NASA Contractor Report 172399

ICASE REPORT NO. 84-33

ICASE

AUGMENTING COMPUTER NETWORKS

Shahid H. Bokhari
and
A. D. Raza

Contracts Nos. NAS1-17070, NAS1-17130
July 1984

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

AUGMENTING COMPUTER NETWORKS

Shahid H. Bokhari
Institute for Computer Applications in Science and Engineering
and
University of Engineering and Technology, Lahore, Pakistan

and

A. D. Raza
Telegraph & Telephone Department
Lahore, Pakistan

Abstract

Three methods of augmenting computer networks by adding at most one link per processor are discussed.

1. A tree of N nodes may be augmented such that the resulting graph has diameter no greater than $4\left\lceil \log_2\left(\frac{N+2}{3}\right)\right\rceil - 2$. This $O(N^3)$ algorithm can be applied to any spanning tree of a connected graph to reduce the diameter of that graph to $O(\log N)$.

2. Given a binary tree T and a chain C of N nodes each, C may be augmented to produce C' so that T is a subgraph of C'. This algorithm is $O(N)$ and may be used to produce augmented chains or rings that have diameter no greater than $2\left\lceil \log_2\left(\frac{N+2}{3}\right)\right\rceil$ and are planar.

3. Any rectangular two-dimensional 4 (8) nearest neighbor array of size $N = 2^k$ may be augmented so that it can emulate a single stage shuffle-exchange network of size $N/2$ in 3 (2) time steps.

This research was supported by a grant from the Division of Research Extension and Advisory Services, University of Engineerinig and Technology, Lahore, Pakistan.

The work of the first author was further supported by the National Aeronautics and Space Administration under NASA Contracts Nos. NAS1-1-17070 and NAS1-17130 while he was in residence at the Institute for Computer Applications in Science and Engineering, NASA

I. INTRODUCTION

We show how the capabilities of an existing computer network can be improved by adding at most one communication link per processor. In particular, we show how at most one edge per node need be added to an arbitrary N node connected graph in order to reduce its diameter to $4\left\lceil \log_2 \left(\frac{N+2}{3}\right)\right\rceil - 2$. Since the diameter of a network determines the maximum time required to communicate between a pair of nodes, this result allows us to improve the connectivity of a network in a very crucial fashion. This is a generalization of a previously reported algorithm that was applicable only to special types of graphs [4]. The cost of this improvement is at most one I/O port per processor and thus no more than $N/2$ additional communication links. This is discussed in Section III of this paper.

In Section IV we describe how a chain or ring of processors can be augmented by adding at most one edge per node so that a given binary tree may be perfectly mapped on it. This algorithm has complexity $O(N)$ and generates graphs that are planar. It is thus a significant improvement over the previously reported algorithm [4], which was $O(N^3)$ and did not guarantee planarity of the augmented graph. This allows us to reduce the diameter of a ring from $N/2$ to $2\left\lceil \log_2 \left(\frac{N+2}{3}\right)\right\rceil$ thereby speeding up the execution of those algorithms that require global data operations, such as sorting. This algorithm can obviously be applied to any graph that is Hamiltonian and can therefore reduce the diameter of a k-dimensional nearest-neighbor array from $O(N^{1/k})$ to $O(\log N)$. For the case of two-dimensional arrays the extra edges

require only one additional layer of interconnect. These results permit us to construct array processors that combine all the advantages of nearest-neighbor arrays as well as those of tree machines.

Section V describes how a 4 (8) nearest neighbor array may be augmented so that it can execute the shuffle-exchange permutation [12] in 3 (2) time steps. This allows us to combine the benefits of nearest neighbor arrays and permutation networks in one machine.

Section VI contains a discussion of our results. We start with a section on definitions.

II. DEFINITIONS

We will consider only undirected connected graphs in what follows [5]. A graph is denoted $G = \langle V,E \rangle$, where V is the set of nodes and E the set of edges. The distance $d(x,y)$ between any two nodes x,y contained in V is the length of the shortest path joining x and y. The diameter of a graph G is the maximum distance between any two nodes in the graph. That is diameter $= \max_{x,y \in V} d(x,y)$.

A tree is a connected undirected acyclic graph with N nodes and $N-1$ edges. A rooted tree is a tree with an explicitly designated root node. Each edge of a rooted tree connects a father node to a son node, where the father occurs before the son in the path connecting the root to the son. All nodes without descendants are called leaf nodes.

A chain is a tree whose degree is constrained to 2. It has exactly two leaves and all of its non-leaf nodes have degree exactly 2.

A binary tree is a rooted tree in which the maximum number of sons of any node is 2. The two sons of a node (if they exist) are called leftson and rightson.

A Moore 3-tree is a rooted tree in which all non-leaf nodes have degree exactly 3. The root thus has 3 sons and all other non-leaf nodes have 2 sons each.

The height h of a rooted tree is the maximum distance from the root nodes to any leaf node. The diameter of a rooted tree cannot exceed 2h or be less than h.

In a complete binary tree of N nodes the distances from the root node to any two left nodes can differ by at most 1 and there can be no more than one non-leaf node with only one son (Fig. 1). The height of this tree is $h = \lceil \log_2(N+1) \rceil - 1$ and its diameter is no more than 2h.

Similarly, in a complete Moore 3-tree of N nodes, the distances from the root to any two leaf nodes can differ by at most one and there can be no more than one non-leaf node with only one son (Fig. 2). The height of such a tree is $h = \left\lceil \log_2\left(\frac{N+2}{3}\right) \right\rceil$ and its diameter is no more than 2h.

Of all trees with N nodes and degree constraint 3, a complete Moore 3-tree has minimum diameter.

A graph G_1 is said to have been mapped onto another graph G_2 if: 1) G_1 and G_2 have the same number of nodes and 2) each node of G_1 has been assigned to a node of G_2 in a one-to-one onto fashion. If every edge of G_1 falls on some edge of G_2 then the mapping is called a perfect mapping. In any case, the number of edges of G_1 that fall on edges of G_2 is called the cardinality of the mapping [2].

