International Journal of Foundations of Computer Science
© World Scientific Publishing Company

FAULT TOLERANCE ON STAR GRAPHS [*]

SHUO-CHENG HU

and

CHANG-BIAU YANG
Department of Applied Mathematics
National Sun Yat-sen University
Kaohsiung, Taiwan 804, R.O.C.
cbyang@math.nsysu.edu.tw

Received (received date)
Revised (revised date)
Communicated by D. Frank Hsu

ABSTRACT

The capability of fault tolerance is one of the advantages of multiprocessor systems. In this paper, we prove that the fault tolerance of an n-star graph is $2n-5$ with restriction to the forbidden faulty set. And we propose an algorithm for examining the connectivity of an n-star graph when there exist at most $2n-4$ faults. The algorithm requires $O(n^2 \log n)$ time. Besides, we improve the fault-tolerant routing algorithm proposed by Bagherzadeh et al. by calculating the cycle structure of a permutation and the avoidance of routing message to a node without any nonfaulty neighbor. This calculation needs only constant time. And then, we propose an efficient fault-tolerant broadcasting algorithm. When there is no fault, our broadcasting algorithm remains optimal. The penalty is $O(n)$ if there exists only one fault, and the penalty is $O(n^2)$ if there exist at most $n-2$ faults.

Keywords: interconnection network, star graph, fault tolerance, routing, broadcasting.

1. Introduction

A multiprocessor system consists of a set of processing units, each of them having its own local memory. The processing units in a multiprocessor system are linked in some topology. The topology of an interconnection network can be modeled by a graph $G(V, E)$, where V is the set of processing units and E represents the communication links connecting pairs of processors. In the past, many researchers have worked on various interconnection networks, such as linear array, mesh, ring, tree, hypercube and so on. In this paper, we shall study the fault tolerance of star graphs [2,3].

[*]This research work was partially supported by the National Science Council of the Republic of China under contract NSC 83-0408-E-110-010.

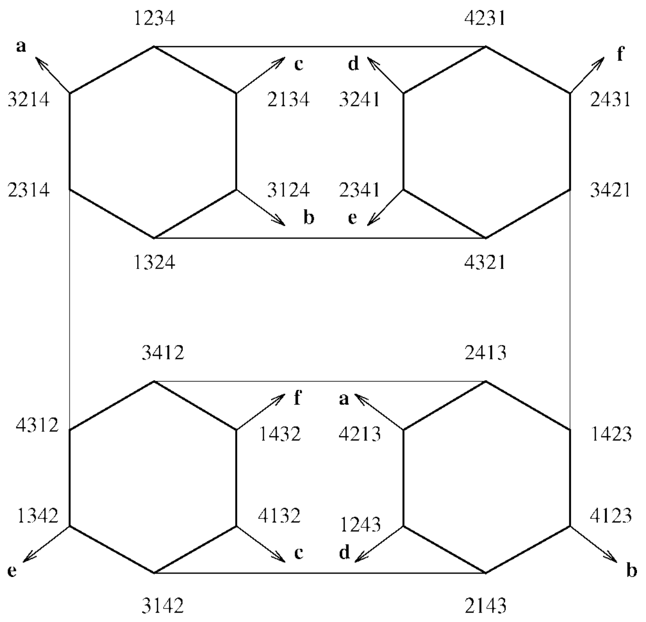

Figure 1: A 4-dimensional star graph, S_4.

An *n-dimensional star graph* is represented by $S_n = (V_n, E_n)$, where V_n consists of $n!$ nodes in which each node is identified by one of the permutations of $\{1, 2, \cdots, n\}$. For any $u, v \in V_n$, $(u, v) \in E_n$ if and only if there exists some i, $2 \leq i \leq n$, such that u and v are identified as $(p_1\ p_2\ \cdots\ p_{i-1}\ p_i\ p_{i+1}\ \cdots\ p_n)$ and $(p_i\ p_2\ \cdots\ p_{i-1}\ p_1\ p_{i+1}\ \cdots\ p_n)$ respectively. We write $g_i(u) = v$ where $g_i = (1\ i)$, $2 \leq i \leq n$, is a generator in the generating set of S_n. In this paper, the notation $g_i(g_j(u))$ is simplified as $g_i g_j(u)$. The notation S_m^b denotes an m-substar in S_n in which the $(m+1)$th position symbol is b. For example, S_4 is illustrated in Figure 1.

The star graph is comparable to the hypercube in many aspects. For example, both of them are edge symmetric, node symmetric, strongly hierarchical, bipartite and optimally fault tolerant. But star graphs offer a better degree and diameter than hypercubes. So the star graph structure has been considered as an attractive alternative to the hypercube structure.

The common measure of fault tolerance in interconnection networks is based upon the connectivity of the underlying graphs since the fault tolerance is the connectivity minus one. Esfahanian [8] assumed that in a hypercube, some related processors, called the *forbidden faulty set*, do not fail at the same time. And, under this assumption, he gave a measure of fault tolerance in the hypercube. In this paper, we shall give a similar measure of fault tolerance on an S_n by restricting the forbidden faulty set not to fail at the same time. Here, for a node, the forbidden faulty set consists of all of its neighbors. An efficient algorithm, with $O(n^2 \log n)$ time, for examining the connectivity of a star graph with at most $2n-4$ faults will also be proposed.

Data routing and broadcasting are two essential issues in interconnection networks. Akers et al.[2] and Mendia et al.[9] have proposed optimal routing and broadcasting algorithms on fault-free star graphs. However, one of the advantages of multiprocessor systems is its capability of fault tolerance. There was little research on routing and broadcasting on faulty star graphs in the past. Sur et al.[11] and Bagherzadeh et al.[5] proposed two fault-tolerant routing algorithms. Both of them are based on the depth first search strategy combined with the optimal routing algorithm. In this paper, we will improve the fault-tolerant routing algorithm proposed by Baghezadeh et al.[5]. And then, we shall propose an efficient broadcasting algorithm on faulty star graphs. When there is no fault, our broadcasting algorithm remains optimal. In an n-star graph, the penalty is $O(n)$ if there exists only one fault, and the penalty is $O(n^2)$ if there exist at most $n-2$ faults.

