ISBN 978-93-84422-22-6
Proceedings of 2015 International Conference on Image Processing, Production and Computer Science (ICIPCS'2015)
Istanbul (Turkey), June 3-4, 2015 pp. 89-96

Finding and Displaying the Shortest Paths in Hyper-Rings

Raghda Alqurashi[1] and Tom Altman[1]

[1]University of Colorado Denver, USA

Abstract: *A graph G= (V, E) is called a hyper-ring with N nodes (N-HR for short) if V = {0, ..., N-1} and E = {(u,v)| v-u modulo N is a power of 2}. In other words, a hyper-ring with N nodes is a graph in which all of the edges connect nodes that are at distances of powers of 2 from each other. Hyper-rings are a multi-machine organization that is, in a sense, a generalization of the standard hyper-cube architecture. One of the most powerful properties of the hyper-rings is that the node-connectivity of an N-HR has been proven to be equal to its degree. This paper discusses hyper-rings, their construction and properties, and an interactive software that allows the user to generate and display hyper-rings of arbitrary size and dimension. The algorithm also finds and displays, under 3D rotations, all node-disjoint shortest paths between any two user-specified nodes.*

Keywords: *Hyper-ring network, graph search, connectivity.*

1. Introduction

A graph is a mathematical abstraction that may be used to represent the architecture of an interconnection network. Here, the nodes represent processors and edges represent communication links between pairs of processors [8]. Different network topologies have been used for the interconnection of large scaled parallel and distributed systems. Hyper-ring is a powerful topology that has applications in networking and parallel processing [2]. A computer network can be built using the hyper-ring design that allows for fast and dependable communication. Hyper-rings (HRs for short) have appeared in literature under several different names, including an optimal broadcasting scheme [5] and binary jumping networks [6]. Hyper-rings have a number of desirable properties for interconnection networks.

We describe an interactive computer program that constructs a hyper-ring of a user-specified size. Also, this program finds the shortest paths, as well as all existing node-disjoint paths between a pair of user-specified nodes, called the *source* and *destination*. A variation of the breadth-first search algorithm [8] is used to find these paths. The constructed HR, its node-disjoint paths, and other information can then be interactively displayed graphically in a 3D rotation.

The paper is organized as follows. In Section II, HRs are formally introduced and some of the known construction methods for HRs are revisited. The advantages of the HRs structure are also briefly discussed. A modified version of the Breadth-First Search algorithm and node-connectivity properties of HRs are presented in Section III. Section IV addresses the implementation, methodology, and the capabilities of the program. Brief concluding remarks are given in Section V.

2. Hyper-Ring

Definition 1. A graph $G = (V, E)$ is called a *hyper-ring* with N nodes (N-HR for short) if $V = \{0, 1, \ldots, N-1\}$ and $E = \{(u, u) \mid [v-u]_N$ is a power of 2$\}$, where $[m]_r$, is m modulo r [2]. If k denotes the power of 2, then k is an integer and $0 \leq k \leq \lfloor log\, N \rfloor$ [2].

Hyper-rings (HRs) are a multi-machine organization that is, in a sense, a generalization of the standard hyper-cube (HC) architecture [1]. Examples of a 15-HR, 34-HR, and a 73-HR are shown in Figure 1.

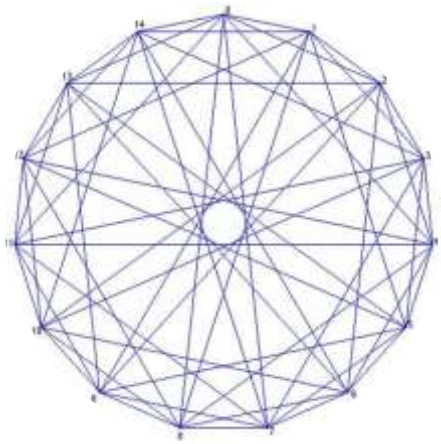

Fig. 1(a): A 15-HR.

