ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.7, July 2007, pp.1818−1830
DOI: 10.1360/jos181818
© 2007 by *Journal of Software*. All rights reserved.

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 NPV 广义超立方体最佳容错路由算法*

田绍槐[1+], 陆应平[2], 张大方[1]

[1](湖南大学 软件学院,湖南 长沙 410082)

[2](Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA)

An NPV-Based Optimal Fault-Tolerant Routing Algorithm for Generalized Hypercube

TIAN Shao-Huai[1+], LU Ying-Ping[2], ZHANG Da-Fang[1]

[1](Software School, Hunan University, Changsha 410082, China)

[2](Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA)

+ Corresponding author: Phn: +86-731-8823264, E-mail: tianshaohuai163@tom.com

Tian SH, Lu YP, Zhang DF. An NPV-based optimal fault-tolerant routing algorithm for generalized hypercube. *Journal of Software*, 2007,18(7):1818−1830. http://www.jos.org.cn/1000-9825/18/1818.htm

Abstract: Optimal fault-tolerant routing is imperative for large hypercube systems in the existence of large number of faulty links or nodes. This paper first defines hypercube network with a large number of faulty nodes and/or links to be generalized hypercube and illustrates that many non-hypercube systems can be transformed to Generalized Hypercube systems. It then proposes node path vector (NPV) to capture the complete optimal and sub-optimal routing information for a generalized hypercube system. To reduce the computation iterations in solving NPV, it also introduces the concept of relay node technique. Based on NPV and relay node technique, this paper further proposes optimal fault-tolerant routing scheme (OFTRS) to derive shortest path for any communication pair in a generalized hypercube system. With an example of large number of faulty nodes or faulty links, it is illustrated that the previous algorithm could omit up to 60% routing paths, while this approach achieves all optimal and sub-optimal routing paths. Compared to previous work, OFTRS has a significant improvement in obtaining routing information for optimal and sub-optimal, i.e. no matter how many faulty nodes or links exist, it is guaranteed to route through the optimal or sub-optimal path as long as a path between the source-destination pair exists. In addition, the proposed scheme is distributed and relying only on non-faulty neighboring nodes since it only requires the information of non-faulty neighbor nodes in computing NPV, thus it has high applicability, especially when some non-hypercube systems can be transformed into Generalized Hypercube systems.

Key words: fault-tolerant routing; generalized hypercube; node path vector; relay node technique

摘 要: 在网络可靠性研究中,设计较好的容错路由策略、尽可能多地记录系统中最优通路信息,一直是一项重要的研究工作.超立方体系统的容错路由算法分为可回溯算法和无回溯算法.一般说来,可回溯算法的优点是容错能力强:只要消息的源节点和目的节点有通路,该算法就能够找到把消息传递到目的地的路径;其缺点是在很多情况下传

* Supported by the National Natural Science Foundation of China under Grant No.60473031 (国家自然科学基金); the Scientific Research Foundation of Education Department of Hunan Province under Grant No.03C036 (湖南省教育厅科研基金项目)

Received 2006-03-14; Accepted 2006-05-18

递路径不能按实际存在的最短路径传递.其代表是深度优先搜索(DFS)算法.无回溯算法是近几年人们比较关注的算法.该算法通过记录各邻接节点的故障信息,给路由算法以启发信息,使消息尽可能按实际存在的最短路径传递.这些算法的共同缺点是只能计算出 Hamming 距离不超过 n 的路由.在 n 维超立方体系统连通图中,如果系统存在大量的故障,不少节点对之间的最短路径大于 n,因此,这些算法的容错能力差.提出了一个实例说明采用上述算法将遗失 60%的路由信息.另外,由于超立方体的结构严格,实际中的真正超立方体系统不多.事实上,不少的网络系统可转换为具有大量错误节点和错误边的超立方体系统.因此,研究能适应具有大量错误节点和错误边的超立方体系统的容错路由算法是一个很有实际价值的工作.研究探讨了:(1) 定义广义超立方体系统;(2) 在超立方体系统中提出了节点通路向量(NPV)概念及其计算规则;(3) 提出了中转点技术,使得求 NPV 的计算复杂度降低到 $O(n)$;(4) 提出了基于 NPV 的广义超立方体系统最佳容错路由算法(OFTRS),该算法是一种分布式的和基于相邻节点信息的算法.由于NPV 记录了超立方体系统全部最优通路和次最优通路的信息,在具有大量故障的情况下,它不会遗漏任何一条最优通路和次最优通路信息,从而实现了高效的容错路由.在这一点上,它优于其他算法.

关键词: 容错路由;广义超立方体;节点通路向量;中转点技术
中图法分类号: TP393 文献标识码: A

1 Introduction

Hypercube topology for distributed-memory system has received phenomenal attention in the past decade due to its simple and regular structure, easy and efficient routing, and built-in robustness. An n-dimensional hypercube system comprises 2^n nodes or processors that can be addressed in n-bit binary numbers. As the size of dimension increases, the number of nodes and/or links also grows drastically. Accordingly, the likelihood of faults resulting from these nodes or links also increases. Therefore, it is very important to design an efficient optimal routing scheme when faults occur so that a source node can still route via a shortest path to the destination node.

