JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 34, 1353-1366 (2018)
DOI: 10.6688/JISE.201811_34(6).0001

Fault-Tolerant Routing Based on Routing Capabilities in a Hyper-Star Graph

YO NISHIYAMA[1], YUKO SASAKI[1], YUKI HIRAI[2], HIRONORI NAKAJO[1] AND KEIICHI KANEKO[1,+]
[1]*Graduate School of Engineering*
Tokyo University of Agriculture and Technology
Koganei-shi, Tokyo 184-8588, Japan
[2]*Adimissons Center*
Shinshu University
Matsumoto-shi, Nagano 390-8621, Japan
[+]*E-mail: k1kaneko@cc.tuat.ac.jp*

A hyper-star graph $HS(n, k)$ provides a promising topology for interconnection networks of parallel processing systems because it inherits the advantages of a hypercube and a star graph. In this paper, we focus on fault-tolerant routing in an $HS(n, k)$ graph with faulty nodes and propose an algorithm to establish a fault-free path between a pair of non-faulty nodes. The algorithm uses limited global information called routing capabilities. Though routing capabilities were originally invented for a hypercube, we extend their notion so that they can be applied to an $HS(n, k)$ graph, which is asymmetric. We have proved that the time complexity to calculate routing capabilities with respect to all the distances at each node is $O(n^2)$. In addition, we present the results of a computer experiment to verify that our algorithm attains high reachability to the destination nodes.

Keywords: interconnection network, parallel processing, hypercube, star graph, faulty node

1. INTRODUCTION

Recent research has proposed many novel topologies for interconnection networks [1-7] as alternatives for simple topologies such as a ring, a mesh, a torus, a hypercube [8], and so on. A hyper-star graph $HS(n, k)$ [9] provides a such new topology. Since a hyper-star graph and its variant, a folded hyper-star graph, inherit advantages of an n-dimensional hypercube and an (n, k)-star graph, they have useful properties. For example, each node in the graph is represented by a simple n-bit sequence instead of a permutation, and the graph can interconnect an arbitrary number of nodes making it incrementally expandable. Hence, they have been studied extensively [10-18]. In a massively parallel system, algorithms should be designed and developed presuming the existence of faulty elements. Another advantage of the hyper-star graph is that its useful properties are generally kept even if there are some faulty elements in the system. Hence, we focus in this paper on fault-tolerant routing in a hyper-star graph $HS(n, k)$ with faulty nodes, and propose an adaptive fault-tolerant routing algorithm between non-faulty nodes.

Optimal fault-tolerant routing is possible if each non-faulty node collects information of all faulty nodes in the graph as global information. However, this approach is impractical because it requires space and time complexities whose orders are equal to the

Received March 9, 2017; revised June 29, 2017; accepted July 26, 2017.
Communicated by Xiaohong Jiang.

number of nodes in the graph. On the other hand, if every non-faulty node collects the status of its neighbor nodes only as local information for fault-tolerant routing, high reachability with shorter paths cannot be expected. Hence, many approaches collect the compressed global information to attain a high reachability with shorter paths. The information is referred to as limited global information.

In the fault-tolerant routing based on the limited global information, to avoid the message flooding, each non-faulty node exchanges fault information with its neighbor nodes synchronously at the power-on state. The amount of faulty information and the number of exchanges are, respectively, restricted to be $O(1)$ and a polynomial order of the degree. Each non-faulty node stores limited global information regarding its neighbor nodes and uses it for fault-tolerant routing. However, the fault-tolerant routing algorithm may cause infinite loops. Therefore, it is necessary to keep track of the number of misroutings and/or hops in each message to detect infinite loops. If a message cannot reach at the destination node because of an infinite loop, the routing algorithm is switched to the depth-first search (for example), which needs longer paths but ensures arrival to the destination node. If the message cannot reach at the destination node even by the depth-first search, the source node can conclude that the destination node is not connected.

For a hypercube, there are several fault-tolerant routing algorithms based on the limited global information. Chiu and Wu have proposed an efficient fault-tolerant routing algorithm [19] by recursively classifying non-faulty nodes into safe, ordinary unsafe, and strongly unsafe nodes depending on the classification of neighbor nodes. To improve the algorithm by Chiu and Wu, Chiu and Chen introduced the routing capabilities obtained by classifying the safety nodes with respect to the Hamming distances to the destination nodes [20]. Wu has also proposed a similar fault-tolerant routing algorithm independently by introducing safety vectors [21]. Moreover, Kaneko and Ito have proposed a fault-tolerant routing algorithm based on classification of ordinary and strongly unsafe nodes with respect to the Hamming distances as well as an efficient method to obtain their classification [22].

For fault-tolerant routing in a star graph, Yeh *et al.* have proposed an algorithm based on safety vectors to attain efficient fault-tolerant routing [23]. Since the routing in a star graph is more complicated than that in a hypercube, the safety vectors on a star graph are based on routing patterns while those on a hypercube are based on distances.

Though routing capabilities were originally invented for a hypercube [20], we extend their notion in this paper to attain fault-tolerant routings of high reachability with shorter paths in a generic non-regular hyper-star graph. For this purpose, we investigate new relationship among the number of neighbor nodes, the distance to the destination node, and the number of neighbor nodes on the shortest paths to the destination in a hyper-star graph. We explain the details of routing capabilities and discuss the statistical results of the computer experiments.

The rest of this paper is structured as follows. First, in Section 2, a definition of a hyper-star graph is introduced with an example, and some properties are proved. Next, in Section 3, we introduce a definition of routing capabilities and describe the fault-tolerant routing algorithm based on them. Then, in Section 4, we experimentally verify the effectiveness of our algorithm. Finally, in Section 5, we present conclusions and a suggestion for future research.

2. PRELIMINARIES

In this section, we give a definition of a hyper-star graph and lemmas about its properties.

