PAPER

# An Efficient Adaptive Routing Algorithm for the Faulty Star Graph

**Leqiang BAI**[†], *Nonmember*, **Hiroyuki EBARA**[††], **Hideo NAKANO**[†††], *and* **Hajime MAEDA**[†], *Members*

**SUMMARY** This paper introduces an adaptive distributed routing algorithm for the faulty star graph. The algorithm is based on that the $n$-star graph has uniform node degree $n-1$ and is $n-1$-connected. By giving two routing rules based on the properties of nodes, an optimal routing function for the fault-free star graph is presented. For a given destination in the $n$-star graph, $n-1$ node-disjoint and edge-disjoint subgraphs, which are derived from $n-1$ adjacent edges of the destination, can be constructed by this routing function and the concept of Breadth First Search. When faults are encountered, according to that there are $n-1$ node-disjoint paths between two arbitrary nodes, the algorithm can route messages to the destination by finding a fault-free subgraphs based on the local failure information (the status of all its incident edges). As long as the number $f$ of faults (node faults and/or edge faults) is less than the degree $n-1$ of the $n$-star graph, the algorithm can adaptively find a path of length at most $d+4f$ to route messages successfully from a source to a destination, where $d$ is the distance between source and destination.

***key words:*** *star graph, node-disjoint subgraph, node-disjoint path, fault-tolerance, adaptive routing*

## 1. Introduction

With the advance in VLSI, a lot of research has been initiated to design large multiprocessors computing systems using special topologies. In fact, by using a non-trivial topology, we can interconnect many processors without increasing the cost. The most popular mode is the hypercube which has been drawn considerable attention from both academic and industrial communities. The star graph in [1] claims to possess topological superiority over the hypercube. Similar to the hypercube, the star graph possesses rich recursive structure, symmetrical properties and simple routing on the fault-free star graph. In addition, it has a lower diameter and degree, and a smaller average diameter for a given size than the hypercube.

There are two different algorithms for message routing: static and adaptive. Static routing algorithms use only a single path to route messages, whereas adaptive routing algorithms allows more freedom in selecting the paths to route messages. If there are faults on networks, adaptive routing algorithms are necessary because static routing algorithms cannot insure messages being routed successfully. But the flexibility of adaptive routing may cause deadlock and livelock problems. A deadlock occurs when a message waits for an event that will never happen. In contrast, a livelock keeps a message moving indefinitely without reaching the destination. If any node only knows the condition of its incident edges, a node fault will easily cause deadlock and livelock problems than an edge fault in communication because a node fault corresponds to more than one edge fault on the interconnection network.

The study of fault-tolerant routing algorithm is very popular in the field of parallel computation, because the effective execution of parallel tasks depends on the reliable communication among processors. The problems with different fault models for the hypercube have been studied in [7], [11], [15], [16]. Fault tolerance of the star graph has been discussed in [1], [9], [13]. The question of simulating a completely healthy $n$-star graph with a degraded one (one with some faulty nodes) has been discussed in [4]. Given a set of at most $n-2$ faulty nodes, node-to-node and set-to-set fault tolerant routing algorithms for the star graph have been presented in [7], [8]. Fault-tolerant routing algorithms for the star graph, based on the local failure information, have been developed subject to faulty edges in [3], [5]. The shortcoming of the algorithms listed above for the star graph is that all these algorithms are only directly subject to node faults or edge faults. Therefore, it is necessary to develop a routing algorithm which can directly tolerate node faults and/or edge faults for the star graph.

In this paper, we present an adaptive distributed routing algorithm that has no deadlock and livelock for the faulty star graph. The algorithm is based on that the $n$-star graph has uniform node degree $n-1$ and is $n-1$-connected. By giving two routing rules based on the properties of nodes, we present a routing function by which the shortest path between source and destination can be found for the fault-free star graph. For a given destination in the $n$-star graph, $n-1$ node-disjoint and edge-disjoint subgraphs, which are derived

Manuscript received April 30, 1997.
Manuscript revised November 25, 1997.

[†]The author is with the Faculty of Engineering, Osaka University, Suita-shi, 565-0871 Japan.
[††]The author is with the Faculty of Engineering, Kansai University, Suita-shi, 564-8680 Japan.
[†††]The author is with the Media Center, Osaka City University, Osaka-shi, 558-8585 Japan.

from $n-1$ adjacent edges of the destination, can be determined by using this routing function and the concept of Breadth First Search. When faults are encountered, according to that there are $n-1$ node-disjoint paths between two arbitrary nodes, the algorithm can always route messages to the destination by finding a fault-free subgraphs based on the local failure information and the properties of nodes. It is not necessary to judge the types of faults that are encountered. The algorithm can tolerate at most $n-2$ faults (node faults and/or edge faults) to route messages successfully for the faulty $n$-star graph.

The remainder of this paper is organized as follows. The star graph and notations throughout this paper are given in Sect. 2; The routing function for the fault-free star graph, and the fault-tolerant routing algorithm and performance comparisons with the other routing algorithms are described in Sect. 3; Conclusions are drawn in Sect. 4.

## 2. Preliminaries

Let $V$ denote the set of $n!$ permutations of symbols $\{1,2,...,n\}$. An $n$-star graph interconnection network on $n$ symbols, denoted by $S_n = (V,E)$, is an undirected graph with $n!$ nodes. The nodes of $S_n$ are in a 1-1 correspondence with the permutation $p = p_1p_2...p_n$ of $\langle n\rangle = \{1,2,...,n\}$. Two nodes of $S_n$ are connected by an edge if and only if the permutation of one node can be obtained from the other by interchanging the first symbol $p_1$ with the $i$th symbol $p_i$, $2 \leqq i \leqq n$. Obviously, every node has $n-1$ incident edges, corresponding with $n-1$ symbols which the symbol in the first position can be interchanged with. Thus, $S_n$ is a regular graph of degree $n-1$ and is $(n-1)$-connected. $S_n$ possesses a number of properties that are desired by interconnection networks. These include node and edge symmetry, maximal fault tolerance, and strong resilience. $S_n$ has a high recursive structure, and is made up of $n$ copies of $n-1$-star graph. Figure 1 is one 4-star graph. It can be partitioned into four 3-star graphs. $S_3(i)$ denotes a 3-star graph induced by all the nodes with the same last symbol $i$, $i \in \langle 4\rangle$, and $S_4 = E_3 \cup \{S_3(i) | 1 \leqq i \leqq 4\}$, where $E_3$ is the set of edges among four 3-star graphs.

