Journal of Korea Multimedia Society Vol. 16, No. 7, July 2013(pp. 897-904)
http://dx.doi.org/10.9717/kmms.2013.16.7.897

Design of a set of One-to-Many Node-Disjoint and Nearly Shortest Paths on Recursive Circulant Networks

Ilyong Chung†

ABSTRACT

The recursive circulant network G(N,d) can be widely used in the design and implementation of parallel processing architectures. It consists of N identical nodes, each node is connected through bidirectional, point-to-point communication channels to different neighbors by jumping d^i, where $0 \le i \le \lceil \log_d N \rceil - 1$. In this paper, we investigate the routing of a message on $G(2^m,4)$, a special kind of RCN, that is key to the performance of this network. On $G(2^m,4)$ we would like to transmit k packets from a source node to k destination nodes simultaneously along paths on this network, the i^{th} packet will be transmitted along the i^{th} path, where $1 \le k \le m-1$, $0 \le i < m-1$. In order for all packets to arrive at a destination node quickly and securely, we present an $O(m^4)$ routing algorithm on $G(2^m,4)$ for generating a set of one-to-many node-disjoint and nearly shortest paths, where each path is either shortest or nearly shortest and the total length of these paths is nearly minimum since the path is mainly determined by employing the Hungarian method.

Key words: Recursive circulant network, Parallel routing algorithm, Hungarian method

1. INTRODUCTION

Research in interconnection networks has used graph-theoretic properties for its investigations and has produced various interconnection schemes. Many of these schemes have been designed to optimize important parameters such as degree, diameter, fault-tolerance, hardware cost, and to address the needs of particular applications [1,2].

The recursive circulant network (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 d^i, where $0 \le i \le \lceil \log_d N \rceil - 1$. Due to these properties, the RCN can be widely used in the design and implementation of parallel processing architectures.

$G(2^m,4)$, a special kind of RCN, has degree m, has maximum connectivity, and compares favorably with the hypercube Q^m[3,4]. Due to the above mentioned properties, $G(2^m,4)$ can also be widely used in the design and implementation of parallel processing architectures.

The routing of messages is thus key to performance of this network. We look for algorithms that are capable of handling multiple data items simultaneously transmitted from a source node to k destination nodes ($1 \le k < m$). In order for all packets to arrive at destination nodes quickly and securely, each path must be disjoint from all other paths.

Disjoint paths can be categorized into three classes-one-to-one, one-to- many, and many-to-many. The first class considers the disjoint paths from a source node to a destination node, the second from a source node to k destination nodes and the third from k source nodes to k destination nodes. One-to-one disjoint paths were constructed

※ Corresponding Author : Ilyong Chung, Address : Dept. of Computer Engineering, Chosun University, Seosoeg-Dong, Dong-Gu, Gwangju, 501-759, Korea, TEL : +82-62-230-7712, FAX : +82-62-230-7754, E-mail : iyc@chosun.ac.kr
Receipt date : Mar. 14, 2013, Approval date : Apr. 16, 2013
† Department of Computer Engineering, Chosun University
*This study wae supported by research funds from Chosun University, 2013.

on several networks such as hypercubes[5], k-ary n-cubes[6] and star graphs[7]. One-to-many disjoint paths were designed on hypercubes [8,9] and star graphs [10]. For the generation of many-to-many disjoint paths some work has been done [11,12].

In this paper, we would like to transmit k packets from a source node to k destination nodes simultaneously along paths on this network, the i^{th} packet will be transmitted along the i^{th} path ($0 \leq i < m-1$). In order for all packets to arrive at these destination nodes quickly and securely, we should construct a set of k node-disjoint paths with a minimum total length. To accomplish this, we employ the operations of nodes presented in the Cayley Graph [13] and employ the Hungarian method [14,15].

2. DESIGN OF THE SHORTEST PATH

Let A and B be any two nodes on $G(2^m,d)$. One of the paper's objectives is to find an algorithm that will facilitate the transmission of data from node A to node B on this network. In order for the data to traverse from node A to node B, it must cross a number of intermediate nodes along a path in succession.