Definition 1: (hyper-star graph $HS(n, k)$) In an $HS(n, k)$ graph, each node is an n-bit sequence that contains k '1' bits. There is an edge between two nodes, $\boldsymbol{a}$ and $\boldsymbol{b}$ ($\boldsymbol{a}, \boldsymbol{b} \in V(HS(n, k))$), if and only if $a_1 \neq b_1$, $a_1 = b_i$, $a_i = b_1$ ($i \geq 2$) and $a_j = b_j$ ($2 \leq j (\neq i) \leq n$). ❑

Fig. 1 shows an example of $HS(6, 2)$ graph where a node $(a_1, a_2, \ldots, a_6)$ is abbreviated into $a_1\ a_2 \ldots a_6$. Each node consists of 6 bits where exactly 2 bits of them are equal to 1 and the remaining 4 bits are equal to 0. For example, the neighbor nodes of the node (1, 1, 0, 0, 0, 0) are obtained by exchange the first bit 1 with one of the other bits 0: (0, 1, 1, 0, 0, 0), (0, 1, 0, 1, 0, 0), (0, 1, 0, 0, 1, 0), and (0, 1, 0, 0, 0, 1). On the other hand, the neighbor nodes of the node (0, 0, 1, 0, 0, 1) are (1, 0, 0, 0, 0, 1) and (1, 0, 1, 0, 0, 0). Hence, an $HS(6, 2)$ graph is not regular.

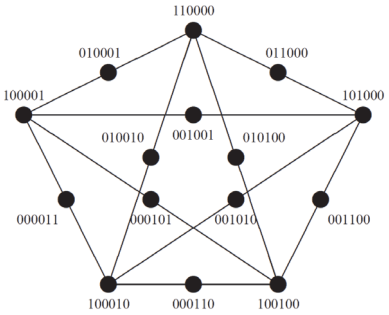

Fig. 1. An example of a hyper-star graph $HS(6, 2)$.

Table 1. Comparison of a hyper-star graph $HS(n, k)$ with a hypercube Q_n, a pancake graph P_n, and a star graph S_n.

topology	#nodes	degree	connect.	diameter
$HS(n, k)$ $(n < 2k)$	${}_nC_k$	$n-k$	$n-k$	$2(n-k)$
$HS(n, k)$ $(n = 2k)$	${}_nC_k$	$k = n-k$	$k = n-k$	$2k-1$
$HS(n, k)$ $(n > 2k)$	${}_nC_k$	k	k	$2k$
Q_n	2^n	n	n	n
P_n	$n!$	$n-1$	$n-1$	†
S_n	$n!$	$n-1$	$n-1$	$\lfloor 3(n-1) / 2 \rfloor$

†: $\leq \lfloor 5(n + 1) / 3 \rfloor$ from [24]

Table 1 shows comparison of a hyper-star graph $HS(n, k)$ with a hypercube Q_n, a pancake graph P_n, and a star graph S_n. The cost of a topology is defined by the product of its degree and its diameter. Let us compare the four topologies: an S_{20}, a P_{20}, an $HS(66, 27)$, and a Q_{61} having a similar number of nodes. The number of nodes of an S_{20}, a P_{20}, an $HS(66, 27)$, and a Q_{61} are 2.43×10^{18}, 2.43×10^{18}, 2.45×10^{18}, and 2.31×10^{18} and the costs are, respectively, 532, $\leq$665, 1458, and 3721. Hence, a star graph and a pancake

graph are the best topologies with costs, a hyper-star graph is the second best, and a hypercube is the worst. However, a shortest-path routing algorithm of polynomial time of n is not known so far with a P_n. Though an S_n has a shortest-path routing algorithm, it is much more complicated than that for an $HS(n, k)$ graph. Also, an $HS(n, k)$ graph has higher incremental expandability than other topologies. That is, it can connect an arbitrary number of nodes to implement interconnection networks.

For two nodes $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ in an $HS(n, k)$ graph, the distance between them $dist(\boldsymbol{a}, \boldsymbol{b})$ is given by $\sum_{i=2}^{n} a_i \oplus b_i$ where the operator $\oplus$ represents an exclusive logical-or operation. In addition, let $N(\boldsymbol{a})$ be the neighbor nodes of $\boldsymbol{a}$. Then, let $Pre(\boldsymbol{a}, \boldsymbol{b}) = \{\boldsymbol{nbr} \mid \boldsymbol{nbr} \in N(\boldsymbol{a}), dist(\boldsymbol{nbr}, \boldsymbol{b}) = dist(\boldsymbol{a}, \boldsymbol{b}) - 1\}$ and $Spr(\boldsymbol{a}, \boldsymbol{b}) = \{\boldsymbol{nbr} \mid \boldsymbol{nbr} \in N(\boldsymbol{a}), dist(\boldsymbol{nbr}, \boldsymbol{b}) = dist(\boldsymbol{a}, \boldsymbol{b}) + 1\}$ be the neighbor nodes that are on the shortest paths from $\boldsymbol{a}$ to $\boldsymbol{b}$ and those on the detour paths from $\boldsymbol{a}$ to $\boldsymbol{b}$, respectively. We call $Pre(\boldsymbol{a}, \boldsymbol{b})$ and $Spr(\boldsymbol{a}, \boldsymbol{b})$ as the preferred neighbor nodes of $\boldsymbol{a}$ to $\boldsymbol{b}$ and the spare neighbor nodes of $\boldsymbol{a}$ to $\boldsymbol{b}$, respectively.

