Vol. 36 No. 7 Transactions of Information Processing Society of Japan July 1995

Regular Paper

A Routing Algorithm in Faulty n-Rotator Graph and Its Performance Evaluation

PETER M. YAMAKAWA,† HIROYUKI EBARA †† and HIDEO NAKANO †

The star graph and the hypercube are receiving attention among researchers as attractive models for highly parallel distributed computing, because they have regular structure, and connect many processors with small diameter and degree. Recently, an interesting interconnection network called Rotator graph was proposed and presents some advantages over the star graph and the hypercube. Namely, Rotator graph has very small diameter and average distance, and simple routing algorithm. We present a fault tolerant routing algorithm for n-Rotator graph. This graph has only one shortest path between any two nodes which is not a good fault tolerant property, but it possesses many short paths. From this fact, we develop an algorithm that looks for the shortest path or a short path by exploiting the network properties of the n-Rotator graph. We show the mathematical expression for the probability of finding the shortest path or the second shortest path (of length equal to the shortest path +1) in the presence of faulty components (links or nodes). The results show that the algorithm finds a very short path with high probability. They enhance the rich topological properties of the n-Rotator graph.

1. Introduction

The hypercube and the star graph due their regular structure, small diameter and degree are attractive models for use in highly parallel communication. Only a few years ago, Corbett[4] proposed a good alternative to these models called Rotator graph. This is a set of directed permutation graphs that has small diameter and average distance, and a simple routing algorithm.

Since the effective execution of parallel tasks depends on the reliable communication among processors, the study of fault tolerant routing algorithms is very popular in the field of parallel computation. In fault tolerant routing algorithms, we can consider two approaches:

The first one is that every node has global information of the system and the algorithm routes the message according to this information. If the shortest path exists, the algorithm will find this path. But, we have the necessity of broadcasting the condition of every component, which is expensive specially for large networks. The second one is that every node keeps only limited information of the system (for example, the condition of the adjacent links and nodes), and makes its routing decision. Since every node does not possess a total view of the system, the algorithm might not lead the message via the feasible shortest path. In spite of this disadvantage, this approach seems to be more practical.

We must design the routing algorithm by exploiting the network properties for the improvement of its efficiency. Fault tolerant routing algorithms have been proposed on many topologies as hypercube[5], star graph[12], arrangement graph[17], and others.

Chen and Shin[5] presented a routing algorithm based on depth first search technique for hypercube. Sur and Srimani[12),13)], with similar idea, proposed an algorithm for star graph. Furthermore, both of them show the performance of their algorithms for the probability of routing the message via the shortest path when the source and destination are at maximum distance.

The n-Rotator graph has better average distance and smaller diameter $n-1$ for a graph with $n!$ nodes than either star graph or hypercube. Since the n-Rotator graph has excellent properties for communications, our interest is to investigate the problem of routing the message in the presence of faults for this topology. We propose a simple algorithm that guides the message via a short path in the presence of faulty components (links or nodes). We assume that each node knows the condition of its adjacent components (distributed manner).

It is worth mentioning that n-Rotator graph presents a unique shortest path. But, it also

† Faculty of Engineering, Osaka University
†† Faculty of Engineering, Kansai University

possesses many short paths (For example, a path of length equal to the length of the shortest path +1). Our algorithm intends to look for any of these short paths by considering the network properties of the n-Rotator graph. We also use a memory which keeps the information of the route traversed by our algorithm (the visited nodes and visited links) in order to avoid loops.

It is shown that our algorithm finds the shortest path or the second shortest path (of length equal to the length of the shortest path +1) with very high probability in the presence of faulty components.

The paper is organized as follows. We introduce some useful definitions and notations in Section 2. Our fault tolerant routing algorithm is shown in Section 3. The performance analysis of the algorithm is given in Section 4. We conclude in Section 5.

2. Preliminaries

We show basic definitions for n-Rotator graph and its routing algorithm for the fault-free situation.

2.1 Basic Definitions

Definition 1 *Let $a_1 \cdots a_n$ be a permutation of n symbols. For $1<j\leq n$, we define a rotator operation as $Rot_j(a_1a_2\cdots a_n)=a_2a_3\cdots a_ja_1a_{j+1}\cdots a_n$.*

For example, $Rot_2(123)=213$ and $Rot_3(4321)=3241$.