A permutation can be viewed as a set of cycles, i.e., a cyclically ordered set of symbols with the property that each symbol's correct position is occupied by the next symbol in the set. Since the $n$-star graph is node symmetric, the routing between two arbitrary nodes reduces to the routing from an arbitrary node to the special node labeled with the identity permutation $I = 12...n$. To reach $I$ from a node $p$ in $S_n$, it suffices to use one of the following two rules [1] repeatedly, until $I$ is reached:

**R1.** If symbol 1 is in the first position, then exchange it with any symbol not in its position.

**R2.** If symbol $i$ $(i \neq 1)$ is in the first position, then move it to its correct position.

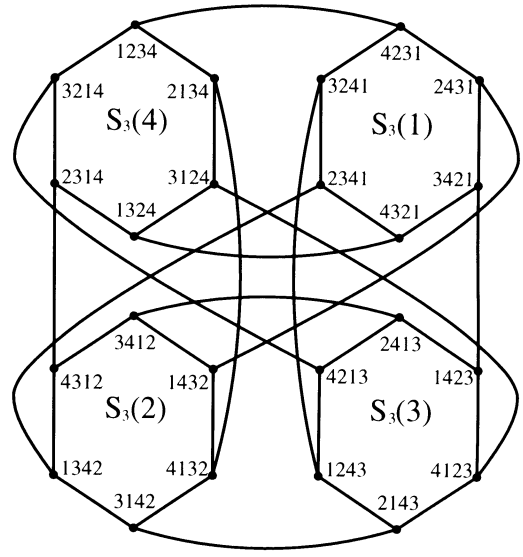



**Fig. 1** The 4-star graph viewed as four 3-star graphs.

Let *p-cycles* be the set of the cycles of length at least 2, the rules **R1** and **R2** insure a path of the minimum distance $d(p,I)$ from $p$ to the identity permutation $I$:

$$d(p,I) = c + m - \begin{cases} 0 & \text{if 1 is in the first position in } p, \\ 2 & \text{otherwise.} \end{cases}$$

where $c$ denotes the number of cycles in *p-cycles* and $m$ is the number of symbols in the cycles of *p-cycles*.

**Example 1:** Here, we will explain some properties of cycles and how to route messages by the rules **R1** and **R2**. Let $p = 13254$ be the source node, and let $I = 12345$ be the destination node in $S_5$. Compared with $I = 12345$, since $p = 13254$ has the symbol 3 in the correct position 2 of the symbol 2 and the symbol 2 in the correct position 3 of the symbol 3, the symbols 2 and 3 form the cycle $(23) : 2 \rightarrow 3 \rightarrow 2$. In the cycle (23), the next symbol of the symbol 2 is the symbol 3, the next symbol of the symbol 3 is the symbol 2. Similarly, the symbols 4 and 5 form the cycle $(45) : 4 \rightarrow 5 \rightarrow 4$. Therefore, *p-cycles* $= \{(23),(45)\}$, where $c = 2$ and $m = 4$. Since $p_1 = 1$, $d(13254,12345) = c + m = 6$. In the same way as described above, the node 23154 contains the cycle $(123) : 1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ and the cycle $(45) : 4 \rightarrow 5 \rightarrow 4$. Let $p_i \leftrightarrow p_j$ denote interchanging the symbol $p_i$ with the symbol $p_j$ of $p$. Applying the two rules described above repeatedly, messages can be sent at 6 steps from $p$ to $I$ through any one of 8 shortest paths as follows.

$$p=13254 \ (\mathbf{R1}) \begin{cases} \left.\begin{array}{l} \xrightarrow{1\leftrightarrow 2} 23154 \xrightarrow{2\leftrightarrow 3} 32154 \xrightarrow{3\leftrightarrow 1} \\ \xrightarrow{1\leftrightarrow 3} 31254 \xrightarrow{3\leftrightarrow 2} 21354 \xrightarrow{2\leftrightarrow 1} \end{array}\right\} 12354 \\ (\mathbf{R2}) \\ \left.\begin{array}{l} \xrightarrow{1\leftrightarrow 4} 43251 \xrightarrow{4\leftrightarrow 5} 53241 \xrightarrow{5\leftrightarrow 1} \\ \xrightarrow{1\leftrightarrow 5} 53214 \xrightarrow{5\leftrightarrow 4} 43215 \xrightarrow{4\leftrightarrow 1} \end{array}\right\} 13245 \end{cases}$$

$$12354 \left\{ \begin{array}{l} \stackrel{1\leftrightarrow 4}{\longrightarrow} 42351 \stackrel{4\leftrightarrow 5}{\longrightarrow} 52341 \stackrel{5\leftrightarrow 1}{\longrightarrow} \\ \stackrel{1\leftrightarrow 5}{\longrightarrow} 52314 \stackrel{5\leftrightarrow 4}{\longrightarrow} 42315 \stackrel{4\leftrightarrow 1}{\longrightarrow} \end{array} \right\} I = 12345.$$

$$13245 \left\{ \begin{array}{l} \stackrel{1\leftrightarrow 2}{\longrightarrow} 23145 \stackrel{2\leftrightarrow 3}{\longrightarrow} 32145 \stackrel{3\leftrightarrow 1}{\longrightarrow} \\ \stackrel{1\leftrightarrow 3}{\longrightarrow} 31245 \stackrel{3\leftrightarrow 2}{\longrightarrow} 21345 \stackrel{2\leftrightarrow 1}{\longrightarrow} \end{array} \right\}$$

(**R1**) (**R2**)

**Definition 1:** Let $p$ be a node in $S_n$, and let $\{i_1, i_2, ..., i_{k-1}, i_k\} \in \langle n \rangle$. Then, $p^{(i_1)}$ denotes the node that is obtained by interchanging the first symbol with the symbol $i_1$ of the node $p$, specially $p^{(p_1)} = p$. And, $p^{(i_1,i_2,...,i_{k-1},i_k)}$ denotes the node that is obtained by interchanging the first symbol with the symbol $i_k$ of the node $p^{(i_1,i_2,...,i_{k-1})}$.

**Example 2:** Let $p = 54231$, then $p^{(3)} = \underline{3}42\underline{5}1$ and $p^{(3,2,1)} = \underline{3}42\underline{5}1^{(2,1)} = \underline{2}4\underline{3}51^{(1)} = \underline{1}435\underline{2}$.

## 3. Routing Algorithm for $S_n$

In this section, based on that the $n$-star graph has uniform node degree $n-1$ and is $n-1$-connected, we present an adaptive distributed fault-tolerant routing algorithm for $S_n$ with at most $n-2$ node faults and/or edge faults. Then, we prove its correctness, and analyze its properties. We make the following assumptions:

**Assumptions.** A fault can be a node fault or an edge fault. If a node is faulty, all the edges incident to it are treated as faulty edges. Each edge is bidirectional. If an edge is faulty, both directions are faulty. The total number of faults is less than degree $n-1$ of $S_n$. Any node only knows the status of all its incident edges that is called the local failure information. Both source and destination are fault-free.

### 3.1 Routing Function for the Fault-Free $S_n$

If messages are routed based on **R1** and **R2**, it is very difficult for an arbitrary node $p$ to know through which adjacent node of $I$ messages are sent to $I$. Let $p$-*cycles* $= \{C_1, C_2\}$, where $C_1$ with $|C_1| \leq 1$ denotes the set of the cycle that contains the symbol 1 and $C_2$ denotes the set of the cycles that don't contain the symbol 1. Let $M_1$ denote the set of the symbols in the cycle of $C_1$, and let $M_2$ denote the set of the symbols in the cycles of $C_2$. To route messages from any node $p$ to $I$, through a given adjacent node of $I$, two routing rules **RD1** and **RD2** are proposed as follows:

**RD1.** If $C_2 \neq \emptyset$, then exchange $p_1$ with the minimum symbol in $M_2$.

**RD2.** If $C_2 = \emptyset$, then exchange $p_1$ with the symbol in position $p_1$ in $M_1$.

The rules **RD1** and **RD2** mean that the routing is performed based on the properties of the cycles $C_1$ and $C_2$ of the nodes. If $C_2 \neq \emptyset$, by using the rule **RD1**, the number of the cycles $C_2$ is reduced to $C_2 = \emptyset$ and $C_1 \neq \emptyset$. If $C_2 = \emptyset$ and $C_1 \neq \emptyset$, by using the rules **RD2**, the number of the symbols in $C_1$ is reduced to $C_1 = \emptyset$. Obviously, $p_1 = 1$ and $p^{(1)} = p$ if $C_1 = C_2 = \emptyset$.