Lemma 1: Let $diam(HS(n, k))$ be the diameter of an $HS(n, k)$ graph. Then, it is given as follows:

$$diam(HS(n,k)) = \begin{cases} 2(n-k) & (n < 2k) \\ 2k-1 & (n = 2k) \\ 2k & (n > 2k) \end{cases}.$$

Proof: Consider two nodes $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ in an $HS(n, k)$. Then, if $n < 2k$, the distance between the nodes, $dist(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=2}^{n} a_i \oplus b_i$, is at most $2(n - k)$. This is attained by, for example, $a_1 = a_2 = \ldots = a_k = 1$, $a_{k+1} = \ldots = a_n = 0$, $b_1 = 1$, $b_2 = b_3 = \ldots = b_{n-k+1} = 0$, $b_{n-k+2} = \ldots = b_n = 1$. Hence, the diameter is $2(n - k)$. From symmetric property, if $n > 2k$, the diameter is equal to $2k$. If $n = 2k$, the distance between the nodes is at most $n - 1 = 2k - 1$, and it is attainable by, for example, $a_1 = a_2 = \ldots = a_k = 1$, $a_{k+1} = \ldots = a_n = 0$, $b_1 = b_2 = \ldots = b_k = 0$, $b_{k+1} = \ldots = b_n = 1$. ❑

Lemma 2: For a node $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ in an $HS(n, k)$ graph, the number of neighbor nodes of $\boldsymbol{a}$, $|N(\boldsymbol{a})|$, is equal to k if $a_1 = 0$ while it is equal to $n - k$ if $a_1 = 1$.

Proof: There are k 1's in $a_2, a_3, \ldots, a_n$ if $a_1 = 0$. Hence, $|N(\boldsymbol{a})| = k$. Otherwise, if $a_1 = 1$, there are $(n - k)$ 0's in $a_2, a_3, \ldots, a_n$. Therefore, $|N(\boldsymbol{a})| = n - k$. ❑

Lemma 3: For two distinct nodes $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ in an $HS(n, k)$ graph, $|Pre(\boldsymbol{a}, \boldsymbol{b})| = \lceil d/2 \rceil$ and $|Spr(\boldsymbol{a}, \boldsymbol{b})| = |N(\boldsymbol{a})| - \lceil d/2 \rceil$ where $d = dist(\boldsymbol{a}, \boldsymbol{b})$.

***Proof*:** If $d = \sum_{i=2}^{n} a_i \oplus b_i$ is even, $a_1 = b_1$ because $\sum_{i=1}^{n} a_i \oplus b_i$ is always even. Therefore, among d bits of $a_2, a_3, \ldots, a_n$ that are different from corresponding $b_2, b_3, \ldots, b_n$, $d/2$ bits are equal to $\overline{a}_1$. Thus, $|Pre(\boldsymbol{a}, \boldsymbol{b})| = d/2$, and $|Spr(\boldsymbol{a}, \boldsymbol{b})| = |N(\boldsymbol{a})| - d/2$. On the other hand, if d is odd, $a_1 = \overline{b}_1$. Therefore, among d bits of $a_2, a_3, \ldots, a_n$ that are different from corresponding $b_2, b_3, \ldots, b_n$, $(d + 1)/2$ bits are equal to $\overline{a}_1$. Hence, $|Pre(\boldsymbol{a}, \boldsymbol{b})| = (d + 1)/2$ and $|Spr(\boldsymbol{a}, \boldsymbol{b})| = |N(\boldsymbol{a})| - (d + 1)/2$. To recap, $|Pre(\boldsymbol{a}, \boldsymbol{b})| = \lceil d/2 \rceil$ and $|Spr(\boldsymbol{a}, \boldsymbol{b})| = |N(\boldsymbol{a})| - \lceil d/2 \rceil$. ❑

3. ROUTING CAPABILITIES AND FAULT-TOLERANT ROUTING ALGORITHM

For a node $\boldsymbol{a}$ in an $HS(n, k)$ graph and a distance d ($1 \le d \le diam(HS(n, k))$), we introduce a routing capability $R_d(\boldsymbol{a})(\in \{0, 1\})$. For any non-faulty node which is located with distance d from the node $\boldsymbol{a}$, $R_d(\boldsymbol{a}) = 1$ ensures that a fault-free path of length d from $\boldsymbol{a}$ to the node can be established.

Definition 2: For a node $\boldsymbol{a}$ in an $HS(n, k)$ graph with a set of faulty nodes F, a routing capability $R_d(\boldsymbol{a})$ with respect to a distance d is recursively defined as follows:

1. $R_d(\boldsymbol{a}) = 1$ if $\boldsymbol{a} \notin F$, $d = 1$,
2. $R_d(\boldsymbol{a}) = 1$ if $\boldsymbol{a} \notin F$, $d \ge 2$, and for any $J(\subset N(\boldsymbol{a}))$ such that $|J|=\lceil d / 2 \rceil$, there exists $\boldsymbol{nbr}(\in J)$ such that $R_{d-1}(\boldsymbol{nbr}) = 1$,
3. $R_d(\boldsymbol{a}) = 0$ otherwise. ❑

Note that routing capabilities $R_d(\boldsymbol{a})$ are defined from $d = 1$ to $d = diam(HS(n, k))$.

Table 2 shows an example of routing capabilities in an $HS(6, 2)$ graph with a faulty node set $F = \{(0, 0, 0, 0, 1, 1), (0, 1, 0, 0, 1, 0), (1, 1, 0, 0, 0, 0)\}$. Note that routing capabilities do not monotonically decrease with the distance.

Table 2. Routing capabilities $R_i(a)$ ($1 \le i \le 4$) of a node a in $HS(6, 2)$ with 3 faulty nodes, which can be identified by $R_1 = 0$.