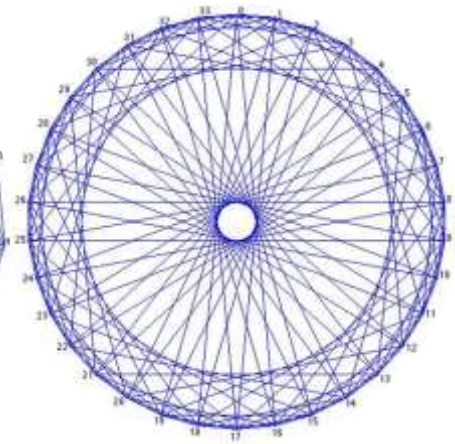

Fig. 1(b): A 34-HR.

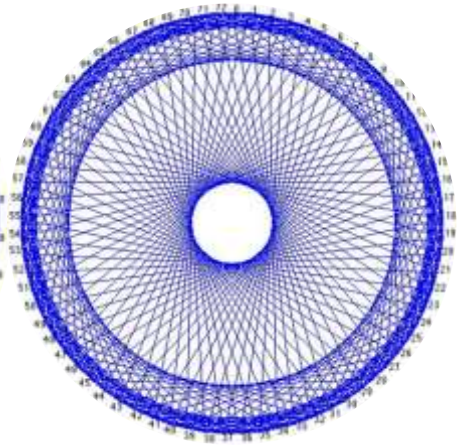

Fig. 1(c): A 73-HR.

2.1 Hyper-Ring Construction

With the same number of nodes, the number of edges in an HR is roughly twice the number of edges of the same size Hyper-cube (HC). However, unlike HCs, the HRs may be constructed with any number of nodes, i.e., it does not have to be a power of 2. The HRs also have an interesting recursive construction. There are two basic approaches to build HRs: a non-recursive and a doubling construction. We will need to study the structure of N itself in order to study the recursive structure of HRs with N nodes, or N-HRs.

2.1.1 Non-Recursive Construction

Definition 2. Let N be a positive integer. The *binarity* of N, written $bin(N)$, is a function whose range is between 0 and $\lceil logN \rceil$, that returns the largest integer k, such that N is divisible by 2^k. (In other words, $bin(N)$ tells us how many consecutive rightmost bits of a binary representation of N are zero) [1]. Observe that a positive integer N may always be represented as the following product

$$N = 2^{bin(N)} M, \tag{1}$$

where M is an odd number.

Definition 3. The *positional distance* between node s and node t of N-HR is defined to be $min\{[t\text{-}s]_N, [s\text{-}t]_N\}$ [4].

Let us first examine the structure of non-recursive N-HRs. The following procedure shows the non-recursive construction of N-HRs [2]:

for *k:=0 to N-1* **do**
  for $i := 0$ *to* $\lfloor log\ N \rfloor$ **do**
    if there is no edge between node k and node $[k + 2^i]_N$
    then connect node k to nodes labeled $[k \pm 2^i]_N$

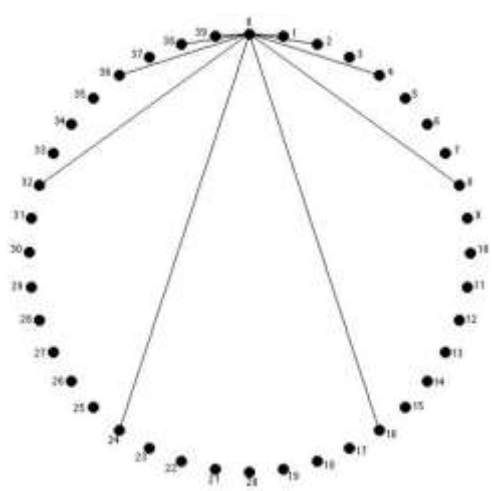

Fig. 2: The connections for one node of an HR are constructed by the above procedure.

While this procedure is effective for constructing an HR, it does not give us any insight about the number of edges in the HR, nor does it give any information about its recursive structure. Any two nodes in an HR are connected if they are either $2^{\lfloor logN \rfloor}$ or $2^{\lfloor logN \rfloor - 1}$ positional distance away from each other, which means the nodes in N-HR are exactly $\pm 2^i$, i= 0,…, $\lfloor logN \rfloor$, positions away from each other. In other words, the distance is a power of 2 from node s to node t in the clockwise or counterclockwise directions. See edges (0,4) and (0, $[-4]_{15}$=11) in Figure 1; what appears to be a connection of positional distance 11 is really a counterclockwise connection of positional distance 4.

