# 재귀원형군에서 병렬 경로 알고리즘의 설계

배 용 근[†] · 박 병 권[††] · 정 일 용[†††]


## 요 약

재귀원형군은 마이크로 프로세스의 모델로서 활발하게 연구되고 있으며 특히 슈퍼컴퓨팅 분야에서 많은 관심을 불러 일으키고 있다. 본 논문에서는 재귀원형군에서 메시지의 경로 설정을 연구하는데 이는 네트워크의 성능 평가에 중요한 기준이 된다.

재귀원형군에서 출발 노드에서 목적 노드까지 m개의 패킷을 m개의 경로를 따라서 동시에 전송하고자 한다. 이 때 i번째의 패킷은 i번째의 경로를 따라서 전송된다($0 \le i \le m-1$). 모든 패킷들이 목적 노드에 신속하고 안전하게 도달하기 위해서 i번째의 경로는 disjoint해야 한다. 이들 경로들을 설계하기 위해서 Hamiltonian Circuit Latin Square(HCLS)를 재귀원형군에 적용시켜서 $O(n^2)$ 병렬 경로 알고리즘을 제안한다.


# The Design of Parallel Routing Algorithm on a Recursive Circulant Network

Yongkeun Bae[†] · Byungkwon Park[††] · Ilyong Chung[†††]


## ABSTRACT

Recursive circulant graph has recently developed as a new model of multiprocessors, and drawn considerable attention to supercomputing. In this paper, we investigate the routing of a message in recursive circulant, that is a key to the performance of this network.

On recursive circulant network, we would like to transmit m packets from a source node to a destination node simultaneously along paths, where the ith packet will traverse along the ith path ($0 \le i \le m-1$). In oder for all packets to arrive at the destination node quickly and securely, the ith path must be node-disjoint from all other paths. For construction of these paths, employing the Hamiltonian Circuit Latin Square(HCLS), a special class of ($n \times n$) matrices, we present $O(n^2)$ parallel routing algorithm on recursive circulant network.


## 1. Introduction

The rapidly growing and intense interest in interconnection network used graph-theoretic properties[1] for its investigations and produced various interconnection schemes. Many of these schemes have been derived to optimize important parameters such as degree, diameter, fault-tolerance, hardware cost, and the needs of particular applications. Owing to low degree and diameter, and the relative ease in mapping different graph configurations (ring[3], liner arrays[3], meshes[4] and trees[5]) into recursive circulant network (RCN) proposed by Park[3], these multicomputers have

※이 논문은 1997년도 조선대학교 학술연구비 지원에 의해 연구되었음.

† 정 회 원:조선대학교 전자계산학과
†† 정 회 원:서강전문대학 전자계산학과
††† 종신회원:조선대학교 전자계산학과

논문접수:1997년 1월 20일, 심사완료:1997년 10월 6일

naturally drawn considerable attention to supercomputing[3]. The RCN G(N, d) consists of N identical processors(nodes). Each processor, provided with its own sizable local memory, is connected through bidirectional, point-to-point communication channels to different neighbors by jumping dn. Due to the above mentioned properties, the recursive circulant topology has been considered by many as an ideal parallel architecture.

The routing of message is thus a key to the performance of such networks. There are routing algorithms using well-known methods, such as the Shortest Path Algorithm(the Forward Algorithm)[6], the Backward Algorithm[7], the Spanning Tree Algorithm[11]. These algorithms provide for only sequential transmission, from the source node to the desired node in a short time. We now look for algorithms that are capable of handling, multiple data items simultaneously transmitted from the staring(source) node to the destination node. There are a few algorithms on the n-dimensional hypercube network[8]-[10] that allow us to locate n disjoint paths such as the Hamiltonian path Algorithm [13], the Rotation Algorithm using Tree Structure[11], the Disjoint Path Algorithm[11], and the Routing Algorithms[12].

Generally speaking, the discovery of the maximum number of node-disjoint paths on a random network is computationally different. However, it has been proven that these paths exist in a specific network[12]. From this fact, the Routing Algorithm for finding these paths on the hypercube network has been designed.