However, due to the rigorousness of hypercube topology, truly hypercube systems have found limited applications. Thus in this paper, we investigate a Generalized Hypercube, which is a standard hypercube or transformed hypercube that possesses faulty nodes or links. There have been considerable researches on providing efficient paths for message passing in a faulty hypercube system. However, these extant schemes fail to provide complete routing information. As illustrated in Section 7, 60% routing paths are omitted in the existing schemes. To address this deficiency, we propose Node Path Vector (NPV) that maintains all routing information for the shortest routing paths. We also propose relay node technique and define ways to reduce the computation complexity in solving NPV. Furthermore, we propose Optimal Fault-Tolerant Routing Scheme (OFTRS) based on NPV. As illustrated by examples, our scheme contains much more routing information and can find more optimal fault-tolerant routing paths than the existing schemes. In a hypercube system with a large number of faulty nodes and/or links, OFTRS scheme possesses the merits of both strong fault-tolerance found in the depth search first (DSF) algorithm[3] and short distance in the non-backtracking routing algorithms. As long as a path of the shortest distance exists between a pair of nodes in a generalized hypercube, OFTRS guarantees to find a shortest path for the node pair.

The rest of paper is organized as follows. Section 2 reviews the related work. Section 3 defines the concept of the Generalized Hypercube and notations used in this paper. Section 4 introduces our proposed NPV (node path vector) and defines an iteration algorithm to generate NPV. In Section 5, we present relay node technique to reduce the iteration times for computing NPV and illustrate the technique through examples. In Section 6, we propose Optimal Fault-Tolerant Routing Scheme (OFTRS) based on the computed NPV in a Generalized Hypercube System. We evaluate the proposed scheme in Section 7. Finally we conclude this paper in Section 8.

2 Related Work

Two types of fault-tolerant routing algorithms exist in a hypercube system: backtracking and non-backtracking. In general, backtracking algorithms have stronger fault-tolerant capability: it normally finds a valid path provided that a path exists. A typical example is the DFS algorithm[3]. However, these algorithms cannot guarantee to find a shortest path in most cases. As a result, non-backtracking algorithms have received more attention.

Non-backtracking algorithms attempt to find a shortest path for a node-pair by retaining the fault information of neighboring nodes. However, this type of algorithm has limited fault-tolerance. Reference [1] shows that when the number of faulty nodes is less than $n/2$ in an n-dimension hypercube, a message can be routed through non-faulty nodes with a path length less than the Hamming distance between the two nodes plus 2. Reference [2] relaxes the number of faulty nodes to $n-1$, but the path length between two non-faulty nodes can reach to their Hamming distance plus 4.

Other non-backtracking algorithms include Wu's Safety Vectors (SVs) and fault-tolerant routing algorithm based on SVs[7]. SVs have a good property whose element can reflect whether an optimal path exists between source and destination nodes within a Hamming distance of k. However, Safety Vectors lack sufficient information for optimal routing. Gao *et al*. extends the definition of Safety Vectors[8,17]. They propose the Extended Safety Vectors (ESVs) and a corresponding routing algorithm. ESVs has revised the definition of routing distance of 2 and thus extended the definition of SVs. As a result, ESVs and the corresponding routing scheme can find more routing paths than the routing scheme based on SVs.

Gao, *et al*.[9] further proposes Optimal Path Matrices (OPMs) by using matrices to record link status between adjacent nodes. Reference [12] further extends OPMs and proposes Extended Optimal Path Matrices (EOPMs). EOPMs have more routing information for faulty hypercube system. EOPMs are further improved in Ref.[13], where Optimal Path Set and Optimal-Path-Set Based Optimal Path Matrices (OPSBOPMs) routing algorithm are proposed. Other work[14−17] further extend the previous schemes. For example, Ref.[15] proposes Maximum Safety Path Matrices that extends OPMs and EOPMs. Even though there are considerable improvements in these newer schemes, as illustrated in Section 7, a lot of communication paths are still omitted in these algorithms, especially for hypercube with a large number of faulty links.

The aforementioned schemes use either local or global link information of node faults and/or link faults to compute deterministic path vector for each node. The deterministic path information is then used to derive an optimal or sub-optimal routing. Other schemes[10,11] adopt a different approach. Instead of calculating a deterministic path vector, they create a probability path vectors that represent the likelihood whether an optimal path exists for a destination node within a Hamming distance of K. Recent researches have also studied the application of hypercube routing into peer-to-peer (P2P) networks[18,19] where the P2P network is converted to a hypercube topology. Broadcasting or multicasting is used to provide proper routing.

From the review of the previous work, we observe that:

(1) Previous schemes all tried to store as much routing information as possible, but failed to record all of them. As the number of the faulty nodes and faulty links in the Hypercube system increases, more routing information is omitted. Consequently, it is more likely that optimal paths are missed in the actual routing.

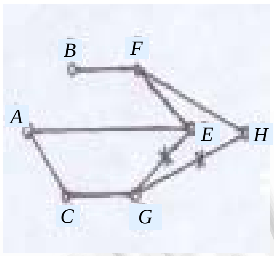
(2) Due to the rigorousness of the hypercube structure, the applicability of most of the research results into practical application is pretty limited. In fact, many network structures in practical use can be converted into a hypercube network with a large number of faulty nodes and faulty links.

Our approach differs from the previous work in that our proposed Node Path Vector (NPV) can capture all routing information. Consequently, the proposed routing algorithm based on NPV can derive all optimal or sub-optimal paths for any communication pair in a generalized hypercube system.

3 Preliminaries and Notations

Definition 1. A hypercube system with a large number of faulty nodes and faulty links, or a connected hypercube graph with nodes and/or links removed is called a *Generalized Hypercube System*.