Definition 2 *The n-Rotator graph is an directed graph $G(V, E)$ given by:*

$V=\{a_1a_2\cdots a_n | a_1a_2\cdots a_n$ is a permutation of n symbols$\}$,

$E=\{(x, y)|x, y\in V$ and $y=Rot_j(x)$ for $j, 1<j\leq n\}$,

For example, the node 1234 is connected to the nodes 2134, 2314, and 2341 for the 4-Rotator graph in Fig. 2. The 3-Rotator graph and the

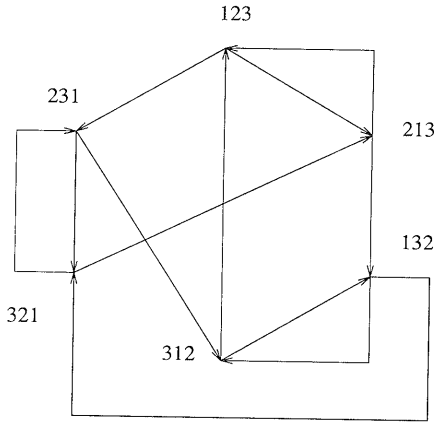


Fig. 1 3-Rotator graph.

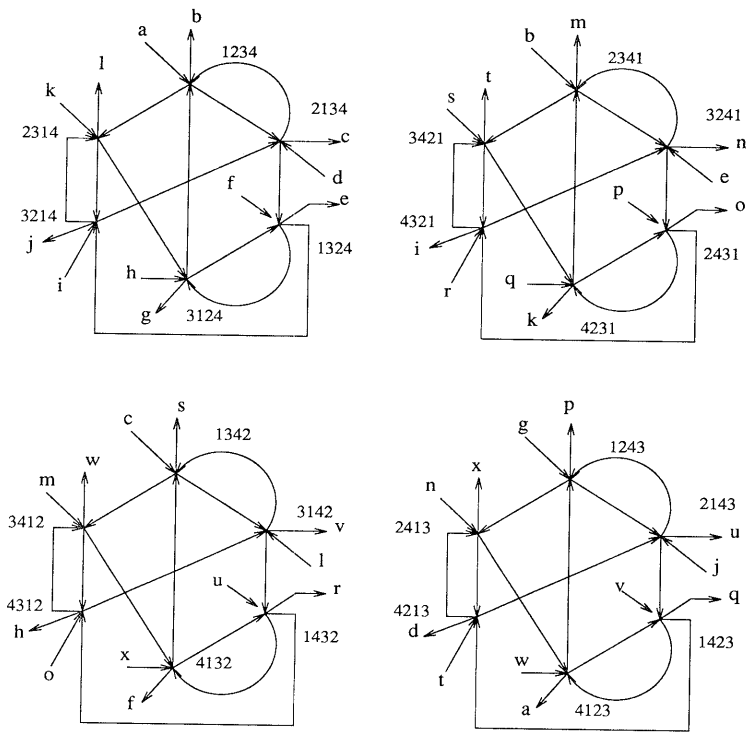


Fig. 2 4-Rotator graph.

4-Rotator graph are shown in **Fig. 1** and **Fig. 2** respectively.

2.2 Basic Properties

The n-Rotator graph is a directed graph with $n!$ nodes. The in-degree and the out-degree is $n-1$ respectively, and the diameter is $n-1$. This graph is node symmetric. It permits us to consider the distance between two arbitrary nodes as the distance between the source node and the identity node $I=123\cdots n$, by renaming the nodes. For example, let 43215 be the source and 32145 be the destination. We can map the destination node to the identity node by making the following changes as $3\mapsto 1, 2\mapsto 2, 1\mapsto 3, 4\mapsto 4, 5\mapsto 5$, and the source and the destination can be considered as 41235 and 12345 respectively. Then, the path between the original source and the destination node becomes isomorphic to the path between the node 41235 and the identity node.

2.3 Routing Algorithm

The routing algorithm in fault free mode was proposed by Corbett[4], and is based on the insertion sort of symbols. Since Rotator graph is node symmetric, we consider the destination as the identity node I.

Definition 3 *Let $s=a_1a_2\cdots a_ib_{i+1}\cdots b_n$ be an arbitrary node in n-Rotator graph, where $a_i>b_{i+1}<b_{i+2}<\cdots<b_n$. The node is divided into two regions of symbols : A leading unsorted region $a_1a_2\cdots a_i$ and a trailing sorted region $b_{i+1}b_{i+2}\cdots b_n$.*

The trailing sorted region is a sequence of symbols arranged in increasing order, and the leading unsorted region is all other symbols. For example, consider the node 342156. The leading unsorted region is 342 and the trailing sorted region is 156.

Consider n-Rotator graph with n symbols. Let $s=a_1a_2\cdots a_ib_{i+1}\cdots b_n$ be the source, and the destination be the identity permutation $12\cdots n$. The routing algorithm in fault free mode[4] is given by :

1. Move a_1 into its correct position l so that $b_l<a_1<b_{l+1}$, or $l=i$ and $a_1<b_{i+1}$, until s becomes theidentity node.

For example, let 4321 be the source and 1234 be the destination. According to the routing algorithm, we make the following moves :

$4321\mapsto 3214\mapsto 2134\mapsto 1234$.