In this paper, we propose the algebraic approach to the routing of message on the RCN. As described above, m packets are simultaneously transmitted from the starting(source) node to the destination node. In this case, the $i^{th}$ packet is sent along the $i^{th}$ path from the starting node to the destination node. In order for all packets to arrive at the destination node quickly and securely, the $i^{th}$ path must be node-disjoint from all other paths. To accomplish this, we employ the operations of nodes presented in Cayley Graph[16] and the special matrix called as Hamiltonian Circuit latin Square(HCLS)[13], which is used to find a set of n disjoint paths on hypercube network.

This paper is organized as follows:Section Ⅱ describes basic definitions and design the shortest paths between arbitrary two nodes. In Section Ⅲ, by employing the shortest path obtained from previous section and the HCLS, we propose the parallel routing algorithm for the design of m node-disjoint paths on the recursive circulant network. This paper concludes with Section Ⅳ.

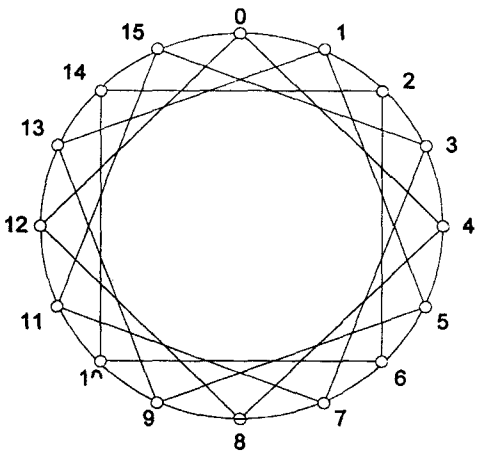
## 2. Design of the Shortest Path

Let A and B be any two nodes in the recursive circulant network(RCN). The paper's objective is to find algorithms that will facilitate the transmission of data from node A to B in that network. In order for the data to traverse from node A to node B, it must cross, successively, intermediate nodes along a path. Each address of these nodes in the path that is successively traversed by the data, is obtained by modifying the value of the bit of the address that represent the node, in the node traversal sequence along the path. In this paper, a node will be often used as the address of that node and we will often use the terms: the bit position in the node's address or the dimension of the space, interchangeably, indicating in both cases the bit position of address associated with two nodes, and subsequently identifying the link between these nodes - the link over which data is moved in its way from the source node to its neighboring node.

Definition 1 : The RCN G(N, d) is defined as follows: Let $V=\{0, 1, 2, ..., N-1\}$ as a set of nodes and $E=\{(v, w) \mid v+d^i=w(\mathrm{mod}\ N), d\geq 2\}$ as a set of edges, where $0\leq i\leq \lceil \log_d N \rceil -1$.

Researches on RCN are actively performed in graphic-theoretical area such as embedding and fault-tolerance.

In this paper, we focus on parallel routing algorithm for $G(2^m, 4)$, where $m=2k$. Node A on $G(2^m, 4)$ has an address $(a_{n-1}a_{n-2}\cdots a_i\cdots a_1a_0)$, $a_i \varepsilon \{0, 1\}$, $0\le i\le m-1$. Therefore, an address of arbitrary node starts from $(00\cdots 00)$ to $(00\cdots 01)\cdots(11\cdots 11)$. This address can be described as $(00\cdots 00)$, $(-1-1\cdots -1-1)$ $(00\cdots 0-1)$ on (mod $2^m$) computation. As mentioned earlier, a node on Cayley Graph can traverse to another node by performing a certain operation. We now employ these operations to RCN.




(Fig. 1) A recursive circulant graph G(16, 4)

Definition 2: A positive operation and a negative operation are executed on an even dimension of G $(2^m, 4)$. Then, the routing function $R_{\pm g_{2k}}$ of RCN for the $2k^{th}$ dimension is as follows:

node address $A=(a_{m-1}a_{m-2}\cdots a_1a_0)$, $a_i\in\{0, 1\}$, $0\le i\le m-1$

$$R_{\pm g_{2k}}(A)=A\pm 2^{2k}(\text{mod } 2^m),\ 0\le k\le \frac{m}{2}-1$$

Node A is physically connected to m neighboring nodes and these m paths are disjoint. The distance of Node A and its neighboring node is one. To understand the concept of Definition 2 easily, Example 1 is given.

Example 1: Let the address of starting node be (0010). According to Definition 2, four disjoint paths, each has length one, and operations are described below, where $g_i$ is an operation of $i^{th}$ dimension.

