HYPERTREE
A MULTIPROCESSOR INTERCONNECTION TOPOLOGY

by

James R. Goodman and Carlo H. Sequin

Computer Sciences Technical Report #427

April 1981

HYPERTREE, A MULTIPROCESSOR INTERCONNECTION TOPOLOGY

James R. Goodman and Carlo H. Séquin

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, California, 94720

ABSTRACT

A new interconnection topology for incrementally expansible multiprocessor systems is described which combines the easy expansibility of tree structures with the compactness of an n-dimensional hypercube. The addition of n-cube links to the binary tree structure provides direct paths between nodes which have frequent data exchange in algorithms such as sorting and Fast Fourier Transforms. The derivation of a family of such Hypertree structures is outlined, and the basic properties such as average path length, uniformity of the distribution of message traffic and routing algorithms are analyzed.

KEYWORDS: Multiprocessors, Communication networks, Message traffic, Routing algorithms, Tree structure, Hypercube.

1. INTRODUCTION

One of the problems that is still impeding the emergence of general purpose multiprocessors is interprocessor communication. For effective cooperation of several processors on the same task, or for fast access to distributed data, high communications bandwidth is typically required. If all processors are simply connected onto one and the same bus, this shared resource becomes a bottleneck preventing simultaneous communication between different pairs of processors, and the effective throughput of the system may actually go down as the number of processors is increased. A suitable interconnection network is thus needed which provides as much bandwidth as possible between any pair of processors. The classical crossbar switch between separate banks of processors and banks of memories, requires a high pin-to-logic ratio, and is therefore not very amenable to VLSI exploitation. One possible approach is to combine one processor and its memory with one node of the switching network, thus creating a regular network of computers.

1.1. Classical Switching Networks

Many people have addressed the problem of switching networks. Much work has been done in particular to allow multiple, simultaneous connections between processor banks and memory banks to permit sharing of data or concurrent cooperation on the same tasks. Among the better known networks are lattice structures [Bouknight 72], the flip net [Batcher 76], the Omega net [Lawrie 75], the indirect n-cube [Pease 77] [Benes 65], the perfect shuffle [Golomb 61] [Stone 71], the augmented data manipulator [Feng 74], the de Bruijn net [Schlumberger 74] [de Bruijn 46], the generalized connection network [Thompson 78], the Banyan partitioner [Goke 73], and the n-cube network [Batcher 76] [Siegel 77].

It is understood that one of the chief properties of many of these networks is the efficient interconnection of nodes in the n-dimensional hypercube, or n-cube, configuration. The n-cube is particularly compact. The worst case distance between any two nodes is only n, the dimension of the structure, and nodes which differ in their node addresses by only one bit are direct neighbors. In addition, this logical structure is extremely useful because of the wide range of algorithms that fit it particularly well. For many problems such as Fast Fourier Transforms or sorting, it is possible to map the logical structure of the problem directly onto the physical structure, and large quantities of data are exchanged between nearest neighbor pairs.

For the more general switching network, where many arbitrary pairs of elements may wish to communicate simultaneously, the n-cube topology is also suitable. The routing algorithm to reach a particular node is trivial: For every bit in the target node address which differs from the current node address, a corresponding link that complements that bit has to be traversed.

1.2. Tree-Structured Networks

The emergence of a technology for very large scale integration (VLSI), which within five years will allow the fabrication of a single silicon chip containing approximately one million active devices, also changes the constraints on a multiprocessor interconnection network. With the advances of this technology, the active devices themselves get smaller, faster and consume less power. Unfortunately, at the same time the disparity increases between the speeds of signals completely internal to the chip and signals which have to pass through the package pins. Proper systems partitioning onto several chips becomes ever more important, and high bandwidth signal paths should be kept completely internal to the silicon chips as far as possible. Thus a proper VLSI building block may be a self-contained computer on a single integrated circuit [Sequin 79].

Once a complete computer fits on a chip, such computers can be placed at all interconnection points of a switching network. This makes particular sense with VLSI technology since the switch by itself tends to have a low logic-to-pin ratio, making it unattractive to VLSI. The switching network then is no longer the link between a discrete bank with p processors and a separate bank with m memory modules, and we no longer need to consider only switching networks with p connection points on one side and m ports on the other. A whole set of networks such as lattices or trees can now also be considered. Since we believe that future computing systems should be easily expansible these latter structures look even more attractive. It is therefore not surprising that recently a lot of interest has been generated in tree-structured networks. [Swan 77] [Despain 78] [Mago 79] [Browning 79].

1.3. Requirements for a new structure

The disadvantage of the previously mentioned n-cube network, from this point of view, is the fact that it is not truly expansible. Whenever the number of nodes grows beyond a power of two, all nodes have to be changed since they have to be provided with an additional port. Thus the "module" of this network is not a constant, predefinable building block. Moreover, a useful expansion of this structure has to occur by doubling the number of nodes. An incompletely populated n-cube lacks some of the above mentioned properties which make the n-cube attractive in the first place. Tree structures are expansible in a natural way, and even unbalanced trees still retain most of the properties that make trees attractive.

The number of package pins does not grow at the same rate as the number of active devices within. The package periphery thus represents another physical bottleneck, and we should look at interconnection topologies which need relatively few ports per node. If the number of ports per node has to be limited,

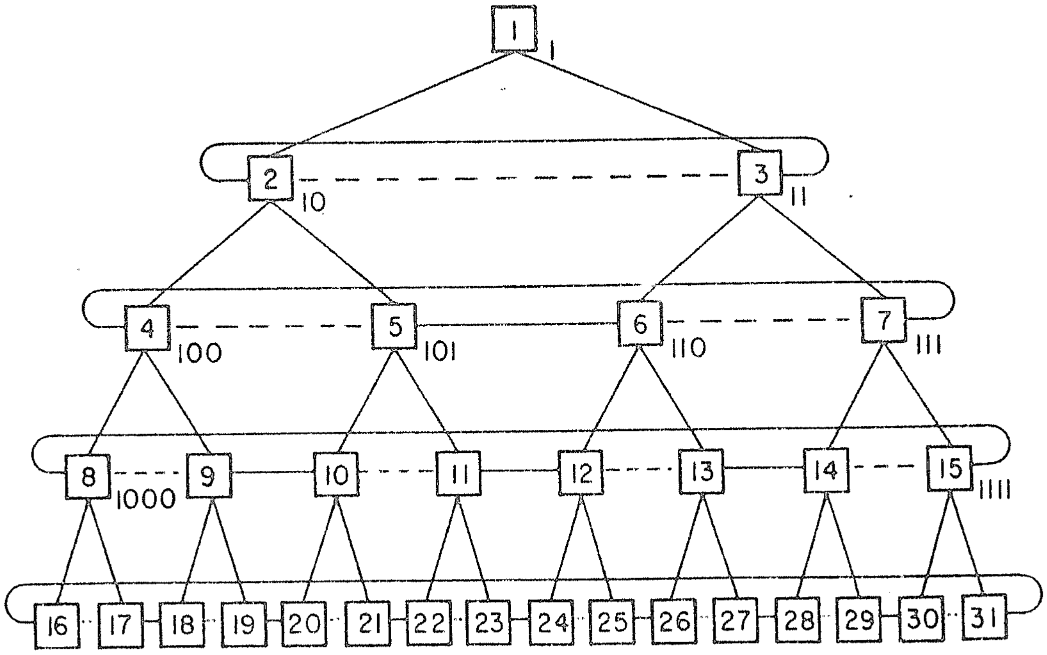


Figure 1. Binary tree with full-ring connections. When the dashed branches are omitted, a half-ring tree is obtained. The chosen numbering scheme gives all nodes on the same level addresses of the same length. The root of the tree has node address "1", and the children of node x have node adresses 2x and 2x+1 respectively.

a binary tree structure, requiring only 3 ports per node, looks particularly attractive. Additional links, however, are required to provide redundant paths which are the basis for a system with some fault tolerance in message routing. At the same time these links can be used to reduce the average distance between nodes and to provide a more uniform message density in all links. An extensive search for the optimal placement of these additional links has shown the half-ring and full-ring binary trees (see Fig 1) to be attractive contenders, primarily because of their simple routing algorithms [Sequin 78].

For tasks in which remote leaves have to communicate extensively with one another, however, the ringed tree structures have their disadvantages. There is a lack of direct long distance connections at the lower levels of the tree. This forces messages between remote leaves to travel rather high up in the tree in order to reach their target along the shortest paths, which in turn can lead to serious congestions of some of the links below the third level of the tree. It is conjectured that a more optimal placement of the additional links could alleviate those problems. Hypertree is the result of the search for such a structure. It combines the best features of the binary tree and of the n-dimensional hypercube.

2. THE HYPERTREE STRUCTURE

The basic skeleton of Hypertree is a binary tree structure. In the following we will assume that the nodes are numbered as shown in Fig. 1. The root has node address 1. Addresses of left and right children are formed by appending a "0" and "1" respectively to the address of the parent node; i.e. the children of node x are numbered $2x$ and $2x+1$. As in the half-ring or full-ring structures, additional links in Hypertree are horizontal, and connect nodes which lie in the same level of the tree. In particular, they are chosen to be a set of n-cube connections, connecting nodes which differ by only one bit in their addresses. We will discuss the cases of nodes with four and five ports, permitting connections of one and two additional horizontal links at each node, respectively. Thus each level of the tree can accommodate either one or two sets of n-cube connections, and the resulting structures will be called Hypertree I and II. It should be obvious from the following description how the concept can be expanded to more than two sets of n-cube connections per level.