When G_1 is a tree, the perfect mapping of G_1 onto G_2 is the same as the spanning of G_2 by G_1. G_1 is then a spanning tree of G_2 in the conventional sense [5].

III. REDUCING THE DIAMETER OF A NETWORK

Given a tree T of N nodes, we will show how to augment it by adding no more than one edge per node so that the resulting graph has diameter no greater than $4\left\lceil \log_2 \left(\frac{N+2}{3}\right)\right\rceil - 2$. This algorithm can clearly be applied to any spanning tree of an arbitrary connected graph of N nodes to obtain $O(\log N)$ diameter.

Let T be the given tree of N nodes. Construct a complete Moore 3-tree M of N nodes. Let U be a set of trees that initially contains T.

```
1.  done:=false;
2.  while not empty(U) and not done do
       begin
3.        select the tree u in U that has the
             maximum diameter d;
4.        remove the chain of nodes c from u
             which lies along this diameter;
5.        return u-c to U;
6.        remove the longest chain m from M,
             let its length be p;
7.        if d < p then done:=true
          else
             begin
8.              place nodes 1 to p of c on nodes
                   1 to p of m;
9.              return the chain formed by nodes
                   p+1 to d to U;
             end
       end
10. reconnect all edges removed in step 4;
11. add edges between all non-adjacent nodes of T
       that were mapped on adjacent edges of M in step 8;
```

This algorithm attempts to map successively smaller chains from T onto successively smaller chains of M. If this algorithm terminates without the condition of line 7 being satisfied, then all of T will have been mapped on all of M with at most one edge added to each node of T in step 11. In this case the diameter of the augmented graph will be no greater than that of M, that is $2\left\lceil \log_2\left(\frac{N+2}{3}\right)\right\rceil$. This will happen, among other cases, if T is a chain, when our algorithm reduces to an inverse of the algorithm described in [4] for mapping degree constrained trees onto chains.

The interesting case is when the condition of line 7 is satisfied and the diameter of the longest remaining chain in T is less than the length of the longest chain in M. Now a part of T, say T_m, has been mapped onto a part of M and, after adding edges,

will become T_m' with diameter no greater than $2\left\lceil \log_2(\frac{N+2}{3}) \right\rceil$. No component in the remaining portion will have diameter greater than the longest remaining chain in M. This is maximum when the condition is satisfied on the second pass through the while loop. (If this is true during the first pass, then the diameter of T is less than the diameter of M and there is no point in running this algorithm.) The diameter of the longest remaining component C_m in $T-T_m$ is thus at most $2\left\lceil \log_2(\frac{N+2}{3})-1 \right\rceil-1$.

The overall diameter of the augmented graph cannot exceed $dia(C_m) + dia(T_m')$. This is proved by contradiction. Suppose the diameter exceeds this amount. Then there exists some component of $T-T_m$, say C_n, such that the diameters of C_m, T_m' and C_n lie along a chain that has length greater than $dia(C_m) + dia(T_m')$, (Fig. 3). This is impossible because when mapping nodes from chains of T onto chains of M we always start at one end of the chain. Thus when extracting the last successfully mapped chain from T, we would have started at an extreme end of C_m or C_n and would have have been left with a chain of length $dia(C_n) + dia(C_m)$, thereby contradicting our assumption that the condition of line 7 is satisfied. The overall diameter of the augmented graph is thus no greater than
$4\left\lceil \log_2(\frac{N+2}{3}) \right\rceil - 2$.

The running time of this algorithm is $O(N^3)$ since it requires the calculation of distances between all pairs of nodes (an $O(N^2)$ process) as much as N times.

IV. MAPPING BINARY TREES ONTO CHAINS

We now describe an algorithm that, given an arbitrary binary tree T and a chain C (each of N nodes) will specify the mapping of T onto C such that no more than one edge per node need be added to C to produce an augmented chain C′ which has T as a spanning tree. In other words, T can be perfectly mapped on C′. This algorithm has complexity O(N) and the mappings that it produces are such that C′ is always planar.

This algorithm starts at the root of the tree and proceeds by threading each node and its sons in a linked list. When this algorithm concludes, all nodes of the tree have been threaded by this linked list which thus specifies the order in which the nodes of T are to be mapped on nodes of C.

The given binary tree is assumed to be stored in an array with each node having a pointer to its leftson, rightson and a back pointer to its father. Two more pointers are required for the linked list. Finally, each node has a label which specifies the order in which it is threaded. This is a crucial notion in the development of our algorithm. Given a node that has already been threaded by the list, the two sons of the node can be added to the list in three possible ways: inorder, postorder and preorder, which correspond to the well known methods of tree traversal [8]. This operation is illustrated in Figures 4 and 5. The way in which a node x is labelled depends on how the father of x was linked and whether x is a leftson or a rightson of its father. A call to procedure link will thread the sons

of a node. Thus link(x, predorder) corresponds to the transition from Figure 4 to Figure 5(b). Note that the father is labelled with the order in which it is linked to its sons.

Procedure link is not recursive, however procedure tree-link (described below) which calls link is. The complete tree threading algorithm is given on the following page. The function father-linked(x) returns label[father[x]], that is the order in which the father of x was threaded.

Figures 6(a), (b) and (c) show successive stages in the threading of a tree and Figure 6(d) shows the fully threaded tree. Figure 7(a) shows how a 25 node complete binary tree is threaded. The resultant augmented chain in shown in Figure 7(b).

```
procedure tree_link(x:range);

begin

     if x=nil then {return}
     else
        begin
           if x is the leftson of its father then
              case father_linked(x) of
                 inorder: link(x, postorder);
                 postorder: link(x, inorder);
                 preorder: link(x, preorder);
              end
           else {x is the rightson of its father}
              case father_linked(x) of
                 inorder: link(x, preorder);
                 postorder: link(x, postorder);
                 preorder: link(x, inorder);
              end;
           tree_link(leftson[x]);
           tree_link(rightson[x]);
        end;
end;

{main program}

begin
     start:=root;
     link(root,inorder);
     tree_link(leftson[root]);
     tree_link(rightson[root]);
end.
```

V. AUGMENTING NEAREST NEIGHBOR ARRAYS TO ALLOW SHUFFLE-EXCHANGES

Figure 8 shows a single stage recirculating shuffle-exchange network of size $N = 8$ [12]. This consists of shuffle interconnections and $N/2$ 2×2 switches that route the shuffled outputs back to the inputs after each shuffle operation. The switches can route their inputs to their outputs in either straight through or interchange fashion.