Let *FFR* denote the routing function that returns a node based on the rules **RD1** and **RD2**.

**function** *FFR* ($p, I$: node): node;
**var** $i, p_1$: integer; $C_2$: the set of cycles;
**var** $M_2$: the set of the symbols in the cycles of $C_2$;
**begin**
  **if** $C_2 \neq \emptyset$ **then begin** $p_i := minimum(M_2)$;
    **return**($p^{(p_i)}$) **end** /* **RD1** */
  **else begin** $i := p_1$; **return**($p^{(p_i)}$) **end** /* **RD2** */
**end**

**Example 3:** Let $s = 42351$, $p = 23154$ and $q = 14523$ be three nodes, and let $I = 12345$ be the destination node in $S_5$. Then, the cycles of each node and the shortest path from each node to $I$, which is decided by *FFR*, can be described as follows.

**1.** $s = 42351$: Since the symbols 2 and 3 are in their correct positions and $4 \rightarrow 5 \rightarrow 1 \rightarrow 4$ forms the cycle (145), *s-cycles* $= \{C_1\}$, where $C_1 = (145)$ and $C_2 = \emptyset$.
The shortest path; $s \rightarrow s^{(5)} \rightarrow s^{(5,1)} = I$.

**2.** $p = 23154$: Since $2 \rightarrow 3 \rightarrow 1 \rightarrow 2$ forms the cycle (123) and $5 \rightarrow 4 \rightarrow 5$ forms the cycle (45), *p-cycles* $= \{C_1, C_2\}$, where $C_1 = (123)$ and $C_2 = (45)$.
The shortest path; $p \rightarrow p^{(4)} \rightarrow p^{(4,5)} \rightarrow p^{(4,5,2)} \rightarrow p^{(4,5,2,3)} \rightarrow p^{(4,5,2,3,1)} = I$.

**3.** $q = 14523$: Since $4 \rightarrow 2 \rightarrow 4$ forms the cycle (24) and $5 \rightarrow 3 \rightarrow 5$ forms the cycle (35), *q-cycles* $= \{C_2\}$, where $C_1 = \emptyset$ and $C_2 = \{(24), (35)\}$.
The shortest path; $q \rightarrow q^{(2)} \rightarrow q^{(2,3)} \rightarrow q^{(2,3,5)} \rightarrow q^{(2,3,5,2)} \rightarrow q^{(2,3,5,2,4)} \rightarrow q^{(2,3,5,2,4,1)} = I$.

Now, we show some properties of the function *FFR*, which are very important for us to develop our fault-tolerant routing algorithm in $S_n$ with less than $n-1$ faults.

**Lemma 1:** For the fault-free $S_n$, the path from any node $p$ with $p \neq I$ to $I$, which is decided using repeatedly the function *FFR*, is the shortest and unique.

**Proof:** Let $q = FFR(p, I)$, then $d(q, I) = d(p, I) - 1$ based on the properties of cycles. By induction, it is easy to prove that Lemma 1 is correct. □

**Example 4:** For any node $p$ with $p \neq I$ in the fault-free $S_n$, *p-cycles* $= \{C_1, C_2\}$ must belong to one of following three cycles: $\{C_1\}$, $\{C_1, C_2\}$ and $\{C_2\}$. Considering $S_4$ as shown in Fig. 1, and $I = 1234$, for given any node $p$ with $p \neq I$ in $S_4$, $d(q, I) = d(p, I) - 1$ is shown as follows.

**1.** For $C_1 \neq \emptyset$ and $C_2 = \emptyset$:

Let $p = 2314$, then $q = FFR(2314, 1234) = 3214$ and $d(3214, 1234) = d(2314, 1234) - 1 = 1$.

**2.** For $C_1 \neq \emptyset$ and $C_2 \neq \emptyset$:

Let $p = 2143$, then $q = FFR(2143, 1234) = 3142$ and $d(3142, 1234) = d(2143, 1234) - 1 = 3$.

**3.** For $C_1 = \emptyset$ and $C_2 \neq \emptyset$:

Let $p = 1342$, then $q = FFR(1342, 1234) = 2341$ and $d(2341, 1234) = d(1342, 1234) - 1 = 3$.

**Lemma 2:** For the fault-free $S_n$, the shortest path from any node $p$ with $p \neq I$ to $I$, which is decided using repeatedly the function *FFR*, passes through a given adjacent node of $I$, which can be decided only by the properties of *p-cycles*.

**Proof:** Let $C_2 = \{c_1, ..., c_j, ..., c_{k-1}, c_k\}$, then *p-cycles* $= \{C_1, c_1, ..., c_j, ..., c_{k-1}, c_k\}$. Let $p^{c_j}$ denote the minimum symbol in the cycle $c_j$. Without loss of generality, assume that $p^{c_1} < ... < p^{c_j} < ... < p^{c_{k-1}} < p^{c_k}$, then $minimum(M_2) = p^{c_1}$. When $C_1 = \emptyset$, using the function *FFR* repeatedly, the node $p^{(p^{c_1},...,p^{c_j},...,p^{c_{k-1}},p^{c_k})}$ is the $k$th node in the shortest path from $p$ to $I$. Let the symbol $p^{c_1}$ be in position $i$ of $p$, then the symbol 1 is in position $i$ of $p^{(p^{c_1})}$. Since there are one $C_1$ cycle and $k - j$ $C_2$ cycles in the nodes of the path from $p^{(p^{c_1})}$ to $p^{(p^{c_1},...,p^{c_j})}$, where $1 \leq j \leq k - 1$, the position of the symbol 1 in the nodes of the path from $p^{(p^{c_1})}$ to $p^{(p^{c_1},...,p^{c_j},...,p^{c_{k-1}},p^{c_k})}$ is not changed. Since there is only one $C_1$ cycle in the nodes from $p^{(p^{c_1},...,p^{c_j},...,p^{c_{k-1}},p^{c_k})}$ to $I$, there is only one shortest path from $p^{(p^{c_1},...,p^{c_j},...,p^{c_{k-1}},p^{c_k})}$ to $I$, which passes through the given adjacent node $I^{(i)}$ of $I$. Lemma 2 holds. □

Let *Dim* be a function that can get the position of the given symbol that is the symbol 1 of $M_1$ or the minimum symbol of $M_2$ for the node $p$.

**function** *Dim*($p, I$: node): integer;
**var** $C_1, C_2$: the set of cycles;
**var** $M_1, M_2$: the set of the symbols in $C_1$ and $C_2$;
**begin**
  **if** $C_1 = \emptyset$ and $C_2 \neq \emptyset$ **then**
    **return**(the position of the symbol $min(M_2)$);
  **if** $C_1 \neq \emptyset$ **then**
    **return**(the position of the symbol 1 in $M_1$)
**end**

**Lemma 3:** For any node $p$ in the fault-free $S_n$, the shortest path from $p$ with $p \neq I$ to $I$, which is decided using repeatedly the function *FFR*, passes through the node $I^{(i)}$, where $i = Dim(p, I)$.

**Proof:** As shown in the proof of Lemma 2. □

**Example 5:** Let $s = 42351$, $p = 23154$ and $q = 14523$ be three nodes, and let $I = 12345$ be the destination node in $S_5$. Referred to Example 3, the properties of the cycles of the nodes $s$, $p$ and $q$ can be known. By the function *Dim*, we can get the position $i$ of the given symbol that corresponds to the node $I^{(i)}$.