Definition 1. $G(2^m,d)$ is defined as follows: Let $V = \{0,1,2,...,2^m-1\}$ be a set of nodes and $E = \{(v, w) | v + d I = w \pmod{2^m}\ 0 \leq i \leq \lceil \log_d 2^m \rceil - 1, d \geq 2\}$ be a set of edges.

Research on the graph-theoretical subjects of networks and systems such as embedding and fault tolerance have been performed[16,17]. In this paper, we focus on a parallel routing algorithm on $G(2^m,4)$. Node A on $G(2^m,4)$ has an address $(a_{m-1}a_{m-2}...a_i...a_1a_0)$, $a_i \in \{0,1\}$, $0 \leq i \leq m-1$. Therefore, possible addresses for an arbitrary node are from (00...00) to (00...01)...(11...11). This address can be described as (00...00), $(-1\ -1...\ -1\ -1)$...$(00...0\ -1)$ on (mod 2^m) computation. A packet on any node can move to another node by performing a routing operation g_i. It is executed on only even bit of $G(2^m,4)$ since node v is connected to w, $w = v+4^i \pmod{2^m}$. The routing function is described in Definition 2.

Definition 2. The routing function of $G(2^m,4)$ for the i^{th} bit is defined as follows: $R_{g\pm i}(A) = A \pm 2^i \pmod{2^m}$ $0 \leq i \leq m-1$, where i is an even number.

Node A is connected to its m neighboring nodes by performing m distinct operations. Among these m operations, $\frac{m}{2}$ operations are in a positive direction, while $\frac{m}{2}$ operations are in a negative direction. Employing positive operations - g_0 and g_2 on $G(16,4)$, (0010) is connected to (0010) $+2^0$ and (0010)$+2^2$, respectively. (0010) is connected to (0010) -2^0 and (0010) -2^2 employing negative operations - g_{-0} and g_{-2}, respectively. Let us consider routing operations for an odd bit. The routing for the $(i+1)^{th}$ bit can be described using the routing function for the i^{th} bit in Definition 2, that is $R_{g\pm i}(R_{g\pm i}(A)) = A \pm 2^i \pm 2^i \pmod{2^m}$. For example, we would like to send a packet from (0000) to (1000). The routing for the 3^{rd} bit should be performed. $R_{g2}(R_{g2}(0000))$, this means that the packet is transmitted from (0000) to (0100) by executing g_2, and then to (1000) by executing the same operation.

The i^{th} packet is transmitted from a source node along the i^{th} path. An intermediate node in this path is obtained from the routing function described in Definition 2. To do this, the relative address of a starting node and a destination node is computed below.

Definition 3. The relative address r of nodes A and B on $G(2^m,4)$ is computed as the absolute value of the difference between A and B, $r=|A-B|$.

Definition 4. Let $T(A,S)$ be the path of data starting from node A to a destination node, where

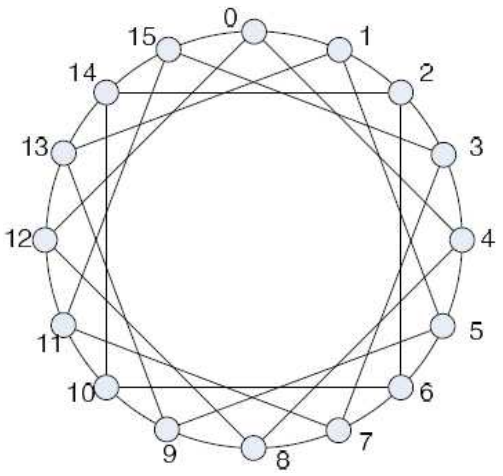

Fig. 1. The recursive circulant graph $G(2^4,4)$.

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

Given node A and sequence S, the next example will illustrate how to transmit to a destination node via intermediate nodes.

Example 1. Let node A and sequence S be (001101) and $<g_0,g_{-2},g_4>$. The traversal of the data along the path is (001101) →(001110) → (001010) →(000110) →(010110).