The following propositions are taken from Ref. 4).

Proposition 1 *The routing algorithm gives a shortest path between any two nodes in the n-Rotator graph.*

Table 1 Comparison of Star graph, Hypercube, and Rotator graphs.

	Nodes	Degree	Diameter	Average Distance
5-Star	120	4	6	3.7
7-Hypercube	128	7	7	3.5
5-Rotator	120	4*	4	3.28
7-Star	5040	6	9	5.9
12-Hypercube	4096	12	12	6
7-Rotator	5040	6*	6	5.28
9-Star	362,880	8	12	8.1
18-Hypercube	262,144	18	18	9
9-Rotator	362,880	8*	8	7.28

* in-degree and out-degree respectively.
[Note] The data are taken from Ref. 3), 4).

Proposition 2 *The n-Rotator graph has a unique shortest path between any two nodes.*

Proposition 3 *The diameter of n-Rotator graph is $n-1$.*

In **Table 1**, we compare the n-Rotator graph with the hypercube and the star graph. For a given size, the table shows that the n-Rotator graph has smaller diameter and average distance than the hypercube and the star graph.

3. The Fault Tolerant Routing Algorithm

In this section, we present an adaptive routing algorithm in the presence of faulty components for the n-Rotator graph. We consider that every node knows only the condition of its adjacent links.

Our work has been realized based on the following observations :

- Since the n-Rotator graph is node symmetric, we can always consider the destination as the identity node.
- The visited nodes and the visited links must be recorded into a memory to avoid loops or to detect the termination of the algorithm.
- Each node is divided in two regions : the *leading unsorted region* and *the trailing sorted region.*
- The routing algorithm in fault free mode is based on insertion sort.
- The sorted region and the unsorted region are the dominant factors for the efficiency of the routing algorithm. Thus, we can exploit fully the properties of the network by working with these two regions efficiently.

Proposition 4 *Let $s=a_1a_2\cdots a_ib_{i+1}b_{i+2}\cdots b_n$ be*

the source node and the identity node be the destination, where $a_i > b_{i+1} < b_{i+2} < \cdots < b_n$. *If we move* a_1 *into its correct position* l, *where* $b_l < a_1 < b_{l+1}$, *or* $l = i$ *and* $a_1 < b_{i+1}$; *the length of the trailing sorted region will be lengthened, and the distance to the destination will be shortened.*

For example, consider the node 1423. If we move 1 into its correct position, we have the node 4123 and the distance to the destination will be shortened.

Proposition 5 *Let* $s = a_1 a_2 \cdots a_i b_{i+1} \cdots b_n$ *be the source node and the identity node be the destination, where* $a_i > b_{i+1} < b_{i+2} < \cdots < b_n$. *If we move* a_i *into the jth position, where* $j < i$, *or* $j = i$ *and* $a_1 > b_{i+1}$; *the length of the trailing sorted region and the distance to the destination will be kept the same.*

For example, consider the node 4213. If we move 4 into the second position, we have the node 2413 and the distance to the destination will be kept the same.

Proposition 6 *Let* $s = a_1 a_2 \cdots a_i b_{i+1} \cdots b_n$ *be the source node and the identity node be the destination, where* $a_i > b_{i+1} < b_{i+2} < \cdots < b_n$. *If we move* a_1 *into the jth position and the lth position is its correct position, where* $j < l$ *and* $b_l < a_1 < b_{l+1}$, *the length of the sorted region will be shortened to* $n - j$ *and the distance to the destination will be increased to* j.

For example, consider the node 621345. If we insert 6 into the third position, we have the node 216345. The length of the sorted region will be shortened to $(6-3)$, and the distance to the destination will be increased to 3.

Proposition 7 *Let* $s = a_1 a_2 \cdots a_i b_{i+1} \cdots b_n$ *be the source node and the identity node be the destination, where* $a_i > b_{i+1} < b_{i+2} < \cdots < b_n$. *If we move* a_1 *into the jth position and the lth position is its correct position, where* $j > l$ *and* $b_l < a_1 < b_{l+1}$, *the length of the sorted region will be shortened to* $n - j + 1$ *and the distance to the destination will be increased to* $j - 1$.

For example, consider the node 42135678. If we insert 4 into the fifth position, we have the node 21354678. The length of the sorted region will be shortened to $(8-5+1)$, and the distance to the destination will be increased to 4.

Definition 4 *The routing information RI stores the visited nodes and the visited links by keeping the number* j *in rotator operation* Rot_j.

For example, let 3421 and 1234 be the source and destination respectively. We make the following rotator operations:

$3421 \mapsto 4213$, $4213 \mapsto 2134$, $2134 \mapsto 1234$,

and the $RI = \{4, 4, 2\}$.

Definition 5 *The inverse rotator operation is defined as* $Rot_i^{Inv}(a_1 \cdots a_i \cdots a_n) = a_i a_1 \cdots a_n$.