A single stage recirculating shuffle-exchange network of size $N/2$ can be formed out of any rectangular 4 (8) nearest neighbor array of size $N = 2^k$ $(k > 2)$ by adding at most one more link per processor. The added links are needed for the shuffle interconnections. Switch functions can be emulated by the existing array interconnections.

We index the columns of a rectangular 4 or 8 nearest neighbor array of size $N = 2^k$ with the numbers $0,1,2,\cdots$ and partition the array into two halves consisting of even and odd-numbered columns. Shuffle interconnections are added between the processors of the two halves. Each group of four processors which straddles evan and odd-numbered columns can emulate any switch function in no more than 2(1) time steps for 4(8) nearest neighbor arrays. Thus the complete augmented array can emulate a shuffle-exchange operation in 3(2) time steps, respectively. Figures 8 and 9 illustrate this process.

VI. DISCUSSION

The network augmenting algorithm described in Section III is capable of reducing the diameter of any connected graph to O(log N) by adding at most one edge per node. This algorithm will be useful for improving the performance of an existing computer network whose interconnection structure has evolved over time without regard for diameter. By reducing the diameter we reduce the maximum communication latency of the network. Our augmentation scheme requires at most one additional communication port per node and thus no more than N/2 additional links. This algorithm is applicable to any connected graph and is thus more general than the algorithms described in [4] which are applicable only to Hamiltonian graphs or graphs partitionable into cliques of size 3 or greater.

The algorithm of Section IV for mapping binary trees onto rings or chains allows us to solve efficiently binary tree structured problems on one-dimensional nearest-neighbor arrays. This algorithm indirectly solves the mapping problem [2] for the special case of binary trees onto rings or chains. This allows us to maximize usage of nearest neighbor links in a chain that has a global bus superimposed on it (a problem similar to that discussed in [3]). This algorithm may also be used to solve approximately the problem of mapping binary trees onto Hamiltonian graphs with cardinality guaranteed at no worse than 2/3 of optimal.

This algorithm may also be used to augment a chain or ring so that it contains a complete binary tree and thus suggests new

interconnection structures that have all the advantages of binary trees as well as those of rings. With a minor modification, it can be used to map a complete Moore 3-tree onto a chain or ring and thus obtain diameter $2\left\lceil \log_2 \left(\frac{N+2}{3}\right)\right\rceil$. This algorithm yields augmented chains or rings that are "one-sided" planar (Figure 7(b)) in $O(N)$ time and is thus superior to the algorithm given in [4] which is $O(N^3)$ and does not guarantee planarity.

Since our augmented rings require the addition of only one edge per node and have logarithmic diameter, they are superior to the chordal rings of Arden and Lee [1] which have square root diameter and to the augmented rings proposed by Pradhan and Reddy [10] which have logarithmic diameter but require the addition of two edges per node.

The algorithm of Section IV can obviously be applied to any Hamiltonian graph. Although the problem of finding Hamiltonian paths in graphs is intractable in general [6], it is trivial for most nearest neighbor arrays. Many array processors have nearest-neighbor interconnections. Examples include the Illiac-IV, the Finite Element Machine [9], and PACS [7]. The $n \times n$ nearest neighbor array lends itself to the efficient solution of many interesting problems [11], [13] but has the disadvantage of an $O(n)$ diameter which results in poor execution of global operations such as sorting or finding maximum. We can use our algorithm to augment such arrays to obtain networks with all the advantages of nearest neighbor arrays as well as those of tree machines. It is interesting to note that only one additional layer of interconnecting wires is required for this purpose.

Finally, we showed in Section V how the powerful perfect-shuffle interconnection can be superimposed on a two-dimensional nearest neighbor array. This gives us an interconnection pattern with all the advantages of nearest neighbor arrays as well as those of the perfect shuffle.

Acknowledgment

The authors wish to thank Professor K. E. Durrani and Dr. R. G. Voigt for their encouragement of this research.

References

[1] B. W. Arden and H. Lee, "Analysis of chordal ring networks," IEEE Trans. Comput., Vol. C-30, No. 4, April 1981, pp. 291-295.

[2] S. H. Bokhari, "On the mapping problem," IEEE. Trans. Comput., Vol. C-30, No. 3, March 1981, pp. 207-214.

[3] S. H. Bokhari, "Finding maximum on an array processor with a global bus," IEEE. Trans. Comput., Vol. C-34, No. 2, February 1984, pp. 133-139.

[4] S. H. Bokhari and A. D. Raza, "Reducing the diameters of arrays," Research Report EECE-83-01, Computer Engineering Division, Department of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan, June 1983.

[5] N. Deo, Graph Theory with Applications to Engineering and Computer Science, Englewood Cliffs, NJ: Prentice-Hall, 1974.

[6] M. R. Garey and D. S. Johnson, Computers and Intractability, San Francisco: Freeman, 1979.

[7] T. Hoshino, T. Kawai, T. Shirikawa, J. Higashino, A. Yamaoka, H. Ito, T. Sato, and K. Sawada, "PACS: A parallel microcomputer

systems for scientific calculations," ACM Trans. Computer Systems, Vol. 1, No. 3, August 1983, pp. 195-221.

[8] D. E. Knuth, The Art of Computer Programming, Vol. 1, Fundamental Algorithms, Reading, MA: Addison Wesley, 1973.