2.1. Selection of n-cube Interconnections

With only one or two ports per node available for the n-cube connections, a choice has to be made as to which set should be chosen at each level. In a heuristic manner we start at the root of the binary tree and progress in a top down manner. We select at each level the set(s) of connections from which the most benefit can be derived: For each pair of nodes for which the addresses differ in exactly one bit, the path length, expressed in number of links, is calculated in the Hypertree structure existing above this level. Figures 2 and 3 show these distances for a binary tree. The longest connection(s) on each level are circled, and in these places a direct n-cube interconnection is introduced. Any such connection will effectively reduce the number directly below it to the value 3 , because the corresponding connections in this lower level can now be made by traveling up one level, accross the n-cube connection and down again. Correspondingly the number two levels below an n-cube connection can be no larger than 5, and so on. This has to be taken into account in determining the longest path between n-cube pairs on each level. This path is then bypassed by the addition of the corresponding set of n-cube connections. While it is clear that this selection procedure will result in a local optimization, it will be shown later, that the selected n-cube interconnections also optimize the overall tree structure in a global sense.

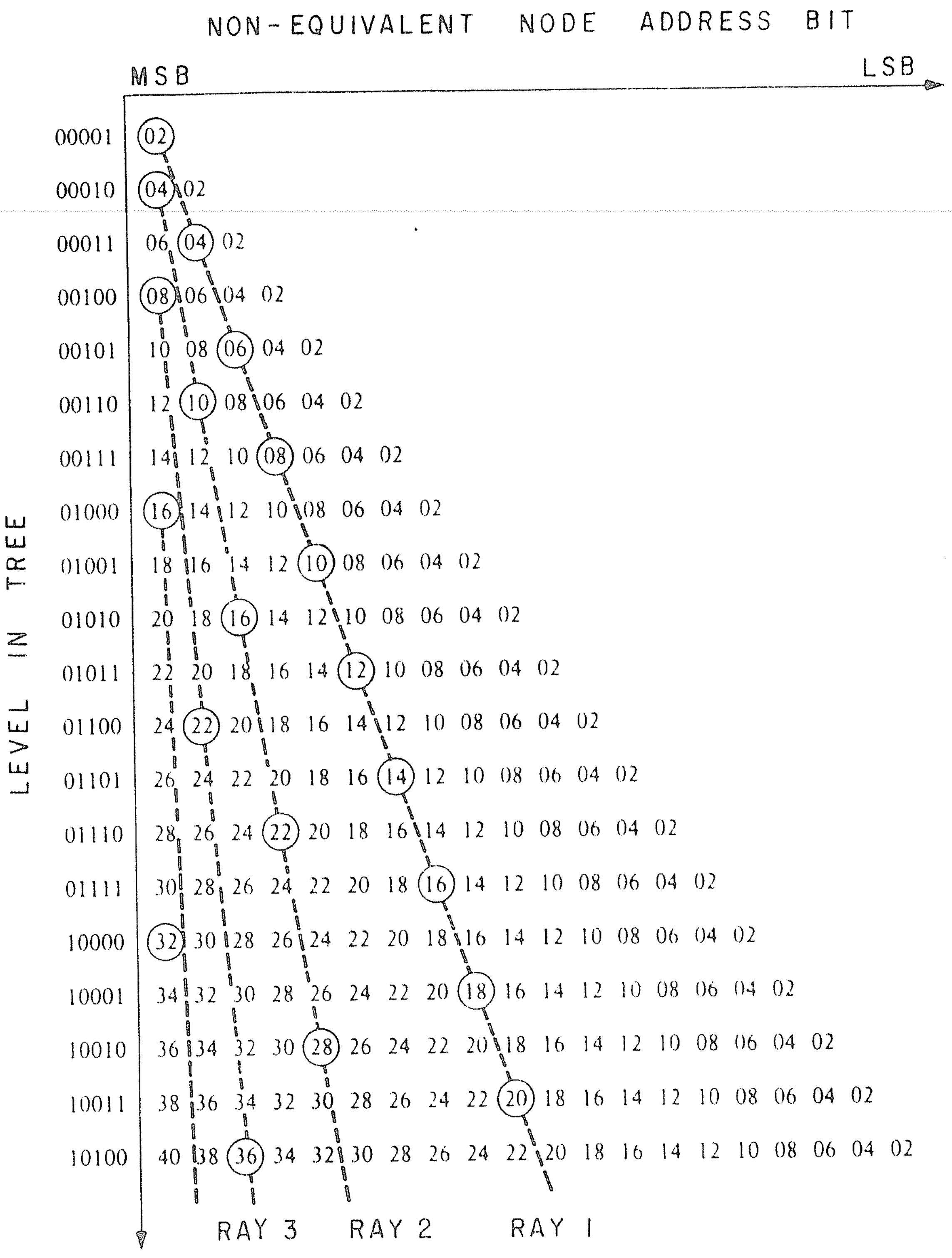


Figure 2. Selection of n-cube connections in Hypertree I. Shown are the distances between nodes on the same level which differ only by one bit in their addresses for the ordinary binary tree. Each circle represents a set of n-cube connections, chosen in such a manner that the longest distance between a pair of nodes with Hamming distance 1 at that level gets reduced to one.

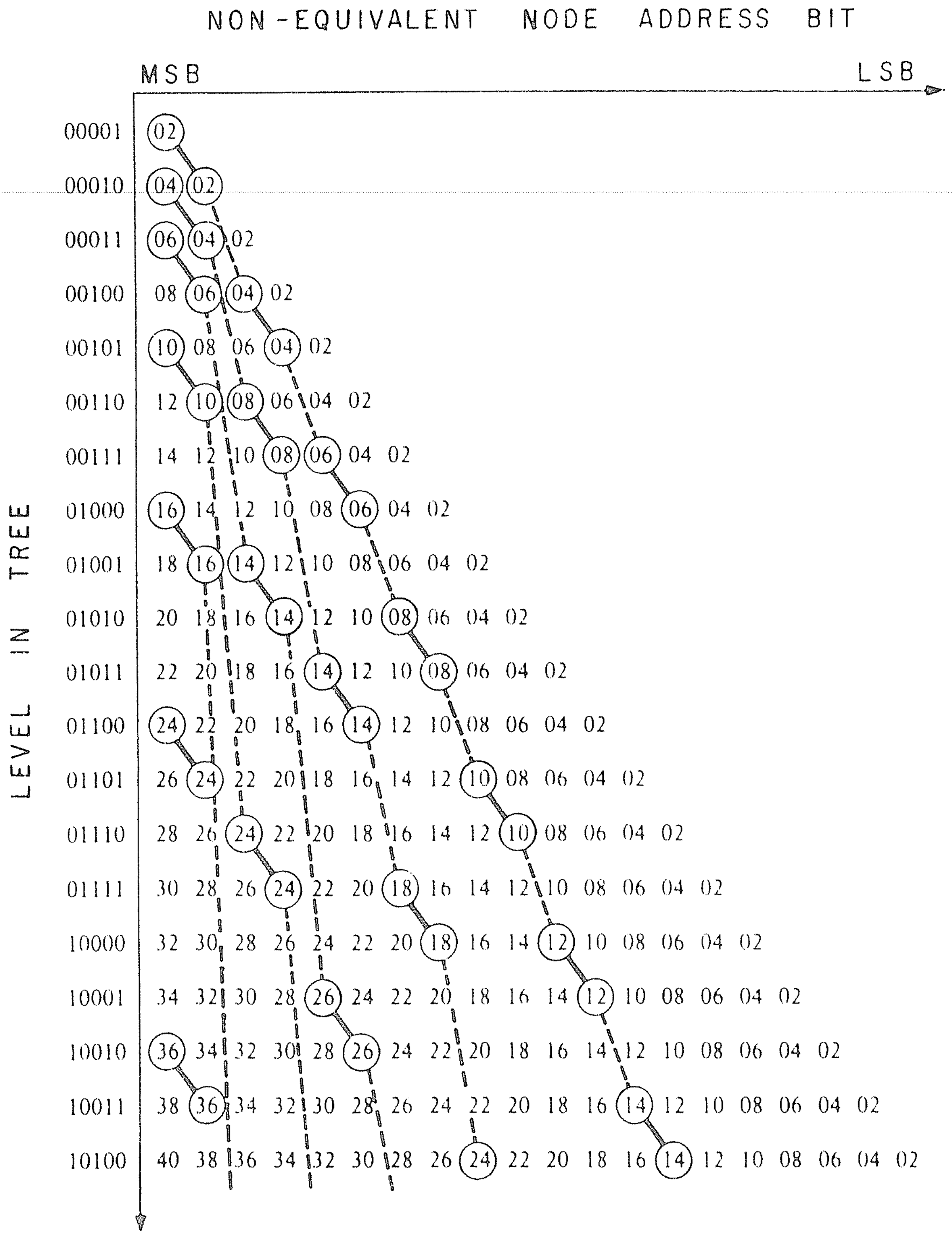


Figure 3. Selection of n-cube connections in Hypertree II. Each circle represents a set of n-cube connections as in Fig. 2.

In Hypertree II there are occasionally levels where a choice exists for the placement of the horizontal link, when two longest paths are of equal length. However, even from the links which are placed with no free choice, a regular pattern of these interconnects appears, and the choice is always made in order to follow this pattern. For Hypertree I all n-cube connections fall on slanted lines or "rays", starting at the location of the most significant bit on levels which are a power of two (Fig. 2). The slope of subsequent rays doubles and is equal to twice the level number on which it starts. For Hypertree II the pattern is more complicated. All n-cube connections appear in diagonal pairs, which themselves fall again on slanted lines (see Fig 3). The resulting interconnection pattern for a small Hypertree I is shown in Fig. 4.