Figure 1 shows a network topology that excludes node *D* and link *AB* with 2 faulty links. After adding the missed node *D* and link *AB*, the network is transformed into a 3-dimension Generalized Hypercube Network with 6 faulty links as illustrated in Fig.2. In the transformed network topology, node *D* and link *AB* are deemed to be faulty.



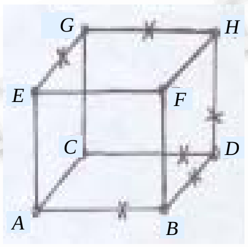


Fig.1 Network with missing nodes and links Fig.2 Transformed into Generalized Hypercube

Notations: The following are the notations in this paper.

Q_n: Denote an *n*-dimension Generalized Hypercube

REL(*A*,*B*): Denote the relative address between nodes *A* and *B*

Dist(*A*,*B*): The Hamming distance between node *A* and node *B*

nei(*A*,*C*)=1: Node *C* is the non-faulty neighbor node of node *A*;

nei(*A*,*C*)=0: Node *C* is the faulty neighbor node of node *A*;

Src, Dest, Cur: Denote a source node, destination node, and current node respectively.

Max-*P*: Denote the maximum distance between any two nodes in hypercube Q_n.

4 Node Path Vectors (NPV)

4.1 Definition of node path vectors

Definition 2. In Q_n, each node uses a 1-dimension array with 2^n elements to store the information of the optimal paths to every other node, where its *k*-th element records the length of the optimal or sub-optimal paths to node *k*. We refer this array to Node Path Vector (NPV). $NPV_A^i(k)$ is used to denote the value of the *k*-th element in node *A*’s NPV at the *i*-th iterative calculation.

Definition 3. In Q_n, let the node *B* correspond to the *k*-th element, the calculation rule of $NPV_A^i(k)$ is as Follows:

When *i*=1,

$$NPV_A^i(k)=\begin{cases}0, & \text{if } Dist(A,B)=0\\ 1, & \text{if } nei(A,B)=1\\ NULL, & \text{if } nei(A,B)=0 \,||\, Dist(A,B)>1\end{cases};$$

When 1<*i*≤Max-*P* && $NPV_A^{i-1}(k)=NULL$,

$$NPV_A^i(k)=\begin{cases}i, & \text{if there exists node } C \text{ satisfying } nei(A,C)=1\,\&\&\,NPV_C^{i-1}(k) \text{ is NOT NULL}\\ NULL, & \text{else}\end{cases}.$$

4.2 Example of NPV calculation

Example 1. Calculate $NPV_A^i(k)$ of nodes (0000) (1000) in the 4-dimension Generalized Hypercube in Fig.3.

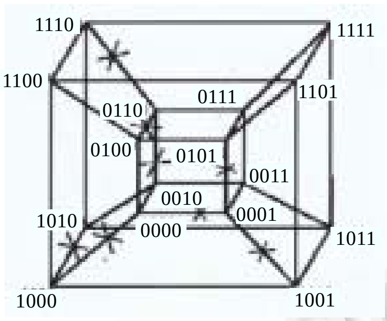


Fig.3 A 4-dimension generalized hypercube with 8 faulty links

Solution:

When *i*=1, the $NPV_A^1(k)$ of nodes (0000) and (1000) is as follows:

Node	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0		1		1											
1000									0	1			1			

When *i*=2, the calculation of $NPV_A^2(k)$, ② is the new value generated in this iteration.

Node	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0		1	②	1	②					②		②			
0010	1		0	1							1					
0100	1				0	1							1			

Node	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
1000					②				0	1		②	1	②	②	
1001									1	0		1		1		
1100					1				1				0	1	1	

…

When *i*=5, the calculation of $NPV_A^5(k)$ of nodes (1000), ⑤ is the value in this calculation.

Node	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
1000	3	4	4	3	2	3	⑤	4	0	1	3	2	1	2	2	3
1001	4	3	3	2	3	2	4	3	1	0	2	1	2	1	3	2
1100	2		3	4	1	2	4	3	1	2	2	3	0	1	1	2

After 4 rounds of synchronous information exchanges, the NPV of two nodes are as follows:

Node	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0	3	1	2	1	2	4	3	3	4	2	3	2	3	3	4
1000	3	4	4	3	2	3	5	4	0	1	3	2	1	2	2	3

4.3 Properties of NPV

Theorem 1. NPV for a node maintains all the routing information of the optimal paths and the sub-optimal paths.

Proof: we prove this theorem by induction.

We first show that the value of *k*-th element of NPV $NPV_A^i(k)$ is the length of the optimal or sub-optimal paths from node *A* to node *B*.

When *i*=1, by the calculation rule of Definition 3, if *nei*(*A*,*B*)=1, then $NPV_A^1(k)=1$. The conclusion is obviously established.

When i=2, if and only if there exists a non-faulty neighboring node C, and its $NPV_C^1(k)=1$ and $NPV_A^1(k)=$ *NULL*, then $NPV_A^2(k)=2$. Clearly, the conclusion holds.

Suppose when $i=m\geq 2$, it holds. If there is a non-faulty neighbor node C of node A, $NPV_C^m(k)$ records the length of the optimal path and the sub-optimal path from node C to the k-th node, and at the same time, $NPV_A^m(k)$ is null. Obviously, $NPV_C^m(k)+1$ is the length of optimal or sub-optimal path from node A to the k-th node. By the calculation rule in Definition 3, $NPV_A^{m+1}(k)=NPV_C^m(k)+1=m+1$. Thus when $i=m+1$ the conclusion also holds. □

We then show that $NPV_A^i(k)$ does not leave out any path of the optimal path or sub-optimal path from node A to node B which corresponds to the k-th node.