Node $\boldsymbol{a}$	R_1	R_2	R_3	R_4
(0, 0, 0, 0, 1, 1)	0	0	0	0
(0, 0, 0, 1, 0, 1)	1	1	1	1
(0, 0, 0, 1, 1, 0)	1	1	1	1
(0, 0, 1, 0, 0, 1)	1	1	1	1
(0, 0, 1, 0, 1, 0)	1	1	1	1
(0, 0, 1, 1, 0, 0)	1	1	1	1
(0, 1, 0, 0, 0, 1)	1	0	0	0
(0, 1, 0, 0, 1, 0)	0	0	0	0
(0, 1, 0, 1, 0, 0)	1	0	1	1
(0, 1, 1, 0, 0, 0)	1	0	1	1
(1, 0, 0, 0, 0, 1)	1	0	0	0
(1, 0, 0, 0, 1, 0)	1	0	0	0
(1, 0, 0, 1, 0, 0)	1	1	1	1
(1, 0, 1, 0, 0, 0)	1	1	1	1
(1, 1, 0, 0, 0, 0)	0	0	0	0

If $n = 2k$, $\sum_{d=1}^{2k-1} {}_kC_{\lceil d/2 \rceil} = O(2^k)$ holds. Similarly, if n is close to $2k$, it takes an exponential time to calculate routing capabilities at each node based on Definition 2. Hence, we introduce a simple calculation method based on the following lemma.

Lemma 4: In an $HS(n, k)$ graph with a set of faulty nodes F, for $\boldsymbol{a}$ node $\boldsymbol{a}(\notin F)$ and a distance $d(\ge 2)$, the following two conditions are equivalent:

1. For any $J(\subset N(\boldsymbol{a}))$ such that $|J| = \lceil d/2 \rceil$, there exists a node $\boldsymbol{nbr}(\in J)$ such that $R_{d-1}(\boldsymbol{nbr}) = 1$.
2. $|\{\boldsymbol{nbr} \mid \boldsymbol{nbr} \in N(\boldsymbol{a}), R_{d-1}(\boldsymbol{nbr}) = 1\}| \geq |N(\boldsymbol{a})| - \lceil d/2 \rceil + 1$.

***Proof*:** Because $|J|=\lceil d/2 \rceil$, $|N(\boldsymbol{a}) \setminus J| = |N(\boldsymbol{a})| - \lceil d/2 \rceil$ holds. Therefore, Condition 2 implies Condition 1. Hence, sufficiency is proved. For necessity, we assume that Condition 2 does not hold. Then, from $|\{\boldsymbol{nbr} \mid \boldsymbol{nbr} \in N(\boldsymbol{a}), R_{d-1}(\boldsymbol{nbr}) = 1\}| \leq |N(\boldsymbol{a})| - \lceil d/2 \rceil$, there exists $J(\subset N(\boldsymbol{a}))$ such that $|J| = \lceil d/2 \rceil$ and $\{\boldsymbol{nbr} \mid \boldsymbol{nbr} \in N(\boldsymbol{a}), R_{d-1}(\boldsymbol{nbr}) = 1\} \subset N(\boldsymbol{a}) \setminus J$. Hence, Condition 1 does not hold, either. Necessity is also proved. The lemma is proved from the above discussion. ❑

In an $HS(n, k)$ graph with a set of faulty nodes F, for a non-faulty node $\boldsymbol{a}$ and a distance d, from Lemma 4, we can compare $\sum_{\boldsymbol{nbr} \in N(\boldsymbol{a})} R_{d-1}(\boldsymbol{nbr})$ with $|N(\boldsymbol{a})| - \lceil d/2 \rceil + 1$ to judge sufficiency of Condition 2 in Definition 2. Fig. 2 shows an algorithm as Procedure **RC** to calculate routing capabilities $R_d(\boldsymbol{a})$ $(1 \leq d \leq diam(HS(n, k)))$ at a node $\boldsymbol{a}$. The procedure must be executed in sync at all the nodes.

```
procedure RC(a, F)
begin
  for d := 1 to diam(HS(n, k)) do
    if a ∈ F then R_d(a) := 0
    else if d = 1 then R_d(a) := 1
    else if Σ_{nbr∈N(a)} R_{d-1}(nbr) >= |N(a)|−⌈d/2⌉+1 then R_d(a) := 1
    else R_d(a) := 0
end
```

Fig. 2. Algorithm to calculate routing capabilities.

Theorem 1: At each node in an $HS(n, k)$ graph with a set of faulty nodes F, the time complexity to calculate routing capabilities with respect to all distances d $(1 \leq d \leq diam(HS(n, k)))$ is $O(n^2)$.

***Proof*:** From Lemma 4, to calculate a routing capability $R_d(\boldsymbol{a})$ with respect to a distance $d(\geq 2)$ at a node $\boldsymbol{a}$, it is necessary to collect $R_{d-1}(\boldsymbol{nbr})$ from each node $\boldsymbol{nbr}$ in neighbor nodes $N(\boldsymbol{a})$ of $\boldsymbol{a}$ and sum up them. This process requires $O(n)$ time complexity. Therefore, it takes $O(n^2)$ time in total to calculate routing capabilities for all distances d $(2 \leq d \leq diam(HS(n, k)))$. ❑

From Theorem 1, it takes $O(n^2)$ time to calculate the routing capabilities with respect to all distances at each node. If all the nodes calculate the routing capabilities in sync, the total time complexity is $O(n^2)$.

In an $HS(n, k)$ graph with a set of faulty nodes F, a fault-tolerant routing algorithm based on routing capabilities is shown in Fig. 3 as Procedure **FTR**. A non-faulty node ***src*** can try to send a message to a non-faulty node ***dst*** by calling this procedure as **FTR(*src*, *dst*, F)**. At each intermediate node ***cur***, which holds the message, the neighbor nodes on the shortest paths from ***cur*** to the destination node ***dst*** are checked to find the node ***nxt****

that has $R_{d-1}(\boldsymbol{nxt}^*) = 1$ and ensures delivery to ***dst*** by the shortest path. If it is found, the message is forwarded to it. Otherwise, the neighbor nodes on the detour paths are checked to find the node $\boldsymbol{nxt}^*$ that has $R_{d+1}(\boldsymbol{nxt}^*) = 1$ and ensures delivery to ***dst*** by the detour path. If it is found, the message is forwarded to it. Otherwise, any non-faulty neighbor node on the shortest path is selected to forward the message. Unfortunately, if there is no non-faulty neighbor node on the shortest paths, a non-faulty node on the detour path is selected and the message is forwarded to it.

```
procedure FTR(cur, dst, F)
begin
  d := dist(cur, dst);
  if d = 0 then deliver the message to cur
  else if d = 1 then FTR(dst, dst, F)
  else if ∃nxt* ∈ {nbr | nbr ∈ Pre(cur, dst), R_{d-1}(nbr) = 1} then
    FTR(nxt*, dst, F)
  else if ∃nxt* ∈ {nbr | nbr ∈ Spr(cur, dst), R_{d+1}(nbr) = 1} then
    FTR(nxt*, dst, F)
  else if ∃nxt* ∈ Pre(cur, dst) \ F then FTR(nxt*, dst, F)
  else if ∃nxt* ∈ Spr(cur, dst) \ F then FTR(nxt*, dst, F)
  else error('delivery failed')
end
```