2.2. Formal Description of Hypertree I

If we number the levels downward, starting with the root as 0, we notice that the level number, n, for each node is the number of binary digits after the most significant one in its address. An inspection of Fig. 2 shows that the number of consecutive trailing zeros, z, in the level number, expressed in binary, defines the ray number,

$$r = z + 1, \tag{1}$$

on which the set of n-cube connections on this particular level lies, e.g. on level 10100 the n-cube connections belong to ray 3. The specific bit b which is affected by the n-cube connections at level, m, can then be determined by the intersection of the proper ray with this level. For ray r,

$$b = \frac{m}{2^r} + \frac{1}{2} \tag{2}$$

where b is the bit number, counting from the left with the redundant, most significant "1" in the node address counted as bit 0.

Several observations can be made which follow an inspection of Fig. 2. "Ray

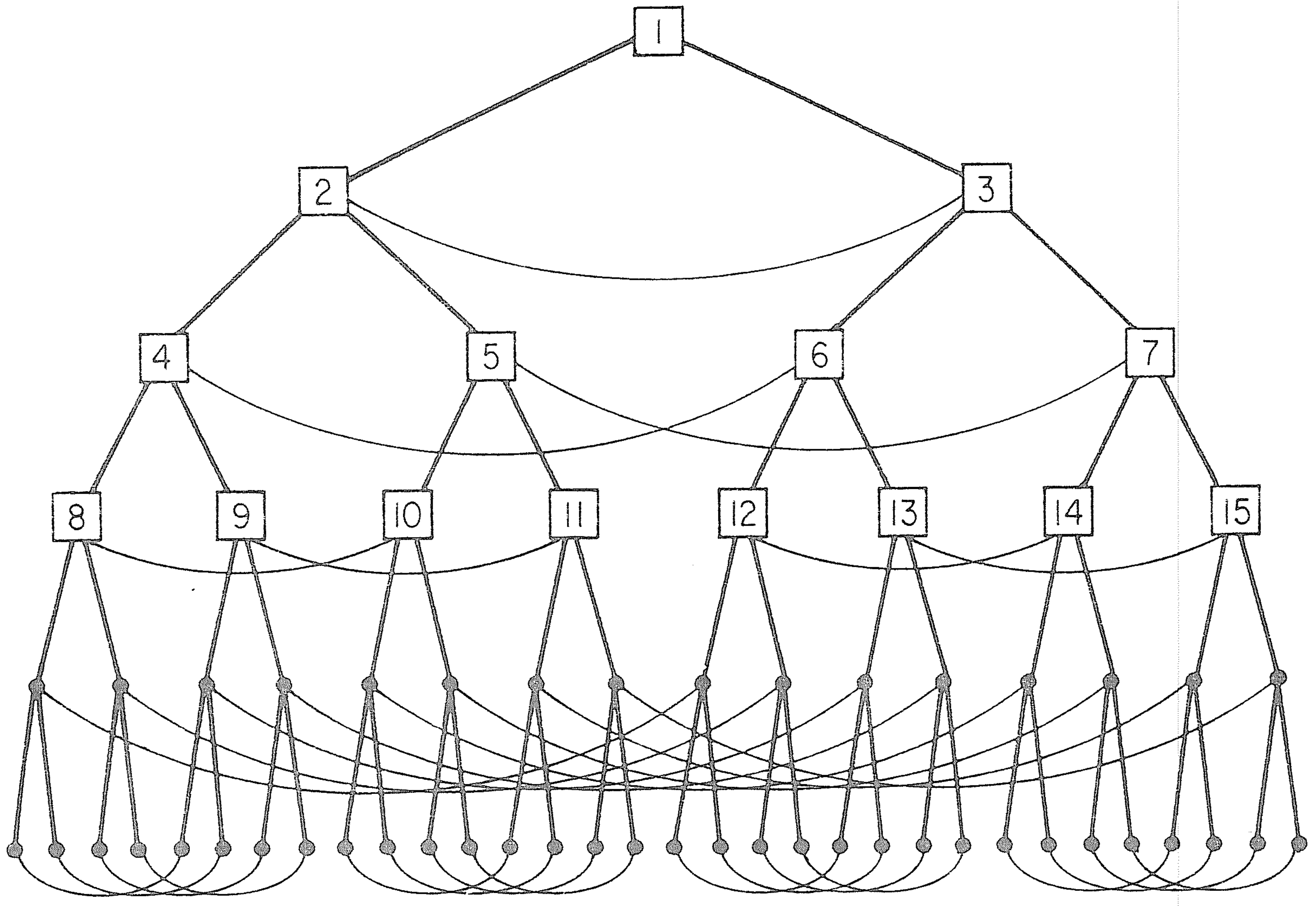


Figure 4. Interconnections in Hypertree I.

level	Hypertree I above	Hypertree I whole	Hypertree II above	Hypertree II whole
1	1	1	1	1
10	2	2	1	1
11	3	3	2	2
100	4	3	3	3
101	5	4	3	3
110	6	5	4	3
111	7	5	5	3
1000	8	6	5	4
1001	9	7	6	5
1010	10	7	7	5
1011	11	8	7	5
1100	12	9	8	5
1101	13	9	9	6
1110	14	10	9	7
1111	15	11	10	7
10000	16	11	11	7
10001	17	12	11	7
10010	18	13	12	8
10011	19	13	13	9
10100	20	14	13	9

Table I. Worst case distances between nodes with Hamming distance 1 for Hypertree I and II. Two cases are considered: "above", where messages only travel above the two nodes, as is appropriate for leaf-to-leaf traffic, and "whole", where messages can find the best path above or below, assuming an infinitely large tree.

1" goes through the middle bit of the significant part of the node address on every level with an odd number of significant bits (excluding the redundant leading one). This means that the n-cube connections always complement a bit in the more significant half of the node address, while the structure relies upon the skeleton of the binary tree for changes in the less significant half. In addition it turns out that for a tree of any size, exactly one representative of each such an n-cube connection is found in the lower half of the tree. Since the lower half of the tree also contains all the necessary connections to make any change in the least significant half of the address, it follows that messages originated at a leaf node need never travel higher than half the height of the tree in order to reach any other leaf.

3. PROPERTIES OF HYPERTREE

In this section some of the relevant properties of Hypertree as an interconnection network will be discussed and compared to other previously described multiprocessor topologies.

3.1. Worst Case Distances

One important measure of the power of an interconnection network is the distance messages must travel in the network. It is advantageous to make this parameter as low as possible, since it will not only reduce travelling time for messages, but also minimize message density in the links. Absolute worst case distances between any two nodes in Hypertree I can be based on the observation made in section 2.2: it is equal to m, the number of levels above the lower of the two nodes. Worst case distances between n-cube pairs can be determined by inspection of Fig. 2 and 3 and are summarized in Table I. For Hypertree I and II worst case distances are listed for a) the case where only connections higher in the tree can be used, as is the case for leaf-to-leaf communication, and b) the case where we assume to be in the middle of a large tree, so that messages can find suitable n-cube connections above as well as below the current level. For Hypertree I the longest distance is 1/2 and 1/3 of the distance in a simple binary tree, and for Hypertree II they are reduced to 1/3 and 1/5, respectively.

3.2. Average Distances

Of even more importance may be the *average* path length travelled by all messages. In order to obtain a meaningful comparison between different networks, some normalization has to be made, in particular if networks between processors with different numbers of ports per node are considered. With no limits on the number of ports, a fully interconnected network could be designed, which would lead to an average distance of one. In order to take into account realistically the limitations in the number of pins and in the amount of power available to drive communication lines, Despain [Despain 79] has proposed that, in the context of single-chip computers, a constant communications bandwidth per node should be assumed. Under this assumption, the bandwidth available through each port is then B/p, where p is the numer of ports and B is the total bandwidth available from that processor. Using the same idea, we propose that the average message path length, L, be normalized by multiplying it with the number of ports, p, in order to permit a meaningful comparison of graphs of different degrees. Thus the normalized average message path length, L′, then becomes

$$L' \equiv L \times p. \tag{3}$$

Another factor influencing the average message path length is the distribution of pairs of communicating nodes. In the absence of any specific information about the communication patterns required by particular task, one might assume a uniform distribution in which all nodes send messages with equal probability to all other nodes. More appropriately, for tree-structured networks

Structure	L	L′
Binary Tree	$2n - 2 + \frac{2}{N}$	$6n - 6 + \frac{6}{N}$
Half-Ring*	$2n - \frac{139}{32} + \frac{31}{4N}$	$8n - \frac{139}{8} + \frac{31}{N}$
Full Ring†	$2n - 5 + \frac{8}{N}$	$10n - 25 + \frac{40}{N}$
n-Cube	$\frac{n}{2}$	$\frac{n^2}{2}$
Hypertree I†	$\frac{5n}{4} - \frac{4}{3} + \frac{4}{3N} - \frac{n \bmod 2}{12}$	$5n - \frac{16}{3} + \frac{16}{3N} - \frac{n \bmod 2}{3}$

Table II. Average distance between nodes for various networks as a function of the number of leaf nodes $N \equiv 2^n$. L is the average distance between every pair of nodes (including each node with itself). L' is the normalized distance obtained by multiplying the average distance by the maximum number of ports per node. traffic.

---

* for n>2.

† for n>1

where all I/O points and all secondary memory are connected to the leaves of the tree [Despain 78], a set of messages running between all pairs of *leaf* nodes will be studied.