$(0010)\rightarrow(0011): g_0$
$(0010)\rightarrow(0001): -g_0$
$(0010)\rightarrow(0110): g_2$
$(0010)\rightarrow(1110): -g_2$

We now examine the operations occurring on odd dimensions of $G(2^m, 4)$. Even operations do not exist on this dimension physically, they can be considered to occur when operations are executed twice in the same direction. Because of modular computation with carry, this idea can be accomplished.

In this paper, data is transmitted from source node along the $i^{th}$ path, which is physically connected to. The path above is selected by the routing function described in Definition 2. To do this, the relative address of starting node and destination node can be obtained below.

Definition 3: The relative address $r$ of nodes A and B on G(N, d) is computed as the absolute value of difference between A and B.

$r=|A-B|$, where $A=(a_{m-1}a_{m-2}\cdots a_1a_0)$,
$B=(b_{m-1}b_{m-2}\cdots b_1b_0)$ and $r=(r_{m-1}r_{m-2}\cdots r_1r_0)$

Let two addresses of node A and node B be (0010) and (1100). What is the relative address of two nodes? The decimal value of address of node A is 2 and that of node B is 12. So, the decimal value of the relative address is 10, which is (1010) in binary notation.

Definition 4: Let T(A, S) be the logical transmission path of data starting from node A to the destination node B, where S is a multiset and a sequence of

operations, via which data can reach at the destination node. T(A, S) is determined by the order of the elements in the set S.

Given the starting node A and a multiset S, we would like to transmit to the destination node via intermediate nodes. Suppose that node A and a set S be (001101) and $\langle g_0, -g_2, -g_2, g_4\rangle$, repectively. The traversal of the data along the path outlined by the sequence of nodes is (001101)→(001110)→(001010)→(000110)→(010110). The path from node A to a destination node is obtained from T(A, S) specified in Definition 4, that is ((001101), (001110), (001010), (000110), (010110)).

A set of the two consecutive bit $(r_{2k+1}r_{2k})$ of the relative address is {00, 01, 10, 11}. By applying operations obtained from the set, data can arrive at the destination node. However, since the operations are various, a number of paths are also made. Since the paper's objective is to find algorithms that will facilitate the fast transmission of data from a starting node to a destination node, those operations which are appropriate for this objective are defined in Definition 5.

Definition 5: Let hk be the shortest distance between $(r_{m-1}r_{m-2}r_{2k+1}r_{2k}\cdots r_1r_0)$ and $(r_{m-1}r_{m-2}\cdots 00\cdots r_1r_0)$. $h_k$ and a sequence S of operations are obtained as follows:

$(r_{m-1}r_{m-2}\cdots 00\cdots r_1r_0)\rightarrow(r_{m-1}r_{m-2}\cdots 00\cdots r_1r_0)$: $\langle\rangle$, $h_k=0$;
$(r_{m-1}r_{m-2}\cdots 01\cdots r_1r_0)\rightarrow(r_{m-1}r_{m-2}\cdots 00\cdots r_1r_0)$: $\langle g_{2k}\rangle$, $h_k=1$;
$(r_{m-1}r_{m-2}\cdots 10\cdots r_1r_0)\rightarrow(r_{m-1}r_{m-2}\cdots 00\cdots r_1r_0)$:
$\langle g_{2k}, g_{2k}\rangle$, $h_k=2$ or $\langle g_{2k+2 \bmod m}, -g_{2k}, -g_{2k}\rangle$, $h_k=2$;
$(r_{m-1}r_{m-2}\cdots 11\cdots r_1r_0)\rightarrow(r_{m-1}r_{m-2}\cdots 00\cdots r_1r_0)$:
$\langle g_{2k+2 \bmod m}, -g_{2k}\rangle$, $h_k=1$;

Among the patterns of $(r_{2k+1}r_{2k})$ explained above, the interesting things are (10) and (11). The reason is that these patterns could change the shortest distance of $(2k+2)^{th}$ dimension, if available. In case of $(r_{2k+3}r_{2k+2})\in\{(10), (11)\}$, the shortest distance of $(2k+2)^{th}$ dimension is changed because $\langle g_{2k+2}\rangle$ is added to a sequence of operations on $(2k+2)^{th}$ dimension. A sequence of operations generated in Definition 3 are performed on positive direction if the address of a destination node is greater than that of a source node, otherwise on negative direction. Let the source node and the destination node be (00011010) and (00000001), respectively. $h_k$ and a sequence S of operations described in Definition 5 are 4 and $\langle g_0, g_2, g_2, g_4\rangle$. Since the address of the destination node is greater, a sequence of operations becomes $\langle -g_0, -g_2, -g_2, -g_4\rangle$.