The final node of $T(A,<g_i,g_i,g_i,g_i,>)$ is the same as that of $T(A,<g_{i+2}>)$. Also, the final node of $T(A,<g_2,g_{-2},g_4>)$ is the same as that of $T(A,<g_4>)$. A packet move to $A+2^2$ by performing g_2 and then move to $A+2^2-2^2$ by performing g_{-2}. It means that a packet comes back to the original node A. By applying operations obtained from this sequence, data can arrive at the destination node. By reordering the operations involved in the shortest path from node A to node B, various other paths may be constructed. Since this paper's objective is to find algorithms that will facilitate the fast transmission of data from a starting node to a destination node, these operations which are appropriate for this objective are defined in Definition 5.

Definition 5. Let h_k be the shortest distance between $(r_{m-1}r_{m-2}...00...r_1r_0)$ and $(r_{m-1}r_{m-2}...r_{2k+1}r_{2k}...r_1r_0)$, where h_k is the number of operations consisting of g_{2k} and g_{-2k}. h_k, S_k, h_{k+1} and S_{k+1} are determined as follows:

$(r_{m-1}r_{m-2}...00...r_1r_0) \rightarrow (r_{m-1}r_{m-2}...00...r_1r_0)$:
  $S_k = <>$, $h_k=0$;
$(r_{m-1}r_{m-2}...00...r_1r_0) \rightarrow (r_{m-1}r_{m-2}...01...r_1r_0)$:
  $S_k = <g_{2k}>$, $h_k=1$;
If $((2k + 2) < m)$ then {
$(r_{m-1}r_{m-2}...00...r_1r_0) \rightarrow (r_{m-1}r_{m-2}...10...r_1r_0)$:
  $S_k =<g_{2k},g_{2k}>$, $h_k=2$
  or $S_{k+1} =<g_{2k+2}>$, $S_k =<g_{-2k},g_{-2k}>$, $h_{k+1}=1$, $h_k=2$;
$(r_{m-1}r_{m-2}...00...r_1r_0) \rightarrow (r_{m-1}r_{m-2}...11...r_1r_0)$:
  $S_{k+1} =<g_{2k+2}>$, $S_k =<g_{-2k}>$, $h_{k+1}=1$, $h_k=1$ }

If $(m = (2k + 2))$ then {
$(00r_{m-3}r_{m-4}...r_1r_0) \rightarrow (10r_{m-3}r_{m-4}...r_1r_0)$:
  $S_k =<g_{2k},g_{2k}>$, $h_k=2$;
$(00r_{m-3}r_{m-4}...r_1r_0) \rightarrow (11r_{m-3}r_{m-4}...r_1r_0)$:
  $S_k =<g_{-2k}>$, $h_k=1$ }

If $(m = (2k + 2))$ then {
$(0r_{m-2}r_{m-3}...r_1r_0) \rightarrow (0r_{m-2}r_{m-3}...r_1r_0)$:
  $S_k =<>$; $h_k=0$;
$(0r_{m-2}r_{m-3}...r_1r_0) \rightarrow (1r_{m-2}r_{m-3}...r_1r_0)$:
  $S_k =<g_{2k}>$, $h_k=1$; }

If the address of a source node is greater than that of a destination node, a sequence of operations are performed in the reverse direction. Let the source node and the destination node be (11000010) and (00000001). Then the relative address of these nodes and the sequence of operations are (11000001) and $<g_0,g_{-6}>$. Since the address of the source node is greater, the sequence of operations becomes $<g_{-0},g_6>$. The following proposition, which is needed to compute the shortest distance between two nodes, is taken from Kim and Chung[7].

Proposition 1. The shortest distance d(A, B) between two nodes A and B is

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

3. Design of a set of One-to-Many Node-Disjoint and Nearly Shortest Paths on Recursive Circulant Networks

In this section, we would like to construct a set of k node-disjoint and nearly shortest paths on $G(2^m,4)$ in order to transmit k packets securely and quickly, where $1 \leq k < m$. First, these packets residing at a starting node are sent to its k neighboring nodes by employing k different operations. Then these packets are transmitted to k destination nodes along k node-disjoint paths employing the Hungarian method, where the i^{th} packet is transmitted to the i^{th} destination node. The Hungarian method is a combinatorial optimization algorithm which solves the assignment problem in cubic time $O(m^3)$. In this paper, this method models an assignment problem as an (k×k) communication cost matrix, each element of which represents the cost of communicating a packet from one node to another node. Here, communication cost means the distance between two nodes on $G(2^m,4)$.

