# An Efficient Algorithm for Hamiltonian Path Embedding of -Ary -Cubes Under the Partitioned Edge Fault Model

Hongbin Zhuang ©, Xiao-Yan Li D, Jou-Ming Chang (C), and Dajin Wang (C)

Abstract-The -ary -cube is one of the most important interconnection networks for building network-on-chips, data center networks, and parallel computing systems owing to its desirable properties. Since edge faults grow rapidly and the path structure plays a vital role in large-scale networks for parallel computing, fault-tolerant path embedding and its related problems have attracted extensive attention in the literature. However, the existing path embedding approaches usually only focus on the theoretical proofs and produce an -related linear fault tolerance since they are based on the traditional fault model, which allows all faults to be adjacent to the same node. In this paper, we design an efficient fault-tolerant Hamiltonian path embedding algorithm for enhancing the fault-tolerant capacity of -ary -cubes. To facilitate the algorithm, we first introduce a new conditional fault model, named Partitioned Edge Fault model (PEF model). Based on this model, for the -ary -cube with and odd , we explore the existence of a Hamiltonian path in with large-scale edge faults. Then we give an algorithm, named HP-PEF, to embed the Hamiltonian path into under the PEF model, where is the number of nodes in . The performance analysis of HP-PEF shows the average path length of adjacent node pairs in the Hamiltonian path constructed by HP-PEF. We also make comparisons to show that our result of edge fault tolerance has exponentially improved other known results. We further experimentally show that HP-PEF can support the dynamic degradation of average success rate of constructing Hamiltonian paths when increasing faulty edges exceed the fault tolerance.

Index Terms- -ary -cubes, algorithm, fault-tolerant embedding, Hamiltonian path, interconnection networks.

# NOMENCLATURE

NoC Network-on-Chip

TSV Through silicon via

IC Integrated circuit

VLSI Very large scale integration

TRC Tours routing chip

DCN Data center network

H-path Hamiltonian path

PEF Partitioned edge fault

-Ary -cube

APL Average path length

SD Standard deviation

FT Fault tolerance of when embedding the H-path under the traditional model

FP Fault tolerance of when embedding the H-path under the PEF model

ASR Average success rate

# I. INTRODUCTION

NETWORK-ON-CHIPS (NoCs) have emerged as a promis- ing fabric for supercomputers due to their reusability and scalability [1], [2]. This fabric allows a chip to include a large number of computing nodes and effectively turn it into a tiny supercomputer. Therefore, it alleviates the bottlenecks faced by the further development of supercomputers. With the rapidly increasing demand for computing capacity, the number of on-chip cores increases quickly, which results in a high average internode distance in two-dimensional NoCs (2D NoCs). Consequently, 2D NoCs exhibit high communication delay and power consumption as the scale of networks increases [3]. Hence, 3D have been designed to solve the scalability problem of 2D NoCs. In 3D NoCs, the so-called through silicon via (TSV) links are used to connect various planes or layers. Though the fabrication cost of TSV links is quite high, 3D NoCs can reduce the probability of long-distance communication while still maintaining high integration density.

However, since the additional expense is required for incorporating more processing nodes, NoCs demand a robust fault tolerance. For example, 3D integrated circuit (IC) fabrication technology improves the power density of modern chips, which results in a thermal-intensive environment for 3D NoCs. High core temperatures reduce chip lifetime and mean time to failure, as well as resulting in low reliability and high cooling costs. The faults in NoCs are mainly divided into two categories, namely, transient faults and permanent faults. Generally speaking, permanent fault deserves more attention [4] since it seriously affects the transmission of more packets. With the rapid increase in the number of processing nodes, NoCs may encounter many permanent faults problems [5], and more reliability threats accompanied by permanent faults will also appear [6]. Therefore, we mainly discuss the permanent faults in this paper.

[[1]](#footnote-22)

A well-designed fault-tolerant routing algorithm can address the fault tolerance challenges of NoCs by bypassing the faults when delivering packets. Routing algorithms should be not only fault-tolerant but also deadlock-free. The Hamiltonian path (H-path for short) strategy is a powerful tool for deadlock avoidance. Since the H-path traverses every node in the network exactly once and contains no cycle structure, the deadlock can be easily prevented by transmitting the packets along the H-path [7], [8], [9], [10], [11], [12], [13]. In recent years, this excellent strategy is utilized for designing fault-tolerant routing algorithms. For instance, the HamFA algorithm [7] is one of the most famous fault-tolerant routing algorithms using the H-path strategy. It constructs two directed subnetworks through the -path strategy and limits packets to be routed in a single subnetwork so that the deadlock can be avoided. Simultaneously, HamFA can tolerate almost all one-faulty links. The FHOE algorithm [8] is also a fault-tolerant routing algorithm based on the H-path strategy for 2D NoCs. It fully combines the advantages of traditional odd-even turn model and HamFA strategy, and consequently can provide higher adaptivity and more choices of minimal paths compared to HamFA. Considering the importance of fault tolerance and extensive applications of the H-path in NoCs, it’s natural to investigate the existence of the H-path in NoCs (i.e., the problem of embedding the H-path into NoCs), especially when faults occur (i.e., the fault-tolerant problem of embedding the H-path into NoCs). However, it is well-known that the problem of embedding an -path into a network is NP-complete, even when no fault exists.