When i=1, by the calculation rule in Definition 3, if *nei*(A,B)=1, then $NPV_A^1(k)=1$. Obviously, it does not omit the optimal path from node A to its non-faulty neighbor node B.

When i=2, if and only if there exists a non-faulty neighboring node C, and its $NPV_C^1(k)=1$ is mistaken to be $NPV_C^1(k)=$*NULL*, then the optimal path from node A to node B (the node corresponding to the k-th element of NPV or the k-th node) can be omitted. Clearly, this is not true.

Suppose when $i=m\geq 2$, the conclusion holds. That is, for all nodes in Q_n, it does not omit any optimal path or sub-optimal path from this node to all other nodes at the m-th iteration calculation. If there exists any path omission in $NPV_A^{m+1}(k)$ (the number of optimal or sub-optimal paths from node A to the k-th node) at the m+1-th calculation, by the calculation rules in Definition 3, then at least there exists a non-faulty neighboring node C of node A that has omitted the k-th node at the m-th calculation. Clearly, this contradicts the assumption. So the conclusion is established.

Corollary 1. If $NPV_A(B)=Dist(A,B)$, then there exists an optimal path between nodes A and B.

Corollary 2. If $NPV_A(B)>Dist(A,B)$, then there exists a sub-optimal path between node A and node B.

Theorem 2. The value of every element in $NPV_A(B)$ is continuous in integer within the range [0,Max-P], where Max-P is the maximum length of optimal and sub-optimal path from node A to all other nodes.

Proof: Let x be an integer, $0<x\leq$Max-P. If $NPV_A(B)=x$, clearly there exists an adjacent node C such that $nei(B,C)=1$ and $NPV_A(C)=x-1$. i.e. all element's values in NPV_A are continuous in integer within the range [0,Max-P].

5 Reducing NPV's Calculation Complexity

According to Definition 3, we observe that it takes Max-P rounds of synchronous message exchanges to compute every node A's $NPV_A()$ for a n-dimension hypercube Q_n. This computation complexity increases radically as the number of dimension and the number of faulty links grows. Therefore, it is very important to reduce the computation complexity in computing NPV. We propose to use Relay Node Technique to alleviate the computation time.

Definition 4. Node A's Path Bit Vector PBV_A is defined as:

$$PBV_A(k)=\begin{cases}1, & \text{if } NPV_A^1(k)=1\\ 0, & \text{otherwise}\end{cases},\ 0\leq k\leq 2^n-1.$$

Example 2. Figure 4 is a 4-dimension hypercube system with 15 faulty links. Derive the PBV for nodes (0110), (0011), (1000) and (1110).

Solution:

$$PBV_{0110}=(0000000100000000),$$

PBV_{0011}=(0110000100010000),

PBV_{1000}=(0000000001001000),

PBV_{1110}=(0000000000100001).

Clearly, before the first round of synchronous message exchange occurs, every node A's PBV_A in Q_n can be computed directly from $NPV_A^1()$. This value will be attached to node A's message and sent out to its non-faulty adjacent nodes. Following this way, after n rounds of synchronous exchanges, we know that for any node A, if $NPV_A^{n+1}(k) \neq NULL$, then PBV_k has been sent to $NPV_A^{n+1}(k)$. Table 1 shows the results for each node in Fig.4.

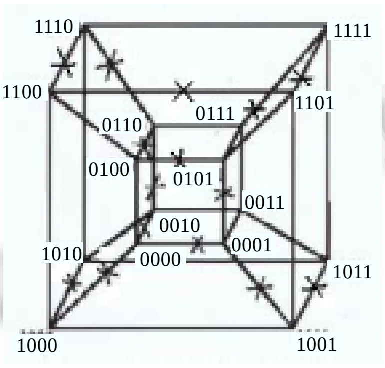


Fig.4 A hypercube with 15 faulty links

Table 1 Results of NPV after four rounds of synchronous exchanges in Fig.4

Node	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0				1				3	4			2	5		
0001		0	2	1		3	3	2		5	3	2		4	4	3
0010		2	0	1		3	3	2		5	1	2		4	2	3
0011		1	1	0		2	2	1	5	4	2	1		3	3	2
0100	1				0	5			2	3			1	4		
0101		3	3	2	5	0	2	1	3	2	4	3	4	1	5	4
0110		3	3	2		2	0	1	5	4	4	3		3	5	4
0111		2	2	1		1	1	0	4	3	3	2	5	2	4	3
1000	3			5	2	3	5	4	0	1			1	2		
1001	4	5	5	4	3	2	4	3	1	0		5	2	1		
1010		3	1	2		4	4	3			0	1		5	1	2
1011		2	2	1		3	3	2		5	1	0		4	2	1
1100	2				1	4		5	1	2			0	3		
1101	5	4	4	3	4	1	3	2	2	1	5	4	3	0		5
1110		4	2	3		5	5	4			1	2			0	1
1111		3	3	2		4	4	3			2	1		5	1	0

Definition 5. Let SBV_A denote node A's State Bit Vector after calculating $NPV_A^{n+1}()$, i.e.

$$SBV_A(k)=\begin{cases}1, & \text{if } NPV_A^{n+1}(k)=NULL\\ 0, & \text{else}\end{cases},\ 0\le k\le 2^n-1.$$

Example 3. Based on Table 1, solve the SBV of nodes (0110), (0011), (1000), (1110) in Fig.4.