3.2.1. Leaf-to-leaf traffic

Table II shows the average distances between leaves for the binary tree, the half-ring tree, the full-ring tree, Hypertree I, and the average distance between all nodes for the n-cube. The n-cube is assumed to have $N \equiv 2^n$ processors, and all tree structures have $N \equiv 2^n$ leaves and thus a total of $2N-1$ processors. The derivation of the average path length is trivial for the binary tree and the n-cube, but for Hypertree it is rather tedious and thus deferred to appendix A. Average distances for the half-ring and full-ring trees have been determined through an exhaustive count on the computer. Table II includes the normalized average distances, L′, for the same structures, which are also plotted in Fig. 5 as a function of n. Note that the normalized average path length in Hypertree is always shorter than in the binary tree, and eventually becomes even less than that of the n-cube.

3.2.2. N-cube nearest neighbors

The n-cube interconnection, i. e. the paths between nodes that have a Hamming distance of 1, can play an important role in problems such as Fast Fourier Transforms and sorting. These connections are the basis for many interconnection networks in SIMD architectures [Batcher 73] [Siegel 76]. It is therefore worthwhile to study these special interconnection paths in the above structures. For balanced binary trees we use the node numbering scheme of Fig. 1, so that the leaf nodes are numbered from 2^n to $2^{n+1}-1$. We then define n-cube nearest neighbors to be those pairs of leaf nodes which have a Hamming distance of one, and assume uniform traffic among them. The average distance between those

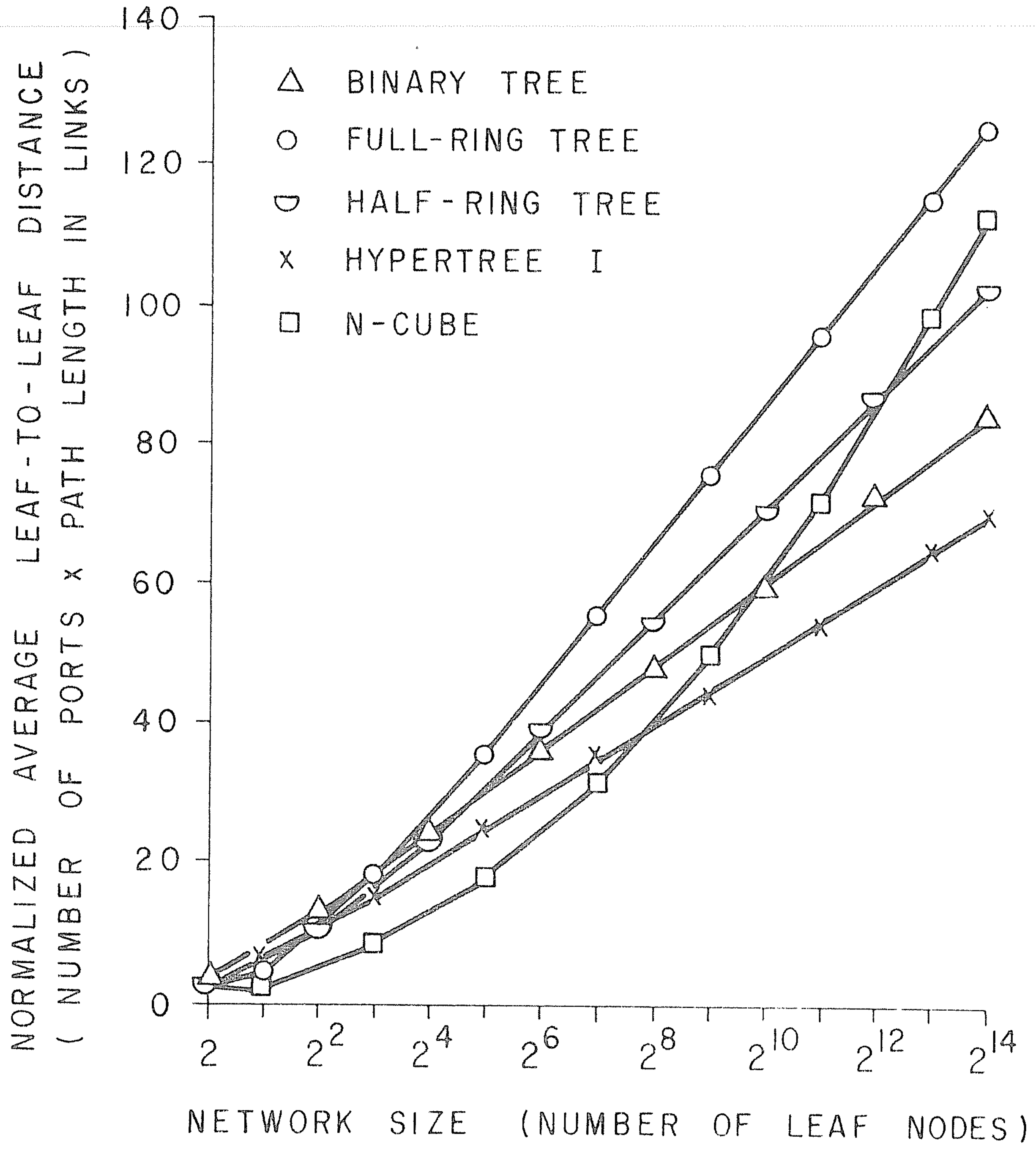


Figure 5. Average path length in various structures as a function of the number of processors. It is assumed that every leaf sends with equal likelihood to every leaf.

n-cube nearest neighbor nodes for the binary tree turns out to be n. For Hypertree I, this value is n/2, while for the n-cube, of course, it is 1. Normalizing these numbers as before with the number of ports per node gives:

binary tree:	3n
Hypertree I:	2n
n-cube:	n

Thus the Hypertree I structure is only a factor of two worse than the n-cube, which, in this particular case, is the ideal structure. For the binary tree the above numbers are valid also for n-cube nearest neighbors within the tree, while for Hypertree the number given is only an upper bound; paths through the lower parts of the tree may provide a more direct connection.

3.2.3. Consideration of locality in traffic

It seems reasonable to assume that an efficient and practical multiprocessor system will exhibit much greater traffic over short distances than over long communication paths, since interaction among small clusters of processors may cause a large portion of the total traffic. This may result from the fact that tasks that can be broken up into smaller subtasks would normally be assigned to neighboring processors. Furthermore, the way that multiple processors are interconnected often reflects the need of the communication patterns in the first place and takes into consideration the fact that communication costs increase with distance. Under such conditions, the effective average message path length may be much shorter than the numbers given in Table II.

Without such locality in the message traffic, the amount of message communication handled per processor will drop off in most multiprocessor networks as the number of nodes increases. This becomes clear from the following analysis: We assume a system with N nodes with a uniform message density close to the limit defined by the bandwidth of the links so that throughput is limited by communication bandwidth. Given the constant bandwidth assumption, if the number of nodes in such a system doubles, and we assume that all nodes communicate equally with all others, then the number of messages sent among the processors quadruples, while the number of links in the system has only doubled. Since the number of nodes that a processor can address has also doubled, the message rate between any specific pair of processors can be at best half the rate in the original system. This rate however can only be achieved if the average path length remains the same. Any increase in the path length will result in a further reduction in the message rate, so that each processor is actually sending and receiving fewer messages. With sufficient locality the longer paths obtain a much lower weight, and thus the increase in average distance can become insignificant.

We have been unable to derive a locality function for any of the enhanced tree structures for which a closed form expression of the average path length could be determined. To get some qualitative understanding of how locality affects the increase of path length with increasing tree size, we will take a look at some crude and extremely simplistic models.

As one extreme, we can assume that communication between pairs of nearest neighbors is most important. In order to provide highest bandwidth for this case, within the constraints of constant total bandwidth per node, the degree of the graph of the interconnection network should be chosen as low as possible. In the extreme this would lead to a set of isolated pairs of nodes, and the average path length then remains constant at the value 1.

In order to get some approximation for a more practical situation, we will assume an arbitrary network in which each node sends one half of its maximum possible message traffic to nodes which are one link away. Half that many messages are sent to nodes at a distance of two links, and in general an amount of traffic $T(d)=T_{\max}2^{-d}$ is going to nodes at distance d away from the sender. For a graph of infinite extension, the average message path length then becomes 2, while for graphs with a finite diameter k, the average path length will be $2-2^{-k}$. Thus, with the above assumtions on locality in the message traffic, the average path length is almost independent on the size of the network. Thus a primary goal must be to try to map a problem onto the topology of the network, so that the direct connections are used as often as possible, or alternatively, to provide a network with a structure of the local interconnections for which this is easily possible for many important problems.

3.3. Routing Algorithms

One of the desirable requirements for a large network of processors is that messages can be routed by each intermediate processor without total knowledge of all the details of the network, since the storage of that information within each node can use up an exorbitant amount of memory space. Both, the n-cube and the binary tree have simple routing algorithms which involve only the addresses of the current location of the message and of the target processor. In the n-cube, links are selected which reduce the Hamming distance by one, until the target is reached. In the binary tree a message is routed upwards in the tree (towards the root) until the target node is a descendent of the current node; from there it is routed down. Both the half-ring and full-ring tree structures also have simple routing algorithms which are optimal in the sense that they always find the shortest path [Sequin 78]. Both follow basically the binary tree algorithm, but use the horizontal links whenever the path length can be reduced.