Proposition 1: The shortest distance d(A, B) between two nodes A and B is defined as follows:

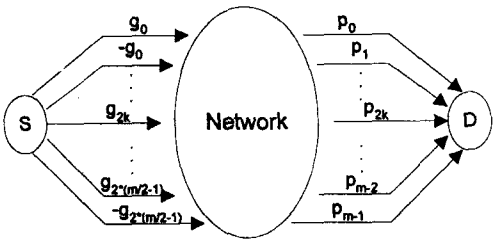
$$d(A, B)=\sum_{i=0}^{\frac{m}{2}} h_k$$

proof: Since RCN has vertex-symmetric structure, the number of links between every two dimensions is the same. If the shortest distance of two nodes is represented as a sum of the shortest distance on each dimension. Therefore a sum of $h_k$, the shortest distance of $(2k)^{th}$ dimension obtained in Definition 5, is the shortest distance of two nodes A and B.

## 3. Application of the Hamiltonian Circuit Latin Square to the Parallel Routing Algorithm on a Recursive Circulant Network

The m packets are transmitted from a source node to a destination node on $G(2^m, 4)$. According to Menger's Theorem[4], a set of m link-disjoint paths exists. In this section, we would like to construct a set of m node-disjoint and shortest paths in order to transmit these packets safely and quickly. First, m packets residing at a node on RCN are sent to its m neighboring nodes along a set of m disjoint paths(referred to Definition 2). These m paths are generated by employing m different operations at the beginning step and

by performing m different operations at the last step in order to arrive at a destination node. The figure below illustrates the operations applied to generate m paths from a source node to a destination node.




(Fig. 2) The operations applied at the first step and the last step.

The $i^{th}$ packet is transmitted along the $i^{th}$ path, the first intermediate node of which is obtained from applying the $i^{th}$ operation($g_i$) at a starting node and the last intermediate node transmits the packet to a destination node by applying the $i^{th}$ operation($p_i$). In some cases, $g_i$ and pi can be the same.

Definition 6: Let $O^s$ be a set of operations occurring at a starting node when m packets are transmitted simultaneously and Let $O^d$ be a set of operations occurring at a destination node when m packets arrive. These sets are defined as follows:

$$O^s = \{\pm g_{2k} \mid 0 \le k \le \frac{m}{2} - 1\},$$

$$O^d = \{p_0, p_1, \cdots, p_{m-2}, p_{m-1}\}, \quad O^s = O^d$$

We now apply the HCLS(Hamiltonian Circuit Latin Square) to find a set of m shortest and node-disjoint paths. A latin square is a square matrix with $m^2$ entries of m different elements, none of the elements occurring twice within any row or column of the matrix. The integer m is called the order of latin square. The next definition describes the HCLS.

Definition 7: The HCLS $M^1$ is constructed as follows: Given distinct m points $a_0, a_2, \cdots, a_{m-2}, a_{m-1}$, a Hamiltonian circuit $a_i \rightarrow a_j \rightarrow \cdots \rightarrow a_k \rightarrow a_i$ is randomly selected. On the circuit each row of M can be obtained from the Hamiltonian path, starting at any position $a_k (0 \le k \le m-1)$, under the condition that no two rows begin at the same position. If a Hamiltonian path is $a_i \rightarrow a_j \rightarrow \cdots \rightarrow a_k$, then the row obtained from it is $[a_i\ a_j \cdots a_k]$

From the definition of the HCLS given in Definition 7, the MHCM(Modified Hamiltonian Circuit Matrix) is constructed below.

Definition 8: Given the HCLS $M^1 = [a_{i,j}]$, the MHCM $M^2$ is constructed as follows: $M^2 = [A_{i,j}]$, $A_{i,j} = \{a_{i,0}, a_{i,1}, \cdots, a_{i,j-1}, a_{i,j}\}$, $0 \le i, j \le m-1$.