2.1.2 Doubling Construction

The doubling construction can improve the network's fault tolerance [6]. In order to construct a 2N-HR from a given constructed N-HR, we use the following procedure that takes an N-HR_0 and returns a 2N-HR [4]:

Make a duplicate copy of N-HR_0, call it N-HR_1;
for $i:=0$ *to* $N-1$ **do**
    relabel all nodes in N-HR_0 from i to $2i$ ($0 \leq i \leq N-1$);
    relabel all nodes in N-HR_1 from i to $2i+1$ ($0 \leq i \leq N-1$);
for $i:=0$ to $2N-1$ **do**
    connect node i and node j iff j = i + 1 *modulo* 2N;

The above is called the *doubling construction* procedure. Figure 3 shows an example of a doubling construction. This figure demonstrates the connections for one node of the HR that is constructed by applying the doubling procedure. This procedure gives us the recursive relation for the number of edges in HRs. If $E(N)$ denotes the number of edges in an HR, then

$$E(2N) = 2E(N) + 2N \tag{2}$$

Also, this procedure shows that if δ is the degree of an N-HR, then the degree of a 2N-HR is $\delta + 2$. Altman et al. showed in [1] that doubling of an HR increases its connectivity from k to k+2.

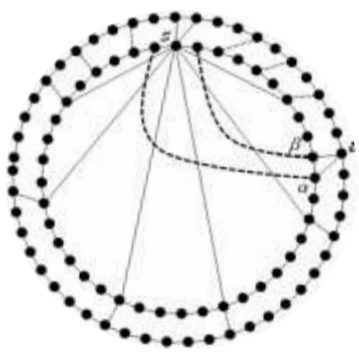

Fig. 3: An HR constructed using the doubling procedure and the two additional node-disjoint paths via α and β.

The doubling construction of an HR can be shown as follows:

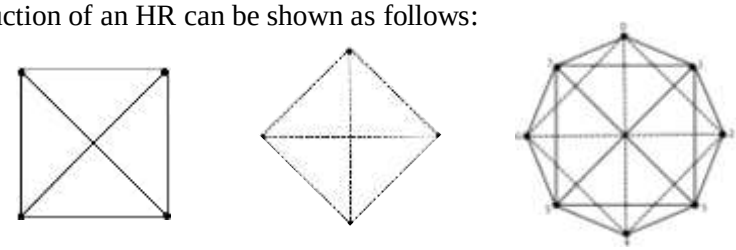

Fig. 4: Doubling construction. (a) A 4-HR. (b) A copy of 4-HR. (c) An 8-HR connected from the two connected 4-HRs.

2.2 The number of HR edges

Surprisingly, the number of edges in HRs does not grow monotonically with the number of nodes [2]. For example, a 15-HR has 61 edges and a 16-HR has 56 edges. From Eq. (1) we can give the exact estimate of the number of edges of an *N*-HR by examining the *M*-HRs, where *M* is an odd positive integer in Eq. (1). To construct such *M*-HRs, we may use the non-recursive construction of *N*-HRs with a trivial modification [2].

Theorem 1 *The number of edges in an N-HR, $N \geq 2$, is given by the following formula:*

$$E(N) = \begin{cases} N\,(\log N - \frac{1}{2}) & \text{if the hamming weight of } N \text{ is } 1 \\ N\,\lfloor \log N \rfloor & \text{if the hamming weight of } N \text{ is } 2 \\ N\,\lceil \log N \rceil & \text{if the hamming weight of } N \text{ is } 3 \text{ or more.} \end{cases}$$

2.3 The advantages of HRs

One of the main goals of network design is to find the best way to increase the reliability of the system. The reliability of a distributed system depends on the reliabilities of its communication links and computer elements and the distribution of its resources as well as on the *connectivity structure* of the network.