We now transmit seven packets from node 0 to nodes 3, 5, 6, 7, 8, 9 and 64 on $G(2^8, 4)$. First, these packets are sent to node 0's 7 neighboring nodes by employing seven distinct operations - g_{-4},g_{-2},g_{-0},g_0,g_2,g_4,g_6 and then reach at nodes 240,252,255, 1,4,16 and 64. To find a set of seven node-disjoint and nearly shortest paths from these intermediate nodes to seven destination nodes, the assignment problem will be employed. Since the 7^{th} packet reaches at the 7^{th} destination node by a single hop, a (6×6) communication cost matrix M^0 is constructed by computing a shortest distance from a neighboring node to a destination node, where $M^0 = (m_{ij}^0)$. Employing the Hungarian method, M^1 is obtained.

$$M^0 = \begin{bmatrix} 3 & 3 & 4 & 4 & 3 & 4 \\ 3 & 5 & 4 & 3 & 2 & 3 \\ 1 & 4 & 3 & 2 & 3 & 4 \\ 3 & 1 & 2 & 3 & 3 & 2 \\ 1 & 1 & 2 & 2 & 1 & 2 \\ 4 & 5 & 4 & 3 & 4 & 5 \end{bmatrix}$$

$$M^1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 3 & 1 & 1 & 0 & 0 \\ 0 & 3 & 1 & 1 & 2 & 2 \\ 2 & 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 & 1 & 1 \end{bmatrix}$$

For example, $m_{00}^0 = 3$ means that the distance of the path from the 0^{th} neighboring node to the 0^{th} destination node is 3. The relative address of these nodes is (11101101) and the sequence of operations is $<g_{-0},g_2,g_4>$. For m_{11}^0, the relative address of these nodes is (11110111) and the sequence of operations is $<g_0,g_2,g_2>$. So, the path is "node 252 → node 253 → node 1 → node 5" However, this path crosses with Path 3 at node 1. In order not to meet at node 1, operation g_{-6} is performed at node 253. At last step, operation g_6 will be performed to reach at node 5. So, the sequence of operations is $<g_0,g_{-6},g_2,g_2,g_6>$ and the path is "node 252 → node 253 → node 189 → node 193 → node 197 → node 5" Therefore, m_{11}^0 is 5, not 3. Applying the Hungarian method, we select the zeros from column numbers 4,5,0,1,2 and 3, respectively. This means that packets are transmitted from nodes 240,252,255,1,4 and 16 to nodes 8,9,3,5,6 and 7, respectively. To reach at the destination nodes, operations described in Definition 5 should be performed. Operations for path 0, path 1, path 2, path 3, path 4, path 5, and path 6 are $<g_2,g_2,g_4>$, $<g_{-6},g_{-2},g_0,g_4,g_6>$, $<g_2>$, $<g_2>$, $<g_{-6},g_0,g_0,g_6>$, and $<g_{-2},g_{-6},g_{-2},g_{-0},g_6>$, respectively. If the first operation is g_i, then the order of operations performed next is from the lowest to the highest. Suppose that the first operation is g_2 and the sequence is $<g_6,g_2,g_{-0},g_{-4}>$. Then the orders of operations is $<g_2,g_{-4},g_{-0},g_2,g_6>$. If arbitrary two paths cross at the same node, g_{-6} is performed just before reaching at the meeting node on one of these

paths to avoid this collusion, and then g_6 is performed at the last step to reach at a destination node. Given the first operation and a sequence of operations obtained from the communication cost matrix M^1, the transmission paths from a starting node are constructed below.