[9] H. J. Jordan, "A special purpose architecture for finite element analysis," Proc. 1978 Conf. on Parallel Processing, August 1978, pp. 263-266.

[10] D. K. Pradhan and S. M. Reddy, "A fault-tolerant communication architecture for distributed systems," IEEE Trans. Comput., Vol. C-31, No. 9, September 1982, pp. 291-295.

[11] A. Rosenfeld, "Parallel image processing using cellular arrays," Computer, Vol. 16, No. 1, January 1983, pp. 291-295.

[12] H. Stone, "Parallel processing using the perfect shuffle," IEEE Trans. Comput., Vol. C-20, No. 2, February 1971, pp. 153-161.

[13] H. S. Stone, "Parallel Computers," in Introduction to Computer Architecture, H. S. Stone, ed., Palo Alto: Science Research Associations, 1980.

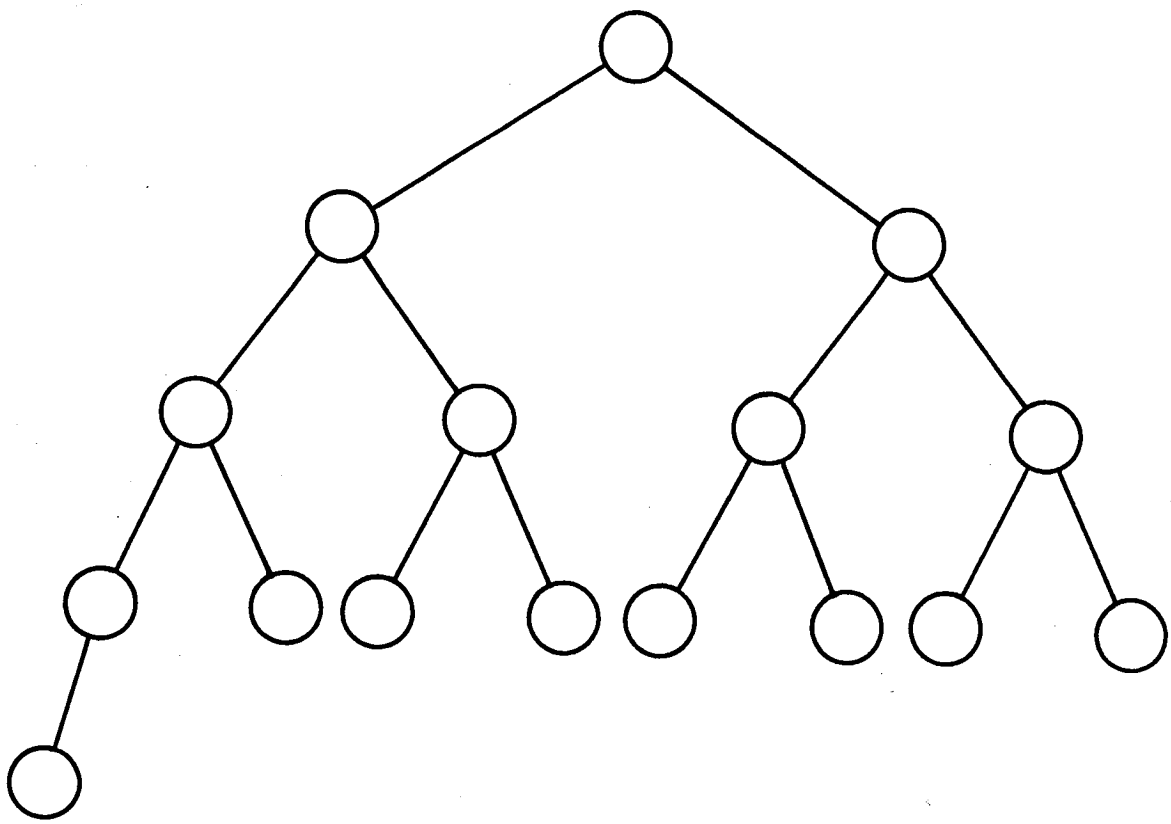


FIGURE 1. A COMPLETE BINARY TREE OF 16 NODES.

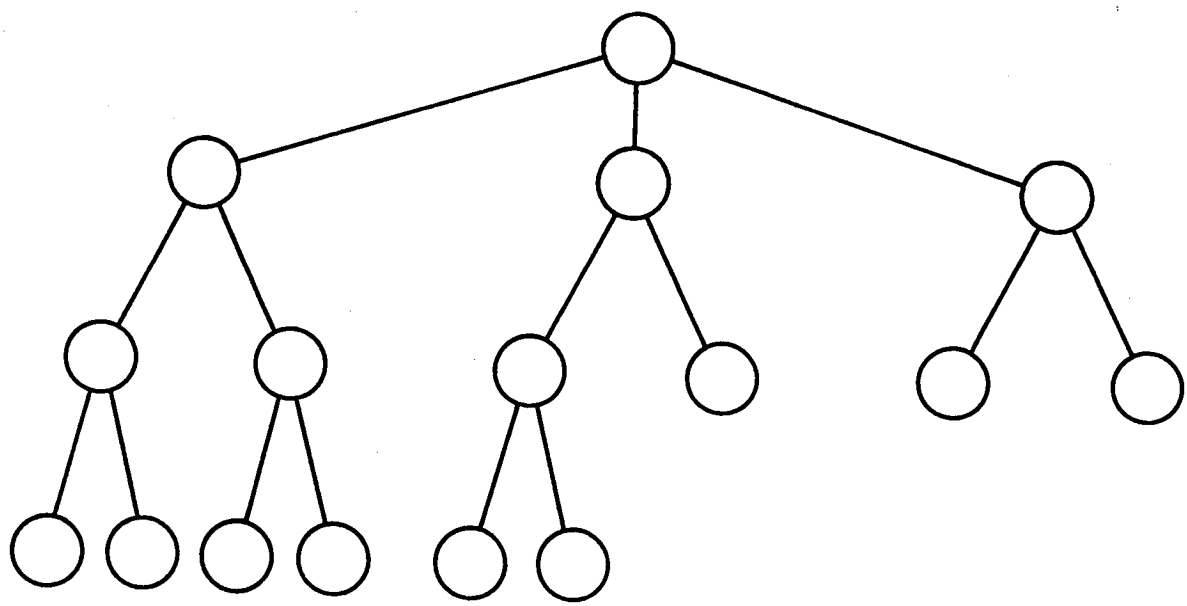


FIGURE 2. A COMPLETE MOORE 3-TREE OF 16 NODES.

