been supported by the Office of Naval Research under contracts N00014-84-K-0043 and N00014-86-K-0564.

References

[1] Ching-Tien Ho and S. Lennart Johnsson. Distributed routing algorithms for broadcasting and personalized communication in hypercubes. In *1986 Int. Conf. Parallel Processing*, pages 640–648, IEEE Computer Society, 1986. Tech. report YALEU/DCS/RR-483, May 1986.

[2] Ching-Tien Ho and S. Lennart Johnsson. *Matrix Transposition on Boolean n-cube Configured Ensemble Architectures.* Technical Report YALEU/DCS/RR-494, Yale University, Dept. of Computer Science, September 1986.

[3] D. Hoey and Charles E. Leiserson. A layout for the shuffle-exchange network. In *International Conference on Parallel Processing*, IEEE Computer Society, 1980.

[4] S. Lennart Johnsson. Communication efficient basic linear algebra computations on hypercube architectures. *Journal of Parallel and Distributed Computing*, 4(2):133–172, April 1987. (Report YALEU/DCS/RR-361, January 1985).

[5] S. Lennart Johnsson and Ching-Tien Ho. *Spanning Graphs for Optimum Broadcasting and Personalized Communication in Hypercubes.* Technical Report YALEU/DCS/RR-500, Yale University, Dept. of Computer Science, November 1986. To appear in IEEE Trans. Computers.

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