The rest of this paper is organized as follows. In Section 2, we shall present previous work. In Section 3, we shall propose a generalized measure of fault tolerance. In Section 4, an improved fault-tolerant routing algorithm will be proposed. In Section 5, an efficient fault-tolerant broadcasting algorithm will be proposed. And finally, this paper will be concluded in Section 6.

2. Previous Work

Some important properties of star graphs are given as follows.

Property 1 [2] *An n-star graph is a regular graph of degree $n-1$. The diameter of an n-star graph is $\lfloor \frac{3(n-1)}{2} \rfloor$.*

Property 2 [10] *An S_n can be decomposed into n S_{n-1}'s in $n-1$ different ways.*

Property 3 [2,3] *The star graph is both edge symmetric and node symmetric.*

Property 4 [7] *There are $n-1$ node disjoint paths between any two nodes in an S_n, and the connectivity of an S_n is $n-1$.*

Property 5 [1,4] *The star graph is optimally fault tolerant.*

Mendia et al. [9] proposed an optimal broadcasting algorithm on fault-free star graphs. Their algorithm for S_n can be divided into the following two phases. In the first phase, the source message is transmitted to $n-2$ neighbors of the source node with the first symbol coinciding with the symbols on the second through the $(n-1)$th positions of the source node respectively. These nodes are called *relay nodes*. In the second phase, by applying g_n, the source node and each of the $n-2$ relay nodes will send the message to a unique sub-star. After these two phases, each S_{n-1} of S_n has a copy of the message in one of its nodes. Each of these nodes is called a *leader node*, which is responsible for broadcasting message in its sub-star. The same skill can be recursively applied until the dimension of the sub-stars becomes 1.

Bagherzadeh et al. [5] proposed a routing algorithm on faulty star graphs, which is based on the greedy optimal routing algorithm combined with depth first search [2,6]. Here, faults are assumed to be in one or more links and a faulty node can be modeled as all links connecting to it are faulty. It is also assumed that a node only know the status of the links and the nodes connecting to it. Since an S_n has the symmetric property, without losing generality, the destination node is assumed to be the identity node, i.e. $(123\ldots n)$. Whenever a node receives a message, it calculates the cycle structure of its identifier. The i-cycle of the cycle structure of a permutation are placed in the following sequence: All i-cycles, $i \geq 2$, are placed on the left of all 1-cycles, the i-cycle, $i \geq 2$, containing 1 is the leftmost cycle and 1 is the rightmost symbol in the cycle. For example, the cycle structure of node (25641873) is (251)(368)(4)(7). In the current node, their algorithm first selects the leftmost symbol in the leftmost i-cycle, $i \geq 2$. The selected symbol j corresponds link j of the current node. If the selected link j is nonfaulty and nonvisited then the message is sent forward along it, i.e. g_j is applied. Otherwise, starting from the rightmost position of the rightmost i-cycle, $i \geq 2$, and proceeding to the left, try to select each symbol in each cycle until some nonfaulty and nonvisited link is found. When no symbol remains in i-cycle, $i \geq 2$, starting from the rightmost cycle and proceeding to the left, try to select each 1-cycle until some nonfaulty and nonvisited link is found. Otherwise, send a return message along the link which it just received message from. When there are at most $n-2$ faults, Bagherzadeh et al. [5] has proved that their routing algorithm has penalty $O(\sqrt{n})$.

3. A Generalized Measure of Fault Tolerance

In this section, we shall generalize the measure by restricting some subsets of nodes not to fail at the same time. Similar research on hypercubes has been done by Esfahanian [8].

Definition 1 [8] *In a multiprocessor system, a subset of system components is said to be a* **forbidden faulty set** *if the set of nodes does not potentially fail at the same time.*

Recall that the connectivity of an S_n is $n-1$[7]. There are $\binom{n!}{n-1}$ node subsets of size $n-1$, only $n!$ of them can disconnect the S_n (This will be proved later.). This ratio is very small, and it is getting smaller as n increases. For a node in an S_n, the *forbidden faulty set* is defined as the set of the $n-1$ neighbors of that node. Before giving our theorems, we define some notations. Let $K'(S_n)$ denote the number of faulty nodes needed to disconnect an S_n with restriction to the forbidden faulty sets. Let $A(G,v)$ denote the set of all nodes adjacent to v in G. And let F^i_{n-1} denote the set of faulty nodes in the S^i_{n-1}.

Theorem 1 $K'(S_n) = 2n-4$.

Proof. Let F denote the set of faulty nodes in S_n. Suppose $|F| = 2n-4$. It is clear that if all of the $2n-4$ nodes are adjacent to either node u or v, where u and v are adjacent, then the S_n is disconnected. In other words, the nodes in F enclose u and v.

Now we want to show that S_n is connected if $|F| \leq 2n-5$. An S_n can be decomposed into n S_{n-1}'s, say $S^1_{n-1}, S^2_{n-1}, \cdots, S^n_{n-1}$. If there is no such i so that$|F^i_{n-1}| \geq n-2$ ($|F^i_{n-1}| \leq n-3$ for all i), then S_n is connected since, by Property 4, each S^i_{n-1} is connected. So we suppose that there exists i such that

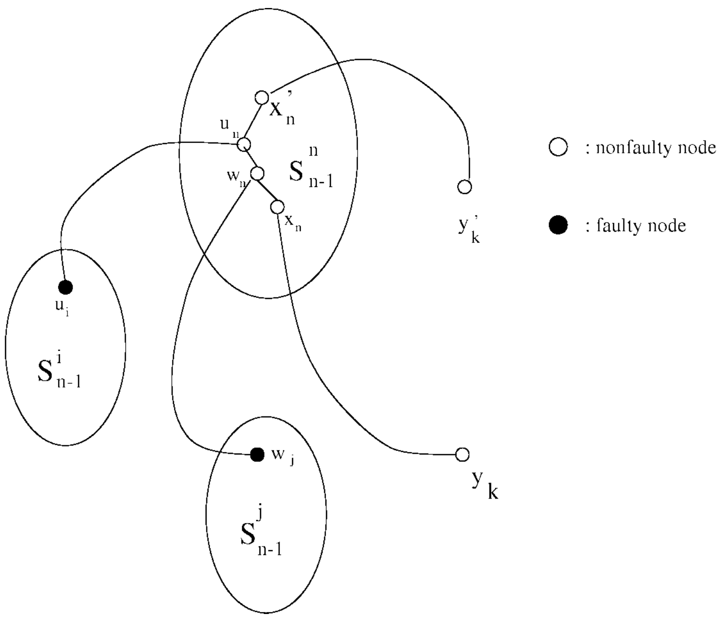

Figure 2: Two kinds of paths connecting u_n to another node y_k in some S_{n-1}.