Referring to [13], the MHCM satisfies the conditions of the MGNDP(Matrix for Generating Node-Disjoint Paths), which is applied to the parallel routing on the hypercube network and a number of node-disjoint paths can be created. Since the HCLS belongs to a latin square, a set of elements in the first column is the same as that of the last column. On the $G(2^m, 4)$, an element in the HCLS is represented as an operation. Also, $O^s$ and $O^d$ in Definition 6 is described as a set of elements in the first column and a set of elements in the last column, respectively. We, intuitively, realize that a set of n shortest and node-disjoint paths is generated if the number of distinct sequences of operations for arriving at an arbitrary node in a short time is $n(n \le m)$. The remaining operations excluding these distinct operations from $O^s$ and $O^d$, should be performed. The following example offers a better understanding of the process explained above.

Example 2: Let A and B be (00011010) and (00110011). According to Definition 5, a sequence S of operations is computed as $\langle g_0, g_2, g_2, g_4 \rangle$ and a set of shortest and node-disjoint paths is generated as follows.

A set of distinct operations in S is $\{g_0, g_2, g_4\}$. Using these operations, $(3\times3)$ HCLS can be obtained from Definition 7.

$$HCLS = \begin{bmatrix} g_0 & g_2 & g_4 \\ g_2 & g_4 & g_0 \\ g_4 & g_0 & g_2 \end{bmatrix}$$

Operations in the $i^{th}$ row of the HCLS generated above are performed for traversal of the $i^{th}$ packet and the remaining operation $g_2$ is also executed at the point except the first and the last points. In order to assure that these paths are node-disjoint, the remaining operation should be executed at the same time. In this example, the running point of the remaining operation $g_2$ is the second. Physical transmission paths from node A to node B are described below.

Path 1: T(A, $\langle g_0, g_2, g_2, g_4\rangle$):
A→(00011011)→(00011111)→(00100011)→B
Path 2: T(A, $\langle g_2, g_2, g_4, g_0\rangle$):
A→(00011110)→(00100010)→(00110010)→B
Path 3: T(A, $\langle g_4, g_2, g_0, g_2\rangle$):
A→(00101010)→(00101110)→(00100011)→B

From Definition 6, $O^s$ and $O^d$ are obtained, that is, $O^s = O^d = \{g_0, -g_0, g_2, -g_2, g_4, -g_4, g_6, -g_6\}$. Excluding the operations $g_0, g_2, g_4$, which are performed in the first and the last steps of these paths above, from $O^s$ and $O^d$, $O^s$ and $O^d$ become $\{-g_0, -g_2, -g_4, g_6, -g_6\}$.

Recall that we deal with the design of eight node-disjoint paths, out of which five node-disjoint paths are now constructed. Examining Definition 2, when a packet traverses in a positive direction on $i^{th}$ dimension and then traverses in a reverse direction on the same dimension, a packet comes back to its original position. This idea is applied to generation of disjoint paths. If $\{g_{2k}, -g_{2k}\}$ exists as a subset of $O^s$ and $O^d$, these operations are executed at the first and the last steps of two paths newly constructed, and operations obtained from the design of the shortest distance previously described at the middle steps of them. In case of $k=3$, Path 4 and Path 5 are constructed below.

Path 4: T(A, $\langle g_6, g_0, g_2, g_2, g_4, -g_6\rangle$):
A→(01011010)→(01011011)→(01011111)→(01100011)→(01110011)→B
Path 5: T(A, $\langle -g_6, g_0, g_2, g_2, g_4, g_6\rangle$):
A→(110110110)→(110110111)→(11011111)→(11100011)→(11110011)→B

Excluding $g_6$ and $-g_6$ from $O^s$ and $O^d$, $O^s$ and $O^d$ become $\{-g_0, -g_2, -g_4\}$. While the operations at the first and the last steps for paths designed so far are not the same, the operations occurring at these steps for the remaining three paths are the same. As described earlier for Path 4 and Path 5, a packet traverses in a positive direction on $i^{th}$ dimension and then traverses in a reverse direction on the same dimension. In this case, a packet traverses twice in the same direction on $i^{th}$ dimension and then traverses twice in the reverse direction on the same dimension, then, a packet comes back to its original position. A sequence of operations for a path is now obtained. Two operations traversed in the same direction are chosen for the first and the last steps. Another two operations in reverse direction and operations for the shortest distance are changed to a sequence of minimum number of operations(referring to Definition 5), and then this sequence of operations is executed for middle steps. Paths 6, 7, 8 are generated by handling cases of $(-g_0, -g_2$ and $-g_4)$, respectively.