Fig. 3. Fault-tolerant routing algorithm based on routing capabilities.

It takes $O(n)$ time to identify the preferred neighbor nodes of ***cur*** to ***dst*** $Pre(\boldsymbol{cur}, \boldsymbol{dst})$ and the spare neighbor nodes of ***cur*** to ***dst*** $Spr(\boldsymbol{cur}, \boldsymbol{dst})$, and to check $R_{d-1}(\boldsymbol{nbr})$ to find the node $\boldsymbol{nxt}^*$. Note that algorithm **FTR** may cause infinite loops. In **FTR**, if the message is forwarded to an intermediate node whose routing capability with respect to the distance to the destination node is equal to 1, the routing becomes successful. Otherwise, if such a node does not exist in the neighbor nodes, the message is forwarded to a non-faulty node. If the distance to the destination node is d, the number of the preferred neighbor nodes is equal to $\lceil d/2 \rceil$. Hence, if d is larger, the probability that a non-faulty node exists in the preferred neighbor nodes is higher, and the message is forwarded to one of them. However, if the message is close to the destination node, that is, d is small, the probability that all of the preferred neighbor nodes are faulty becomes higher. If all of them are faulty, **FTR** finds non-faulty nodes in the spare neighbor nodes, and forwards the message to one of them. In the next step, if **FTR** forwards the message to the previous node again, an infinite loop is formed.

4. COMPUTER EXPERIMENT

We conducted a computer experiment to evaluate the performance of our algorithm **FTR**. Since there is no fault-tolerant routing algorithm for an $HS(n, k)$ graph proposed so far, we use a simple fault-tolerant routing algorithm **SMP** shown in Fig. 4. In this section, we give the details of the experiment and its results. Note that the algorithm **SMP** may also contain infinite loops. The computer experiment was carried out for an $HS(n, k)$

graph, where $(n, k) = (9, 2), (9, 3), (9, 4), (10, 2), (10, 3), (10, 4)$, and $(10, 5)$ changing the proportion of faulty nodes α from 0.0 to 0.9, and we have measured the proportion of successful routings and their path lengths. In general, if the fault-tolerant routing algorithm increases the proportion of successful routings, the lengths of the paths constructed tend to be larger. Therefore, it is necessary for us to check that **FTR** does not cause excessively long routing paths. This is why we have measured the average path lengths along with the proportion of successful routings.

```
procedure SMP(cur, dst, F)
begin
  d := dist(cur, dst);
  if d = 0 then deliver the message to cur
  else if ∃nxt* ∈ Pre(cur, dst) \ F then SMP(nxt*, dst, F)
  else if ∃nxt* ∈ Spr(cur, dst) \ F then SMP(nxt*, dst, F)
  else error('delivery failed')
end
```

Fig. 4. A simple fault-tolerant routing algorithm.

Concretely, first, we randomly selected faulty nodes in an $HS(n, k)$ graph with the proportion α. Next, we selected the source node ***src*** and the destination node ***dst*** from non-faulty nodes randomly. Finally, after checking the connectivity of ***src*** and ***dst***, we applied the fault-tolerant routing algorithms. If ***src*** and ***dst*** are not connected, that is, there is no fault-free path between them, we started over from the selection of faulty nodes. For each pair of (n, k) and α, we executed at least 100,000 trials.

Figs. 5, 7, 9, 11, 13, 15, and 17 show the proportions of successful routings by algorithms **FTR** and **SMP**. Figs. 6, 8, 10, 12, 14, 16, and 18 show the average path lengths by algorithms **FTR** and **SMP**. For comparison, these figures include the average lengths of the fault-free shortest paths between the source and destination nodes.

To verify that the algorithm **FTR** is more effective than the algorithm **SMP**, we conducted a Wilcoxon rank-sum test on the successful routing proportions and the average path lengths obtained by the experiment. Table 3 shows the results where the p value represents the probability that a test statistic, the same as or more extreme than the observed value W, is observed under the null hypothesis. The W value is defined as the sum of ranks of the successful routings or the average path lengths by **FTR** among the corresponding values by **FTR** and **SMP** in this experiment. For example, in an $HS(9, 2)$ graph, we changed the numbers of faulty nodes from 0 to 32 in the experiment, and measured the proportion of successful routings or the average path lengths for each of the faulty node numbers. In the test, we compared the results by the algorithms **FTR** and **SMP** by using the Wilcoxon rank-sum test, where the proportion of successful routings or the average path lengths for each number of faulty nodes were used as samples. From Table 3, we can see a significant difference between the proportions of successful routings by the algorithms **FTR** and **SMP** in $HS(9, 2)$ and $HS(10, 2)$ graphs. On the other hand, there was no significant difference between the average path lengths by the algorithms **FTR** and **SMP**.