Solution:

SBV_{0110}=(1000100000001000),

SBV_{1000}=(0110000000110011),

SBV_{0011}=(1000100000001000),

SBV_{1110}=(1000100011001100).

In summary, for a hypercube Q_n of n-dimension, if we stop the calculation after n rounds of synchronous message exchanges, we have the following five observations:

(1) By Theorem 1, we know that every node's optimal paths and sub-optimal paths with length less than n+2 have all been solved. By Definition 4, all nodes' PBV_k have been solved. Only when $NPV_A^{n+1}(k) \neq NULL$, can the PBV_k be sent to $NPV_A^{n+1}(k)$. Let's assume:

Y={y| $NPV_A^{n+1}(y)$ =$NULL$}, X={x| $NPV_A^{n+1}(x)$ =n+1},

PS_x=PBV_x & SBV_A (PS_x is the result of bit-wise AND between PBV_x and SBV_A),

Y_x^0 ={y| $NPV_x^{n+1}(y)$ =$NULL$, $y\in Y$, $x\in X$}, Y_x^1 ={y| $NPV_x^{n+1}(y)$ =1, $y\in Y$, $x\in X$},

Y_x^2 ={y| $NPV_x^{n+1}(y)$ >1, $y\in Y$, $x\in X$}.

After n rounds of synchronous message exchanges, although the path length between node A and node y has not been solved, by Theorem 2, it can be determined that they have a sub-optimal path with length greater than n+1.

(2) After n rounds of synchronous message exchanges, every node A's SBV_A can be solved.

(3) For every node A, PBV_A, SBV_A are calculated only once, i.e. before the first round of message exchange and after the n-th round of message exchange respectively.

(4) According to Definition 3 and Theorem 2, unless all elements of NPV_A() have been solved, there exists at least one node $x\in X$ and $NPV_A^{n+1}(x)$ =n+1.

(5) According to Theorem 2, if $Y\neq\Phi$, then $Y_x^1\neq\Phi$. i.e. there exists at least one pair of nodes (x,y), $y\in Y$, $x\in X$ and $NPV_A^{n+1}(x)$ =n+1, $NPV_x^{n+1}(y)$ =1. Otherwise, $NPV_A^{n+2}()$ has no element having the value of n+2, which contradicts the conclusion in Theorem 2.

Definition 6. In an n-dimension generalized hypercube Q_n, a node x is called node A's Relay Node if node $x\in X$ and there exists $y\in Y$ and $NPV_x^{n+1}(y)$ =1 after n rounds of synchronous message exchanges.

Clearly, from observation (5), node A certainly has a relay node, but it may have more than one relay node.

If there is only one element x ($x\in X$), and $NPV_A^{n+1}(x)$ =n+1, clearly x is the only relay node of node A. Based on Definition 3 and Theorem 2, since the shortest distance from A to any node y in set Y is greater than n+1, as a result, the shortest path from A to y must traverse node x. According to Theorem 1, clearly, $NPV_A^{n+1}(y) = NPV_A^{n+1}(x) + NPV_x^{n+1}(y)$ =n+1+ $NPV_x^{n+1}(y)$, which is the shortest path between A and y.

Suppose there exists an element x ($x\in X$) and $NPV_A^{n+1}(x)$ =n+1. From Definitions 4 and Definitions 5, if the result of bit-wise AND between PBV_x and SBV_A (PBV_x & SBV_A) contains no element of value of '1', that is, there is no $y\in Y$ such that $NPV_x^{n+1}(y)$ =1. In that case, x is not the relay node of A. If it is assumed to be the relay node, then at least the value of n+2 is missing from all elements of $NPV_x^{n+1}()$, which contradicts Theorem 2.

If there exists more than one element x ($x\in X$) such that $NPV_A^{n+1}(x)$ =n+1, and the result of bit-wise AND between PBV_x and SBV_A contains '1' element(s), from Definition 6, these nodes are all relay nodes. If all the '1' elements in the result of a node x1's PBV_{x1} & SBV_A are overlapped in position with the '1' elements in the result of another node x2's PBV_{x2} & SBV_A, then we refer x1 to be absorbable by node x2. As the connectivity between nodes set Y and a relay node x1 absorbed by other node x2 is already contained by other nodes, thus node x1 can be removed from the relay node set. A master relay node can be selected from the remaining relay nodes whose PBV_x & SBV_A has the largest number of '1' elements. Thereby, we obtain the following relay node selection criteria.

Relay Node Selection Criteria. The following criteria are used to select a relay node:

(1) If there is only one node x satisfying $NPV_A^{n+1}(x)$ =n+1, then x is the relay node of A. For all $y\in Y$, the path for a message from A to y is $A\rightarrow x\rightarrow y$. In particular, if $y\in Y_x^0$, then the path for a message from A to y is $A\rightarrow x\rightarrow x$'s relay $node\rightarrow y$.

(2) If there are more than one node *x* satisfying $NPV_A^{n+1}(x)=n+1$, first, nodes having $PS_x=0$ are removed from consideration. Then those absorbed by other nodes are removed. If there is only one node left, this is the relay node we are looking for. Otherwise, among the remaining nodes, the node whose PBV_X & SBV_A has the largest counts of '1' elements is selected to be the master relay node for node *A*. And if $y\in Y_x^1$, then the node *x* with $PS_x(y)=1$ is selected as the relay node for node *A*. Otherwise, if $y\in Y_x^2$, then the master relay node is selected as the relay node for node *A*. Following (1), the message routing path can also be found.

Example 4. From table 1 and Relay Node Selection Criteria, decide the relay nodes of all nodes in Fig.4.