NoCs usually take interconnection networks as their underlying topology, which inherently affects the performance of NoCs. The -ary -cube is one of the most important interconnection networks for building NoCs owing to its desirable properties, such as regularity, recursive structure, node symmetry, edge symmetry, low-latency, and ease of implementation [14]. The two associated parameters and in provide it the ability to satisfy structural needs in a variety of circumstances. A commercial VLSI chip named the Tours Routing Chip (TRC) was designed early to perform wormhole routing in an arbitrary -ary -cube [15]. Furthermore, many fault-tolerant deadlock-free routing algorithms have been developed in -based NoCs [16], [17],[18]. The desirable properties of have even attracted a lot of research actually to build data center networks (DCNs), such as CamCube [19], NovaCube [20], CLOT [21], and Wave-Cube [22]. Though the scale of DCNs is much larger than that of NoCs, -ary -cubes can easily cope with it. It’s worth pointing out that stronger fault tolerance is necessary for the DCN since it possesses a lot of servers. Moreover, a lot of well-known parallel computing systems like iWarp [23], J-machine [24], Cray T3D [25], Cray T3E [26], and IBM Blue Gene/L [27] all have adopted -ary -cubes as their underlying topologies. These - based architectures usually have high bisection width, high path diversity, high scalability, and affordable implementation cost.

In order to apply the attractive H-path structure in with as many faults as possible, the fault-tolerant problem of embedding the H-path into has been extensively investigated in [28], [29],[30],[31],[32],[33],[34]. A network is Hamiltonian-connected if an H-path exists between any two nodes in . Also, is -edge fault-tolerant Hamiltonian-connected provided it is Hamiltonian-connected after removing arbitrary edges in . Yang et al. [32] proved that for any odd integer is(2n - 3)-edge fault-tolerant Hamiltonian-connected. Stewart and Xiang [33] proved that for any even integer , there is an -path between any two nodes in different partite sets in with at most faulty edges. Yang and Zhang [34] recently showed that for every odd integer admits an -path between any two nodes that avoids a set of faulty edges and passes through a set of prescribed linear forests when . All the above results are obtained under the traditional fault model, which doesn’t exert any restriction on the distribution of faulty edges. However, Yuan et al. [35] and et al. [36] respectively pointed out that this model has many flaws in the realistic situation since it ignores the fact that it’s almost impossible for all faulty nodes (resp. faulty edges) to be adjacent to the same node simultaneously (unless that the node fails). In other words, the fault tolerance assessment approaches under the traditional fault model seriously underestimate the fault tolerance potential of .

The conditional fault model was proposed for tolerating more faulty edges by restricting each node to be adjacent to at least two fault-free edges. Under this model, Wang et al. [29] proved that for any even integer , there is an -path between any two nodes in different partite sets in with at most conditional faulty edges. Though the fault tolerance they obtained is about twice that under the traditional fault model, it remains linearly correlated with . In addition, all the literature mentioned above only provides theoretical proofs about the existence of the -path in , while executable fault-tolerant -path embedding algorithms and their performance analysis are missing. Thus, this may hinder the practical application of the -path on .

In this paper, we pay more attention to the distribution pattern of faulty edges in each dimension of . This consideration is based on the fact that various dimensions of usually possess different faulty features in practical fields. For example, to minimize the fabrication cost of TSV links, -based 3D are often designed with only partial connection in the vertical dimension (i.e., partial TSVs) [37], [38]. Particularly, the TSV density of 3D NoCs was suggested for only 12.5% in [38]. It implies that only 12.5% of vertical links are available, and 87.5% of vertical links can be deemed faulty. In other words, many missed links exist in one dimension inherently when is utilized for building 3D NoCs. In this case, we can deem the vertically partially connected NoC topology as a with many faulty links concentrated at the same dimension.