$Rot_2^{Inv}(1234) = 2134$ *and* $Rot_3^{Inv}(2134) = 3214$ *as for example. We can apply the inverse rotator operations to know the visited nodes and the visited links. The routing information RI and the inverse rotator operation allow us to avoid loops and to terminate the algorithm.*

The fault tolerant routing algorithm

We present our routing algorithm where $s = a_1 a_2 \cdots a_i b_{i+1} \cdots b_l \cdots b_n$, $a_1 a_2 \cdots a_i$ is the leading unsorted region, $b_{i+1} \cdots b_n$ is the trailing sorted region, the *lth* position is the correct position of a_1, and m is the message.

```
/* Reach the destination */
if s is the identity node then
  stop;
/* Intend to send the message to a node so that the distance to the
destination will be shortened */
if the link from s to node Rot_l(s) is not faulty and
the link to this node has not visited before by checking RI then
  send(RI, m) to that node;
  stop;
/* Intend to send the message to a node so that the distance to the destination
will be kept the same or will be lengthened as small as it is possible, and j ≠ l */
for j:=2 to n do
   if the link from s to a node Rot_j(s) is not faulty and
   the link to this node has not visited before by checking RI then
      send(RI, m) to that node
      stop;
writeln ('the message can not be routed')
stop;
```

Consider 4321 and 1234 be the source and the destination respectively. The links between the nodes 4321 and the node 3214, and the link between the node 3421 and the node 4213 are faulty. Our algorithm leads the message to the destination traversing the following route:

Example

$$4321 \mapsto \underset{\boxed{3214}}{3421} \mapsto \underset{\boxed{4213}}{4231} \mapsto 2314 \mapsto 3124 \mapsto 1234$$

For example, we can observe that the node 4321 intends to select the link connected to the node 3214, which is faulty. Then, the algorithm selects the link connected to 3421. Also, the node 3421 intends to select the link connected to the node 4213, which is faulty. Then, it sends to the node 4231.

The execution of the algorithm is as follows:

Table 2 Numerical values of probability for the shortest path or the second shortest path in n-Rotator graph with f faulty links and P equal to the distance traversed by the algorithm.

n	$f=1$	$f=2$	$f=3$	$f=4$	$f=5$	$P=n-1$ or n
4	0.986111	0.970657	0.953756	0.935522	0.916070	3 or 4
5	0.997917	0.995772	0.993568	0.991305	0.988984	4 or 5
6	0.998611	0.999443	0.999162	0.998879	0.998594	5 or 6
7	0.999967	0.999934	0.999901	0.999868	0.999834	6 or 7
10	1.000000	1.000000	1.000000	1.000000	1.000000	9 or 10

Table 2a Numerical values of probability for the shortest path in n-Star graph with f faulty links and with P equal to the distance traversed by the algorithm.

n	$f=1$	$f=2$	$f=3$	$f=4$	$f=5$	$P=[3/2(n-1)]$
4	0.944444	0.887302	0.829132	0.770478	0.711861	4
5	0.991667	0.983229	0.974691	0.966057	0.948518	6
6	0.998889	0.997776	0.996661	0.995544	0.994426	7
7	0.999868	0.999735	0.999603	0.999471	0.999338	9
10	1.000000	1.000000	1.000000	1.000000	1.000000	13

Table 2b Numerical values of probability for the shortest path in hypercube with f faulty links, and with P equal to the distance traversed by the algorithm.

n	$f=1$	$f=2$	$f=3$	$f=4$	$f=5$	$P=n$
3	0.916667	0.818182	0.704545	0.577778	0.443182	3
4	0.968750	0.935484	0.900202	0.862903	0.823599	4
5	0.987500	0.974684	0.961551	0.948101	0.934335	5
6	0.994792	0.989529	0.984211	0.978839	0.973413	6

[Note] The data of Tables 2a and 2b are taken from Ref. 12) and Ref. 5) respectively.

Table 3 Numerical values of probability for the shortest path or the second shortest path in n-Rotator graph with g faulty nodes.

n	$g=14$	$g=2$	$g=3$	$g=4$	$g=5$
4	0.954545	0.900433	0.838961	0.771765	0.699780
5	0.991525	0.982471	0.972864	0.962729	0.952093
6	0.998607	0.997187	0.995740	0.994267	0.992767
7	0.999802	0.999602	0.999402	0.999201	0.998999
10	1.000000	0.999999	0.999999	0.999999	0.999999

First, it looks for the unique shortest path by inserting the first symbol into its correct position (by Proposition 4). If the path is blocked-by a faulty component, it intends to send the message to a non-faulty node which is at the same distance to the destination as the previous sender (by Proposition 5). If all these paths are blocked, the algorithm intends to send to a non-faulty node, which is the available closest node to the destination (by Proposition 6 and by Proposition 7). The order of the execution of the algorithm is very important. Note that we can achieve these selections if the algorithm is begun by inserting the first symbol from the left position to the right position. Hence, the distance to the destination will be kept the same or will be lengthened as small as possible. If all these paths are blocked, the algorithm terminates. The information is represented by (RI, m). We use the routing information RI by keeping the rotator operation.