Solution: Clearly, from table 1 and selection Criteria (1), nodes (1101), (1001), (1001), (1000), (0101), (1100), (1101), (1001), (0111), (1101) are the sole relay nodes for nodes (0000), (0001), (0010), (0011), (0100), (0111), (1010), (1101), (1100), (1111) respectively.

From Table 1 and selection Criteria (2):

Since $NPV_{1000}^5(0011)=NPV_{1000}^5(0110)=5$, PBV_{0110} & SBV_{1000}=(0000000000000000), PBV_{0011} & SBV_{1000}= (0110000000010000), So node (0011) becomes the only relay node of node (1000).

Similarly, node (1000) is the only relay node of node (0110); node (1011) is the only relay node of node (1001); node (0101) is the only relay node of node (1110); node (0100) is the only relay node of node (0101).

Since $NPV_{1101}^5(0000)=NPV_{1101}^5(1010)=NPV_{1101}^5(1111)=5$, PBV_{0000} & SBV_{1101}=(0000000000000000),

PBV_{1010} & SBV_{1101}=(0000000000000010), PBV_{1111} & SBV_{1101}=(0000000000000010), either node (1010) or node (1111) can be a relay node. Each node can absorb the other one. Either one is selected to be the relay node.

6 Optimal Fault-Tolerant Routing Scheme (OFTRS)

6.1 Algorithm for generating NPV

According to the Definitions 3~6, we compute $NPV_A^i()$ of every node *A* through a distributed information exchange. Here is the algorithm to compute NPV of a node.

```
Algorithm create_NPV()
  {Determine NPV and collect link fault information to calculate $PBV_A$ and construct $NPV_A^1()$;
    Send $NPV_A^1()$ and $PBV_A$ via non-faulty link;
    Receive all $NPV_C^1()$ and $PBV_C$ from the non-faulty neighbor node C;
    for (i=2; i≤n; i++)
      {Calculate $NPV_A^i()$ with all $NPV_C^{i-1}()$ && nei(A,C)=1 based on Definition 3;
        Send it out via non-faulty link;
        Receive all $NPV_C^i()$ from the non-faulty neighbor node C;}
    According to Definition 5 calculate $SBV_A$;
    Determine the relay node x from the relay node selection criteria.}
```

6.2 Optimal fault-tolerant routing scheme based on NPV

We further present our Optimal Fault-Tolerant Routing Scheme (OFTRS) based on NPV.

```
Algorithm OFTRS_Route
int main()
  {char c1=cur;
    char c2=dest;
    if ($NPV_{cur}$(dest)<>NULL)
```

```
        route(c1,c2)
    else
        {c3=relay node; route(c1,c3); route(c3,c2) }
  }
int route(cur, dest)
  {if ($NPV_{cur}$(dest)=1) send to this node and return (SUCCESS);
    k=count(nei(cur,C)=1);
    for (j=1; j≤k; j++)
        if ($NPV_j$(dest)=$NPV_{cur}$(dest)−1)
            {send message to this neighboring node j;
                return(SUCCESS);}
  }
```

Example 5. Based on OFTRS, illustrate the message routing path from node (0000) to node (1110) in Fig.3.

Solution: In Fig.3, suppose a message needs to be sent from node (0000) to node (1110). Since NPV_{0000}(1110)=3, node (0000) has 2 non-faulty adjacent nodes (0010) and (0100), and NPV_{0010}(1110)=2, NPV_{0100}(1110)=2, so node (0000) can send the message to either node (0100) or (0010). If the message is sent to node (0100), since node (0100) has 3 non-faulty adjacent nodes (0000), (0101), (1100), and NPV_{0000}(1110)=3, NPV_{0101}(1110)=3, NPV_{1100}(1110)=1, from the routing algorithm, the message can be only delivered to node (1100). After the message reaches to node (1100), since NPV_{1100}(1110)=1, eventually the node (1100) will send the message to node (1110). Similarly, if a message is sent to node (0010), the routing path is (0000)→(0010)→(1010)→(1110).

Example 6. Based on Relay Node Technique and OFTRS, please draw the message path from node (0000) to node (1110) in Fig.4.

Solution: According to Table 2, (0000)→(0100)→(1100)→(1000)→(1001)→[(0000)'s relay node (1101)]→(0101)→(0111)→(0011)→(0010)→[(1101)'s relay node (1010)]→(1110). The path length is 11.

Table 2 NPV values for all nodes in Fig.4

Node	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0	[9]	[9]	[8]	1	[6]	[8]	[7]	3	4	[10]	[9]	2	⑤	[11]	[10]
0001	[9]	0	2	1	[8]	3	3	2	[6]	⑤	3	2	[7]	4	4	3
0010	[9]	2	0	1	[8]	3	3	2	[6]	⑤	1	2	[7]	4	2	3
0011	[8]	1	1	0	[7]	2	2	1	⑤	4	2	1	[6]	3	3	2
0100	1	[8]	[8]	[7]	0	⑤	[7]	[6]	2	3	[9]	[8]	1	4	[10]	[9]
0101	[6]	3	3	2	⑤	0	2	1	3	2	4	3	4	1	5	4
0110	[8]	3	3	2	[7]	2	0	1	⑤	4	4	3	[6]	3	5	4
0111	[7]	2	2	1	[6]	1	1	0	4	3	3	2	⑤	2	4	3
1000	3	[6]	[6]	⑤	2	3	5	4	0	1	[7]	[6]	1	2	[8]	[7]
1001	4	5	5	4	3	2	4	3	1	0	[6]	⑤	2	1	[7]	[6]
1010	[10]	3	1	2	[9]	4	4	3	[7]	[6]	0	1	[8]	⑤	1	2
1011	[9]	2	2	1	[8]	3	3	2	[6]	⑤	1	0	[7]	4	2	1
1100	2	[7]	[7]	[6]	1	4	[6]	⑤	1	2	[8]	[7]	0	3	[9]	[8]
1101	5	4	4	3	4	1	3	2	2	1	⑤	4	3	0	[6]	5
1110	[11]	4	2	3	[10]	⑤	5	4	[8]	[7]	1	2	[9]	[6]	0	1
1111	[10]	3	3	2	[9]	4	4	3	[7]	[6]	2	1	[8]	⑤	1	0