Path 0: node 0 → node 240 → node 244 → node 248 → node 8.
Path 1: node 0 → node 252 → node 188 → node 184 → node 185 → node 201 → node 9.
Path 2: node 0 → node 255 → node 3.
Path 3: node 0 → node 1 → node 5.
Path 4: node 0 → node 4 → node 196 → node 197 → node 198 → node 6.
Path 5: node 0 → node 16 → node 12 → node 204 → node 200 → node 199 → node 7.
Path 6: node 0 → node 64.

The process to find a set of node-disjoint and nearly shortest paths is described above. We now propose an one-to-many parallel routing algorithm on G(2^m,4). In this paper, we will use the term "distance" between two nodes to refer to the number of routing steps (also called the hopcount) needed to send a message from one node to another.

OTM −RCN −Routing

A ← a starting node
N_i ← the i^{th} neighboring node of node A ($0 \le i < m-1$)
D_i ← the i^{th} destination node ($0 \le i < m-1$)

begin
(1) Send k packets from A to its k neighboring nodes by performing k distinct
operations ($1 \le k < m$).
(2) Check that N_i is equal to D_j . If so, then the i^{th} path is determined.
(3) If the number of paths to be determined is k, then given k neighboring nodes and k destination nodes, a ($k \times k$) communication cost matrix M can be constructed. m_{ij} is the length of the path required for transmitting the i^{th} packet from the i^{th} neighboring node to the j^{th} destination node under the condition that the path should not meet at N_a, one of neighboring nodes. If an intermediate node on this path is N_a, this path should be detoured not to reach at N_a. To do this, g_{-6} is performed just before reaching at N_a and g_6 is performed at the last step to reach at a destination node. The distance of the path is the number of operations described in Definition 5.
(4) In order to design a set of node-disjoint and nearly shortest paths, the Hungarian method is applied to the communication cost matrix.
(5) From the cost matrix, we obtain the length of a path between a neighboring node to a destination node, which is the number of operations in the sequence, the order of which is from the lowest to the highest. If arbitrary two paths P_i, P_j cross at the same node, then operations $<g_l, g_{-l}>$ are performed to P_j to avoid this occurrence. On Path P_j , g_{-l} is performed just before reaching at the meeting node and g_l is performed at the last step to reach at a destination node, where if m is even, then $l = m-2$, otherwise $l = m-1$.
(6) We transmit a packet from a starting node to a destination node via the corresponding neighboring node by performing the sequence of operations.

end.

Execution of *OMT −RCN −Routing* is thus fairly straightforward. The time involved in performing Steps (1), (2) and (6) is small compared to the remaining steps. The first, second, and the sixth steps of this algorithm do not, therefore, contribute to an objectionable overhead.

Theorem 1. *OMT −RCN −Routing* can be performed in $O(m^4)$.

Proof. There are three important steps for determining the time complexity requisite for the

Algorithm. The first is to construct a communication cost matrix, which requires $O(m^2)$ in Step (3) of the Algorithm. The second is to execute the Hungarian method in Step (4), which can be computed in $O(m^3)$. The final is to construct a set of k node-disjoint paths in Step (5), which requires $O(m^4)$. Therefore, the time complexity of the Algorithm is $O(m^4)$.

The paper's objective is to design a set of k node-disjoint paths from a single source node to k destination nodes ($1 \le k < m$). The major topological characteristics of $G(2^m,4)$ are considered and the requisite properties of k paths obtained from the Algorithm are proven below.

Theorem 2. The k transmission paths produced by *OTM-RCN-Routing* are node-disjoint and nearly shortest.