HRs are commonly used as a design for communication in a system. The HR topology is very suitable for mid-scale to large-scale multicomputer systems and networks of workstations for its cost-efficiency, its fixed uniform node degree due to its symmetric construction, and the simplicity of any routing schemes [1, 2, 3].

The connectivity of a hyper-ring determines the limits and the performance of any system it represents, such us a computer network. The node-connectivity of an HR provides for and efficient and reliable transmission of information and highly fault-tolerant routing in networks [6].

HR networks provide a fast and dependable node-to-node communication. Hyper-ring is the first connection architecture that allows for dependable communication in time of $\lceil logD \rceil$+1, where *D* is the positional distance between the source and destination nodes that plays an important role for fast and reliable node-to-node communication protocols [3]. In addition, the HR architecture allows for optimal reliable computing schemes and implementation of secret message passing mechanisms [7].

Moreover, HRs provide an *optimal* broadcasting communication scheme [6]. Consider a network where a node can send a message to all of its direct neighbors in one round. The broadcasting in HRs may be carried out as follows:

Procedure *disseminateHR*(*N*)
Repeat
  for round:=0 **to** $\lceil \log N \rceil$ - 1 **do**
    each node *i* send a message to all its neighbors concurrently
forever

This procedure needs $\lceil \log N \rceil$ rounds to broadcast information from any source node to all destination nodes, even in the presence of node failures. All of these make HRs desirable as a network interconnection topology.

Hyper rings are regularly connected graphs. Altman et al. showed that the number of edges in a hyper-ring is roughly twice that of a hyper-cube with the same number of nodes [1]. However, the organization of the hyper-ring has advantages over the hyper-cube. An HR can be constructed with *N* processors, whereas a hypercube must contain exactly 2^n processors for some positive *n* [2].

Although the number of links between processors in a hyper-ring is roughly twice that of a hyper-cube with the same number of processors, the diameter of the hyper-ring, which is a good property for information transfer, is shorter than that of a hyper-cube. Also, the node-connectivity of the HR is greater than that of an HC. The shortness of the diameter can prevent the performance deterioration due to the large scale of the network.

Thus, hyper-rings have advantages over hyper-cubes since the number of processors in an HR does not have to be a power of 2, and the number of edges in an HR is roughly twice that of the same size of HC. Moreover,

Altman et al. in [1] demonstrated that for any *n*, the *n*-dimensional hyper-cube and complete binary trees could be directly embedded in the HRs with 2^n nodes as its subgraph[1] [2].

Hyper-ring and ring topology:

Most interconnected networks support ring topology since ring topology is cheap from an implementation viewpoint. In fact, its cost grows no faster than the number of nodes. Also, its node degree is fixed at two, which leads to a cheap and simple network interface at each node. Broadcasts as other multi-node communication operations are easily implemented over the ring. Thus, the ring is a desired topology for multiprocessors systems for its low cost, ease of implementation, and broadcast support. However, in the case of a large number of nodes, the ring suffers from a large diameter. So, the ring topology is not proper for large-scale systems [9].

Hyper-ring is a hierarchical organization and scalable ring-based topology that eliminates the main disadvantages of large-scale rings. The diameter of the hyper-ring is shorter than that of the ring, which is a good property for information transfer. Also, the connectivity of the hyper-ring is much larger than that of the ring [9].

3. Breadth-First Search Algorithm and Node-connectivity of HR

Breadth-First Search (BFS) is a graph search algorithm that begins at the root node as the source node and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal [8].

Our hyper-ring programs utilize the breadth-first search algorithm in order to find the shortest paths between nodes. They also utilize a modified form of this algorithm to search for all of the node-disjoint paths. Within the BFS algorithm, a tree-trimming step is added in order not to waste time searching for paths that share some ancestor node, as these paths are not node-disjoint. The first step in the algorithm is to begin a tree with the source (root) node. The children are the adjacent nodes of the source node. These nodes are appended to the tree and become the frontiers, or leaves, of the tree. Next, each of these children is checked to see if it is the destination node. If any of these nodes is the destination node, then a path is created containing only the source node and the destination node, and the path length is one edge and is, therefore, the shortest path. This path begins the list of disjoint paths. In order to find the rest of the disjoint paths, the remaining children of this tree are found and appended to the tree whether or not the shortest path was found.