It is worth mentioning that rich topologies like hypercube and star graph have many optimal paths (of length equal to the shortest path), and their routing algorithms intend to look for these paths. The n-Rotator graph has only a unique shortest path between any pair of nodes, but also possesses many very short paths. We are aware from this fact and develop a routing algorithm that looks for these short paths.

4. Performance Analysis of Our Routing Algorithm

We show the mathematical expression for the probability of routing the message via the shortest path or the second shortest path for n-Rotator graph in the presence of faulty components (links or nodes). Our results are compared with those of previous works on hypercube[5] and star graph[12]. We make the following observations and assumptions:

- The n-Rotator graph has $n!$ nodes and $L=n!(n-1)$ links.
- The occurrence of faults are assumed to be equally.
- Let f be the number of faulty links in n-Rotator graph. The number of the possible configurations of f faulty links is C_f^L.
- For a given source node, not all the f faulty links may not affect performance of the routing algorithm to compute the short path or the second shortest path (the faults that are located on any possible shortest path or second shortest path will affect the performance of the algorithm). We denote k as this relevant number of faults.
- Since the number of possible routes for the second shortest paths depends on the representation of the source node, we consider the source as $s=n(n-1)\cdots 1$ and the destination as the identity node. So, these nodes are at maximum distance.
- The algorithm must work effectively with the sorted region and the unsorted region in order to exploit fully the network properties.

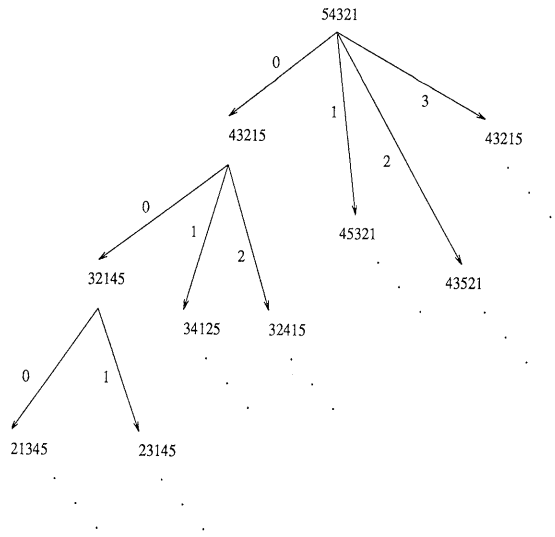


Fig. 3 Routing from 54321 with faulty components.

Let us consider the example in **Fig. 3**. It will illustrate us to analyze the performance of the algorithm. Let 54321 be the source and the identity node be the destination. The label of the edges means the number of faults that is encountered by the algorithm while routing to the destination. According to our algorithm, the source will intend to send the message to 43215, if there is one fault to 45321, if there are two faults to 43521, and so on. The idea is that the algorithm will try to send the message to a node in an specific order according to the number of faults (by inserting the first symbol from the left position to the right position of the given source). This procedure is followed for all nodes until a leaf node is reached. A leaf node is reached if it is the destination or a node where no more faults can be tolerated to generate the second shortest path.

Lemma 1 *If the algorithm starts at distance $n-1$ to the destination and there are f faulty links, the number of fault configurations so that the algorithm guides to the shortest path is given by C_f^{L-n+1}.*

Proof: Since the message has reached the destination by traversing the shortest path, the number of links used by the algorithm is $n-1$. The n-Rotator graph has L links, and the rest of links are $L-n+1$. Hence, the f faulty links are distributed among $L-n+1$ links in C_f^{L-n+1}. □

Theorem 1 *If the algorithm starts at distance $n-1$ to the destination and there are f faulty links, the probability of finding the shortest path is given by* $\dfrac{C_f^{L-n+1}}{C_f^L}$.