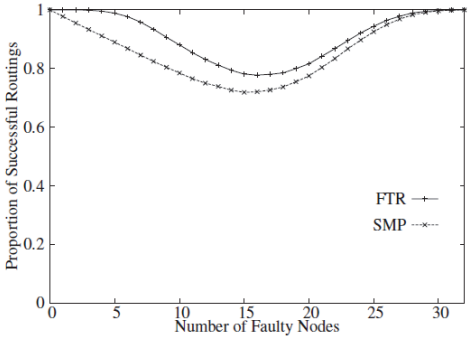

Fig. 5. Proportions of successful routings by algorithms **FTR** and **SMP** in an *HS*(9, 2) graph.

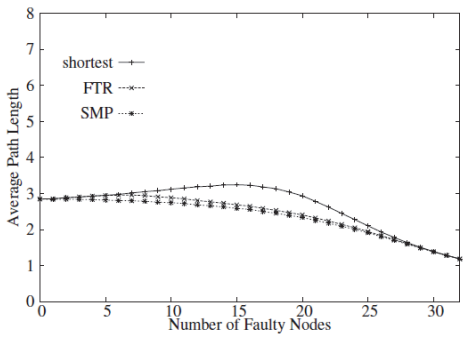

Fig. 6. Average path lengths by algorithms **FTR** and **SMP** and the fault-free shortest paths in an *HS*(9, 2) graph.

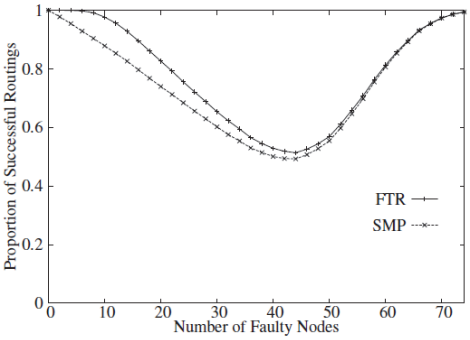

Fig. 7. Proportions of successful routings by algorithms **FTR** and **SMP** in an *HS*(9, 3) graph.

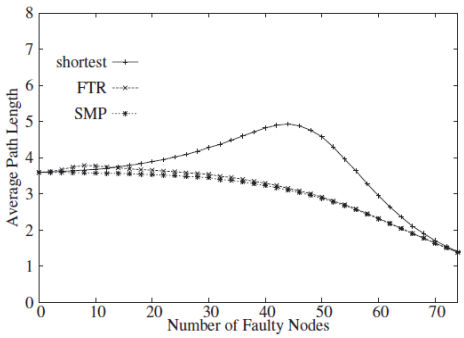

Fig. 8. Average path lengths by algorithms **FTR** and **SMP** and the fault-free shortest paths in an *HS*(9, 3) graph.

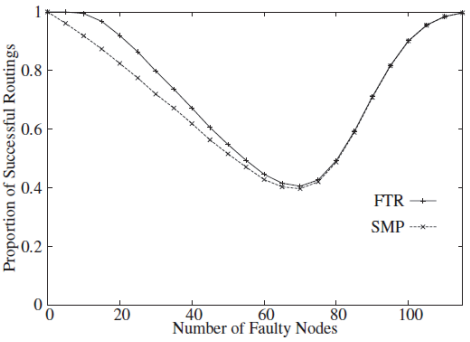

Fig. 9. Proportions of successful routings by algorithms **FTR** and **SMP** in an *HS*(9, 4) graph.

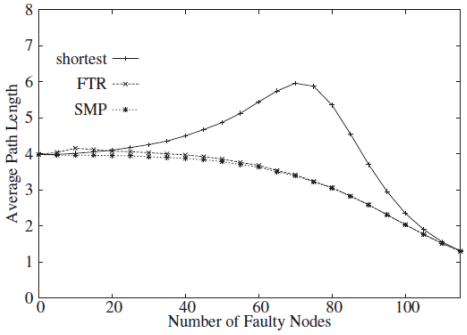

Fig. 10. Average path lengths by algorithms **FTR** and **SMP** and the fault-free shortest paths in an *HS*(9, 4) graph.

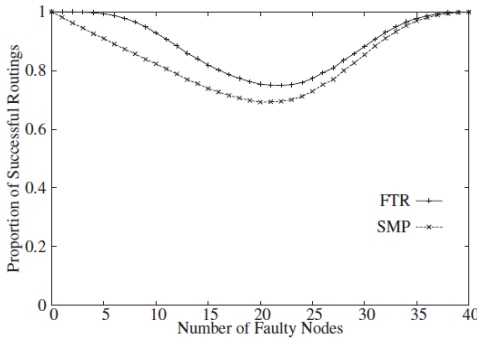

Fig. 11. Proportions of successful routings by algorithms **FTR** and **SMP** in an *HS*(10, 2) graph.

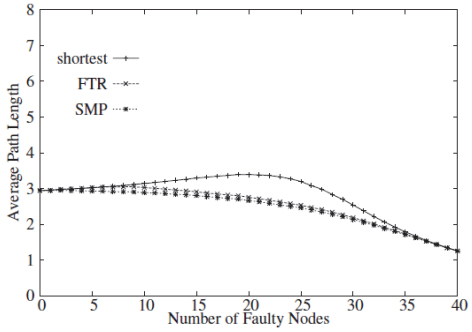

Fig. 12. Average path lengths by algorithms **FTR** and **SMP** and the fault-free shortest paths in an *HS*(10, 2) graph.

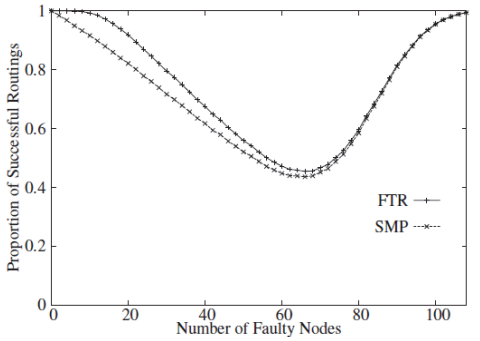

Fig. 13. Proportions of successful routings by algorithms **FTR** and **SMP** in an *HS*(10, 3) graph.