3.3.1. A simple routing algorithm

Similar algorithms exist for the Hypertree structures. They are optimal for messages between leaf nodes in a balanced tree. A simple routing algorithm works as follows:

As in the basic binary tree, messages are routed upwards in the tree until the target is a direct descendent of the current position. In addition, whenever the n- cube connections at a particular level reduce the Hamming distance between the current position and the target, the corresponding link is traversed.

This algorithm will result in a path in which all useful n-cube connections have been traversed before the message has reached the highest point of its path, even though any particular n-cube connection could also have been utilized during the downward phase of the message routing.

3.3.2. Optimal routing algorithm

There are circumstances under which the simple routing algorithm will not lead to the shortest path.

1) For messages between non-leaf nodes, the use of n-cube connections below either source or target node may lead to a shorter path. The simple algorithm will not find these paths since it makes no assumptions about n-cube links below the current node.

2) A similar observation can be made even for traffic between leaf nodes, if the tree is unbalanced. If the target lies below the source, an n-cube link below the target could be used for the shortest possible connection. The simple algorithm may search much higher in the tree for a corresponding interconnection.

3) If the the Hypertree structure is incomplete, and a particular n-cube link is missing, the corresponding bit could be complemented by taking another n-cube link of the same set on the way down. The above algorithm in its simplest form

is not aware of this possibility.

Under the assumptions of a balanced, complete Hypertree, we were able to derive a routing algorithm which always selects a path of minimum distance. It is too complex, to be described here, but the point should be made that an optimum algorithm exists which uses only local information and knowledge of the size of the tree.

An algorithm able to find the optimal path in an unbalanced tree must have knowledge about the extension of the frontier, i.e. the position of the leaves, of the tree. For the same reason, in order to find the shortest path, it would also need global knowledge of any missing links, be it because the tree is incomplete by design or because certain links have failed arbitrarily. Simulation studies have demonstrated that in balanced trees with up to eleven levels (4095 nodes), the simple routing algorithm yields an average path length for all pairs of nodes which is never more than 0.42% greater than the optimal path length. The slight potential improvement must be weighed against a much more complicated algorithm, requiring more detailed global information about the network, and a more complicated routing controller.

3.4. Message Density

Another major goal in the design of an efficient network topology is to distribute traffic as evenly as possible over all existing links. The basic binary tree and the fullring tree both have serious deficiencies in that respect. The binary tree suffers intolerable congestion near the root, since roughly half of all traffic must pass through the highest links in the tree if no locality exists. A similar bottleneck exists in the horizontal links on the third level of the full-ring tree if the simplest routing algorithm is used. Some of the same kind of bottlenecks exist even in the Hypertree, but to a much lesser degree, since, as has been shown in Section 2.2, leaf-to-leaf messages never go more than half way up the tree. The fact that messages are kept in the lower, wider parts of the tree, reduces congestion considerably.

Fig. 6 shows the maximum number of messages routed through the busiest link for the n-cube, the binary tree, half-ring and full-ring tree, and Hypertree as a function of tree size. The data has been normalized by multiplying the derived values with the number of ports per node. For the binary tree, the busiest links are the two top ones, while for the n-cube, all links carry an equal amount of traffic. For Hypertree I with 2^n leaf nodes the horizontal links $\frac{n+1}{2}$ levels below the root are the busiest, if n is odd. If n is even, the load is equally heavy on horzontal links at level $\frac{n}{2}+1$ and on the vertical links immediately above that level. The full-ring tree has a seroius bottleneck at the horizontal links on level 3, and the half-ring has almost as much congestion on the vertical links above level 3. Of all the described tree structures, the Hypertree stucture is clearly the best in that respect. While in the other tree structures the maximum traffic density grows roughly as the square of the number of nodes in the tree, in Hypertree I the growth rate is only $N^{1.5}$.

It should be pointed out, that even for n-cube, communication density grows beyond any fixed bound, making large systems inefficient without the reliance on locality in the message traffic. Similar trends as demonstrated for average path length exist, making the maximum message density rather independent of the size of the network with certain models of locality. Since it is not obvious what a proper model for locality is unless the mapping of a particular application onto the network has been worked out in detail, specific calculations of maximum traffic densities other than with uniform message traffic have not been pursued at this point.

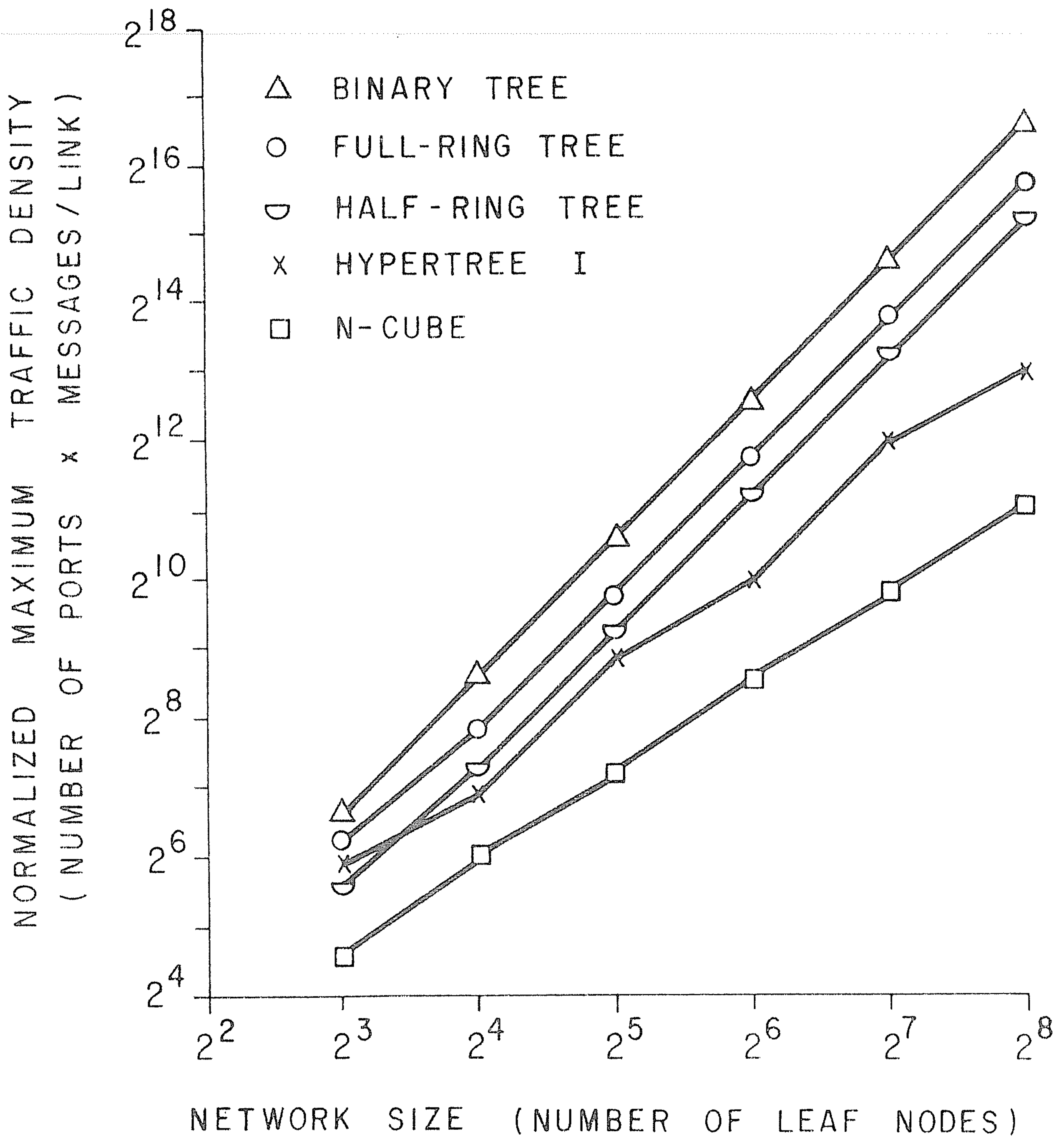


Figure 6. Maximum number of messages routed through a single link as a function of network size for uniform leaf-to-leaf traffic. Numbers have been normalized by multiplying them with the number of ports per node required.

4. PRACTICAL CONSIDERATIONS

4.1. Expansibility

Among the important parameter of a multiprocessor system are its modularity, expansibility, and specifically the smallest increment by which the system can be expanded in a useful way. It is generally unreasonable to demand that a system must remain balanced in all stages of expansion, since this may imply that its size must be increased in large steps, i.e. powers of two. The shortcomings of n-cube with respect to modularity and incremental expansibility have been discussed in Section 1.3.