Proof: Immediately from Lemma 1. □

Lemma 2 *The algorithm guides the message for the second shortest path, if the length of the sorted region is kept the same during only one movement and is lengthened during all other movements.*

Proof: If the algorithm routes the message via the second shortest path, it means that the length of the sorted region is lengthened one time during its execution. On the other hand, the length of the sorted region can not be shortened during any movement because the algorithm leads to a path which is longer than the second shortest path. □

Lemma 3 *Consider the source node which is at distance $n-1$ to the destination. There are $n-2$ possible outgoing links connected to this node so that the algorithm can lead the message for the second shortest path.*

Proof: According to the routing algorithm, it intends to insert the first symbol into the available leftmost position due to the presence of faulty links. We have assumed that the source is at distance $n-1$ to the destination, so the length of the unsorted region is $n-1$ and the algorithm can select $n-2$ positions according to the number of faulty links adjacent to the given source. Other positions lead to the case for the shortest path or the path which is longer than the second shortest path. □

Lemma 4 *Consider the source node which is at distance $n-1$ to the destination. There are $n-2$ possible configurations for the second shortest path starting from this node.*

Proof: Immediately from Lemma 3. □

For example, in Fig. 3, the node 54321 is at distance 4 and there are 3 possible configurations for the second shortest path starting from this node.

From Lemma 4, for a given source node, we can notice that the number of possible configurations for the second shortest path is decreased as long as the distance between the given source and the destination is decreased. This relation can be observed in **Fig. 4**.

For the analysis of the probability of routing the message via the second shortest path, we need to know the total number of configurations for the second shortest path. From Lemma 4 and the description given in Fig. 4, we can see all these possible configurations for the

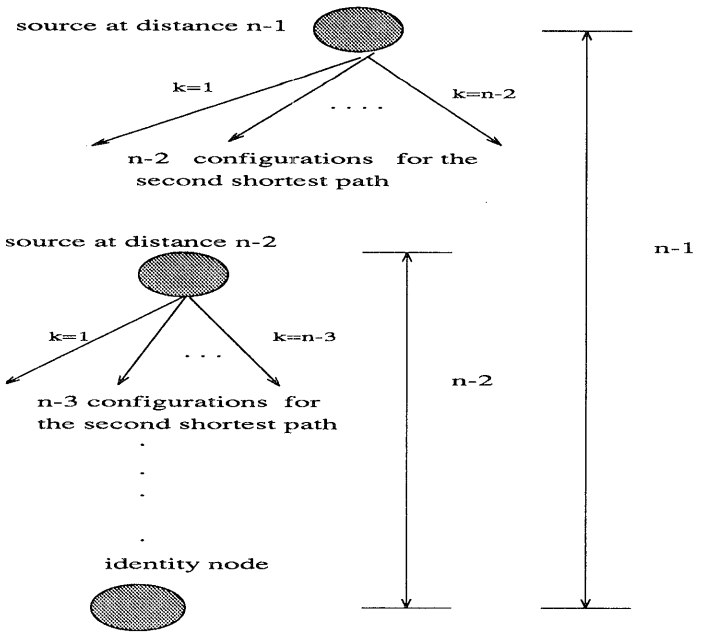


Fig. 4 Configurations for the second shortest path.

Table 4 Configurations for the second shortest path.

distance/faults	$k=1$	$k=2$	$\cdots$	$k=n-3$	$k=n-2$
$n-1$	o	o	$\cdots$	o	o
$n-2$	o	o	$\cdots$	o	
$n-3$	o	o	$\cdots$		
•	o	o	$\cdots$		
•	o	o	$\cdots$		
	o	o	$\cdots$		
2	o		$\cdots$		

second shortest path as the **Table 4**. The presence of the symbol "o" means that there are one possible configuration with k relevant faults.

Lemma 5 *If the algorithm starts at distance $n-1$ to the identity permutation, the number of possible configurations for the second shortest path, which is defined by the function $\alpha(k)$, is given by*

$$\alpha(k)=\begin{cases}(n-1)-k & \text{if } 0<k<n-1\\ 0 & \text{otherwise}\end{cases}$$

where k is the number of relevant faulty links encountered by the algorithm.

Proof: From the Table 4, we can know that there are $n-2$ configurations with one fault, there are $n-3$ routes with two faults, and so on. Thus, the number of configurations is reduced as long as the number of faults k is increased. Hence, for a given source at distance $n-1$ to the destination, the number of configurations with k faults is $n-1-k$. □

Lemma 6 *If the algorithm starts at distance $n-1$ to the destination, the total number of configurations so that the algorithm can lead to the second shortest path is given by*

$$\sum_{k=1}^{n-2}\alpha(k)$$

where k is the number of relevant faulty links encountered by the algorithm.