$|F^i_{n-1}| \geq n-2$. Without loss of generality, we assume that $i = n$. Now we want to prove that each node in $S^n_{n-1} - F^n_{n-1}$ is connected via a path to a node in $S_n - S^n_{n-1}$.

Let u_n be an arbitrary node in graph $S^n_{n-1} - F^n_{n-1}$ and (u_n, u_i) be an edge from S^n_{n-1} to S^i_{n-1} for some $i \neq n$. If $u_i \notin F$, we have done. So assume that $u_i \in F$. Besides, $A(S_n, u_n)$ is not a subset of F due to the forbidden faulty set. Hence $A(S^n_{n-1} - F^n_{n-1}, u_n)$ is not empty. See Figure 2. Let $w_n \in A(S^n_{n-1} - F^n_{n-1}, u_n)$ and (w_n, w_j) be an edge from S^n_{n-1} to S^j_{n-1} for some $j \neq n$. If $w_j \notin F$, we have done. So assume that $w_j \in F$. Let $X = A(s^n_{n-1}, u_n) \cup A(s^n_{n-1}, w_n) - \{w_n, u_n\}$. That is, X is the set of neighbors of w_n and u_n in S^n_{n-1} excluding themselves. Since there are $n-2$ neighbors for a node in S^n_{n-1}. So $|X| = 2(n-2) - 2 = 2n-6$. Let $F' = F - \{u_i, w_j\}$. We have $|F'| \leq 2n-7$. Thus, there must be at least one node in X, say x_n, and (x_n, y_k) is an edge from S^n_{n-1} to S^k_{n-1} for some $k \neq n$ such that both x_n and y_k are nonfaulty. This implies that in $S_n - F$, each node is connected to another node in $S_n - S^n_{n-1}$ via a path of length at most 3. We complete our proof. □

In Theorem 1, we have shown that an S_n in which no node has all faulty neighbors can tolerate $2n-5$ faulty nodes. In the following, we shall propose an algorithm for determining whether an S_n is connected or not if there exist at most $2n-4$ faults. For identifying the correctness of our algorithm, the following three lemmas are needed.

Lemma 1 *Let F be the set of faulty nodes in S_n. If $F = A(S_n, v) \cup \{v\}$ for some v in S_n, then graph $S_n - F$ is connected.*

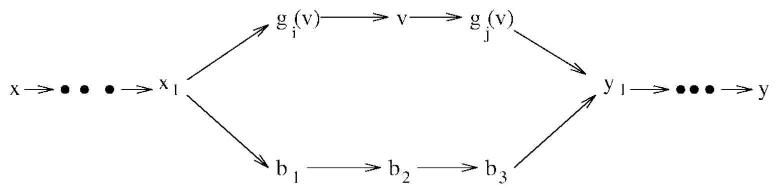

Figure 3: Two paths from x to y.

Proof. We shall prove this lemma by showing that for any pair of nonfaulty nodes in S_n, there exists a path connecting them.

Let x and y be two nonfaulty nodes in S_n. We can construct a path from x to y by applying the optimal routing algorithm. If the path does not contain any node in F, we have done. Otherwise, there is at least one fault on the path. It must be one of the following two cases:
$x \rightarrow \cdots \rightarrow x_1 \rightarrow g_i(v) \rightarrow y_1 \rightarrow \cdots \rightarrow y$
and
$x \rightarrow \cdots \rightarrow x_1 \rightarrow g_i(v) \rightarrow v \rightarrow g_j(v) \rightarrow y_1 \rightarrow \cdots \rightarrow y$
where nodes $g_i(v)$, v and $g_j(v)$ are faulty. In the first case, the path from x_1 to y_1 can be substituted by $x_1 \rightarrow a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow y_1$ where x_1, a_1, a_2, a_3, y_1 and $g_i(v)$ form an S_3. And in the second case, the distance between x_1 and y_1 is 4. If x_1, $g_i(v)$, v, $g_j(v)$ and y_1 are not in an S_3, then they must be in a cycle of eight nodes. We can find nodes b_1, b_2 and b_3 such that the path $x_1 \rightarrow b_1 \rightarrow b_2 \rightarrow b_3 \rightarrow y_1$ is fault-free. (See Figure 3.) We complete our proof. $\Box$

Lemma 2 *Let F be the set of faulty nodes in S_n. Suppose $|F| = n-1$. $S_n - F$ is disconnected if and only if $F = A(S_n, v)$ for some node v in S_n. Under this situation, $S_n - F$ has exactly two components, one consisting of v, and the other consisting of $S_n - F - \{v\}$.*

Proof. It is trivially true that $S_n - F$ is disconnected if $F = A(S_n, v)$ for some node v in S_n.
($\Rightarrow$) Suppose there exists u, a neighbor of v, is nonfaulty. It is not hard to see that u connects to all other nodes in $S_n - F$, by an argument similar to the proof of Lemma 1. Thus $S_n - F$ is connected. This is a contradiction. We complete our proof. $\Box$

Lemma 3 *Let F be the set of faulty nodes in S_n. Suppose that $|F| = 2n-4$ and for every node in S_n, $A(S_n, v)$ is not a subset of F. $S_n - F$ is disconnected if and only if there exists an edge (u, v) in S_n such that $F = A(S_n, v) \cup A(S_n, u) - \{u, v\}$.*