**1.** $s = 42351$: $i = Dim(s, I) = 5$ that is the position of the symbol 1 since $C_1 \neq \emptyset$.

**2.** $p = 23154$: $i = Dim(p, I) = 3$ that is the position of the symbol 1 since $C_1 \neq \emptyset$.

**3.** $q = 14523$: $i = Dim(q, I) = 4$ that is the position of the symbol 2 since $C_1 = \emptyset$ and $C_2 \neq \emptyset$.

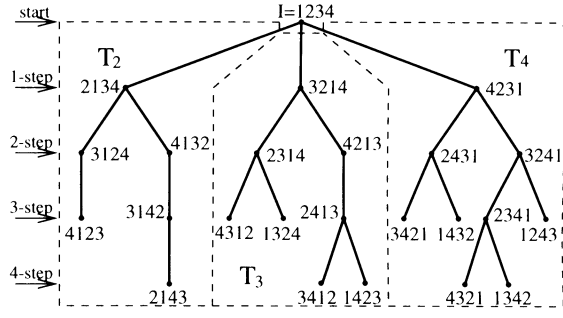



**Fig. 2** The breadth-first spanning tree $T$ from $I = 1234$ in $S_4$.

**Definition 2:** Let $I$ be a node on $S_n$, a spanning tree derived from the node $I$, which is constituted by Breadth First Search, is called a breadth-first spanning tree derived from $I$ on $S_n$.

**Lemma 4:** For the fault-free $S_n$, the tree $T$ that is constructed using the function *FFR* as the parent function is a breadth-first spanning tree derived from $I$. Let $T_i$ denote the subgraph that contains the edge $(I, I^{(i)})$ and the subtree derived from the node $I^{(i)}$ of $T$, then $T_i \cap T_j = \emptyset$ for $2 \leq i \neq j \leq n$ and $T = I \cup \{T_i | 2 \leq i \leq n\}$.

**Proof:** Based on Breadth First Search, Lemma 1, 2 and 3. □

Figure 2 is the breadth-first spanning tree $T$ derived from $I = 1234$ in the fault-free $S_4$, which satisfies Lemma 4. It is obvious that $T_2$, $T_3$ and $T_4$ are node-disjoint and edge-disjoint, and $T = I \cup T_2 \cup T_3 \cup T_4$.

Lemma 4 shows a method that can construct a breadth-first spanning tree from $I$. By applying the function *FFR* to a node $p$ except $I$ in $S_n$, the node $p$ decides a node $q$ and an edge $(p, q)$, where $FFR(p, I) = q$ and $d(q, I) = d(p, I) - 1$. Let $E_T$ denote the set of the edges that are created by applying the function *FFR* to all the nodes except $I$ in $S_n$. Since a node $p$ creates only an edge $(p, q)$ with $(p, q) \in E_T$ based on the function *FFR*, $|E_T| = |V| - 1$. Since $FFR(p, I) = I$ for each node $p$ with $d(p, I) = 1$, $T = (V, E_T)$ construct a breadth-first spanning tree from $I$. By finding a fault-free subtree $T_i$ for $2 \leq i \leq n$, we will develop an adaptive fault-tolerant routing algorithm for $S_n$. Now, we give a method for a node through node-disjoint paths to find the different subtrees $T_i$, $2 \leq i \leq n$, of the breadth-first spanning tree $T$ from $I$.

**Lemma 5:** Let $p$ be a node in some $T_i$, then there are $n - 2$ node-disjoint paths of length at most 3 that pass not through the node $FFR(p, I)$ and connect respectively $p$ to the other $n - 2$ $T_j$ for $2 \leq j \neq i \leq n$.

**Proof:** For $2 \leq j \neq i \leq n$,

1. if $p_1 = 1$, then $p \rightarrow p^{(p_j)} \in T_j$.
2. if $p_1 \neq 1$,
   a. if $p^{(p_j)} = FFR(p, I)$, then $p \rightarrow p^{(1)} \rightarrow p^{(1,p_j)} \in T_j$.
   b. if $p^{(p_j)} \neq FFR(p, I)$, then $p \rightarrow p^{(p_j)} \rightarrow p^{(p_j,1)} \rightarrow p^{(p_j,1,p_1)} \in T_j$.

□

**Example 6:** As shown in Fig. 2, two node-disjoint paths from the node $p = 1324$ or $q = 4312$ in $T_3$ to $T_2$ and $T_4$ can be given by:
A. From the node $p = 1324$ with $p_1 = 1$ to $T_2$ and $T_4$,

1. $p = 1324 \rightarrow 1324^{(p_2)} = 1324^{(3)} = 3124 \in T_2$,
2. $p = 1324 \rightarrow 1324^{(p_4)} = 1324^{(4)} = 4321 \in T_4$.

B. From the node $q = 4312$ with $p_1 \neq 1$ to $T_2$ and $T_4$, since $q^{(q_4)} = 4312^{(2)} = FFR(q, I)$,

1. $q = 4312 \rightarrow 4312^{(q_2)} \rightarrow 4312^{(q_2,1)} \rightarrow 4312^{(q_2,1,q_1)} = 4312^{(3,1,4)} = 4132 \in T_2$,
2. $q = 4312 \rightarrow 4312^{(1)} \rightarrow 4312^{(1,q_4)} = 4312^{(1,2)} = 2341 \in T_4$.

### 3.2 Routing Algorithm for the Faulty $S_n$

Given the destination node $I$, since the total number of faults is less than degree $n-1$ of $S_n$, there is at least one fault-free subgraph $T_i$ for $2 \leq i \leq n$. Since $S_n$ is $n-1$-connected, there are $n-1$ node-disjoint paths between two arbitrary nodes in the $n$-star graph. By finding a fault-free $T_i$ for $2 \leq i \leq n$ based on Lemma 5 for determining a fault-free path from a source node to a destination node, we can develop a fault-tolerant routing algorithm for $S_n$ with less than $n-1$ faults. Now, we present a routing algorithm called **ROUTING** that can tolerate at most $n-2$ faults that are node faults and/or edge faults.

Let $M = \{messages, I, F, Path\}$ denote a sending request. $F$ is the set of the invalid nodes which are relative to the faults and are treated as the faulty nodes, $Path$ denotes the path from $p$ to a given $T_i$ which contains no nodes in $F$. Let $S$ be the set of invalid subgraphs $T_i$ that contain the nodes in $F$. When a faulty edge is encountered in the course of **ROUTING**, $F$, $S$ and $Path$ are updated by the function $FSP(p, I.F)$ in the following steps.