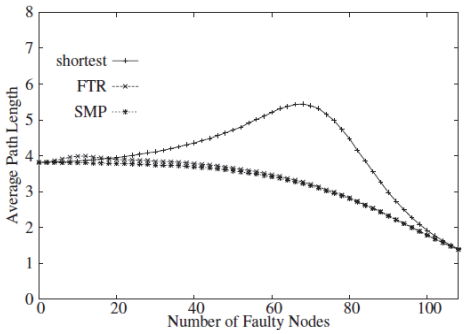

Fig. 14. Average path lengths by algorithms **FTR** and **SMP** and the fault-free shortest paths in an *HS*(10, 3) graph.

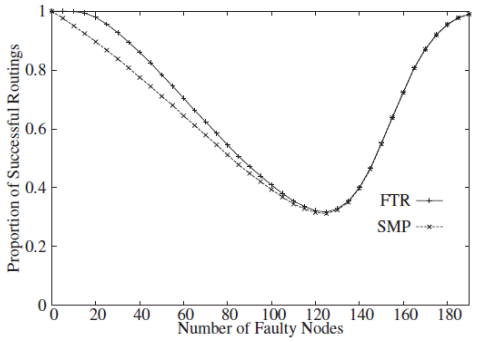

Fig. 15. Proportions of successful routings by algorithms **FTR** and **SMP** in an *HS*(10, 4) graph.

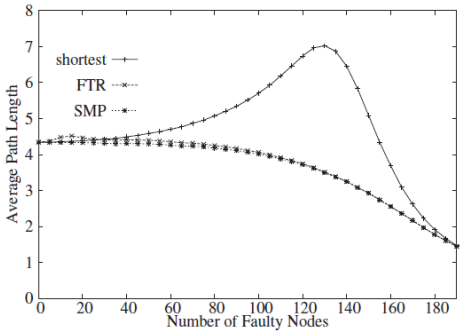

Fig. 16. Average path lengths by algorithms **FTR** and **SMP** and the fault-free shortest paths in an *HS*(10, 4) graph.

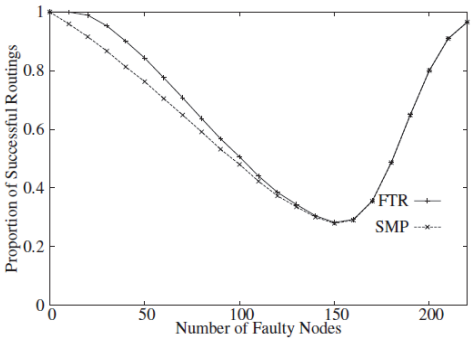

Fig. 17. Proportions of successful routings by algorithms **FTR** and **SMP** in an *HS*(10, 5) graph.

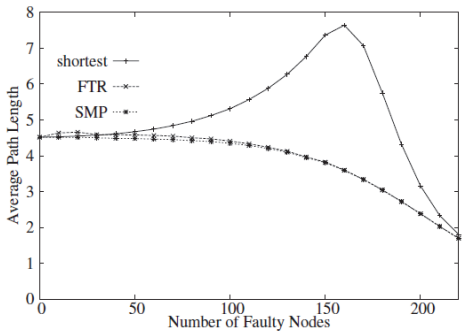

Fig. 18. Average path lengths by algorithms **FTR** and **SMP** and the fault-free shortest paths in an *HS*(10, 5) graph.

Table 3. Results of a Wilcoxon rank-sum test.

HS	#nodes	#observations (#faulty nodes)	Wilcoxon rank-sum test (**FTR** vs **SMP**)			
			Succ. routing props.		Average path lengths	
			W value	*p* value	W value	*p* value
(9, 2)	36	33 (0, 1, …, 32)	1272	0.03*	1208	0.19
(9, 3)	84	38 (0, 2, …, 74)	1575	0.24	1590.5	0.19
(9, 4)	126	24 (0, 5, …, 115)	620	0.51	638	0.30
(10, 2)	45	41 (0, 1, …, 40)	1949.5	0.02*	1854.5	0.16
(10, 3)	120	55 (0, 2, …, 108)	3260	0.21	3305.5	0.13
(10, 4)	210	39 (0, 5, …, 190)	1614	0.46	1665	0.21
(10, 5)	252	24 (0, 10, …, 230)	612	0.62	636	0.32

Based on this statistical analysis, we can say that the algorithm **FTR** performs better than the algorithm **SMP**, with a small amount of additional cost, especially when the degrees of nodes are relatively unbalanced. When the proportion of faulty nodes becomes higher, it becomes more difficult to find a pair of the source and destination nodes that are non-faulty and connected. Thus, the found pairs become closer and closer. Eventually, the proportion of successful routings becomes higher as the additional cost becomes lower.

5. CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have introduced the concept of routing capabilities in a hyper-star graph *HS*(*n*, *k*) and proposed a fault-tolerant routing algorithm. We have proved that the time complexity to calculate routing capabilities with respect to all the distances at each node is $O(n^2)$. In addition, we have conducted a computer experiment to verify that our algorithm attains a high reachability to the destination nodes.

In the future research, we suggest incorporating a stochastic framework into our method.

ACKNOWLEDGEMENTS

We would like to express special thanks to Dr. Bipin Indurkhya for his proofreading. Also, we appreciate the Associate Editor, Dr. Xiaohong Jiang, and the anonymous reviewers for their insightful comments and constructive suggestions. This study was partly supported by a Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science under Grant No. 17K00093.