Proof. It is trivially true that $S_n - F$ is disconnected if $F = A(S_n, v) \cup A(S_n, u) - \{u, v\}$ for some edge (u, v) in S_n.
($\Rightarrow$) Suppose there is no (u, v) in S_n such that $F = A(S_n, v) \cup A(S_n, u) - \{u, v\}$. We want to prove that $S_n - F$ is connected. If no S_{n-1} has more than or equal to $n-2$ faults, $S_n - F$ is connected. And if only one S_{n-1} has more than or equal to $n-2$ faults, applying the proof technique of Theorem 1, we can show that $S_n - F$ is connected. So we suppose that there are two S_{n-1}'s having $n-2$ faults. Without

loss of generality, we assume the two S_{n-1}'s are S_{n-1}^1 and S_{n-1}^n. By Lemma 2, each of the two S_{n-1}'s is disconnected if and only if there exist nodes $u \in S_{n-1}^1$ and $v \in S_{n-1}^n$ such that $A(S_{n-1}^1, u)$ and $A(S_{n-1}^n, v)$ are all faulty. According to our assumption, (u, v) is not an edge in S_n. Thus, u or v can start from itself via a path, containing a node in $S_{n-1}^n - F - \{v\}$ or $S_{n-1}^1 - F - \{u\}$, to reach a node in $S_n - S_{n-1}^1 - S_{n-1}^n$. So $S_n - F$ is connected. This is a contradiction. We complete our proof. $\square$

By Lemma 2 and Lemma 3, if there are at most $2n - 4$ faulty nodes, only two forms of disconnection can occur: (1) An individual node is isolated. (2) A pair of connected adjacent nodes are isolated. Now we shall propose an algorithm to determine whether $S_n - F$ is connected when $|F| \leq 2n - 4$. In the algorithm, it is assumed that we have a list of faulty nodes.

Algorithm 1. Determine if S_n is connected or not when $|F| \leq 2n - 4$.

Case 1. $|F| < n - 1$. $S_n - F$ is connected.

Case 2. $(n-1) \leq |F| < (2n-4)$. Check if there exists a node $v \in S_n$ such that $v \notin F$ and $A(S_n, v)$ in S_n is a subset of F. If the answer is yes, then $S_n - F$ is disconnected; otherwise $S_n - F$ is connected.

Case 3. $|F| = 2n - 4$. Check if there exists an edge $(u, v) \in S_n$ such that $F = A(S_n, v) \cup A(S_n, u) - \{u, v\}$. If the answer is yes, then $S_n - F$ is disconnected; otherwise, $S_n - F$ is connected.

The detail of Case 3 in Algorithm 1 is as follows.

Algorithm 2. Algorithm of Case 3.

Step 3.1 Select an arbitrary node x from F.

Step 3.2 If F contains a node y whose distance to x is 3, then go to the next step. Otherwise return NO.

Step 3.3 Find one pair of nodes, say u and v, between x and y.

Step 3.4 If $F = A(S_n, v) \cup A(S_n, u) - \{u, v\}$, then return YES; otherwise return NO.

Now we will analyze the time complexity of Algorithm 1. In Case 2, there are at most $(n-1)|F|$ nodes neighboring to at least one faulty node. We can collect all neighbors of each faulty node. Then sort these neighbors. Now the same node will occupy consecutive locations in the sorted list if it appears more than once. If the number of appearance of a node is $n - 1$, then all of its neighbors are faulty. So Case 2 needs $O(((n-1)|F|) \log(n-1)|F|) = O(n^2 \log n)$ time. It is not difficult to see that Case 3 requires $O(n^2)$ time. Thus, the time complexity of Algorithm 1 is $O(n^2 \log n)$.

4. An Improved Fault-tolerant Routing Algorithm

Before proposing our algorithm, we need the following definition.

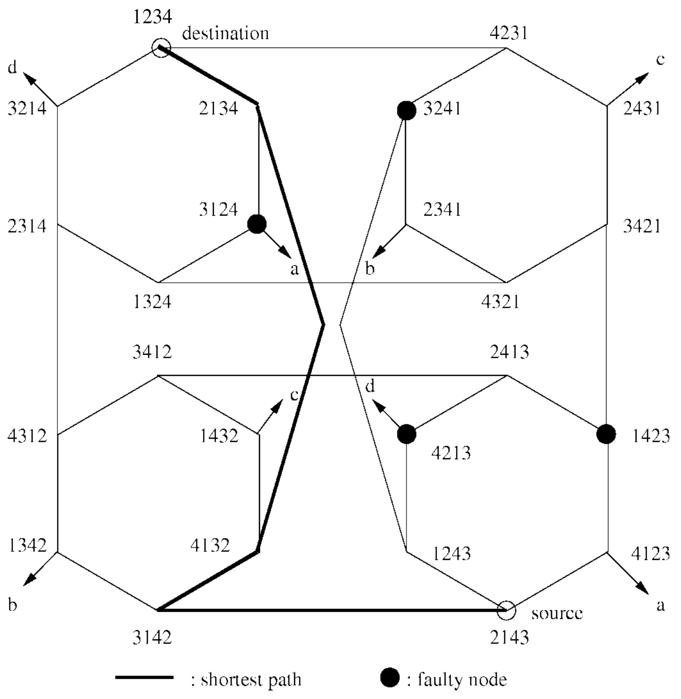

Figure 4: Examples of blocked nodes, 1243 and 4123.

Definition 2 *A node which has a copy of message is said to be* **blocked** *if it cannot forward message any further, that is, all of its neighboring nodes, except the one it received message from, is faulty.*

Examples of blocked nodes are shown in Figure 4.

In our fault-tolerant routing algorithm, there are two assumptions: (1) Faults are assumed to be in one or more nodes, and with slight modification, we can also take link failures into consideration. (2) A node only knows the status of its neighbors. The status includes the information which one is faulty and which one is nonfaulty. Each node also maintains the status of all neighbors of each of its neighbors. With this scheme, the blocked neighbors can be easily found out. Besides, without losing generality, we assume that the destination node is the identity node.