**Step 1.** Since the node $p$ is not invalid node, remove $p$ from $F$ if $p \in F$, and reset $S$ and $Path$.

**Step 2.** Update $F$ based on the local failure information.

**Step 3.** Update $S$ based on $F$, the function $Dim$, Lemma 3 and 4.

**Step 4.** Update $Path$ based on $F$, $S$ and Lemma 5.

```
function FSP(p, I: node;
             var F: the set of invalid nodes): path;
var i, p_i: integer; q: node; Path: path;
var S: the set of invalid subgraphs;
begin
  F := F - p; S := ∅; Path := ∅ /* Step 1 */
  for i := 2 to n do begin /* Step 2 */
    if the edge (p, p^(p_i)) is faulty and p^(p_i) ≠ I
      then F := F ∪ p^(p_i);
    if the edge (p, p^(p_i)) is faulty and p^(p_i) = I
      then F := F ∪ p;
  end;
  for all q ∈ F do /* Step 3 */
    begin i := Dim(q, I); S := S ∪ T_i end;
  for i := 2 to n do begin /* Step 4 */
    if T_i ∉ S then
      if p_1 = 1 then
        begin /* p^(p_i) ∈ T_i based on Lemma 5 */
          Path := p → p^(p_i); return(Path)
        end;
      if p^(p_i) = FFR(p, I) and p^(1) ∉ F then
        begin /* p^(1,p_i) ∉ F */
          Path := p → p^(1) → p^(1,p_i); return(Path)
        end;
      if p^(p_i) ≠ FFR(p, I) and p^(p_i) ∉ F
           and p^(p_i,1) ∉ F then
        begin /* p^(p_i,1,p_1) ∉ F */
          Path := p → p^(p_i) → p^(p_i,1) → p^(p_i,1,p_1);
            return(Path)
        end
  end
end;
```

Based on the functions *FFR*, $FSP$ and the local failure information, we can develop the algorithm **ROUTING**. Let $p$ be the node that has received $M$ and is ready to forward $M$. The following is the skeleton of the algorithm.

**Phase 1.** $Path = \emptyset$: decide a node $z$ by *FFR*. If the edge $(p, z)$ is non-faulty, send $M$ to $z$. Otherwise, go to Phase 3.

**Phase 2.** $Path \neq \emptyset$: decide a node $z$ by $Path$. If the edge $(p, z)$ is non-faulty, send $M$ to $z$. Otherwise, go to Phase 3.

**Phase 3.** *Updating*: update $Path$ by $FSP$, and send $M$ to $z$ that is decided by $Path$.

**Algorithm ROUTING**
**Input**: A sending request $M$.
**Output**: A node $z$ to receive the request $M$.

```
var p, z, I: node; Path: path;
var F: the set of invalid nodes;
begin
```

/* $p$ is ready to send $M$, $z$ is ready to receive $M$ */
**if** $Path = \emptyset$ **then begin** /* Phase 1 */
$z := FFR(p, I)$;
**if** edge $(p, z)$ is non-faulty **then goto** *Sending*
**else goto** *Updating*
**end**;
**if** $Path \neq \emptyset$ **then begin** /* Phase 2 */
decide $z$ by $Path$;
**if** $z$ is the terminal node of $Path$ **then** $Path := \emptyset$;
**if** edge $(p, z)$ is non-faulty **then goto** *Sending*
**else goto** *Updating*
**end**;
*Updating*:
**begin** /* Phase 3 */
$Path := FSP(p, I, F)$; decide $z$ by $Path$
**end**;
*Sending*: **if** $p \neq I$ **then** send $M$ to $z$
**end**;

For convenience, we call a pair of sending and receiving a step, and denote the length of path by the number of steps. Let $f \leq n-2$ denote the number of faults in $S_n$. Now, we prove the algorithm **ROUTING**, analyze its properties, and give the length of the path that is decided by the algorithm **ROUTING** as well as its message complexity.

**Lemma 6:** If $f \leq n-2$ in $S_n$, then $|F| \leq n-2$ and $|S| \leq n-2$; If $|S| = n-2$, then $|F| = n-2$.

**Proof:** Since a fault induces only an invalid node in $F$, $|F| \leq f \leq n-2$. Since all the invalid nodes in $T_i$ induce only the invalid $T_i$ in $S$, $|S| \leq |F| \leq n-2$. When $|S| = n-2$, since $|F| \geq |S| = n-2$ and $|F| \leq f \leq n-2$, $|F| = n-2$. □

**Lemma 7:** If $f \leq n-2$ in $S_n$, the algorithm **ROUTING** can always find a node in $T_i \notin S$.

**Proof:** We prove Lemma 7 by induction. It is obvious that Lemma 7 is correct when $f = 1$.

Assume Lemma 7 is correct when $f = k \leq n-3$. Let $x$ be the node that encounters $k$ faults, then $|F| = k$ in $M$ that is sent out from $x$.

When $f = k+1$, let $p$ be the node that encounters $(k+1)$th faults, then $|F| = k+1 \leq n-2$ and $|S| \leq |F| \leq n-2$ for the node $p$. Since $|F| \leq n-2$ and $|S| \leq n-2$ and there are $n-1$ node-disjoint paths that connect respectively $p$ to the other $n-2$ $T_j$ for $2 \leq j \neq i \leq n$ based on Lemma 5, the function $FSR$ can decide $Path$ that connects $p$ to some of the other $n-2-|S|$ $T_j \notin S$ for $2 \leq j \neq i \leq n$. Since all $k+1$ faults have been encountered before $M$ is sent out from $p$, there are no faults in $Path$ from $p$ to $T_j \notin S$. Through this fault-free $Path$, $M$ can be sent to a node in $T_j \notin S$. Lemma 7 holds. □

**Theorem 1:** The algorithm **ROUTING** can adaptively find fault-free path to route messages successfully from a source to a destination in $S_n$ with less than $n-2$ faults.

**Proof:** Without loss of generality, let $s$ denote the source node, and $I$ denote the destination node. Before faults are encountered, according to Lemma 1, $M$ is routed through an optimal path. After faults are encountered, according to Lemma 7, $M$ can always be sent to a node $p$ in $T_i$ where there are no faults if $f \leq n-2$. According to Lemma 1, $M$ can be sent from $p$ to $I$ through an optimal path. Since the function $FSP$ in the course of the algorithm **ROUTING** always decides $Path$ to $T_i \notin S$ based on the local failure information, this algorithm is adaptive when faults are encountered. □

**Corollary 1:** If $f \leq n-2$ in $S_n$, the algorithm **ROUTING** is deadlock-free and livelock-free.

**Theorem 2:** The algorithm **ROUTING** can take at most $d(s, I)+4f$ steps to route messages from the source $s$ to the destination $I$ in $S_n$ with $f \leq n-2$ faults.

**Proof:** We prove the correctness of Theorem 2 by induction. It is obvious that Theorem 2 is correct when no faults are encountered in routing.

When $f = 1$, let $p$ be the node that encounters the fault, and let $i = Dim(p, I)$, then $p$ is in $T_i$. Based on Lemma 5, we need to prove that Theorem 2 holds for $2 \leq j \neq i \leq n$ in the following cases:

**Case 1:** $p_1 = 1$. $p \rightarrow p^{(p_j)} \in T_j$.

**Case 2:** $p_1 \neq 1$,

**Case 2.1:** $p^{(p_j)} = FFR(p, I)$. $p \rightarrow p^{(1)} \rightarrow p^{(1,p_j)} \in T_j$.

**Case 2.2:** $p^{(p_j)} \neq FFR(p, I)$. $p \rightarrow p^{(p_j)} \rightarrow p^{(p_j,1)} \rightarrow p^{(p_j,1,p_1)} \in T_j$.

**Case 1.** Since only $T_i \in S$, it takes one step to send $M$ from the node $p$ to the node $p^{(p_j)}$ in $T_j \neq T_i$. From $d(p^{(p_j)}, I) \leq d(p, I) + 1$, $d(s, p) + 1 + d(p^{(p_j)}, I) \leq d(s, p) + d(p, I) + 2 = d(s, I) + 2$. It takes at most $d(s, I) + 2$ steps to send $M$ from $s$ to $I$.