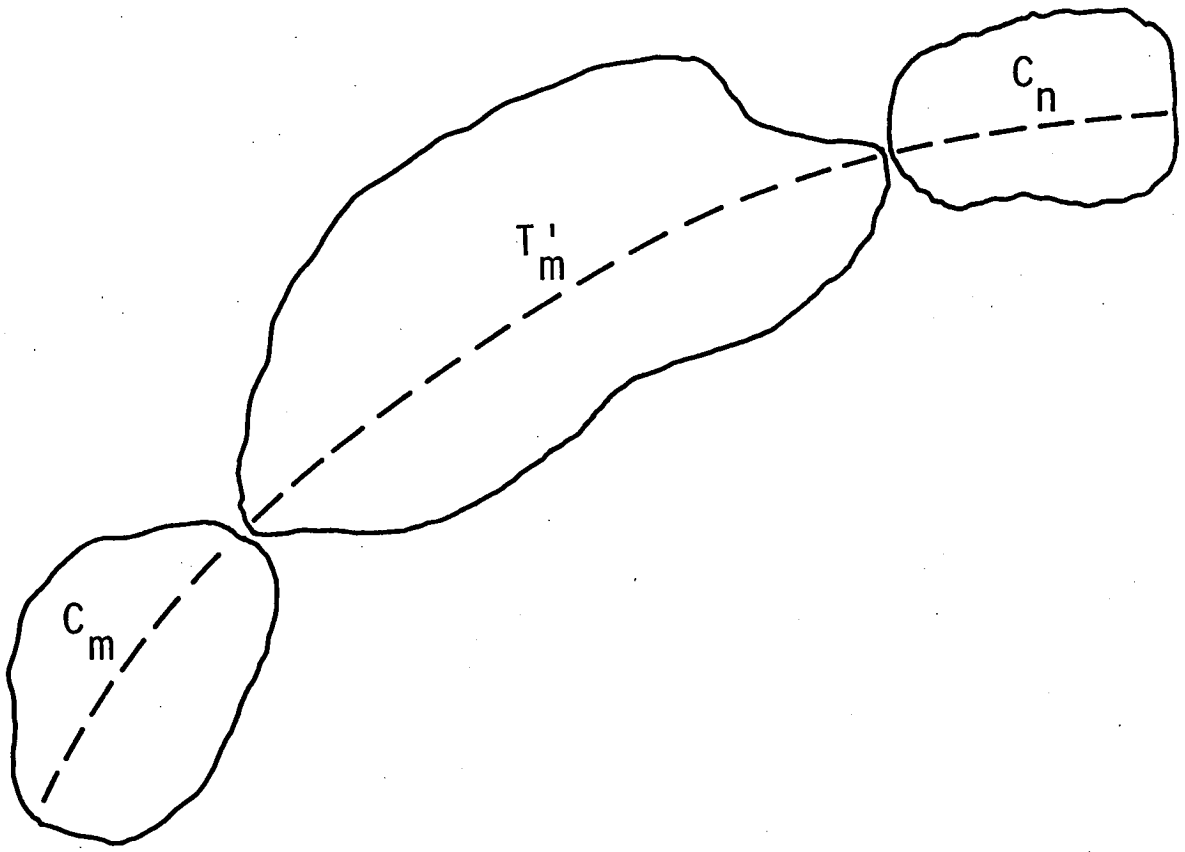


FIGURE 3. PROOF OF CORRECT TERMINATION OF NETWORK AUGMENTING ALGORITHM.

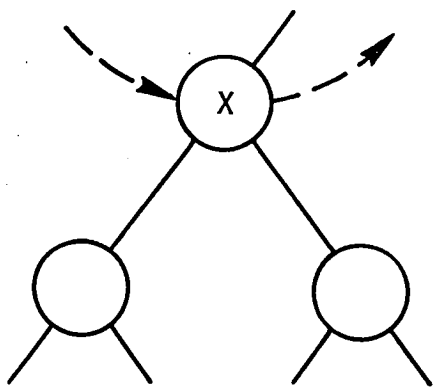
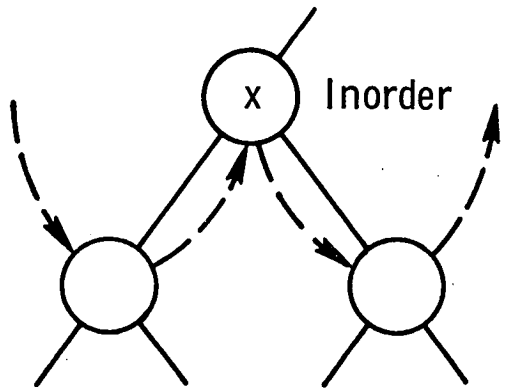
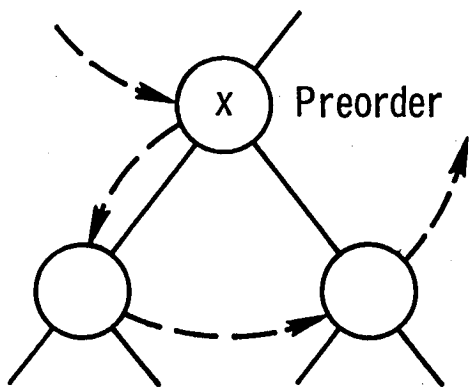


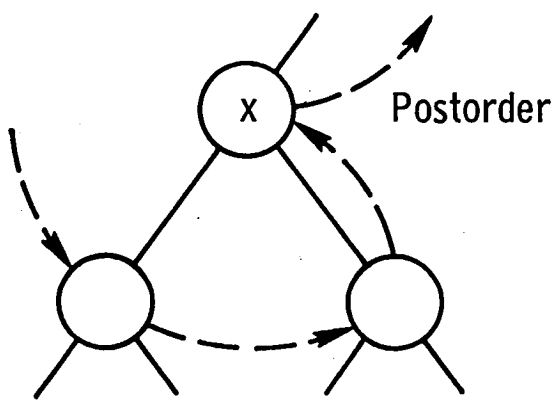
FIGURE 4. THE SONS OF NODE X HAVE NOT BEEN THREADED.