The fault-tolerant routing algorithm proposed by Bagherzadeh et al.[5] is based on the depth first search strategy. The major overhead of this kind of algorithm is backtracking. What we focus on is how to decrease backtracking under limited information. Consider Figure 4. The process of routing from 2143 to 1234, according to their algorithm, is shown in Table 1. It takes eight hops. However if we select link 4 first, it takes only 4 hops. In other words, the path with minimum length is selected. The extra hops are needed because of blocked nodes. We shall modify their algorithm to avoid routing along blocked nodes.

Besides, in their algorithm, whenever a node p' of S_n receives a copy of message from another node p, it has to compute the cycle structure of p'. To achieve this

Table 1: The process of routing from 2143 to 1234 in S_4.

node	cycle structure	link	comment
2143	(21)(43)	2	sent
1243	(43)(1)(2)	4	faulty
		3	faulty
backtracking			
2143	(21)(43)	3	sent
4123	(4321)	4	faulty
		2	faulty
backtracking			
2143	(21)(43)	4	sent
3142	(3421)	3	sent
4132	(421)(3)	4	sent
2134	(21)(3)(4)	2	sent
1234	(1)(2)(3)(4)		destination

purpose, we have to scan all n symbols of p'. In fact, the identifier of p' can be obtained from p by exchanging the first symbol and one selected symbol on another position. Note that if the selected symbol is j, it means that $p' = g_j(p)$. Thus, this can be done more efficiently if we calculate the cycle structure of p' on node p and send the cycle structure with the message from node p to node p'. Two cases have to be considered. In the first case, 1 is the first symbol of p, and in the second case, 1 is not in the first position of p. These two cases are as follows.

Case 1: The cycle structure of p is $(a_1\ a_2\ \cdots\ a_k)(1)(c)$

case 1.1: The selected symbol is $a_j,\ 1 \le j \le k$.
$p' = g_{a_j}(p) = (a_{j+1}\ a_{j+2}\ \cdots\ a_k\ a_1\ a_2\ \cdots\ a_j\ 1)(c)$

case 1.2: The selected symbol is already in its correct position, say c.
$p' = g_c(p) = (c\ 1)(a_1\ a_2\ \cdots\ a_k)$

Case 2: The cycle structure of p is $(a_1\ a_2\ \cdots\ a_r\ 1)(b_1\ b_2\ \cdots\ b_q)(c)$

case 2.1: The selected symbol is a_1.
$p' = g_{a_1}(p) = (a_2\ a_3\ \cdots\ a_r\ 1)(b_1\ b_2\ \cdots\ b_q)(c)(a_1)$

case 2.2: The selected symbol is in the first cycle, a_1 and 1. Without loss of generality, the selected symbol is assumed to be $a_j,\ 2 \le j \le r$.
$p' = g_{a_j}(p) = (a_{j+1}\ a_{j+2}\ \cdots\ a_r\ 1)(a_2\ a_3\ \cdots\ a_j\ a_1)(b_1\ b_2\ \cdots\ b_q)(c)$

case 2.3: The selected symbol is $b_j,\ 1 \le j \le q$.
$p' = g_{b_j}(p) = (b_{j+1}\ b_{j+2}\ \cdots\ b_q\ b_1\ b_2\ \cdots\ b_j\ a_1\ a_2\ \cdots\ a_r\ 1)(c)$

case 2.4: The selected symbol is already in its correct position, say c.
$p' = g_c(p) = (c\ a_1\ a_2\ \cdots\ a_r\ 1)(b_1\ b_2\ \cdots\ b_q)$

In our algorithm, the cycle structure of the next node is calculated by the current node and sent as a part of the message. The calculation requires only constant time. For the convenient representation, the selected node is denoted by p'. Our fault-tolerant routing algorithm is described as follows.

Algorithm 2. IDFR (Improved Depth-First Routing)

Step 1. If the current node is the destination, then stop.

Step 2. Select the leftmost symbol in i-cycles, $i \geq 2$. If p' is nonfaulty, nonvisited and nonblocked or p' is blocked and it is destination, then send the message to it and stop.

Step 3. Starting from the rightmost position of the rightmost i-cycle, $i \geq 2$, proceeding to the left, select each symbol in each i-cycle. If p' is nonfaulty, nonvisited and nonblocked, then send the message to it and stop. When no symbol remains, go to the next step.

Step 4. Starting from the rightmost position, proceeding to the left, select the symbol in each cycle of length 1. If p' is nonfaulty, nonvisited and nonblocked, then send the message

step 5 At this time, the destination cannot be reached from the current node. Send a return message along the link which it just received message from, and stop.

Our fault-tolerant routing algorithm can avoid entering blocked nodes. The improvement is especially significant when many blocked nodes are selected before a nonblocked neighbor in the algorithm proposed by Bagherzadeh et al. [5]. The penalty (extra routing steps) for their routing algorithm executed in an S_n with at most $n-2$ faults is $O(\sqrt{n})$. And, the penalty of our algorithm is less than that of their algorithm.

5. A Fault-tolerant Broadcasting Algorithm

In the first phase of the broadcasting algorithm on fault-free star graphs proposed by Mendia et al.[9], the relay nodes are embedded on a tree, which is called the *relay tree*. By preserving the leftmost symbol of each node and omitting the other symbols, the relay tree of an S_{17} is shown in Figure 5(a). In the relay tree, a node labeled as i is one node whose leftmost symbol is the same as the i-th symbol of the source node. Recall that in the second phase of the broadcasting algorithm, for a relay node v in an S_n, it should send the broadcast message to node $g_n(v)$. Thus, each sub-star S_{n-1} has a copy of the message in one of the nodes. To achieve this goal, the relay tree of an S_n must consist of $n-1$ nodes which are labeled from 1 to $n-1$. Since the root of the relay tree represents the source node, the label of the root is always 1. In the first phase of the broadcasting algorithm[9], the relay tree is constructed as follows. After a node v has received the message from node $g_l(v)$, it sends the message to the node $g_{l+2^{i-1}}(v)$ at step i if $l + 2^{i-1} \leq 2^{\lceil log(n-1) \rceil}$ and