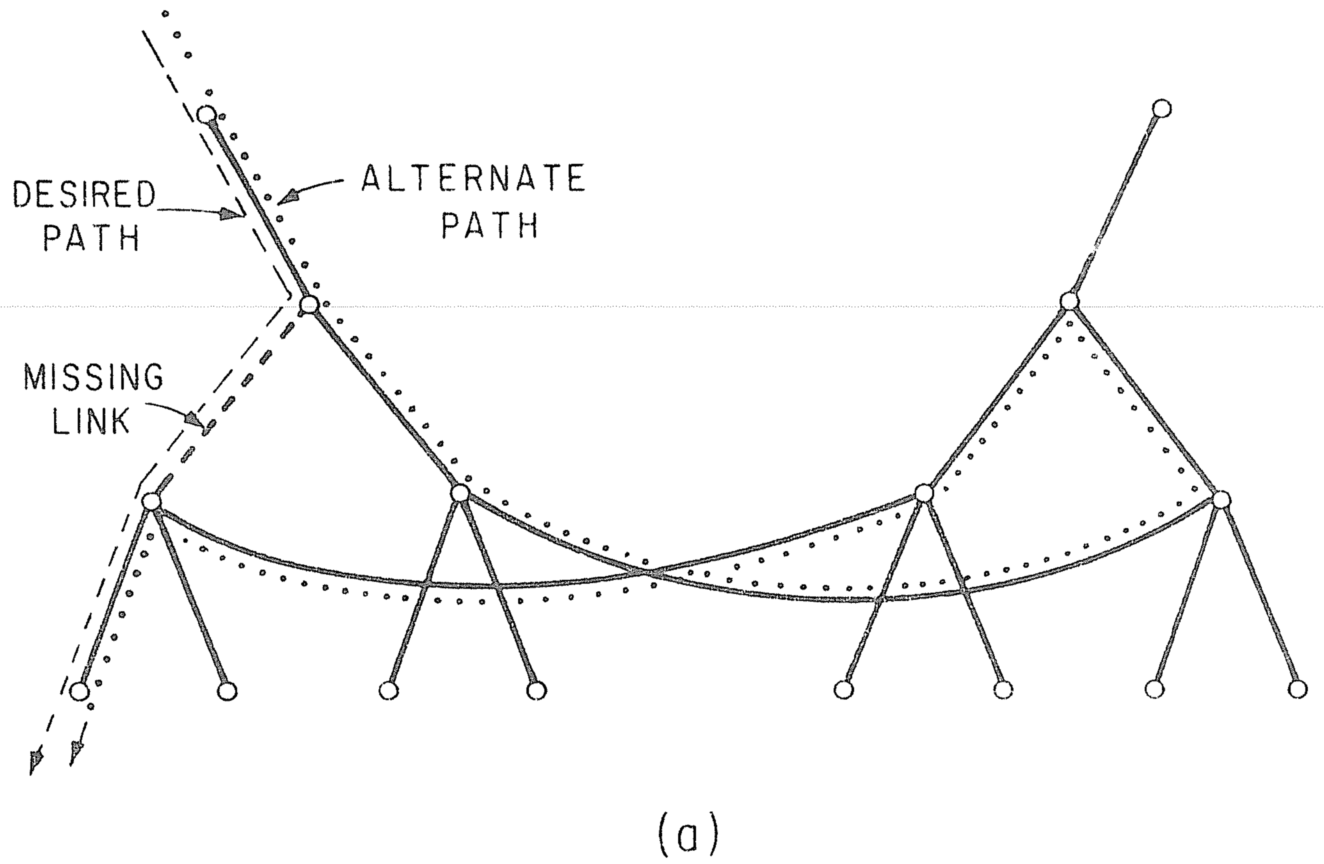
Binary trees also require a doubling of size in order to remain balanced. However, tree structures do not lose too much of their attractiveness if they are not perfectly balanced. The routing algorithms discussed still reach their target in an efficient way even if the tree is somewhat unbalanced, as long as the skeleton of the binary tree is still present. Although routing in such a system may not be optimal for all circumstances, such a structure still has its usefulness if it reflects the nature of the problems that it is handling, and provides the links as required by the nature of the communication patterns in that context.

Two basic approaches could readily be imagined by which Hypertree is expanded in a "natural" way. If the particular installation is a single, cohesive system, additional nodes should be placed on the last incomplete level, before a new level is started. The imbalance will then never exceed one level, an amount which causes no problems for the routing algorithm. On the other hand, if the system needs to grow organically with the needs of a widespread and diverse user community, many individual subtrees may develop which are connected through a shared main tree containing the root of the system. In both cases, n-cube interconnects could be added as soon as both the nodes, differing only in the corresponding bit that is to be complemented at that particular level, are present. However, the sequence of n-cube interconnects on subsequent levels, as derived in section 2.1, is really designed for a single cohesive system. If the system contains many disjoint subtrees, connected in only a few points to a main tree, it may be better to start a new sequence of n-cube interconnects within each subtree, thereby optimizing local communications. This would be worth the price of storing the list of these sets as a function of the level in each routing controller. Since the lack of interconnection between individual subtrees is a consequence of the way the system has been expanded, it can be assumed that it reflects an absence of the need to communicate between nodes belonging to different subtrees. Thus the lack of those horizontal links should have no adverse effect on the overall performance of the system or on the effective average pathlength.

4.2. Fault Tolerance

The requirements for fault tolerance have greatly increased in recent years as systems have become increasingly complex. Certainly an important feature of structures such as those considered here is that it must continue to work correctly, though perhaps with reduced performance, when one or more components have failed. Specifically we expect such a system to continue to operate properly in the presence of failures of a single link or even a single node with all its attached links, as long as that particular node is not involved in the computation, i.e. is neither the source nor the destination for any messages.

The addition of the n-cube links to the binary tree creates multiple disjoint paths between every pair of nodes. The simple algorithm described in the previous section already has some fault tolerance. It can readily cope with missing links during its upward move through the tree, by using the following alternate choices for missing links:



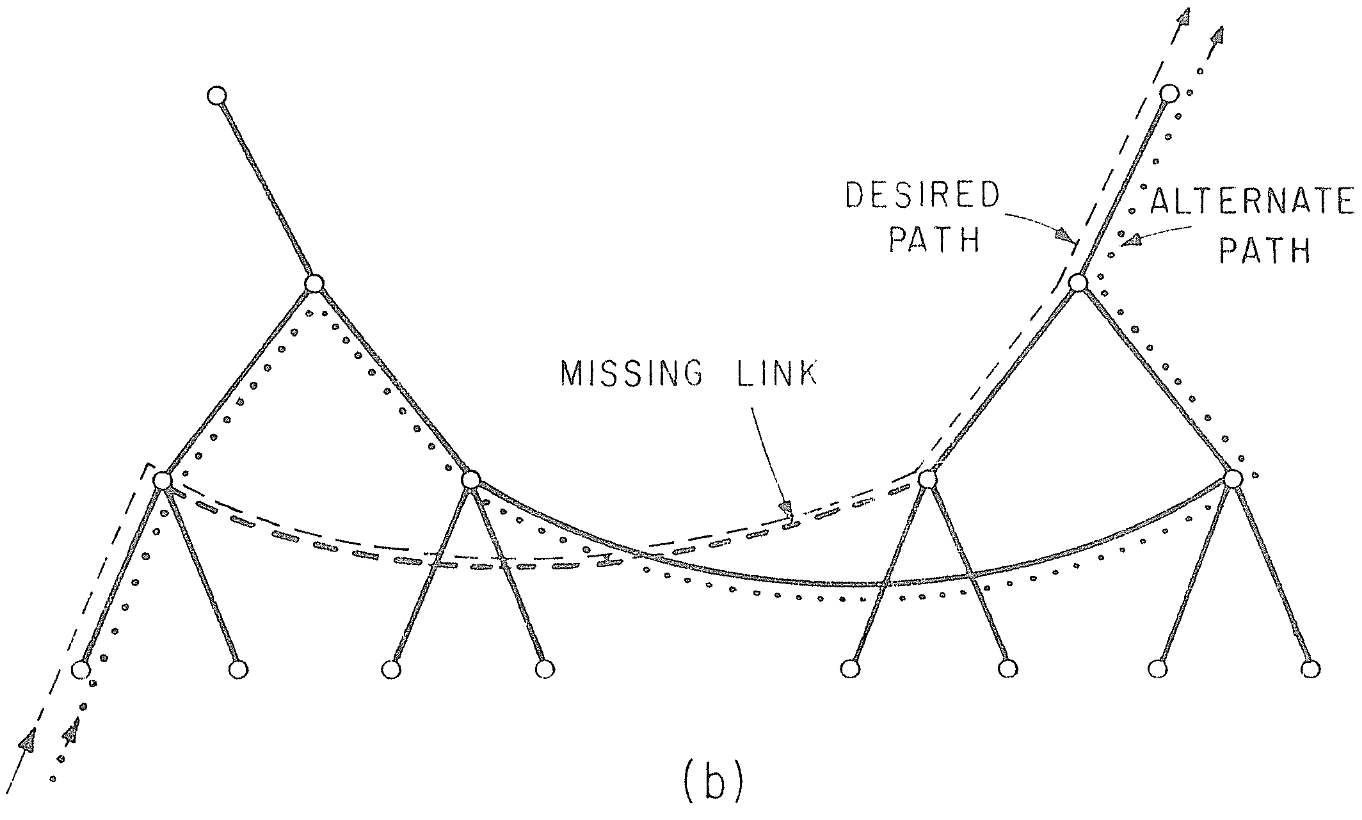


Figure 7. Detours around missing or defective links, a) around a vertical link in the downward direction, b) around missing n-cube link

a) If an n-cube link cannot be passed, go upwards instead.

b) If an upward link cannot be passed, go across n-cube link instead, followed by an upward move.

Difficulties occur on the downward branch if a missing link is encountered. Because of the lack of redundant paths over short distances, a deviation from the proper descending path can not be corrected by traversing links of the binary tree skeleton only. A detour into another plane of the hypercube has to be made (Fig. 7a). Instead of the missing vertical link, the downward link to the sibling node is taken. After traversing the n-cube link at that level, the message is routed to the sibling node in that part of the tree, and subsequently back to the original path across another n-cube link at the same level. During this whole detour the message has to remember that it is being rerouted and can not simply follow the standard routing algorithm. This extra information has to be carried along in the message header. The decoding and proper treatment of this information will increase the complexity of the routing algorithm.