**Case 2.** Without loss of generality, let $p = p_1 ... p_i ... p_j ... p_n$, then $p = p_1 ... 1 ... p_j ... p_n$.

**Case 2.1.** $p^{(p_j)} = FFR(p, I)$. It takes two steps to send $M$ from the node $p$ to the node $p^{(1,p_j)}$ in $T_j \neq T_i$. From $d(p^{(1,p_j)}, I) \leq d(p, I) + 2$, $d(s, p) + 2 + d(p^{(1,p_j)}, I) \leq d(s, p) + d(p, I) + 4 = d(s, I) + 4$. It takes at most $d(s, I) + 4$ steps to send $M$ from $s$ to $I$.

**Case 2.2.** $p^{(p_j)} \neq FFR(p, I)$. It takes three steps to send $M$ from the node $p$ to the node $q = p^{(p_j,1,p_1)} = p_1 ... p_j ... 1 ... p_n$ in $T_j \neq T_i$, where the symbol 1 is in position $j$ of $p^{(p_j,1,p_1)}$. Let $p\text{-}cycles = \{C_1^p, C_2^p\}$, then $d(p, I) = |C_1^p| + |C_2^p| + |M_1^p| + |M_2^p| - 2$. Since there are only three cases: $p_j \in M_1$, $p_j \in M_2$ or $p_j \notin M_1 \cup M_2$ for the symbol $p_j$, we need to prove that $d(q, I) \leq d(p, I)+1$ in the following three cases:

1. $p_j \in M_1$. Without loss of generality, let $\{p_1, x, ..., y, p_j, z, ..., l, 1\} \in C_1^p$, then $\{z, ..., l, p_j\} \in C_2^q$ and $\{p_1, x, ..., y, 1\} \in C_1^q$. Since $|M_1^q| + |M_2^q| = |M_1^p| + |M_2^p|$ and $|C_1^q| + |C_2^q| = |C_1^p| + |C_2^p| + 1$, $d(q, I) = d(p, I) + 1$.

2. $p_j \in M_2$. Without loss of generality, let $\{p_1, x, ..., y, 1\} \in C_1^p$ and $c_j = \{p_j, z, ..., l, p_{c_j}\}$, then $\{p_1, x, ..., y, p_j, z, ..., l, p_{c_j}, 1\} \in C_1^q$. Since $|M_1^q| + |M_2^q| = |M_1^p| + |M_2^p|$ and $|C_1^q| + |C_2^q| = |C_1^p| + |C_2^p| - 1$, $d(q, I) = d(p, I) - 1$.

3. $p_j \notin M_1 \cup M_2$. Without loss of generality, let $\{p_1, ..., 1\} \in C_1^p$, then $\{p_1, ..., p_j, 1\} \in C_1^q$ and $|M_1^q| = |M_1^p| + 1$. Since $|M_1^q| + |M_2^q| = |M_1^p| + |M_2^p| + 1$ and $|C_1^q| + |C_2^q| = |C_1^p| + |C_2^p|$, $d(q, I) = d(p, I) + 1$.

In **Case 2.2**, $d(q, I) \leqq d(p, I) + 1$. Let $L(p, q)$ denote the length of the path from $p$ to $q$, then $L(p, q) = 3$ based on Lemma 5. Since $d(s, p) + L(p, q) + d(q, I) \leqq d(s, p) + d(p, I) + 4 = d(s, I) + 4$, Theorem 2 holds when $f = 1$.

Assume that $M$ can be sent from $s$ to $I$ in $d(s, I) + 4k$ steps when $f = k \leqq n - 3$.

When $f = k + 1$, let $p$ be a node that encounters $k+1$th fault, then it takes at most $4k$ extra steps to send $M$ from $s$ to $p$. Since it takes at most 4 extra steps from $p$ to $I$ based on the proof of $f = 1$, the length of the path from $s$ to $I$ is $d(s, I) + 4(k + 1)$.

In conclusion, the algorithm **ROUTING** takes at most $4f$ extra steps to route messages from the source $s$ to the destination $I$ if $f \leqq n - 2$ in $S_n$. The maximum length of the path is $d(s, t) + 4f$. □

**Corollary 2:** If no faults are encountered in routing, the algorithm **ROUTING** is optimal.

**Corollary 3:** If $f \leqq n - 2$ in $S_n$, the number of $M$ transmitted from the source $s$ to the destination $I$ in the course of **ROUTING** is less than or equal to $d(s, I) + 4f$.

Here, we give two examples to illustrate the routing course of the algorithm **ROUTING**. Example 7 shows that this algorithm needs not to judge the types of faults, because it can tolerate not only node faults but also edge faults. Example 8 shows that this algorithm can take at most $4f$ extra steps to route messages from a source node to a destination node if $f \leqq n - 2$ in $S_n$. Let $x \xrightarrow{M} y$ denote that $x$ sends $M$ to $y$. Referred to Fig. 2, the subgraphs $T_i$ for $2 \leqq i \leqq 4$, where each node is, can be determined.

**Example 7:** As shown in Fig. 3, let $s = 2143$ be the source, and let the node 1324 and the edge $(1234, 2134)$ be faulty. Through the shortest path that is decided using repeatedly $FFR$, $2143 \xrightarrow{M} 3142 \xrightarrow{M} 4132 \xrightarrow{M} 2134$. Since $FFR(2134, 1234) = 1234$ and the edge $(1234, 2134)$ is faulty, $FSP(2134, 1234)$ updates $F = \{2134\}$, $S = \{T_2\}$ and $Path = \{2134 \rightarrow 3124 \rightarrow 1324 \rightarrow 2314\}$ that connects to $T_3$. Through this path, $2134 \xrightarrow{M} 3124$. When the faulty edge $(3124, 1324)$ in $Path$ is encountered, $FSP(3124, 1234)$ updates $F = \{2134, 1324\}$, $S = \{T_2, T_3\}$ and $Path = \{3124 \rightarrow 4123 \rightarrow 1423 \rightarrow 3421\}$ that connects to $T_4$. Through this path, $3124 \xrightarrow{M} 4123 \xrightarrow{M} 1423 \xrightarrow{M} 3421$. Then, $3421 \xrightarrow{M} 2431 \xrightarrow{M} 4231 \xrightarrow{M} 1234 = I$. The length of the path from $s = 2143$ to $I = 1234$ is 10 that is less than $d(s, I) + 4 \times 2 = 12$. It shows that this algorithm can tolerate not only node faults but also edge faults.