If a path to the destination node was found previously, the rest of the edges from the branch on which the destination was found will not be explored. This is where our algorithm differs from the standard BFS. In the standard BFS, any node that is found has exactly one parent. Because our goal is to find all of the node-disjoint paths to the destination node, the destination node must have more than one parent. It will in fact, have as many parents as it will have node-disjoint paths. All nodes that are found will have multiple parents. An *intermediate* node is one on a path that is neither the source nor the destination node. Intermediate nodes and their unused parents must be preserved, even if one of the other parents of the intermediate node is used in one of the disjoint paths. This is because these nodes can still be used in another disjoint path with another parent. The branches do not terminate in unique nodes. For this reason, the search will be limited to branches that did not yet find the destination node. This process continues until all of the tree branches have stopped growing. By using BFS this way, all of the disjoint paths from the source node to the destination node are found.

The breadth-first algorithm to find the shortest path and the node-disjoint paths:

```
degree = number of nodes adjacent to the source node
set root of tree to the source  node
disjoint_paths [ ]
for i = 1 to ⌈log D⌉ +1
        children = column of all nodes adjacent to last tree column
        if any element of children equals destination then
            paths[ ] = paths between the source and the destination nodes
                for j = 1 to number of paths
                    if paths[j] contains no nodes from disjoint_paths then
```

[1] An embedding is defined as a mapping from a guest graph into a host graph [2].

```
                    disjoint_paths = concatenate disjoint_paths with paths[j]
                end
            end
        tree = concatenate children onto tree
        remove branches of tree containing nodes used in disjoint_paths
        if tree cannot grow then stop
end
shortest path = disjoint_paths.
```

4. Implementation

The key idea used here is to create the HR by applying the doubling construction, where each HR graph is constructed from the two previous HRs that have already been built and to properly connect them. We have decided to use MATLAB[2] to create the program, which is deployed to be a standalone application that can run under any operating system without the need to install MATLAB.

The program consists of four main parts. The first part creates all of the nodes of the HR, distributes them evenly on the edge of a circle, and assigns them what are called dimensions and levels within dimensions. Each new dimension has as many levels as the total number of levels below it. These nodes are then drawn in a plot. In Figure 5(d) below, the boxes contain nodes in individual dimensions, specifically dimensions 3 and 4. What can also be seen, is that within each dimension, the nodes are broken up into individual levels. Edges that connect the nodes on a single level are blue, and edges that connect nodes within a dimension, but on different levels, are in green.

We create the total number of nodes based on the equation: number of nodes $\times 2^{\text{dimensions-1}}$. This allows for the number of nodes in any given dimension to be equal to the total number of previous nodes. After this, the nodes are placed into their respective dimensions, and the dimensions are then stratified into "levels". The number of nodes in each level is equal to those in the first dimension.

The second part of the code identifies the distance between the nodes, based on the definition of the HR, i.e., the distance between any node and its adjacent nodes is the node number $\pm 2^i$, $i = 0,\ldots,$ □$logN$□. Next, the edges that define the node connections are drawn. First, the edges are constructed by the non-recursive construction, where the edges for every node are created first before moving on to the next node. We go through each node, and find the nodes with a higher index, or node that is adjacent to it. We then draw the links connecting the adjacent nodes. The final HR graph is drawn in a plot window. HRs in one, or up to five dimensions may be displayed:

```
total_nodes = nodes× 2^(dimensions-1)
for i = dimensions to 2
        place nodes into dimension
        separate current dimension into 2^(i-2) levels
end
limit = ⌈log total_nodes⌉
for i = 1 to total_nodes
        for j = 1 to limit
                num = i + 2^j
                connect node i to node num
                adjacent(i,num) = 1
                adjacent(num,i) = 1
        end
end
```

[2] High-level language and interactive environment for numerical computation, visualization, and programming.