Additional features can be built into the simple routing algorithm in order to enhance its efficiency in the case of single element failures in balanced hypertrees. The unsuccessful attempt to use an n-cube interconnect during the upward branch could be remembered in the message header, and the corresponding bit could be complemented in the downward branch of the message path. Alternatively a local detour similar to the one described above could be built into the path (Fig. 7b). Through the common parent node the message is shipped to the brother of the node with the unavailable n-cube link, from where the the n-cube traversal is now executed. At the other end, the desired path can be reached again in one or two steps by going through the parent node. The trade-offs between the efficiency of these fault tolerant routing algorithms and the complexity of the necesseray hardware to implement them will have to be evaluated individually for each realization of such a system.

In summary, this algorithm will successfully route messages in the presence of no more than one failed link or node, provided that the failed node is not itself the source or target. This is true even in somewhat unbalanced trees, provided that the expansion has been performed in a "natural" manner. This implies that a complete binary tree skeleton exists above the two nodes, that each node has a brother node, and that brother nodes are connected to another pair of sibling nodes through two n-cube links. Thus, at least two disjoint paths should have existed before the failure.

While the addition of the long-distance n-cube interconnections is useful for certain algorithms, the lack of redundant paths to nearby nodes, such as the extra links in a full-ring or half-ring tree, may be disadvantageous if there is a lot of locality in the message traffic and also for generating a simple fault-tolerant routing algorithm. If five ports per node can be afforded, it may therefore be advisable to use only one set of hypercube connections per level and reserve the other extra port for a half-ring connection.

5. RELATION TO OTHER NETWORKS

Other people have studied various structures in the context of the questions addressed in this paper. Schlumberger[74] analyses the de Bruijn network, which may be introduced as the state diagram for an n-bit, k-ary shift register, where k is the number of unidirectional links leading in and out of each node. He has shown that the worst case path length for $N=2^n$ nodes is n and that the average path length, when all nodes communicate equally with one another, is slightly less than that. The structure has an elegant routing scheme, requiring only local information. and also has reasonable fault tolerance. By considering the special case of a binary shift register ($k=2$) and making all links bidirectional paths, we obtain graphs for processor with four ports which have many similarities to the networks discussed here. The de Bruijn network can also be drawn as a binary tree with one additional node and with feedback paths from leave nodes to the higher parts of tree (Fig. 8).

The average distance in this network is about 15 to 25% shorter (for trees with up to eleven levels) than in Hypertree I. The reason for this is that this network uses all of its links for communication and none for input, output or connection to secondary memory. In the tree structures discussed, all leaf nodes have two free ports, while the de Bruijn network has no ports available. For a fair comparison, a fifth port should be added to each node in the de Bruijn network, and this network should then be compared to Hypertree II, not Hypertree I. It turns out that the average path length of Hypertree II is indeed slightly shorter than that of the de Bruijn graph. Alternatively, if we use the two ports at the leaves of Hypertree I for additional connections, and place a perfect shuffle network [Stone 71] at the bottom of the tree, the average path length drops to 5 to 10% below that of the de Bruijn graph.

However, the additional paths in the de Bruijn network break the nice symmetric properties which are inherent in the other tree structures. It is not apparent that algorithms can be derived to take advantage of this unusual network, and it therefore appears to show less promise than Hypertree.

Pease[Pease 77] has proposed an "Indirect Binary n-Cube" array, a structure which has been suggested numerous times in various contexts [Beizer 62] [Benes 65] [Lawrie 75] [Goke 73], for use with microprocessors. In this structure, each stage implements one dimension of the n-cube interconnection, i. e., each level provides the exchange corresponding to one bit in the node address. If each level of edges in the simple binary tree is thought of as one "stage" of a switching network, it will be seen that the binary tree has the same characteristic. The bottom level provides the exchange corresponding to the least

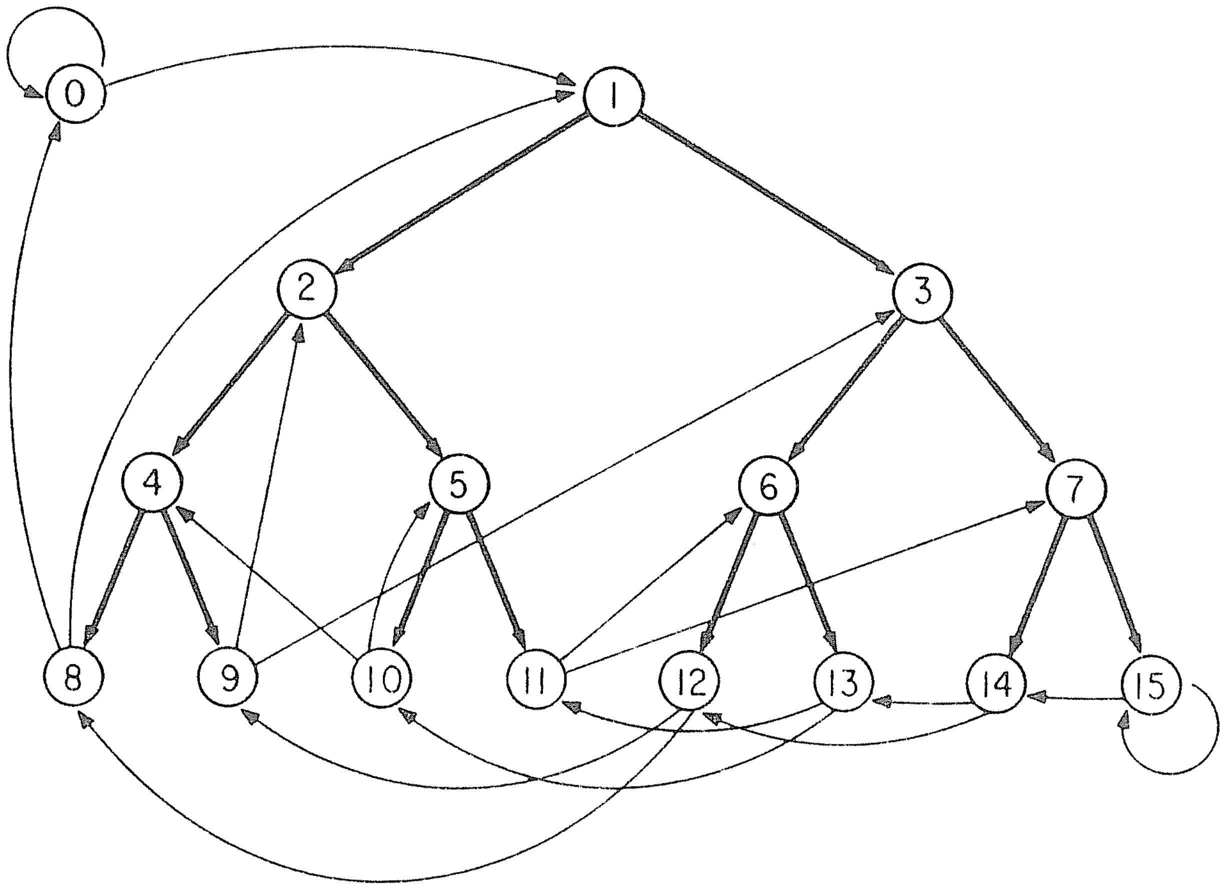


Figure 8. De Bruijn network represented as a binary tree with an extra node and additional feedback links.

significant bit, the next level the next least significant bit, etc. The horizontal connections in Hypertree at each level also provide the exchange corresponding to one bit in the node address. Although Pease did not propose it this way, it would seem possible to implement a network of single-chip computers with switching circuitry in the form of an indirect n-cube. The two ends of the network could be tied together, forming a cylinder, and the links could all be bidirectional. Such a network has some very interesting properties and is being analyzed for its possibilities.

Both of the above mentioned structures contain one property that made them unacceptable in our application: lack of incremental expansibility. A minimum increment in the case of the de Bruijn network requires a doubling of the number of processors, and even worse, a total reconfiguration of the nodes. A minimum increase in the size of the indirect n-cube requires doubling the circumference of the cylinder and increasing its height by one extra stage. Neither structure appears to maintain its nice properties if it is not complete.

6. CONCLUSIONS

A new network topology for multiprocessor systems has been derived in an attempt to combine the best features of easily expansible tree structures and the rather compact n-dimensional hypercube. The two underlying structures permit two distinct logical views of the system. Thus problems which map particularly nicely onto a tree structure can take advantage of the binary tree, while those that can use the symmetry of the n-cube can be assigned to processors in a way that efficiently uses the n-cube links.

The regular structure allows the implementation of simple routing algorithms, which require no detailed knowledge of the network interconnections. With a relatively small additional overhead, a routing algorithm can be constructed that is robust enough, so that messages will arrive at the proper node even for grossly unbalanced trees or in the presence of failing nodes or links. This is a requirement for easy expansibility of the system and for graceful degradation in the presence of communication hardware failures.

The network is readily expansible in an incremental way. All nodes have a fixed number of ports regardless of the size of the network. These two properties make this topology particularly attractive for implementations of multi-microprocessor networks of the future, where a complete computer with a substantial amount of memory can fit on a single VLSI chip.

ACKNOWLEDGMENTS

We would like to thank Al Despain and Dave Patterson for many stimulating and enlightening discussions on multiprocessor network topologies, and Bill Goldberg for producing some results on his simulator, which we could not derive analytically.

This study was sponsored in part by the Joint Services Electronics Program, Contract F44620-76C-0100.