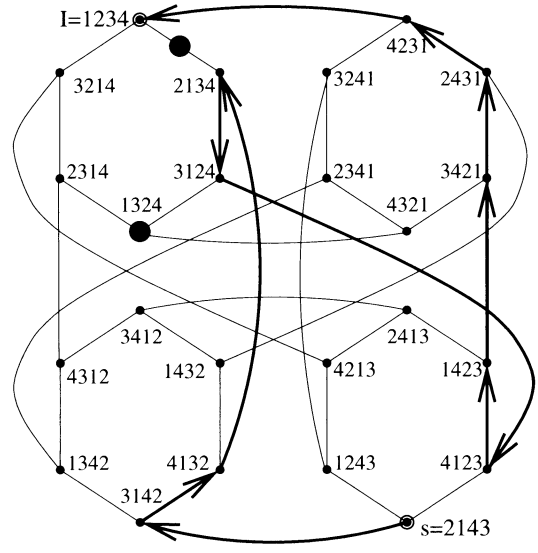



**Fig. 3** Routing from $s$ to $I$ in $S_4$ with the faulty edge $(1234, 2134)$ and the faulty node 1324.

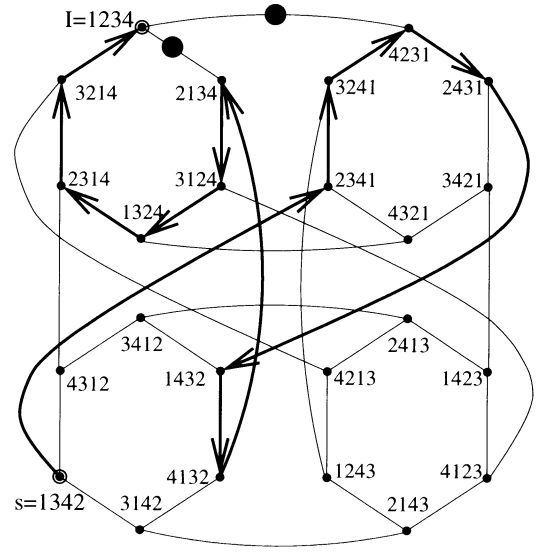



**Fig. 4** Routing from $s$ to $I$ in $S_4$ with the faulty edges $(1234, 2134)$ and $(1234, 4231)$.

**Example 8:** As shown in Fig. 4, let $s = 1324$ be the source, and let the edges $(1234, 2134)$ and $(1234, 4231)$ be faulty. Through the shortest path, $1342 \xrightarrow{M} 2341 \xrightarrow{M} 3241 \xrightarrow{M} 4231$. When the faulty edge $(1234, 4231)$ is encountered, $FSP(4231, 1234)$ updates $F = \{4231\}$, $S = \{T_4\}$ and $Path = \{4231 \rightarrow 2431 \rightarrow 1432 \rightarrow 4132\}$ that connects to $T_2$. Then, $4231 \xrightarrow{M} 2431 \xrightarrow{M} 1432 \xrightarrow{M} 4132$ and $4132 \xrightarrow{M} 2134$. When the faulty edge

**Table 1** Comparison of basic parameters of $n$-hypercube and $n$-star graph with $n > 2$.

| graph | dimension | nodes | degree | diameter | average diameter | shortest cycle |
|---|---|---|---|---|---|---|
| $n$-cube | $n$ | $2^n$ | $n$ | $n$ | $\frac{n}{2}$ | 4 |
| $n$-star | $n$ | $n!$ | $n-1$ | $\lfloor \frac{3}{2}(n-1) \rfloor$ | $n-4+\frac{2}{n}+\sum_{i=1}^{n} 1/i$ | 6 |
| 7-cube | 7 | 128 | 7 | 7 | 3.5 | 4 |
| 5-star | 5 | 120 | 4 | 6 | 3.7 | 6 |
| 12-cube | 12 | 4096 | 12 | 12 | 6 | 4 |
| 7-star | 7 | 5040 | 6 | 9 | 5.9 | 6 |
| 18-cube | 18 | 262144 | 18 | 18 | 9 | 4 |
| 9-star | 9 | 362880 | 8 | 12 | 8.1 | 6 |

$(1234, 2134)$ is encountered, $FSP(2134, 1234)$ updates $F = \{4231, 2134\}$, $S = \{T_4, T_2\}$ and $Path = \{2134 \rightarrow 3124 \rightarrow 1324 \rightarrow 2314\}$ that connects to $T_3$. Then, $2134 \stackrel{M}{\rightarrow} 3124 \stackrel{M}{\rightarrow} 1324 \stackrel{M}{\rightarrow} 2314$, and $2314 \stackrel{M}{\rightarrow} 1234 = I$. The length of the path from $s = 1324$ to $I = 1234$ is 12 that is equal to $d(s, I) + 4 \times 2 = 12$. It shows that the maximum length of the path is equal to $d(s, I) + 4f$ if $f \leq n - 2$ in $S_n$.

Note that the actual path lengths are shorter than the upper boundary given in Theorem 2 in general and the conditions of faults affect the length of the path that is decided by the algorithm **ROUTING**. Though there are the shortest paths between $s$ and $I$, it is obvious that it is impossible to insure to find it only based on the local failure information. For example as shown in Fig. 4, there is the shortest path $1342 \rightarrow 4312 \rightarrow 2314 \rightarrow 3214 \rightarrow 1234$. Since three adjacent edges are all non-faulty and there is the shortest path from 1342 through 2341 to 1234, the node 1342 can route $M$ through the shortest path $1342 \rightarrow 2341 \rightarrow 3241 \rightarrow 4231 \rightarrow 1234$ if the faulty edge $(1234, 4231)$ is unknown. It shows that, for the $n$-star graph, there are not any optimal routing algorithms only based on the local failure information.

As described in [10], the shortest length of cycles of the $n$-star graph is 6. As shown in Fig. 1, let the edge $(1234, 2134)$ be faulty, any algorithm takes at least 4 steps to route messages from 2134 to 3214 or 4231. It shows that any routing algorithm will take at least $4f$ extra steps to complete routing in the worst case when any node only knows the condition of its incident edges. Example 8 shows that it takes $d(s, I) + 8$ steps for $s = 1324$ to send messages to $I = 1234$ when the edges $(1234, 2134)$ and $(1234, 4231)$ be faulty in $S_4$. If any node only knows the condition of its incident edges, we conjecture that any routing algorithm cannot always route messages through the path whose length is less than the length of the path that is decided by the algorithm **ROUTING** for the $n$-star graph with less than $n - 1$ faults.

### 3.3 Performance Comparisons

We show the performance of our algorithm **ROUTING** by comparing it with the algorithm **Unicast_V** proposed by Lan [11] for the hypercube, and the algorithm **DSR** (Depth-Search-Routing) proposed by Bagherzadeh, Nassif, and Latifi [3] for the star graph. We compare with those algorithms since they all make routing decisions adaptively based on the local failure information only.

Both the hypercube and the star graph are regular, node symmetric and edge symmetric. Table 1 shows a summary of the basic parameters of the $n$-dimension hypercube and the $n$-star graph. From Table 1, we can see that, based purely on the degree and diameter requirements, the $n$-star graph is asymptotically superior for a given size.

By applying the property of $n$-bit binary numbers that are correspondence with nodes of the $n$-dimension hypercube, Lan proposed the adaptive fault-tolerant algorithm **Unicast_V** for the $n$-dimension hypercube with less than $n$ faults. It can route messages from a source $u_s$ to a destination $u_d$ in no more than $H(u_s, u_d) + 2F$ steps, where $F$ is the number of faults (edge and/or node faults) and $H(u_s, u_d)$ is the distance from $u_s$ to $u_d$. The set of the extra arguments required for the routing is called message overhead. In the algorithm **Unicast_V**, the message overhead consists of two arguments: the destination node, and an $n$-bit binary vector $T$ that is introduced to deal with faults. Since the $n$-star graph is more complex than the $n$-dimension hypercube, the fault-tolerant routing algorithms, which have been proposed for the $n$-star graph, introduce more message overheads to deal with faults than that for the $n$-dimension hypercube. In our algorithm **ROUTING**, the message overhead consists of three arguments: the destination node, $F$ which is bounded by $n - 2$ elements, and $Path$ which is bounded by 4 elements. The total message overheads are bounded by $n + 3$ elements.