(a) Inorder



(b) Preorder



(c) Postorder

FIGURE 5. THREADING THE SONS OF NODE X IN: (A) INORDER; (B) PREORDER; AND (C) POSTORDER.

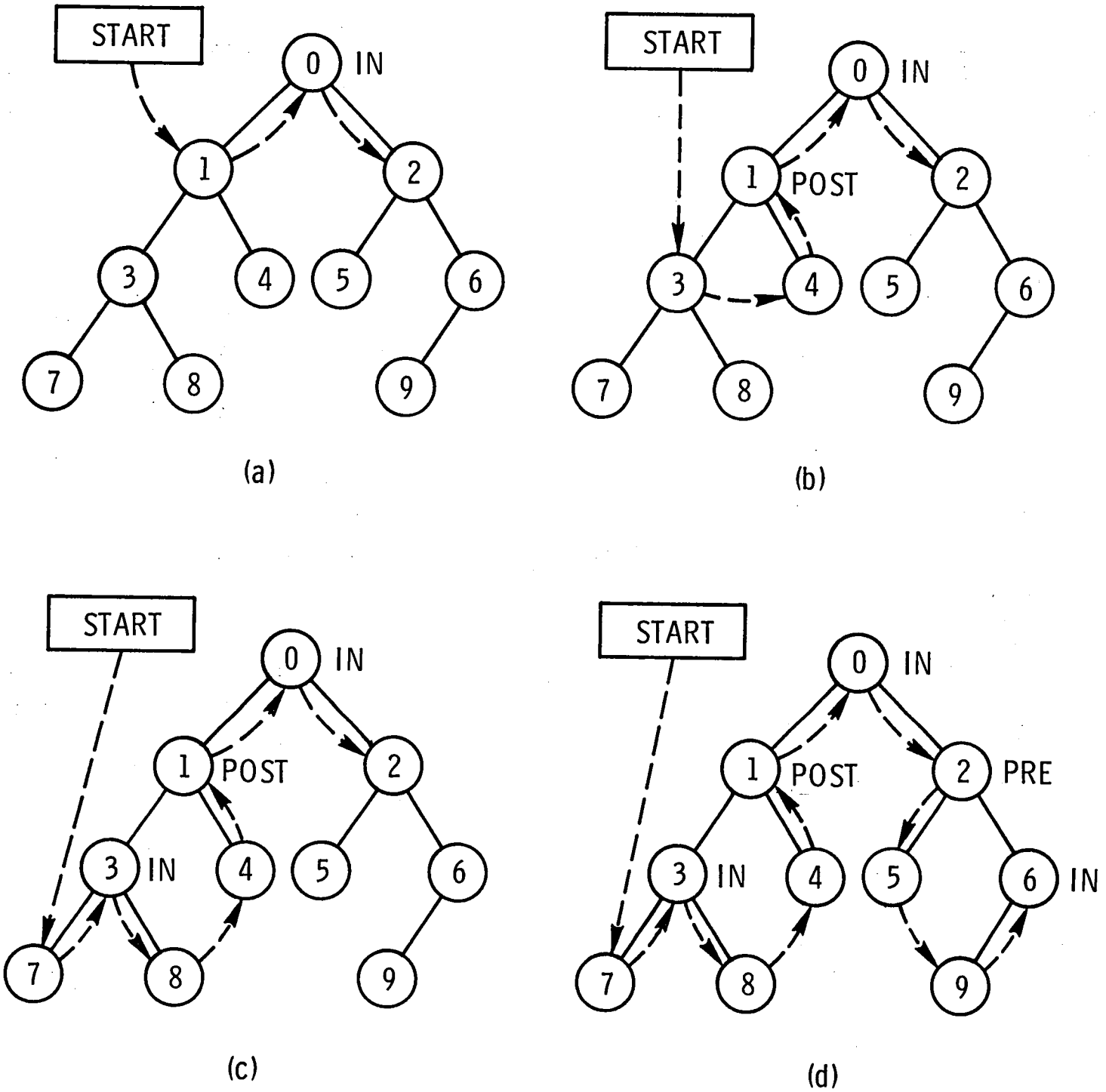


FIGURE 6. (A), (B), (C): SUCCESSIVE STAGES IN THE THREADING OF A TREE.
(D): THE FULLY THREADED TREE.