Proof. Let S_i and S_j be two sequences of operations for sending two packets ets from starting node A to two destination nodes, where $S_i = <g_{1i}, g_{2i},...,g_{ti},g_{(t+1)i},...,g_{xi}>$, $S_j = <g_{1j},g_{2j},...,g_{tj},g_{(t+1)j},...,g_{yj}>$, $1i<1j$. $<g_{2i},...,g_{ti},g_{(t+1)i},...,g_{xi}>$ and $<g_{2j},...,g_{tj},g_{(t+1)j},..., g_{yj}>$ are ordered sequences from the lowest to highest. Suppose that two packets arrive at the same node. In order for this case to occur, $S_{ti}=S_{wj}$, where S_{ti} and S_{wj} are the subsequences of operations performed until time t and time w, respectively, where $S_{ti} = <g_{1i},g_{2i},,...,g_{(t-1)i}, g_{ti}>$, $S_{wj} = <g_{1j},g_{2j},...,g_{(w-1)j},g_{wj}>$, $t \le x$, $w \le y$. However, these sequences do not appear. To prove it, we consider four cases.

Case 1: Suppose that $g_{-1} \notin S_{ti}$ and $g_{-1} \notin S_{wj}$. Intuitively, S_{ti} is not S_{wj} since $S_{ti}-g_{1i}$ and $S_{wj}-g_{1j}$ are ascending-ordered and the length of S_i and S_j obtained from a communication cost matrix is either shortest or nearly shortest. However, if $S_{ti}=S_{wj}$, then g_{-1} and g_1 are added to S_i to avoid this occurrence, where g_{-1} is added between $g_{(t-1)i}$ and g_{ti}, and g_1 is added after g_{xi} in S_i, where if m is even, then $1 = m-2$, otherwise $1 = m-1$. So, S_{ti} is $<g_{1i},g_{2i},...,g_{(t-1)i},g_{-1}>$. It is contradiction since $g_{-1} \in S_{ti}$.

Case 2: Suppose that $g_{-1} \in S_{ti}$ and $g_{-1} \notin S_{wj}$. If $g_{-1} \notin S_{t'i}(t' < t)$, then this case is the same as Case 1. Otherwise, S_{ti} is not S_{wj} since g_{-1} is an element of S_{ti}.

Case 3: Suppose that $g_{-1} \notin S_{ti}$ and $g_{-1} \in S_{wj}$. If $g_{-1} \notin S_{w'j}(w' < w)$, then this case is the same as Case 1. Otherwise, Sti is not S_{wj} since g_{-1} is an element of S_{wj}.

Case 4: Suppose that $g_{-1} \in S_{ti}$ and $g_{-1} \in S_{wj}$. If $g_{-1} \notin S_{t'i}$ or $g_{-1} \notin S_{w'j}$ ($t' < t, w' < w$), then this case is the same as one of three cases above. Otherwise, S_{ti} is not S_{wj} since $S_{t'i} \neq S_{w'j}$, $S_{ti} - S_{(t'+1)i}$ and $S_{wj} - S_{(w'+1)j}$ are ascending-ordered and two packets arrive at the two distinct nodes when performing $S_i - g_{xi}$ and $S_j - g_{yj}$. It means that the two paths generated by performing $S_i - g_{xi}$ and $S_j - g_{yj}$ are node-disjoint just before arriving at the corresponding destination nodes and the final operations g_{xi} and g_{yj} are performed, where $g_1=g_{xi}=g_{yj}$.

The total length of these paths is minimal at most cases since the number of operations is obtained by employing the Hungarian method. However, depending on selecting which elements in the modified cost matrix(see M^1 in Section 3), two arbitrary paths may cross at the same node. It causes these paths not to be node-disjoint. So, one of these paths should be detoured to avoid this occurrence, which makes the total length of them longer. Therefore, the Algorithm constructs a set of k node-disjoint and nearly shortest paths.

4. CONCLUSION

In this paper, we present an algorithm that generates a set of k nearly shortest and node-disjoint paths on $G(2^m,4)$ from a source node to k destination nodes employing the Hungarian method. There

are three important steps for determining the time complexity requisite for the Algorithm. The first is to construct a communication cost matrix, which requires $O(m^2)$. The second is to execute the Hungarian method, which can be computed in $O(m^3)$. The final is to construct a set of k node-disjoint paths, which requires $O(m^4)$. Therefore, we can create an $O(m^4)$ parallel routing algorithm for constructing a set of k node-disjoint and nearly shortest paths. For further researches, we will extend this algorithm to design a set of one-to-many node-disjoint paths on arbitrary recursive circulant networks and on other networks. Also we will look into the construction of a set of many-to-many node-disjoint paths on $G(2^m,4)$.