Path 6: T(A, $\langle -g_0, g_0, g_0, g_0, g_2, g_2, g_4, -g_0\rangle$) T(A, $\langle -g_0, -g_0, -g_2, g_4, g_4, -g_0\rangle$):
A→(00011001)→(00011000)→(00010100)→(00100100)→(00110100)→B
Path 7: T(A, $\langle -g_2, g_2, g_2, g_0, g_2, g_2, g_4, -g_2\rangle$) T(A, $\langle -g_2, g_0, g_4, g_4, -g_2\rangle$): A→(00010110)→(00010111)→(00100111)→(00110111)→B
Path 8: T(A, $\langle -g_4, g_4, g_4, g_0, g_2, g_2, g_4, -g_4\rangle$) T(A,

$\langle -g_4, g_0, g_2, g_2, -g_4, g_6, -g_4\rangle$):
$A \rightarrow (00001010) \rightarrow (00001011) \rightarrow (00001111) \rightarrow$
$(00010011) \rightarrow (00000011) \rightarrow (01000011) \rightarrow B$

The process to find a set of a shortest and node-disjoint paths is described above. We now propose a parallel routing algorithm that generates a set of m minimum-distance and node-disjoint paths for the RCN. In this paper, we will use the term "distance" between two nodes in a interconnection network to refer to the number of routing steps(also called hopcounts) needed to send a message from one node to another.

*RCN_Routing_Algorithm*

A ← an address of a starting node A

B ← an address of a destination node B

$O^s$ ← a set of operations occurring at a starting node A

$O^d$ ← a set of operations requisite for getting to a destination node B

begin

(1) Compute the relative address R of nodes A and B; $R = |A - B|$

(2) Using the relative address R, a sequence S of operations to arrive at node B in a short time are produced.

(3) In order to design a set of shortest and node-disjoint paths, find a set $S_1$ of distinct elements in S. A set of $|S_1|$ shortest and node-disjoint paths are generated. Each path of length is $|S|$,

(3-1) Using the set $S_1$, $(n \times n)$ HCLS is constructed, where $n = |S_1|$.

(3-2) Operations in the $i^{th}$ row of the HCLS are performed for traversal of the $i^{th}$ packet and the remaining operations in S should be executed at the point except the first and the last points.

(3-3) $O^s \leftarrow O^s - S_1$ and $O^d \leftarrow O^d - S_1$.

(4) Construct two node-disjoint paths, each path has length $|S| + 2$.

(4-1) If $O^s = \phi$, the process is finished.

(4-2) If a set of $\{g_{2k}, -g_{2k}\}$ is found in $O^s$, then these operations are performed at the first and the last steps of two paths newly designed, and operations in S at the middle steps of them, otherwise go to (5).

(4-3) $O^s \leftarrow O^s - \{g_{2k}, -g_{2k}\}$, $O^d \leftarrow O^d - \{g_{2k}, -g_{2k}\}$ and go to (4-1).

(5) Generate the remaining paths, each of which has length $\leq |S| + 3$

(5-1) If $O^s = \phi$, the process is finished.

(5-2) Produce a sequence $S_2$ of minimum number of operations by reducing the size of $S \cup \{-g_i, -g_i\}$, $g_i \in O^s$(see Definition 5).

(5-3) Operation gi is performed at the first and the last steps at traversal and operations of $S_2$ are executed at the middle steps.

(5-4) $O^s \leftarrow O^s - \{g_i\}$, $O^d \leftarrow O^d - \{g_i\}$ and go to (5-1).

end.

The routing algorithm of the recursive circulant network is similar to that of the hypercube network. For the RCN, the number of links is determined by the jumping number. Considering the structure of this network, the following proposition is described.

Proposition 2: Let A and B be any two nodes in the RCN and assume the number of distinct elements in S is $|S_1|$. Then there are $|S_1|$ parallel paths of length $|S|$ between A and B.

Proof: Let $|S_1| = k$. Then the operation positions that differ between A and B are $\{p_1, p_2, ..., p_k\}$. We can write k permutation of this set, indicating the different operation positions for k parallel paths of length k. These k permutation are used to design the HCLS. Using this matrix, k parallel paths are obtained automatically. In order for these paths to have length $|S|$, the remaining operations in S are now performed. If each operation is running at the same point except the first and last points and at the same time on k traversals, these paths to B must be node-disjoint.