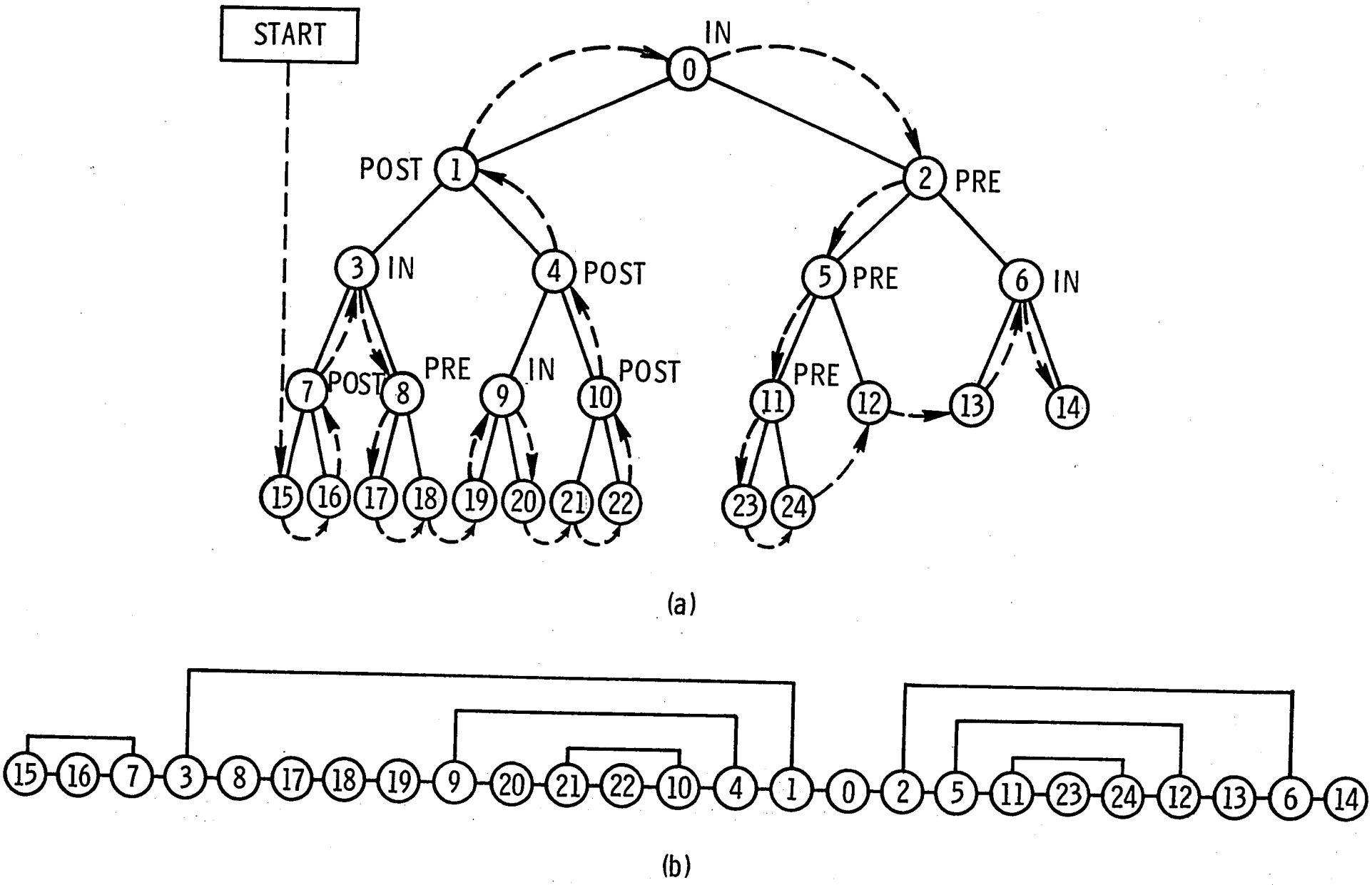


FIGURE 7. (A) THREADDING A 25 NODE COMPLETE BINARY TREE.
(B) THE RESULTANT AUGMENTED CHAIN.

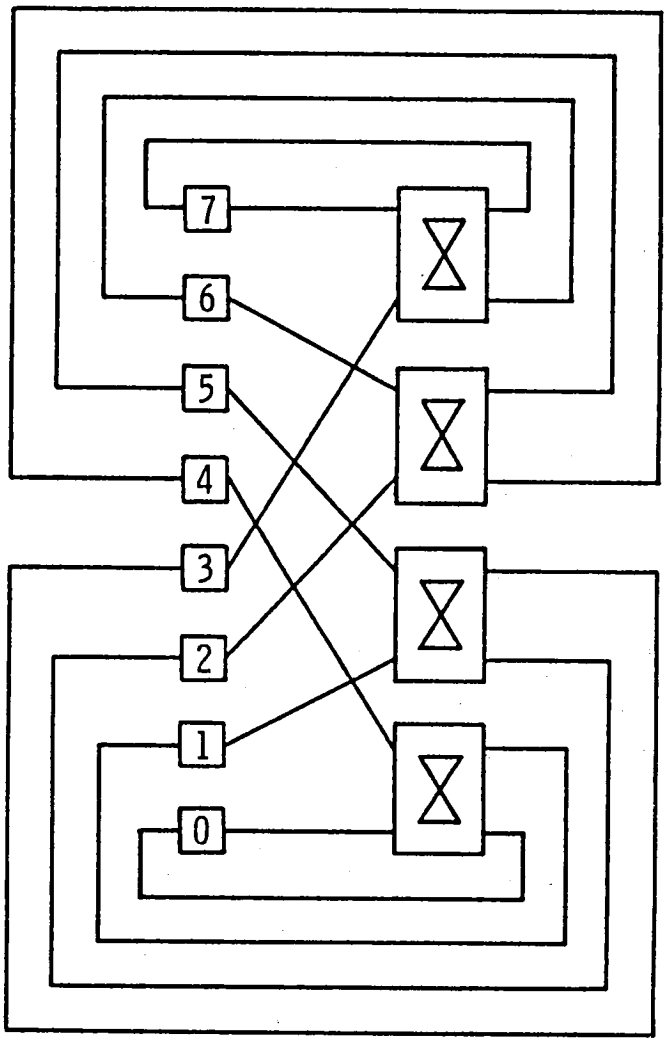


FIGURE 8. SINGLE STAGE RECIRCULATING SHUFFLE-EXCHANGE NETWORK OF SIZE N = 8.