REFERENCES

[1] J. Park, H. Kim, and H. Lim, "Many-to-Many Disjoint Path Covers in the Presence of Faulty Elements," *IEEE Trans. Comput.,* Vol. 58, No 4, pp. 528-540, 2009.

[2] Q. Zhu, J. Xu, and M. Xu, "Reliability of the Folded Hypercubes," *Info. Sci.,* Vol. 177, No. 5, pp. 1782- 1788, 2007.

[3] J. Park, "Panconnectivity and Edge-Pancyclicity of Faulty Recursive Circulant $G(2^m, 4)$," *Theoret. Comput. Sci.,* Vol. 390, No. 1, pp. 70-80, 2008.

[4] S. Tang, Y. Wang, and C. Yi, "Genneralized recursive circulant graphs," *IEEE Trans. Parall. Distr. Syst.,* Vol. 23, No. 1, pp. 87-93, 2012.

[5] I. Chung, "Application of the Special Latin Squares to the Parallel Routing Algorithm on Hypercube," *J. of Korea Info. Sci. Soc.,* Vol. 19, No. 5, pp. 569-578, 1992.

[6] Y. Shih and S. Kao, "One-to-one Disjoint Path Covers on K-ary N-cubes," *Theoret. Comput. Sci.,* Vol. 412, No. 1, pp. 4513-4530, 2011.

[7] K. Day and A. Tripathi, "A Comparative Study of Topological Properties of Hypercubes and Star Graphs," *IEEE Trans. Parall. Distr. Syst.,* Vol. 5, No. 1, pp. 31-38, 1994.

[8] S. Gao, B. Novick, and K. Qiu, "From Hall' Matching Theorem to Optimal Routing on Hypercubes," *J. Combinatorial Theory,* Series B 74, pp. 291-301, 1998.

[9] C. Lai, "Two Conditions for Reducing the Maximal Length of Node-Disjoint Paths in Hypercubes," *Theoret. Comput. Sci.,* Vol. 418, No. 1, pp. 82-91, 2012.

[10] C. Chen and J. Chen, "Nearly Optimal One-to-Many Parallel Routing in Star Networks," *IEEE Trans. Parall. Distr. Syst.,* Vol. 8, No. 12, pp. 1196-1202, 1997.

[11] Q. Gu and S. Peng, "Cluster Fault-Tolerant Routing in Star Graph," *Networks,* Vol. 35, No. 1, pp. 83-90, 2000.

[12] S. Madhavapeddy and I. Sudborough, "A Topogical Property of Hypercubes: Node Disjoint Paths," *IEEE Symp. Parall. Distr. Process.,* pp. 532- 539, 1997.

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

[14] Y. Chen, S. Promparmote, and F. Maire, "MDSM: Microarray Database Schema Matching using the Hungarian Method," *Info. Sci.,* Vol. 176, No. 19, pp. 2771-2790, 2006.

[15] H. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Res. Logist. Quart.,* Vol. 2, No. 1-2, pp. 83-97, 1955.

[16] T. Jeon and C. Kim, "A Real-time Embedded Task Scheduler Considering Falut-tolerant," *J. of Korean Multimedia Society,* Vol. 12, No. 7, pp. 940-948, 2011.

[17] J. Park and K. Chwa, "Recursive Circulants and Their Embedding Among Hypercubes," *Theoret. Comput. Sci.,* Vol. 244, No. 1-2, pp. 35-62, 2000.

Ilyong Chung

received the B.E. degree from Hanyang University, Seoul, Korea, in 1983 and the M.S. and Ph.D. degrees in Computer Science from City University of New York, in 1987 and 1991, respectively. From 1991 to 1994, he was a senior technical staff of Electronic and Telecommunication Research Institute (ETRI), Dajeon, Korea. Since 1994, he has been a Professor in Department of Computer Science, Gwangju, Korea. His research interests are in computer networking, security systems and coding theory.