APPENDIX 1: AVERAGE PATH LENGTH IN HYPERTREE

In this section we will derive the expected path length in Hypertree I for a message between arbitrary leaf nodes. This is equivalent to deriving the average path length when all leaf nodes send messages to all leaf nodes. We shall include the useless case of a node sending a message to itself since it results in simpler formulas if it is left in. If desirable it can easily be removed.

For a graph consisting of a balanced, binary tree plus the additional links defined in this paper as Hypertree I, we number the nodes in the natural way: top to bottom, left to right, starting with the root as "node 1" on "level 0". These node addresses, expressed as binary numbers with coefficients x_j, will then take the form

$$X = x_0\, x_1\, x_2 \cdots x_m$$

where m is the level number of the node, and where $x_0 = 1$ and $x_j = 0$ or 1, for $j = 1,2,3, \cdots m$. In particular, for leaf nodes $m = n$, where n is the number of levels below the root.

We now define a "half-way" level $l = \left\lceil \frac{m}{2} \right\rceil$ and $a = m \; mod \; 2$, so that $l = \frac{m+a}{2}$. One can convince one-self, e.g. by looking at Fig. 2 in the paper, that any bit in a leaf node address can be complemented by following a path which stays in the lower half of the tree. Some bits get complemented by using the skeleton of the binary tree only, while others make use of one of the horizontal n-cube links. Now consider the path between two leaf nodes X_1 and X_2, and define the *address difference* as

$$Y_{1,2} = 0\, y_1\, y_2\, y_3 \;\cdots\; y_n,$$

where y_j, is the modulo-2 sum between x_{j1} and x_{j2}.

We now rename the bits in this address difference so that b_j are bits which can be complemented by staying on the binary tree links in the lower half of the tree, and h_j are bits for which the shortest path leads across a n-cube link:

$$Y_{1,2} = 0 \left\{ h_0 \, h_1 \cdots h_{l-1} \right\} b_{l-a} \; b_{l-a-1} \cdots b_2 \, b_1.$$

The indices of the bits denote how many levels above the leaf the path has to climb in order to make the complementation of that bit possible. Thus if the bit position to be complemented is among bits b_j, i.e. $b_j = 1$, the the shortest path between the two nodes uses only links of the simple binary tree, going up j levels, then back down, and the minimum distance d is $2j$. If the differing bit is among bits h_j, then the shortest path between the two nodes goes up the binary tree j levels, across the horizontal link, then down through the binary tree, and $d = 2j+1$. While the indices of the bits b_j appear in decending order in $Y_{1,2}$, the bits h_j are an unordered set. It is important to note, however, that every term from h_0 to h_{l-1} appears exactly once for a tree with $n = 2l-a$ levels below the root.

In the general case where the source and the target addresses differ in several bits, the minimum path length d is achieved by climbing the tree to the highest level necessary, then going back down, traversing the horizontal links for the necessary exchange of bits h_j on the appropriate levels, either on the way up or on the way down.

The highest level to which the path must climb is determined by t, the largest value of j for which either h_j or b_j is 1. Then, if we define the number of horizontal links to be traversed as

$$H = \sum_{j=0}^{l-1} h_j \, ,$$

the path length becomes

$$d = 2t + H.$$

For random messages among the leaf nodes, we must now calculate the expected values of t and H.

Calculation of H and t

For the random message case, each bit in the address difference is equally likely to be 0 or 1. Since there are l bits of the relative address that result in a traversal of a horizontal link, one can calculate

$$H = \frac{1}{2} \times l = \frac{m+a}{4}.$$

Each h_j is equally likely to be 0 or 1, and similarly b_{l-a} will be 1 with probability ½. Note that t can reach its maximum value l only if $a = 0$, i.e., m is even, and $b_l = 1$. Thus

$$p(t=l) = \frac{(1-a)}{2}.$$

Thus if $a = 0$, $t < l$ with probability ½. Regardless of the value of a, however, if $t < l$, then $t = l-1$ with probability ¾, *i.e.*, when either h_{l-1} or b_{l-1} is 1. Thus

$$p(t=l-1) = \frac{3}{4}[1-p(t=l)] = \frac{3}{4}[1-\frac{1}{2}+\frac{a}{2}] = \frac{3}{8}(a+1)$$

and in general for $j = 1,2,3, \cdots, l-1$,

$$p(t=l-j) = \frac{3}{4}[1-\sum_{i=0}^{j-1} p(t=l-j)] = \frac{3(1+a)}{2 \cdot 4^j}.$$

The expected vertical distance t, is then

$$t = \sum_{j=0}^{l} j \cdot p(t=j) = \sum_{j=1}^{l} j \cdot p(t=j)$$

or more explicitly,

$$t = 1 \cdot \frac{3(1+a)}{2 \cdot 4^{l-1}} + 2 \cdot \frac{3(1+a)}{2 \cdot 4^{l-2}} + \cdots + (l-2) \cdot \frac{3(1+a)}{2 \cdot 4^2} + (l-1) \cdot \frac{3(1+a)}{2 \cdot 4} + l \cdot \frac{1-a}{2}$$

which can be expressed as

$$t = \frac{3(1+a)}{2 \cdot 4^l} \sum_{j=1}^{l-1} j\, 4^j + l \cdot \frac{1-a}{2}.$$

Now, since it holds in general for $q > 1$

$$\sum_{j=1}^{k-1} j \cdot q^j = \frac{q}{(q-1)^2}\left((k-1)q^k - kq^{k-1} + 1\right)$$

it follows that

$$t = \frac{3(1+a)}{2\cdot 4^l}\cdot\frac{4}{9}[(l-1)4^l - l\cdot 4^{l-1}+1]+l\cdot\frac{1-a}{2}.$$

Collecting terms, rearranging and noting that

$$2^a = a+1$$

we get

$$t = \frac{m}{2}-\frac{2}{3}+\frac{2}{3}2^{-m}-\frac{a}{6}.$$

Thus the average leaf-to leaf path length in Hypertree I is

$$d = 2t+H = \frac{5m}{4}-\frac{4}{3}+\frac{4}{3}2^{-m}-\frac{a}{12}$$

This formula has been used to plot the curve in Fig. 5 in the paper.

REFERENCES

[Batcher73] K.E.Batcher, "STARAN/RADCAP hardware architecture," Proc. 1973 Sagamore Computer Conf. Parallel Processing, pp.209-227.

[Batcher76] K.E.Batcher, "The Flip Network in STARAN," 1976 Int. Conf. on Parallel Prcessing, Aug. 1976, pp.65-71.

[Benes65] V.E.Benes, Mathematical Theory of Connecting Networks and Telephone Traffic. New York: Academic Press, 1965.

[Browning79] S.A. Browning: "Computations on a Tree of Processors", VLSI Conference, CALTECH, Pasadena, CA, Jan. 22-24, 1979, Proceedings

[Bouknight72] W.J.Bouknight, S.A.Denenberg, D.E.McIntyre, J.M.Randall, A.H.Sameh, and D.L.Slotnick, "The ILLIAC IV System," Proceedings of the IEEE, Vol.60,No.4, Apr. 1972, pp.369-388.

[de Bruijn46] N.G.de Bruijn, "A combinatorial prblem," Koninklijke Nederlands Akademie van Wetenschappen, Proceedings, Vol.49 (part 2), pp.758-764, 1946.

[Despain78] A.M.Despain and D.A.Patterson, "The computer as a component," submitted to CACM, 1978.

[Feng74] T.Feng, "Data Manipulating Functions in Parallel Processors and Their Implementations," IEEE Trans. Comput., Vol.C-23, No.3, Mar. 1974, pp.309-318.

[Goke73] L.Goke and G.Lipovski, "Banyan Networks for Partitioning Multiprocessor Systems," Proc. 1st Ann. Comput. Architecture Conf., 1973, p.21-28.

[Golomb61] S.W.Golomb, "Permutations by Cutting and Shuffling," SIAM Review, Vol.3, Oct.1961, pp.293-297.

[Lawrie75] D.Lawrie, "Access and alignment in an array processor," IEEE Trans. Comput., Vol.C-24, Dec.1975, pp.1145-1155.

[Mago79] G.A. Mago: "A Cellular Language-directed Computer Architecture", VLSI Conference, CALTECH, Pasadena, CA, Jan. 22-24, 1979, Proceedings

[Pease77] M.C.Pease, "The Indirect Binary n-Cube Microprocessor Array," IEEE Trans. Comput., Vol.C-26,No.5, May 1977, pp.458-473.

[Schlumberger74]M.A.Schlumberger, "De Bruijn Communication Networks," Stanford Ph.D. Dissertation, Computer Science Department, Stanford, California, June 1974.

[Sequin79] C.H. Sequin: "Single-Chip Computers, the New VLSI Building Blocks", VLSI Conference, CALTECH, Pasadena, CA, Jan. 22-24, 1979, Proceedings

[Siegel77] H.J.Siegel, "Analysis Techniques for SIMD Machine Interconnection Networks and the Effects of Processor Address Masks," IEEE Trans. Comput, Vol. C26, No.2, Feb. 1977, pp.153-161.

[Stone71] H.S.Stone, "Parallel Processing with the Perfect Shuffle," IEEE Trans. Comput., Vol.C-20,No.2, Feb.1971, pp.153-161.

[Swan77] R.J.Swan, S.H.Fuller, and D.P.Siewiorek, "Cm* - A modular, multi-microprocessor," 1977 NCC Proceedings, pp. 645-655, June 1977.

[Thompson65]C.Thompson, "Generalized Connection Networks for Parallel Processor Intercommunication," IEEE Trans. Comput., Vol C-27, No.12, Dec. 1978, pp.1119-1125.