RCN_Routing_Algorithm is thus fairly straightfor-

ward. The time involved in performing Steps (1), (2) and (4) is small compared to the remaining steps. The first, second and fourth steps of this algorithm does not, therefore, contribute to an objectionable overhead.

Theorem 1 : The construction of a set of m node-disjoint paths can be performed in $O(n^2)$ time.

Proof : Applying the Algorithm above to generate m node-disjoint paths. Important steps for determining time complexity requisite for the Algorithm are two things. One is to design the HCLS, which requires O (n). The other is to run Step (5) of RCN_Routing_Algorithm. In Step (5), in order to run $g_i$ in $O^s$ at the first and last steps in transmission, a sequence of operations is determined as $S \cup \{-g_i, -g_i\}$, and is reduced by the rules described in Definition 5. Since Reducing process requires O(n) time and the number of elements in $O^s$ is less than n, Step (5) can be computed in $O(n^2)$. Therefore, a set of m node-disjoint paths can be created in $O(n^2)$ time.

The paper's objective is to find a set of m shortest and node-disjoint paths between two nodes. The major topological characteristics of the RCN is considered and the property of m paths obtained from the Algorithm above is proven.

Theorem 2 : The m transmission paths produced by RCN_Routing_Algorithm are shortest and node-disjoint.

Proof : The m packets residing at node A are now transmitted at time $t_0$. These packets reach to its m neighboring nodes at time $t_1$. Then, each packet traverses to a neighboring node of a destination node along a shortest path(referring to Definition 5). Suppose that two packets arrive at the same node except a destination node during transmission. In order for this case to occur, the following condition should be satisfied. Let $S_i$ and $S_j$ be two sequences of operations for sending two packets from a starting node to two arbitrary nodes at time $t_{ij}$, where $t_{ij}$ means that one packet arrives at time $t_i$ and the other arrives at time $t_j$. Then, $S_i$ and $S_j$ should be the same. In other words, if these sequences do not appear, a set of node-disjoint paths can be constructed. According to the Algorithm described above, three classes of paths are generated on this network. We now consider three cases.

Case 1 : Let $S_{1i}$ and $S_{1j}$ be two sequences of operations obtained from design of the shortest distance. Then $S_{1i}$ and $S_{1j}$ must not be the same due to the properties of the MGNDP.

Case 2 : Let $S_{2i}$ and $S_{2j}$ be two sequences of operations acquired by running Algorithm-(4). Then $S_{2i}$ and $S_{2j}$ must not be the same because the first elements of $S_{2i}$ and $S_{2j}$ are $g_k$ and $-g_k$, respectively and the rests of them are the same. In order for paths generated through case 1 and case 2 to be node-disjoint, we prove that $S_{1i}$ and $S_{2j}$ must not be the same. Considering the first element $g_k$ of $S_{2j}$, it is not an element of $S_{1i}$. Therefore, these sequences are different all the times.

Case 3 : Let $S_{3i}$ and $S_{3j}$ be two sequences of operations generated by performing Algorithm-(5), and T be a sequence of operations, which creates a shortest path from a source node to a desired node. Then, the first elements of $S_{3i}$ and $S_{3j}$ do not belong to T, $S_{3i}$ does not contain the first element of $S_{3j}$, and $S_{3j}$ does not contain the first element of $S_{3i}$. So, $S_{3i}$ and $S_{3j}$ are not be the same. To prove that the paths created by all the cases are node-disjoint, $S_{3i}$ and $S_{2j}$ must not be the same. A method of proof is the same as the case 2. Looking into the first element of $S_{2j}$, the element is not a part of $S_{3i}$. Therefore, these sequences are different all the times.

## 4. Conclusion

In this paper, we present the algorithm that generates a set of m shortest and node-disjoint paths on G(N, d), employing the Hamiltonian Circuit Latin Square(HCLS), $N = 2^m$, $d = 4$. Even N and d are fixed

values, the algorithm can be easily extended on arbitrary recursive circulant networks. Important steps for determining time complexity requisite for the algorithm are two things. One is to design the HCLS, which needs O(n). The other is to execute Step (5) of RCN_Routing_Algorithm, which requires $O(n^2)$. Therefore, we can create $O(n^2)$ parallel routing algorithm for constructing m shortest and node-disjoint paths.