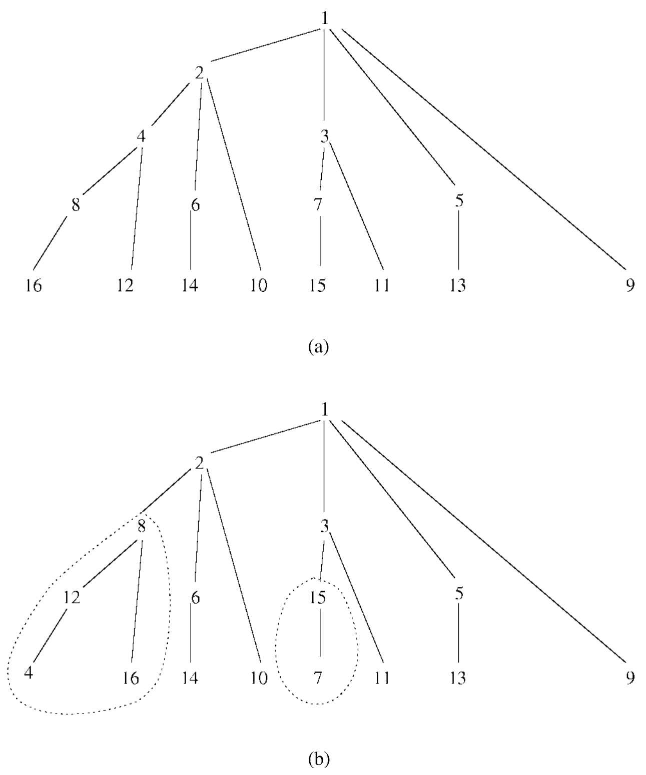

Figure 5: Relay trees. (a) A relay tree of S_{17}. (b) A relay tree obtained from (a) after substituting 4 by 8 and 7 by 15.

$1 \leq i \leq \lceil log(n-1) \rceil$, where the step counter is counted with the beginning step of the root being 1.

To present our broadcasting algorithm, we have to define some terms. A node v in an S_n is said to be *valid* if (1) it is not faulty, (2) $g_n(v)$ is not faulty and (3) at least one of its descendants in the relay tree are nonfaulty. A node is *semi-valid* if it only satisfies the first two conditions. And a node is *invalid* if otherwise.

In fact, the relay tree is not unique. In other words, there are many ways for constructing a relay tree. In our mechanism, when there exists one or more faults, a faulty node can be substituted by any one of its descendants in the tree. After the substitution, for a node v, we can also perform a similar calculation to obtain its next receiver. After node v has received the broadcast message from node $g_l(v)$, it will send the message at step i, $1 \leq i \leq \lceil log(n-1) \rceil$, along the link $(g_{(l+2^{i-1}) mod (2^{\lceil log(n-1) \rceil})})$ if the neighbor is valid. Otherwise we will do the substitution again. For example, in Figure 5(a), if the node labeled by 2 is faulty, any even number smaller than 16 can substitute 2; if the node labeled by 3 is faulty, it can be substituted by 7,11 and 15. Figure 5(b) is the relay tree obtained from Figure 5(a) when nodes labeled by 4 and 7 are faulty. Figure 6 shows an example for relay trees of an S_n where $n-1$ is not a power of 2. It is not hard to see that if

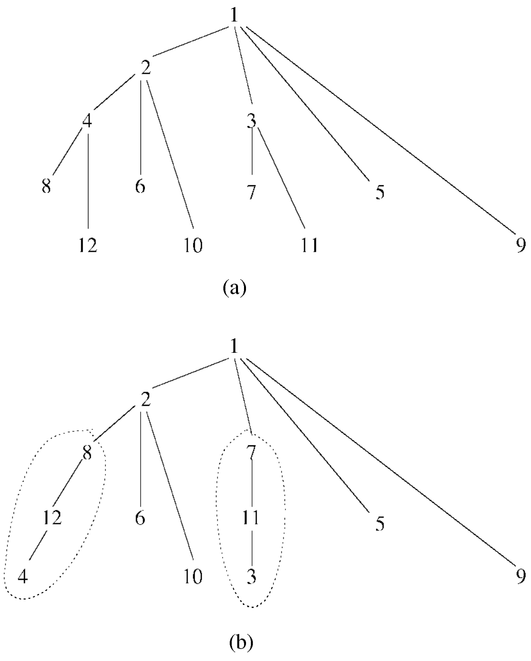

Figure 6: Relay trees. (a) A relay tree of S_{13}. (b) A relay tree obtained from (a) after substituting 4 by 8 and 3 by 7.

the node labeled by i is faulty, after the substitution, the node labeled by i of the new relay tree must be a different node in the S_n.

Nevertheless, the substitution may not work. For example, if a leaf node is invalid, there is no substitution can be done since the node has no descendants in the relay tree. Under this situation, we have to do backtracking. For example, in Figure 5(b), if the node represented by 4 is invalid, the node labeled by 12 will send a backtracking message to its parent node. If the parent node, labeled by 8, has a valid or semi-valid neighbor on link 4, it sends the broadcast message to that node. Otherwise, it sends the backtracking message upward. The same skill is recursively applied until the root node is reached. Besides, when the node labeled by 12 sends a backtracking message, it also informs the node labeled by 8 that it has no valid or semi-valid descendants. Thus, in the next recursion (the broadcasting in an S_{16}), the node labeled by 8 will try to send message along link 4 first.

After the backtracking has been done, the root node will has the collection of all absent symbols if we still cannot build a complete relay tree. Note that a complete relay tree of an S_n should have $n-1$ nodes labeled with 1 through $n-1$. For this situation, the root node has to route the broadcast message to other nodes. Let u be the root node and i be one of the absent symbols, u will route message, via a longer and fault-free path, to $v(= g_n g_i(u))$, which is the closest node to u in S^i_{n-1} and then becomes the leader node in S^i_{n-1}. Note that it is assumed that u knows the status of the neighbors of $g_i(u)$ since $g_i(u)$ is a neighbor of u. The routing to several destinations can be done in pipeline. If v is faulty, we will select a neighbor of v, says v', which is in the same smallest sub-star with v. The reason why we select such a v' is that, in this way, we can localize the influence of v. Note that v' becomes the leader node of S^i_{n-1} if v' is nonfaulty. If u cannot reach v', u will route message to $g_{n-1}g_2(v'), g_{n-1}g_3(v'), \cdots, g_{n-1}g_{n-2}(v'), g_{n-1}(v')$ and a neighboring node of v' in the same S_{n-2} in the next recursion. That is, u will take the responsibility of v' in S^i_{n-1} (play the role of the leader of S^i_{n-1}), and route message to exactly one node in each S_{n-2} of S^i_{n-1}. Our algorithm for broadcasting on an S_n can be divided into two phases as follows.