Proof: Immediately from Lemma 5. Since $1\leq k<n-2$, we have to consider all the possible configurations for the second shortest path for any k relevant faults. □

Lemma 7 *If the algorithm starts at distance $n-1$ to the destination and there are f faulty links. The total number of configurations so that the algorithm leads to the second shortest path is given by*

$$\sum_{k=1}^{min\{f,n-2\}}\alpha(k)C_{f-k}^{L-n-k},$$

Proof: From Lemma 6, we get the number of possible configurations for the second shortest path when the algorithm has been encountered by k faulty links. The number of links traversed by the algorithm are n, and k links are encountered faulty. Hence, there are $L-n-k$ available links. The rest $f-k$ faulty links are distributed uniformly among all possible links. And, we can say that there are C_{f-k}^{L-n-k} configurations of these faulty links. So, we have a total of $\sum_{k=1}^{min\{f,n-2\}}\alpha(k)C_{f-k}^{L-n-k}$ different configurations of faulty links. □

Theorem 2 *If the algorithm starts at distance*

$n-1$ to the destination and there are f faulty links, the probability of finding the second shortest path is given by

$$\frac{\sum_{k=1}^{min\{f,n-2\}}\alpha(k)C_{f-k}^{L-n-k}}{C_f^L}.$$

Proof: Immediately from Lemma 7. □

Theorem 3 *If the algorithm starts at distance $n-1$ to the destination, and there are f faulty links. The probability of finding the shortest path or the second shortest path is given by*

$$\frac{\sum_{k=1}^{min\{f,n-2\}}\alpha(k)C_{f-k}^{L-n-k}+C_f^{L-n+1}}{C_f^L}.$$

Proof: It is easily proved from Theorem 2 and Theorem 1. □

Numerical examples for our algorithm to compute the shortest path or the second shortest path in the presence of faulty links is given in table 2. We can observe that our algorithm guides the message to the destination with very high probability. Moreover, if we consider that the shortest path is $n-1$ for a given size of n-Rotator graph, it shows us that our algorithm also traverse a very short distance. The results show that our algorithm leads the message to the destination via very short distance with very high probability, and presents some advantages in its performance over the algorithms proposed by Sur and Surmani and by Chen and Shin. Namely, we can see that the proposed algorithm routes the message via shorter distance (P) than that of the algorithm on star graph. Also, we can see that the probability of our algorithm guides message via short paths is greater than hypercube. All these results enhance the rich topological properties of the n-Rotator graph.

Now, we state the problem of routing the message via g faulty nodes and present the following theorems:

Lemma 8 *If the algorithm starts at distance $n-1$ to the destination, and there are g faulty nodes. The number of configurations of faults so that the algorithm leads to the second shortest path is given by*

$$\sum_{k=1}^{min\{g,n-2\}}\alpha(k)C_{g-k}^{n!-(1+n)-k}$$

where k is the number of relevant faulty nodes encountered by the algorithm.

Proof: If the algorithm is encountered by k faulty nodes, the number of routes for the second shortest path is $\sum_{k=1}^{g,n-2}\alpha(k)$ from the source (from similar concept to the proof of Lemma 7). Since the algorithm leads to the second shortest path, the number of nodes visited by the algorithm is $n+1$. And, k nodes are encountered faulty by the algorithm. The rest $g-k$ are distributed uniformly among the remaining nodes $n!-(n+1)-k$. Thus, we can say that there are $C_{g-k}^{n!-n-1-k}$ configurations of faulty nodes. □

Theorem 4 *If the algorithm starts at distance $n-1$ to the destination, and there are g faulty nodes. The probability of finding the shortest path or the second shortest path is given by*

$$\frac{\sum_{k=1}^{min\{g,n-2\}}\alpha(k)C_{g-k}^{n!-1-n-k}+C_g^{n!-1-n}}{C_g^{n!-2}}.$$

Proof: We assume that the source and the destination are correct nodes. It is easily proved from similar concept to the proof of Theorem 3. □

5. Conclusions

We think that n-Rotator graph is a good candidate for highly parallel and distributed communication systems, because this topology has many good properties as small diameter, simple routing algorithm, and small average distance.

Our algorithm routes the message by working with the sorted region and unsorted region of an arbitrary node. Using this approach, we could exploit the network properties and face the problem of the presence of faulty components. We also notice that n-Rotator graph has only one shortest path, but also possesses many short path. This idea is the skeleton of the developing of our routing algorithm.

For the performance analysis of our algorithm, we consider that the source and the destination are at maximum distance. Our interest is to find the probability so that the algorithm can lead the message via the shortest path or the second shortest path. The reason is that we think that this analysis could give us a good parameter to evaluate the performance of our algorithm, and that we could discuss with other related works. It is shown that the message is routed to destination by traversing a very short distance with high probability in the presence of faulty components. These results also enhance the rich properties for communication of this topology.

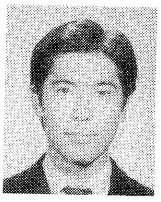
Finally, we say that the developing of routing algorithms is important in the field of parallel computation, and so is the selection of topology. We believe that an efficient fault tolerant routing algorithm in combination with good topology will be the key of success of parallel computation.

Acknowledgments Thanks to Prof. Hajime Maeda from the Department of Communication Engineering at Osaka University for his valuable support.