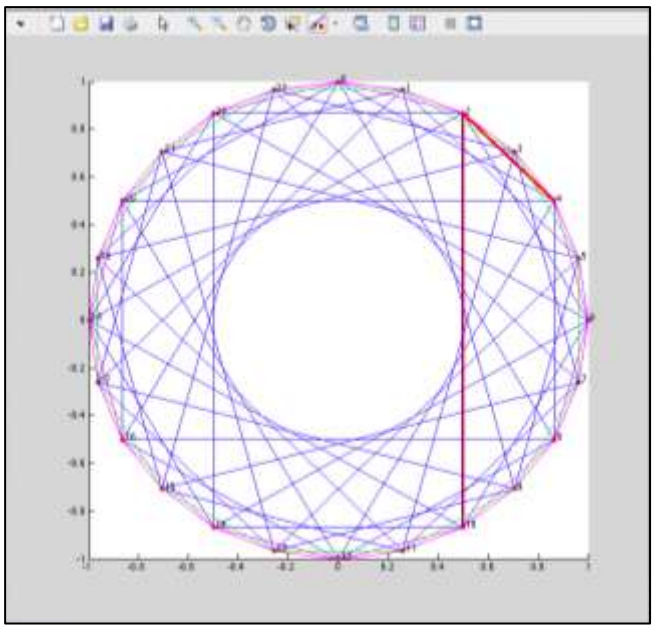

Fig. 5(a): The HR graph with shortest path.

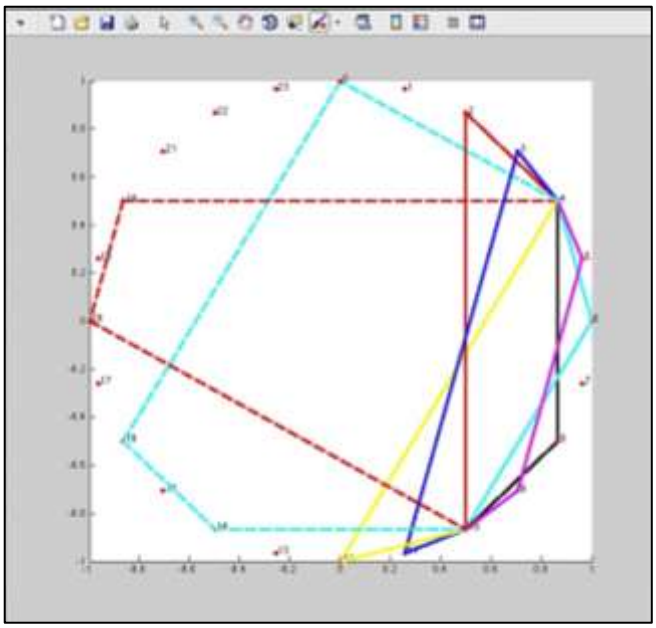

Fig. 5(b): The disjoint paths.