Algorithm FTB (Fault-Tolerant Broadcasting)

Phase 1 :

1.1 If the source (leader) node has only one nonfaulty neighbor then select the nonfaulty neighbor as a new source (leader) node.

1.2 Construct a relay tree. If an invalid node is encountered, try to do a substitution if possible.

1.3 If there is still one or more nodes cannot be reached, keep the absent symbols.

Phase 2 :

2.1 Each node which has collected some absent symbols, sends the absent symbols to its parent. When a node receives such a backtracking message from its son, if it has a neighboring node whose leftmost symbol is the same as one of the absent symbols, forward the message to the neighboring node and remove this absent symbol.

2.2 Each relay node forwards the message to a unique sub-star. If the leader, say v, of a sub-star is faulty, the corresponding relay node routes the message to $g_k(v)$, where $k = min\{i | g_i(v) \text{ is not faulty }\}$.

2.3 For every absent symbol i, the source (leader) node u routes the message to node $v = g_n g_i(u)$ by using Algorithm IDFR. If v is faulty, try to route the message to a neighbor v' of v. If the selected node v' cannot be reached, the source (leader) node routes message to nodes $g_{n-1}g_2(v'), g_{n-1}g_3(v'), \cdots, g_{n-1}g_{n-2}(v'), g_{n-1}(v')$ and a neighboring node of v' in the same S_{n-2} in the next recursion.

For simplifying our description, the numbering of recursions is counted down. The starting recursion is the n-th recursion, which is for S_n to distribute the message to each S_{n-1}. The next recursion is the $(n-1)$-th recursion.

An example for illustrating our FTB is shown in Figure 7, in which the source node is 12345. In the first phase of the fifth recursion, since $21345 = g_2(12345)$ is faulty, it is substituted by $42315 = g_4(12345)$. In the second phase, applying g_5, each S_4 has a unique leader node. We omit the further progress in the

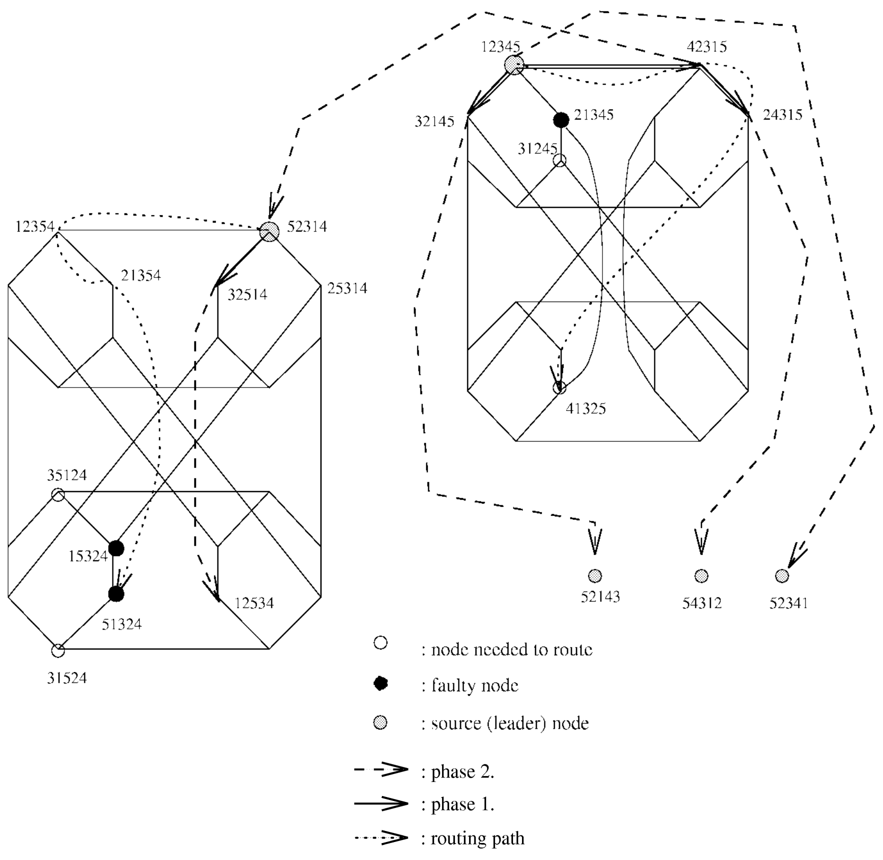

Figure 7: An example of FTB on S_5.

fault-free $S_4's$. In the fourth recursion, 12345 is the leader node of S_4^5. Since 21345 is faulty and no substitution can be done, 12345 needs to route message to $g_4 g_2(12345) = 41325$. Since the leader node of S_4^4 is 52314, and the neighbor on dimension 4 of 25314 is faulty, 52314 should route data to one neighbor of 15324, say $g_2 g_4 g_2(52314) = 51324$. In the 3rd recursion, since 51324 is faulty, 52314 should route to $\{g_3 g_2(51324), g_3(51324)\} = \{35124, 31524\}$. Again, since 21345 is faulty, 12345 should route data to $g_3(21345) = 31245$.

If there is no faulty node in the star graph, the action of FTB is the same as that of the optimal broadcasting algorithm on fault-free star graphs. Thus, under this condition, FTB is optimal.

Theorem 2 *In an S_n, if there is only one faulty node, FTB has penalty at most $2n+2$.*

Proof. We first consider the case: $n = 2^m + 1$ for some positive integer m.