## References


[1] Akers, S.B., and Krishnamurthy, B., "On Group Graphs and Their Fault Tolerance," IEEE Trans. Comput., vol. c-36, no. 7, pp. 885-888, July 1987.

[2] Hwang, K., and Briggs, F.A., Computer Architecture and Parallel Processing. McGraw-Hill, New York, 1984.

[3] Park, J.H., Circulant Graph and Their Application to Communication Network. Ph.D. Thesis, Dept. Computer Science, KAIST, Daejeon, Korea, 1992.

[4] Kim, S.B., Fault Analysis and mesh embedding of recursive circulant graphs, M.S Thesis, Dept. Computer Science, KAIST, Daejeon, Korea, 1992.

[5] Kim, S., Park, J., Chwa, K., "Embedding of Recursive Circulant Graph into Complete Biranry Tree," Proceeding of KISS Fall Conference, J. Korea Info. Sci. Soc., vol. 19, no. 2, pp. 919-922, 1992.

[6] Basse, S., Computer Algorithms: Introduction to Design and Analysis, Addition-Wesley, Reading, MA, 1978.

[7] Stallings, W., Data and Computer Communications. Macmillan Publishing Company, New York, 1985.

[8] Saad, Y. and Schultz, M.H., "Topological Properties of Hypercubes," IEEE Trans. Comput., vol. 37, no. 7, pp. 867-872, July 1988.

[9] Chen, H.-L. and Tzeng, N.-F., "On-Lins Task Migration in Hypercubes Through Double Disjoint Paths," IEEE Trans. Comput., vol. 46, no. 3, pp. 379-384, Mar. 1997.

[10] Wu, J. and Huang, K., "The Balanced Hypercube : A Cube-Based System for Fault-Tolerant Application," IEEE Trans. Comput., vol. 46, no. 4, pp. 484-490, Apr. 1997.

[11] Johnson, S.L. and Ho, C-T., "Optimum Broadcasting and Personalized Communication in Hypercube," IEEE Trans. Comput., vol. 38, no. 9, pp. 1249-1268, Seep. 1989.

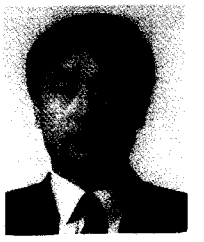
[12] Rabin, M.O., "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," J. ACM, vol. 36, no. 2, pp. 335-348, Apr. 1989.

[13] Chung, Il-Yong, "Application of the Special Latin Squares to the Parallel Routing Algorithm on Hypercube," J. Korea Info. Sci. Soc., vol. 19, no. 5. Sep. 1992.

[14] Gibbons, A., Algorithmic Graph Theory. Cambridge University Press, New York, 1985.

[15] Denes, J. and Keedwell, A.D., Latin Square and Their Applications. Academic Press, New York, 1974.

[16] Stone, H.S., Discrete Mathematical Structures and Their Applications. SRA, Chicago, IL., 1973.


**배 용 근**

1984년 조선대학교 전산기공학과 졸업(공학사)

1986년 조선대학교 전자공학과 졸업(공학석사)

현재 원광대학교 전자공학과 박사과정

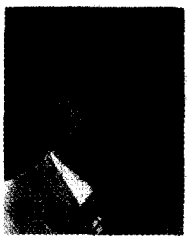
1984년~1988년 조선대학교 전자계산소 근무

1988년~현재 조선대학교 전자계산학과 조교수

관심분야: 마이크로 프로세서 응용, 병렬처리, 멀티미디어

박 병 권

1988년 조선대학교 전자계산학과 졸업(학사)
1990년 조선대학교 대학원 전자계산학과 졸업(석사)
1996년~현재 조선대학교 대학원 전자계산학과 박사과정
1991년~1994년 광주은행 전산업무부 근무
1995년~현재 서강전문대학 전자계산과 전임강사
관심분야:소프트웨어 공학, 객체지향 시스템



정 일 용

1983년 한양대학교 공과대학 졸업(공학사)
1987년 미국 City University of New York 전산학과(전산학석사)
1991년 미국 City University of New York 전산학과(전산학박사)
1991년~1994년 한국전자통신연구소 선임연구원
1994년~현재 조선대학교 전자계산학과 조교수
관심분야:네트워크 관리, 분산시스템 보안, 코딩이론, 병렬 알고리즘