7 Evaluation

To illustrate the effectiveness of computing all optimal and sub-optimal paths of the proposed Node Path Vector (NPV) and its routing scheme, we use Fig.4 as an example and compare the computed path information using different existing schemes. Figure 4 is a 4-dimensional hypercube system with 15 faulty links.

Table 3 shows the lengths of optimal and sub-optimal paths for Fig.4 based on ESVs and EOPMs routing algorithms[*]. Finally, Table 2 shows all nodes' NPV in Fig.4 after 4 rounds of synchronous message exchanges and adopting relay node technique. In this table, the column node corresponding to value ⑤ is the relay node for the corresponding row node. The length values without bracket are the lengths of directly computed optimal or sub-optimal paths. Those values with bracket are the lengths of the indirectly obtained shortest paths considering relay node. It can be shown that these values reflect the entire optimal or sub-optimal paths actually existing.

Table 3 Path information based on ESVs and EOPMS routing algorithms for Fig.4

Node	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0				1								2			
0001		0	2	1		$\underline{3}$	$\underline{3}$	2				2				$\underline{3}$
0010		2	0	1		$\underline{3}$	$\underline{3}$	2			1	2			2	$\underline{3}$
0011		1	1	0		2	2	1			2	1				2
0100	1				0				2				1			
0101		$\underline{3}$	$\underline{3}$	2		0	2	1		2		$\underline{3}$		1		
0110		$\underline{3}$	$\underline{3}$	2		2	0	1				$\underline{3}$				
0111		2	2	1		1	1	0				2		2		$\underline{3}$
1000					2				0	1			1	2		
1001						2			1	0			2	1		
1010		$\underline{3}$	1	2				$\underline{3}$			0	1			1	2
1011		2	2	1		$\underline{3}$	$\underline{3}$	2			1	0			2	1
1100	2				1				1	2			0			
1101						1		2	2	1				0		
1110			2								1	2			0	1
1111		$\underline{3}$	$\underline{3}$	2				$\underline{3}$			2	1			1	0

It can be easily found that ESVs and EOPMs essentially omit a large number of optimal and sub-optimal paths. Table 4 shows the comparison results. Among these three schemes, OFTRS computes all optimal or sub-optimal paths. While ESVs computes 31% paths, paths equal to or longer than 3 have all been omitted (almost 70% omission rate). EOPMs can compute 40% paths (60% omission rate). The omission rate for paths being 3 is 47%. Paths longer than 3 are all omitted. This means that the system may not find a path even there are obvious paths.

Table 4 Comparison between Table 2 and Table 3

Path length	1	2	3	4	5	6	7	8	9	10	11
Actual path number	34	40	38	30	24	20	18	16	12	6	2
Path number by OFTRS	34	40	38	30	24	20	18	16	12	6	2
Path number by ESVs	34	40	0	0	0	0	0	0	0	0	0
Path number by EOPMs	34	40	20	0	0	0	0	0	0	0	0

From the perspective of memory space cost, for each node, ESVs needs to retain n-bits information, EOPMs needs to retain n^2-bits information, and NPV require to stores information of 2^n entries. With the continuous improvement of computing and memory technology, this overhead can be accepted. For example, when n=20, there can be one million nodes, the memory space requirement is a few mega bytes. As current machines are configured with memory of gigabytes, the memory requirement for NPV can be easily satisfied.

8 Conclusions

We have introduced the Generalized Hypercube System Q_n and proposed Node Path Vectors (NPVs) in a Generalized Hypercube Network and its calculation rules. By using Relay Node technique, we are able to significantly reduce the computation complexity of computing NPV. We further present an Optimal Fault-Tolerant Routing Scheme (OFTRS) based on NPV. Since NPV has captured the information of all the optimal and

[*] Since ESVs and EOPMS improve on SVs and OPMs respectively, we only show the path information based on ESVs and EOPMs in table 3. Those values with an underline are for EOPMs only. Others are for both ESVs and EOPMs.

sub-optimal path of a Hypercube System with a large number of faulty links, it will not omit any optimal and sub-optimal routing path. It can ensure that a message traverses through the shortest path as long as one such path exists, thus achieving a highly efficient fault-tolerant routing. As exemplified, it can find more than 60% more optimal and sub-optimal routing paths in a large number of faulty Generalized Hypercube than the existing schemes. Meanwhile, this is a distributed scheme based only on non-faulty neighbor node information, thus it possesses high reliability and practical applicability.

References:

[1] Lee TC, Hayes JP. A fault-tolerant communication scheme for hypercube computers. IEEE Trans. on Computers, 1992,41(10): 1242−1256.

[2] Chiu GM, Wu SP. A fault-tolerant routing strategy in hypercube multicomputers. IEEE Trans. on Computers, 1996,45(2):143−155.