Based on Depth First Search, the algorithm **DSR** always can find a path between two nodes within a bounded number of steps if the two nodes in the star graph are connected. Faults are assumed to be one or more edges in the algorithm **DSR**. If the total number of edge faults is less than degree $n - 1$ of $S_n$, the algorithm **DSR** can take at most $d(s, I) + (2i + 2)f$ steps to route messages from a source $s$ to a destination $I$, where $i = \sqrt{2n - 5.75} + 0.5$. When $n \geq 3$, $2i + 2 \geq 4$ since $i = \sqrt{2n - 5.75} + 0.5 \geq 1$. It shows that the time complexity of the algorithm **DSR** is worse than the time complexity of **ROUTING**. For the $n$-star graph with

$f < n-1$ edge faults, the algorithm **DSR** introduces the following message overhead that consists of four arguments: the destination node, $f/w$ which is one bit, *Visited* which is bounded by $n!$ elements, and *Linklist* which is bounded by $d(s,I)+(2i+2)f$ elements. It is obvious that the total number of the message overheads of the algorithm **DSR** is more than that of our algorithm **ROUTING**.

In order to insure that two non-faulty nodes in $S_n$ are always connected, we assume that the total number of faults is less than degree $n-1$ of $S_n$. Compared with the algorithm **DSR**, our algorithm **ROUTING** has the following advantages:

1. **ROUTING** is better than **DSR** in the sense that **ROUTING** can directly tolerate not only edge faults but also node faults.
2. **ROUTING** is better than **DSR** in the time complexity.
3. **ROUTING** introduces less message overheads than **DSR**.

## 4. Conclusions

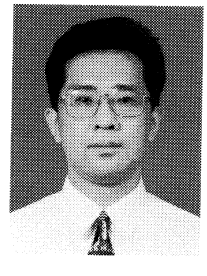
This paper has presented an adaptive distributed routing algorithm without deadlock and livelock for the faulty star graph. Based on the given routing rules and the concept of Breadth First Search, we can determine $n-1$ node-disjoint and edge-disjoint subgraphs, which are derived from $n-1$ adjacent edges of a given destination in the $n$-star graph. When faults are encountered, the algorithm can route messages to the destination by finding a fault-free node-disjoint subgraphs based on the local failure information and the properties of nodes. The judgment of the types of faults is not required. It insures that the routing procedure is deadlock-free and livelock-free. If there are $f \leq n-2$ faults (node faults and/or edge faults) in the $n$-star graph, it can find a path of length at most $d(s,I)+4f$ to route messages from a source $s$ to a destination $I$ successfully.

## Acknowledgment

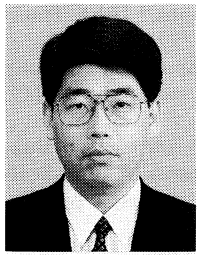
The authors would like to thank the reviewers for their constructive and helpful comments. They have made many helpful revisions and suggestions that have significantly improved the presentation.

## References

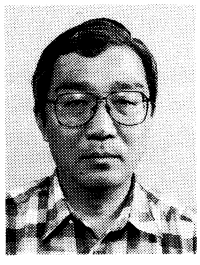
[1] S.B. Akers, D. Harel, and B. Krishnamuthy, "The star graph: An attractive alternative to the n-cube," Proc. Int. Conf. Parallel Proceeding, pp.393–400, 1987.
[2] S.B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," IEEE Trans. Comput., vol.38, no.4, pp.555–566, 1989.
[3] N. Bagherzadeh, N. Nassif, and S. Latifi, "A routing and broadcasting scheme on faulty star graphs," IEEE Trans. Comput., vol.42, no.11, pp.1398–1403, 1993.
[4] N. Bagherzadeh and M. Dowd, "Computation in faulty stars," IEEE Trans. Reliability, vol.44, no.1, pp.114–119, 1995.
[5] L. Chungti, B. Sourav, and T. Jack, "Performance evaluation of fault-tolerant routing on star networks," Proc. Scalable High Performance Computing Conference, pp.650–657, 1994.
[6] K. Day and A. Tripathi, "A comparative study of topological properties of hypercubes and star graphs," IEEE Trans. Parallel & Distrib. Syst., vol.5, no.1, pp.31–38, 1994.
[7] Q.P. Gu and S.T. Peng, "Linear time algorithm for fault tolerant routing in hypercubes and star graph," IEICE Trans. Inf. & Syst., vol.E78-D, no.9, pp.1171–1177, 1995.
[8] Q.P. Gu and S.T. Peng, "Set-to-set fault tolerant routing in star graphs," IEICE Trans. Inf. & Syst., vol.E79-D, no.4, pp.282–289, 1996.
[9] Z. Jovanovic and J. Misic, "Fault tolerance of the star graph interconnection network," Infor. Process. Lett., vol.49, pp.145–150, 1994.
[10] J.S. Jwo, S. Lakshmivarahan, and S.K. Dhall, "Embedding of cycles and grids in star graphs," J. Circuits, Syst. & Comput., vol.1, no.1, pp.43–74, 1991.
[11] Y. Lan, "An adaptive fault-tolerant routing algorithm for hypercube multicomputers," IEEE Trans. Parallel & Distrib. Syst., vol.6, no.11, pp.1147–1152, 1995.
[12] J.A. McHugh, "Algorithmic Graph Theory," Prentice-Hall Inc., 1990.
[13] Y. Rouskov, S. Latify, and P.K. Srimani, "Conditional fault diameter of star graph network," J. Parallel & Distrib. Computing, vol.33, no.1, pp.91–98, 1996.
[14] L. Shahram, "Parallel dimension permutations on star-graph," Proc. IFIP WG10.3 Working on Architectures and Compilation Techniques for Fine and Medium Grain Parallelism, pp.191–201, 1993.
[15] R. Srinivasan, V. Chaudhary, and S.M. Mahmud, "Contention sensitive fault-tolerant routing algorithm for hypercubes," International Symposium on Parallel Architectures, Algorithms and Networks, pp.197–204, 1994.
[16] C.C. Su and K.G. Shin, "Adaptive fault-tolerant deadlock-free routing in mesh and hypercubes," IEEE Trans. Comput., vol.45, no.6, pp.666–683, 1996.



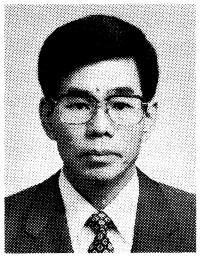
**Leqiang Bai** was born in 1962. He received the B.E. and M.E. degrees all from Northeast University, Shenyang, China, and the D.E. degree from Osaka University, Osaka, Japan, in 1984, 1987 and 1998, respectively. He was a lecture of the Department of Communications Engineering, Northeast University from 1987 to 1990. His research interests include algorithm, parallel processing, and fault-tolerant routing and broadcasting.



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



**Hideo Nakano** was born in 1948. He received the B.E., M.E. and D.E. all degrees from Osaka University, Osaka, Japan, in 1970, 1972 and 1975, respectively. He was a lecture and an associate professor of the Department of Communications Engineering, Osaka University from 1975 to 1995. Since 1995, he has been a professor of the Media Center, Osaka City University. His research interests include combinatorial optimization, security, and Internet. He is a member of IEEE, ACM.



**Hajime Maeda** was born in 1943. He received the B.E., M.E. and D.E. all degrees from Osaka University, Osaka, Japan, in 1966, 1968 and 1971, respectively. He was a lecture and an associate professor of the Department of Communications Engineering, Osaka University. Since 1993, he has been a professor of the Department of Communications Engineering of the same university. His research interests include system and control theory, network theory, and signal theory. He is a member of IEEE.