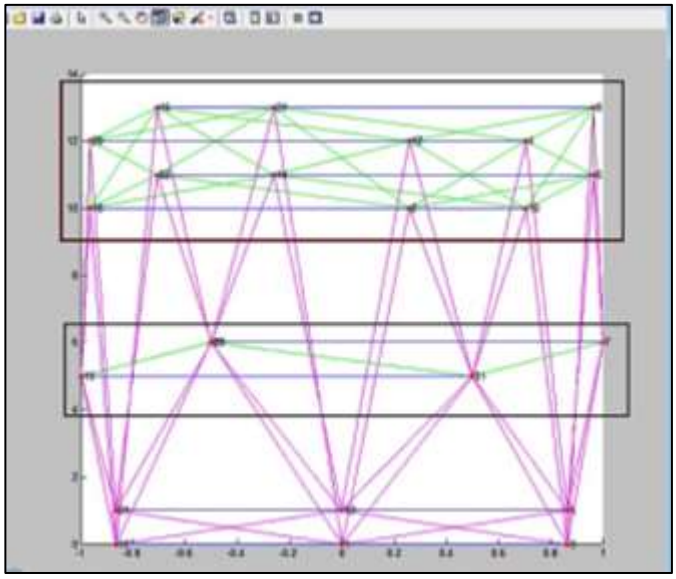

Fig. 5(c): The dimensions and the levels

The next part identifies and draws the shortest path between the two nodes that are specified by the user. The shortest path is drawn on top of the hyper-ring graph that already exists, and on a new plot window that contains only the nodes, without the edges. The purpose of this figure is to clarify the shortest path on the graph without the edges, since the number of edges may get large, as is the case with large HRs. This part of the algorithm runs only if the source and destination nodes are identified by the user. It uses a variation of the breadth-first search algorithm [8]. An adjacency matrix is created during the second section of code (The part of the code that identifies the distance between the nodes). From this matrix, another one is created. Each column in the second matrix is filled with the indices of all of the nodes adjacent to the node with the index equal to the column number. So, column 1 contains all of the indices of the nodes adjacent to node 1, column 2 contains all of the indices of the nodes adjacent to node 2, and so on. By using these matrices, this algorithm creates a tree to search for the shortest path. In this implementation of the algorithm, the tree is a two by two array. The first node (index 1,1) is the source node specified by the user. Each subsequent column of the tree array is comprised of the nodes adjacent to the prior column. The tree stops being created once one of its values is equal to the destination node. The last index of the variable path, which is an array, is filled with the index that corresponds to the

destination node that is user specified. After that, the index is divided by the number of adjacent nodes (this value is the same for every node in a hyper-ring), and rounded up. This value is used as the row index in the column prior to the last column, as the next to last node in the path. This continues until the array path has been filled with the same number of nodes as 'tree' has columns.

5. Conclusion

Hyper-ring topology is very suitable for mid- to large-scale multicomputer systems. It is also useful for parallel/distributed computing due to HRs' properties, such as fixed node degree, low communication cost, symmetry, and the simplicity of any routing schemes. The main goal of this work was to design interactive programs to display the hyper-ring graphs and their node-connectivity (shown through the node disjoint paths). We may construct and display hyper-rings of arbitrary size. The node disjoint paths can be easily visualized on the N-HRs (see Figure 5(b)), which reside in the k-dimensional space, where $k \approx \log N$. Shown in Figures 1(a-c), the dimension of hyper-ring structure can be easily seen and requires no additional computational work in order to be presented.

6. References

[1] T. Altman, Y. Igarashi, and K. Obokata, "Hyper-ring connection machines," *Parallel Comput.*, vol. 21(8), pp.1327–1338, August 1995.
http://dx.doi.org/10.1016/0167-8191(95)00022-G

[2] T. Altman, Y. Igarashi, and K. Obokata, "Hyper-ring connection machines," in *TENCON '94. IEEE Region 10's Ninth Annual International Conference. Theme: Frontiers of Computer Technology. Proceedings of 1994*, 1994, vol.1, pp. 290–294.
http://dx.doi.org/10.1109/tencon.1994.369291

[3] T. Altman, Y. Igarashi, and K. Motegi, "Fast and dependable communication in hyper-rings," in *Proceedings of the 8th Annual International Conference on Computing and Combinatorics*, COCOON '02, 2002, pp. 350–359, Singapore.
http://dx.doi.org/10.1007/3-540-45655-4_38

[4] T. Altman, Y. Igarashi, and K. Motegi, "Semi hyper-rings and enriched semi hyper-rings," *COMP2002*, vol. 102(396), pp.17-22, 2002.

[5] N. Alon, U. Manber, and A. Barak, "On disseminating information reliably without broadcasting," Technical Report TR 0621, University of Wisconsin (Madison, WI), 1985.

[6] Y. Han, Y. Igarashi, K. Kanai, and K. Miura, "Broadcasting in faulty binary jumping networks," *J. Parallel Distrib. Comput.*, vol. 23(3), pp. 462–467, 1994.
http://dx.doi.org/10.1006/jpdc.1994.1157

[7] M. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. of the ACM*, vol.36, pp. 335–348, 1989.
http://dx.doi.org/10.1145/62044.62050

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed. MIT Press, USA: Cambridge, MA, 2001.

[9] F. N. Sibai, "Optimal clustering of hierarchical hyper-ring multicomputers," *J. Supercomput.*, vol.14(1), pp. 53–76, July 1999.
http://dx.doi.org/10.1023/A:1008199214034