[3] Chen MS, Shin KG. Depth-First approach for fault-tolerant routing in hypercube multicomputers. IEEE Trans. on Parallel and Distributed Systems, 1990,1(2):152−159.

[4] Jong K, Shin KG. Deadlock-Free fault-tolerant routing in injured hypercubes. IEEE Trans. on Computers, 1993,42(9):1078−1088.

[5] Chen MS, Shin KG. Adaptive fault-tolerant routing in hypercubes multicomputers. IEEE Trans. on Computers, 1990,39(12): 1406−1416.

[6] Wu J. Reliable unicasting in faulty hypercubes using safety levels. IEEE on Computers, 1997,46(2):241−247.

[7] Wu J. Adaptive fault-tolerant routing in cube-based multicomputers using safety vectors. IEEE Trans. on Parallel and Distributed Systems, 1998,9(4):321−334.

[8] Gao F, Li ZC, Min YH, Wu J. A fault-tolerant routing strategy based on extended safety vectors in hypercube multicomputers. Chinese Journal of Computers, 2000,23(3):248−254 (in Chinese with English abstract).

[9] Gao F, Li ZC. Fault-Tolerant routing in hypercube multicomputers using optimal path matrices. Chinese Journal of Computers, 2000, 23(3):242−247 (in Chinese with English abstract).

[10] Al-Sadi J, Day K, Ould-Khaoua M. A new fault-tolerant routing for *k*-ary *n*-cubes. Microprocessors and Microsystems, 2001,25(5): 239−246.

[11] Al-Sadi J, Day K, Ould-Khaoua M. Probability vectors: A new fault-tolerant routing algorithm for *k*-ary *n*-cubes. In: Proc. of the 17th ACM Symp. on Applied Computing (SAC 2002). Madrid: IEEE Society, 2002. 830−834.

[12] Tian SH. A fault-tolerant routing strategy based on extended optimal path matrices in hypercube multi-computers. Chinese Journal of Computers, 2002,25(1):87−92 (in Chinese with English abstract).

[13] Tian SH, Cai ZX, Tian Z, Tian M. Utilizing OPSBOPMs to realize a fault-tolerant routing strategy in hypercube system. Journal of Central South University of Technology (Natural Science Edition), 2002,33(6):637−642 (in Chinese with English abstract).

[14] Wang GJ, Chen JE, Chen SQ. Designing efficient routing algorithms on hypercube networks with a large number of faulty nodes. Chinese Journal of Computers, 2001,24(9):909−916 (in Chinese with English abstract).

[15] Wang L, Lin YP, Chen ZP, Wen X. Fault-Tolerant routing for hypercube multi-computers based on maximum safety-path matrices. Journal of Software, 2004,15(7):994−1003 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/994.htm

[16] Wang L, Lin YP, Chen ZP, Wen X. Fault-Tolerant routing based on safety path vectors in hypercube system. Journal of Software, 2004,15(5):783−790 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/783.htm

[17] Wu J, Gao F, Li ZC, Min YH. Optimal, and reliable communication in hypercubes using extended safety vectors. IEEE Trans. on Reliability, 2005,54(3):402−411.

[18] Schlosser M, Sintek M, Decker S, Nejdl W. HyperCuP—Hypercubes, ontologies and efficient search on P2P networks. In: Koubarakis M, ed. Proc. of the Int’l Workshop on Agents and Peer-to-Peer Computing (AP2PC). Bologna: Springer-Verlag, 2002. 112−124.

[19] Castro M, *et al*. An evaluation of scalable application-level multicast built using peer-to-peer overlay networks. In: Proc. of the IEEE INFOCOM. San Francisco: IEEE Society, 2003. 1510−1520.

附中文参考文献:

[8] 高峰,李忠诚,闵应骅,吴杰.超立方体多处理机系统中基于扩展安全向量的容错路由.计算机学报,2000,23(3):248−254.

[9] 高峰,李忠诚.用最优通路矩阵实现超立方体多处理机系统的容错路由.计算机学报,2000,23(3):242−247.

[12] 田绍槐.超立方体多处理机系统中基于扩展最优通路矩阵的容错路由.计算机学报,2002,25(1):87−92.

[13] 田绍槐,蔡朝曦,田争,田敏.用 OPSBOPMs 实现超立方体系统的容错路由.中南工业大学学报(自然科学版),2002,33(6): 637−642.

[14] 王国军,陈建二,陈松乔.具有大量错误节点的超立方体网络中的高效路由算法的设计与讨论.计算机学报,2001,24(9):909−916.

[15] 王雷,林亚平,陈治平,文学.超立方体中基于极大安全通路矩阵的容错路由.软件学报,2004,15(7):994−1003. http://www.jos.org.cn/1000-9825/15/994.htm

[16] 王雷,林亚平,陈治平,文学.超立方体系统中基于安全通路向量的容错路由.软件学报,2004,15(5):783−790. http://www.jos.org.cn/1000-9825/15/783.htm

TIAN Shao-Huai was born in 1948. He is a professor at the Hunan Taxation College and Hunan University. His research areas are fault-tolerant computing, fault diagnosis and management information system, etc.

ZHANG Da-Fang was born in 1959. He is a professor at the Hunan University and a CCF senior member. His research areas are dependable systems and networks, network measure, software fault-tolerant, etc.

LU Ying-Ping was born in 1966. He was an associate professor at the Hunan University. Presently he works as a postdoctoral at the University of Minnesota and SGI. His research areas are high performance computing, storage network and object storage, quality of service, etc.