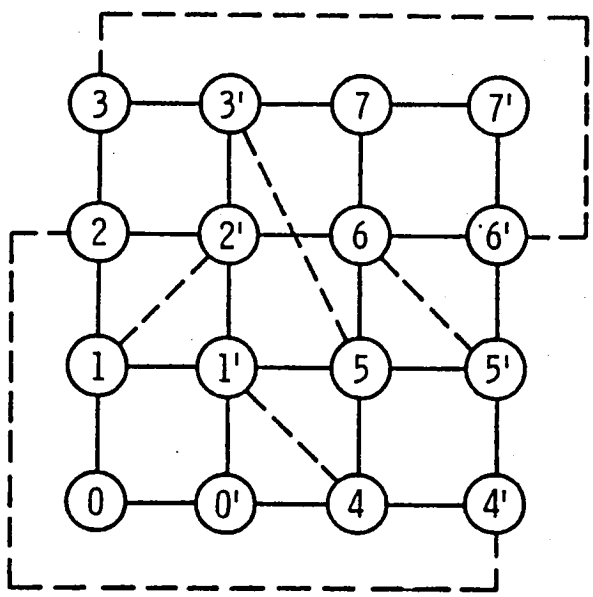


FIGURE 9. SHUFFLE CONNECTION AUGMENTED 4x4 4-NEAREST NEIGHBOR ARRAY THAT CAN EMULATE THE NETWORK OF FIGURE 8 IN CONSTANT TIME.

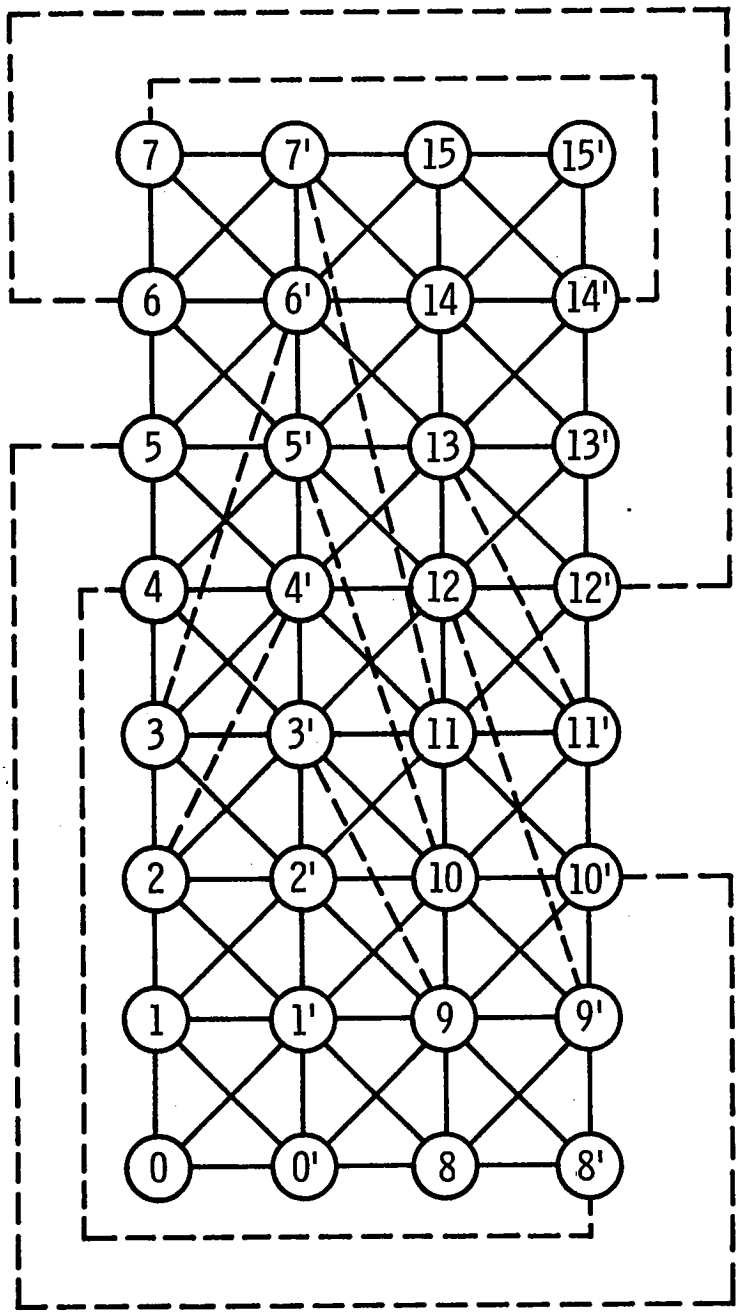


FIGURE 10. AN AUGMENTED 4x8 8-NEAREST NEIGHBOR ARRAY THAT CAN EMULATE A 16 NODE SHUFFLE.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-172399 ICASE Report No. 84-33		

4. Title and Subtitle	5. Report Date
AUGMENTING COMPUTER NETWORKS	July 1984
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Shahid H. Bokhari and A. D. Raza	84-33
9. Performing Organization Name and Address	10. Work Unit No.
Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665	11. Contract or Grant No. NAS1-17070 NAS1-17130
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered Contractor Report
National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code 505-31-83-01

15. Supplementary Notes

Langley Technical Monitor: R. H. Tolson
Final Report

16. Abstract

Three methods of augmenting computer networks by adding at most one link per processor are discussed.

1. A tree of N nodes may be augmented such that the resulting graph has diameter no greater than $4\left\lceil \log_2\left(\frac{N+2}{3}\right)\right\rceil - 2$. This $O(N^3)$ algorithm can be applied to any spanning tree of a connected graph to reduce the diameter of that graph to $O(\log N)$.

2. Given a binary tree T and a chain C of N nodes each, C may be augmented to produce C' so that T is a subgraph of C'. This algorithm is $O(N)$ and may be used to produce augmented chains or rings that have diameter no greater than $2\left\lceil \log_2\left(\frac{N+2}{3}\right)\right\rceil$ and are planar.

3. Any rectangular two-dimensional 4 (8) nearest neighbor array of size $N = 2^k$ may be augmented so that it can emulate a single step shuffle-exchange network of size $N/2$ in $3(t)$ time steps.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Array processors, augmented networks, binary trees, diameter reduction, perfect shuffle	62 - Computer Systems Unclassified - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	25	A02