Recall the relay tree constructed in phase 1 of FTB. The more descendants the faulty node has, the less penalty will be caused by it. Note that the only node which cannot be substituted and cannot do backtracking is the one with leftmost symbol $\frac{n+1}{2}$, denoted as v, which is the rightmost leaf. If v is the faulty node, the source node have to route the message to $g_n(v)$ in this recursion. Thus, in general, in the j-th recursion (broadcasting in an S_j), $\frac{n+3}{2} \leq j \leq n$, since v is the rightmost leaf in the relay tree, we have to route the broadcast message from the source to $g_j(v)$. And in the $(\frac{n+1}{2})$-th recursion, since v is the leader of a sub-star, we have to route the message to $g_2(v)$. The distances from the source to these nodes are all 2. However, v is on the shortest paths from the source to these nodes, and a smallest cycle in the star graph consists of six nodes. Hence, routing to these nodes, which is done in Step 2.3, needs 4 hops. So the total penalty is

$$4(\tfrac{n+1}{2}) = 2n + 2$$

For the case when n is between $2^m + 2$ and $2^{m+1} + 1$ for some positive integer m, the penalty is no more than $2n + 2$. The proof is similar. □

Theorem 3 *In an S_n, if the number of faults is at most $n-2$, FTB has penalty $O(n^2)$.*

Proof. Let r be the total number of nodes to which we should route data from the source node. Let d be the maximum distance from the source (leader) node to the destination for routing and p be the maximum number of penalty hops for routing one data element. Since the routing is performed in pipeline, total routing time needed in Step 2.3 is $O(r + n(d+p))$. And the routing is an extra work to broadcasting. Thus, the total penalty of FTB is $O(r + n(d+p))$.

Our IDFR has penalty at most $O(\sqrt{n})$, so $p = O(\sqrt{n})$. The distance between two nodes in S_n is $O(n)$. Thus $d = O(n)$. Since there are at most $n-2$ faults, we want to find out the upper bound of r. Let u denote the source (leader) node. Suppose i is one of the absent symbols such that $g_i(u)$ and $g_n g_i(u)$ are faulty. In the n-th recursion, u originally has to route data to $g_n g_i(u)$. However, u has to route data to $g_2 g_n g_i(u)$ since $g_n g_i(u)$ is faulty. Suppose $g_2 g_n g_i(u)$ is also faulty. In the $(n-1)$-th recursion, u has to broadcast the message to $n-1$ leader nodes by our IDFR. With this argument, u has to route data to $O(n)$ nodes when there exits one faulty node. It follows that $r = O(n^2)$ since there are at most $n-2$ faults. Thus, the total penalty is $O(n^2)$. □

6. Concluding Remarks

In this paper, we introduced a generalized fault tolerance measure, the forbidden faulty set, for star graphs. The forbidden faulty set on an S_n is the set of $n-1$ neighbors of a node in the S_n. We proved that the fault tolerance of an S_n is $2n-5$ with restriction to the forbidden faulty set. We also proposed an algorithm for determining whether an S_n is connected if a set of $2n-4$ faults is given. The algorithm requires $O(n^2 \log n)$ time.

Data routing and broadcasting are two essential and important issues in multiprocessor systems. We improved the fault-tolerant routing algorithm proposed by Bagherzadeh et al.[5]. We also proposed a mechanism to generate a fault-tolerant relay tree when faults are encountered. In this way, we can localize the influence of faulty nodes. Based on that, we proposed an efficient broadcasting algorithm on faulty star graphs. Our algorithm remains optimal when there is no fault. The penalty is $O(n)$ if there exists only one fault, and the total penalty is $O(n^2)$ if there exist at most $n-2$ faults.

Although the star graph structure has been proposed for a couple of years. There are still a lot of open problems in this field. For instance, up to now, there are no optimal all-to-all broadcasting algorithm, optimal parallel sorting algorithm, optimal FFT algorithm and the fault-tolerant aspect of these problems on star graphs.

References

1. S. B. Akers, D. Harel, and B. Krishnamurthy, "On group graphs and their fault tolerance," *IEEE Trans. Computers*, Vol. C-36, pp. 885–888, July 1987.
2. S. B. Akers, D. Harel, and B. Krishnamurthy, "The star graph: An attractive alternative to the n-cube," *Proc. of the Int'l Conference on Parallel Processing*, pp. 393–400, 1987.
3. S. B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Trans. Computers*, Vol. 38, pp. 555–566, Apr. 1989.
4. B. Alspach, "Cayley graphs with optimal fault tolerance," *IEEE Trans. Computers*, Vol. 41, No. 10, pp. 1337–1339, Oct. 1992.
5. N. Bagherzadeh, N. Nassif, and S. Latifi, "A routing and broadcasting scheme on faulty star graphs," *IEEE Trans. Computers*, Vol. 42, No. 11, pp. 1398–1403, Nov. 1993.
6. D. M. Blough and N. Bagherzadeh, "Near-optimal message routing and broadcasting in faulty hypercubes," *Int. J. Parallel Programming*, Vol. 19, No. 5, pp. 405–423, 1990.
7. K. Day and A. Tripathi, "A comparative study of topological properties of hypercubes and star graphs," *IEEE Trans. Parallel and Distributed Systems*, Vol. 5, No. 1, pp. 31–38, Jan. 1994.
8. A. H. Esfahanian, "Generalized measures of fault tolerance with application to n-cube networks," *IEEE Trans. Computers*, Vol. 38, No. 11, pp. 1586–1591, Nov. 1989.
9. V. E. Mendia and D. Sarkar, "Optimal broadcasting on the star graph," *IEEE Trans. Parallel and Distributed Systems*, Vol. 3, No. 4, pp. 389–396, Jul. 1992.
10. A. Menn and A. K. Somani, "An efficient sorting algorithm for the star graph interconnection network," *Proc. of the Int'l Conference on Parallel Processing*, pp. 1–8, 1990.
11. S. Sur and P. K. Srimani, "A fault tolerant routing algorithm in star graph interconnection networks," *Proc. of the Int'l Conference on Parallel Processing*, Vol. 3, pp. 267–270, 1991.