REFERENCES

1. F. P. Preparata and J. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Communications of ACM*, Vol. 24, 1981, pp. 300-309.
2. S. B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Transactions on Computers*, Vol. 38, 1989, pp. 555-566.
3. P. F. Corbett, "Rotator graphs: An efficient topology for point-to-point multiprocessor networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3, 1992, pp. 622-626.
4. Q. M. Malluhi and M. A. Bayoumi, "The hierarchical hypercube: A new interconnection topology for massively parallel systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, 1994, pp. 17-30.
5. K. Ghose and K. R. Desai, "Hierarchical cubic networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, 1995, pp. 427-435.
6. J.-S. Jwo, "Properties of star graph, bubble-sort graph, prefix-reversal graph and complete-transposition graph," *Journal of Information Science and Engineering*, Vol. 12, 1996, pp. 603-617.
7. S. Latifi and P. K. Srimani, "A new fixed degree regular network for parallel processing," in *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing*, 1996, pp. 152-159.
8. C. L. Seitz, "The cosmic cube," *Communications of the ACM*, Vol. 28, 1985, pp. 22-33.
9. H.-O. Lee, J.-S. Kim, E. Oh, and H.-S. Lim, "Hyper-star graph: A new interconnection network improving the network cost of the hypercube," in *Proceedings of the 1st EurAsian Conference on Information and Communication Technology*, 2002, pp. 858-865.
10. E. Cheng and L. Lipták, "Structural properties of hyper-stars," *Ars Combinatria*, Vol. 80, 2006, pp. 65-73.
11. E. Cheng and M. Shah, "A strong structural theorem for hyper-stars," *Congressus Numerantium*, Vol. 179, 2006, pp. 181-191.
12. J.-S. Kim, C.-H. Cho, and H.-O. Lee, "Analysis of topological properties and embedding for folded hyper-star network," *Journal of Korea Multimedia Society*, Vol. 11, 2008, pp. 1227-1237.
13. J.-S. Yang and J.-M. Chang, "Independent spanning trees on folded hyper-stars," *Networks*, Vol. 56, 2010, pp. 272-281.

14. E. Cheng, P. Hu, R. Jia, and L. Lipták, "Matching preclusion and conditional matching preclusion for bipartite interconnection networks II: Cayley graphs generated by transposition trees and hyper-stars," *Networks*, Vol. 59, 2012, pp. 357-364.
15. L. Lipták, E. Cheng, J.-S. Kim, and S. W. Kim, "One-to-many node-disjoint paths of hyper-star networks," *Discrete Applied Mathematics*, Vol. 160, 2012, pp. 2006-2014.
16. A. Bossard, "A set-to-set disjoint paths routing algorithm in hyper-star graphs," *ISCA International Journal of Computers and Their Applications*, Vol. 21, 2014, pp. 76-82.
17. F. Zhang, K. Qiu, and J.-S. Kim, "Hyper-star graphs: Some topological properties and an optimal neighbourhood broadcasting algorithm," *Concurrency and Computation: Practice and Experience*, Vol. 27, 2015, pp. 4186-4193.
18. J.-S. Yang, S.-S. Luo, and J.-M. Chang, "Pruning longer branches of independent spanning trees on folded hyper-stars," *The Computer Journal*, Vol. 58, 2015, pp. 2972-2981.
19. G.-M. Chiu and S.-P. Wu, "A fault-tolerant routing strategy in hypercube multicomputers," *IEEE Transactions on Computers*, Vol. 45, 1996, pp. 143-155.
20. G.-M. Chiu and K.-S. Chen, "Use of routing capability for fault-tolerant routing in hypercube multicomputers," *IEEE Transactions on Computers*, Vol. 46, 1997, pp. 953-958.
21. J. Wu, "Adaptive fault-tolerant routing in cube-based multicomputers using safety vectors," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, 1998, pp. 322-334.
22. K. Kaneko and H. Ito, "Fault-tolerant routing algorithms for hypercube interconnection networks," *IEICE Transactions on Information and Systems*, Vol. E84-D, 2001, pp. 121-128.
23. S.-I. Yeh, C.-B. Yang, and H.-C. Chen, "Fault-tolerant routing on the star graph with safety vectors," in *Proceedings of the 6th Annual International Symposium on Parallel Architectures*, *Algorithms*, *and Networks*, 2002, pp. 301-306.
24. W. H. Gates and C. H. Papadimitriou, "Bounds for sorting by prefix reversal," *Discrete Mathematics*, Vol. 27, 1979, pp. 47-57.

Yo Nishiyama (西山遥) has finished the master program of Tokyo University of Agriculture and Technology in Japan, and he is now working at CyberAgent, Inc. His main research areas are dependable computing and fault-tolerant systems. He received the B.E. and M.E. degrees from Tokyo University of Agriculture and Technology in 2013 and 2015, respectively.

Yuko Sasaki (佐々木悠子) has finished the master program of Tokyo University of Agriculture and Technology in Japan, and she is now working at ARK Information Systems, Inc. Her main research areas are dependable computing and fault-tolerant systems. She received the B.E. and M.E. degrees from Tokyo University of Agriculture and Technology in 2013 and 2015, respectively.

Yuki Hirai (平井佑樹) is a Lecturer at Shinshu University. His main research areas are educational technologies and computer-supported collaborative learning. He received the B.L.A. and M.Ed. degrees from Tokyo Gakugei University in 2007 and 2009, respectively. He also received the Ph.D. degree from University of Tsukuba in 2012. He is a member of IPSJ and JSAI.

Hironori Nakajo (中條拓伯) is an Associate Professor at Tokyo University of Agriculture and Technology in Japan. His research interests are computer architecture, parallel processing, cluster computing and reconfigurable computing. He received the B.S., M.S., and Ph.D. degrees from Kobe University in 1985, 1987, and 1997, respectively. He is a member of ACM, IEEE CS, IEICE, and IPSJ.

Keiichi Kaneko (金子敬一) is a Professor at Tokyo University of Agriculture and Technology in Japan. His main research areas are functional programming, parallel and distributed computation, partial evaluation and fault-tolerant systems. He received the B.E., M.E. and Ph.D. degrees from the University of Tokyo in 1985, 1987 and 1994, respectively. He is a member of ACM, IEEE CS, IEICE, IPSJ and JSSST.