References

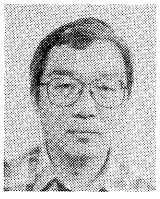
1) Akers, S. B. and Krishnamurthy, B.: A Group Theoretic Model for Symetric Interconnection Networks, *Proc. Int. Conf. Parallel Processing*, pp. 216-223 (1986).
2) Akers, S. B. and Krishnamurthy, B.: The Fault Tolerance of Star Graphs, *Proc. 2nd Int. Conf. Supercomputing*, pp. 31-42 (1987).
3) Akers, S. B., Harel, D. and Krishnamurthy, B.: The Star Graph: An Attractive Alternative to n-cube, *Proc. Int. Conf. Parallel Processing*, pp. 393-400 (1987).
4) Corbett, P. F.: Rotator Graphs: An Efficient Topology for Point-to-Point Multiprocessor Networks, *IEEE Trans. Par. and Dist. Sys.*, Vol. 3, No. 5, pp. 622-626 (1992).
5) Chen, M. S. and Shin, K. G.: Depth-first Search Approach for Fault Tolerant Routing in Hypercube Multicomputers, *IEEE Trans. Par. and Dist. Sys.*, Vol. 1, pp. 152-159 (Apr. 1990).
6) Chen, M. S. and Shin, K. G.: Adaptive Fault Tolerant Routing in Hypercube Multicomputers, *IEEE Trans. Comput.*, Vol. 39, pp. 1406-1416 (Dec. 1990).
7) Day, K. and Tripathi, A.: Characterization of Node Disjoint Paths in Arrangement Graphs, Tech. Rept., TR. 91-43, Dept. of Computer Science, University of Minessota (1991).
8) Day, and Tripathi, A.: Arrangement Graphs: A Class of Generalized Star Graphs, *Info. Process. Lett.*, No. 42, pp. 235-241 (1992).
9) Esfahanian, A. H. and Hakimi, S. L.: Fault Tolerant Routing in DeBrujin Communication Networks, *IEEE Trans. Comput.*, Vol. C-34, No. 9, pp. 777-788 (1985).
10) Gaughan, P. T. and Yalamanchili, S.: Adaptive Routing Protocols for Hypercube Interconnection Networks, *IEEE Comput.*, pp. 12-23 (May 1993).
11) Pradhan, D. K. and Reddy, S. M.: A Fault Tolerant Communication Architecture for Distributed Systems, *IEEE Trans. Comput.*, Vol. C-31, No. 9, pp. 863-870 (1982).
12) Sur, S. and Srimani, P. K.: A Fault Tolerant Routing Algorithm in Star Graph Interconnection Networks, *Proc. Int. Conf. Parallel Processing*, pp. 267-270 (1991).
13) Sur, S. and Srimani, P. K.: A DFS Strategy to Fault Tolerant Routing in Star Graphs and Its Performance Evaluation, *Tech. Report*, Department of Computer Science, Colorado State University (1991).
14) Wu, J. and Yao, K.: Multicasting in Injured Hypercubes Using Limited Global Information, Tech. Report, TR-CSE-93-26, Dept. of Computer Science and Engineering, Florida Atlantic Univ. (1993).
15) Yamakawa, P. M., Ebara, H. and Nakano, H.: A Fault Tolerant Routing Algorithm for Arrangement Graphs, Tech. Report of IECE, COMP 93-49 (Sep. 1993).
16) Yamakawa, P. M., Ebara, H. and Nakano, H.: A Routing Algorithm in a Faulty n-Rotator Graph, *Proc. Joint Symp. Parallel Processing 1994*, pp. 65-72, Tsukuba (May 1994).
17) Yamakawa, P. M., Ebara, H. and Nakano, H.: Fault Tolerant Routing for (n, k)-Arrangement Graph, *Proc. Int. Symp. Parallel Architectures, Algorithms, and Networks*, pp. 213-220, Kanazawa (Dec. 1994).

(Received September 21, 1994)
(Accepted February 10, 1995)



Peter M. Yamakawa received the BS degree in electronic engineering from the National University of Engineering in Peru, and the MS degree in communication engineering from Osaka University. He is currently a doctoral candidate at Osaka University. His research interests include parallel and distributed algorithms, interconnection networks, graph and combinatorial theory.

Hiroyuki Ebara was born in Osaka, on April 28th, 1958. He received the B. E., M. E., and DE. Eng. degrees in communication engineering from Osaka University, Osaka, Japan, in 1982, 1984, 1987, respectively. In 1987 he became Assistant Professor of Osaka University. Since 1994 he has been with Kansai University, where he is a lecturer. His main research interests are in computational geometry, combinatorial optimization, and parallel computing. Dr. Ebara is a member of IPSJ, IEEE, ACM, and IECE.



Hideo Nakano was born in Osaka, Japan, on January 1, 1948. He is now an associate professor at Osaka University. His field of interest are combinatorial optimization, security, and Internet. He is a member of IECE, IEEE, and ACM.