Based on the above concerns, for a class of networks that exhibit different faulty features in each dimension, we introduce another fault model, named the partitioned edge fault model (PEF model for short), to help such networks achieve a better fault-tolerant capacity. In essence, this model imposes different restrictions on faulty edges in each dimension according to flawed features. In fact, these restrictions are similar to the concept recently proposed by Zhang et al. [39], which pointed out that restricting the number of faulty edges in each dimension is quite important for reflecting the actual fault-tolerant capacity of a network and can be utilized to improve network edge connectivity. Thus, we utilize the PEF model to explore the fault tolerance potential of when embedding an H-path into with and odd and evaluate the performance of our approach. Our contributions are presented as follows:

1) We propose a new fault model, the PEF model, to improve the fault tolerance of when we embed an H-path into the faulty .

2) Under the PEF model, we provide a theoretical analysis for proving the existence of the -path in , where is a PEF set (defined in Section III) such that

3) Based on the obtained theoretical results, we design an algorithm, named , for embedding the - path into under the PEF model, where is the number of nodes in . To our knowledge, this is the first time that an algorithm is not only proposed and proved correct, but also actually implemented, for H-path embedding into an edge-faulty .

The implementation of the algorithm afforded us the ability to observe some features of the generated -paths. For example, if an edge connecting nodes and became faulty, then the path length of and in the generated -path can be an indicator of how important the missed edge is. By experimenting with the algorithm, we gather the data of average path lengths for all edges in the generated -path.

Our algorithm is shown to outperform all known similar works in terms of tolerated faulty-edges. In particular, compared to [32], the improvement is from linear(2n - 3) to exponential . We also show that HP-PEF can support the dynamic degradation of average success rate of constructing required -paths even when increasing faulty edges exceed .

Organization: The rest of this paper is organized as follows. In Section II, we provide the preliminaries used throughout this paper. In Section III, we first present the definition of the PEF model, and then give the theoretical proof related to the existence of the H-path in under the PEF model. In Section IV, we design the fault-tolerant H-path embedding algorithm HP-PEF based on the theoretical basis in Section III and offer a detailed example for illuminating the execution process of HP-PEF. In Section V, we evaluate the performance of our method by implementing computer programs. Section VI concludes this paper.

# II. Preliminaries

# A. Terminologies and Notations

For terminologies and notations not defined in this subsection, please refer to the reference [40]. An interconnection network can be modeled as a graph , where represents its node set and represents its edge set. The notations and denote the size of and , respectively. Given a graph , if and , then is a subgraph of . Given a node set , the subgraph of induced by , denoted by , is a graph with the node set and edge set . Let be a faulty edge set of , and be the graph with the node set and the edge set . Given a positive integer , we denote the set as . Moreover, let when and . A graph is called a path if nodes are distinct and is an edge of with . The length of is the number of the edges in . If , then is a Hamiltonian path (H-path for short) of .

# B. -Ary -Cube

Definition II.1. (See [33]). The -ary -cube is a graph with the node set such that two nodes and are adjacent in if and only if there is an integer satisfying and for every . In this case, such an edge(u, v)is called an -dimensional edge for , and the set of all -dimensional edges of is denoted by , or for short.

Hereafter, for brevity, we will omit "(mod )" in a situation similar to the above definition. By Definition II.1, is -regular and contains nodes. In addition, is edge symmetric [41]. The can be partitioned into disjoint subgraphs (abbreviated as along the -dimension for . All these subgraphs are isomorphic to . Given a faulty edge set , let . If there is no ambiguity, we abbreviate to . Each node of has the form . The node is the neighbor of in if , which is denoted by . To distinguish the positions of the subgraphs where different nodes are located, let . That is, . Although the values of and are equal, the notation mainly focuses on the position rather than the dimension of node . Let (abbreviated as ) be the subgraph induced by node set with . Moreover, we have when , and taken modulo , we have when . A path in is denoted by with . For convenience, we abbreviate to when , and to when by taking modulo .

Fig. 1 shows , and . We color each edge according to its dimension. In Fig. 1(c), we partition into 3 disjoint subgraphs , and along the 2-dimension. The subgraph is induced by node set . The node is adjacent to two nodes and . In addition, and . It’s easy to see that and .

In particular, the can be deemed a grid with wraparound edges, where a node is indexed by its row and column . Let be two row indices with . If , we define the row torus to be the subgraph of induced by the nodes on rows , and particularly, all column edges between nodes on row and row are removed when and . Otherwise, if , we define the row torus to be the subgraph of induced by the nodes on rows , and particularly, all column edges between nodes on row and row are removed when . Fig. 2 depicts the , which is obtained by removing the column edges between nodes on row 0 and row 4 from . Throughout, we assume that the addition of row or column indices is modulo .

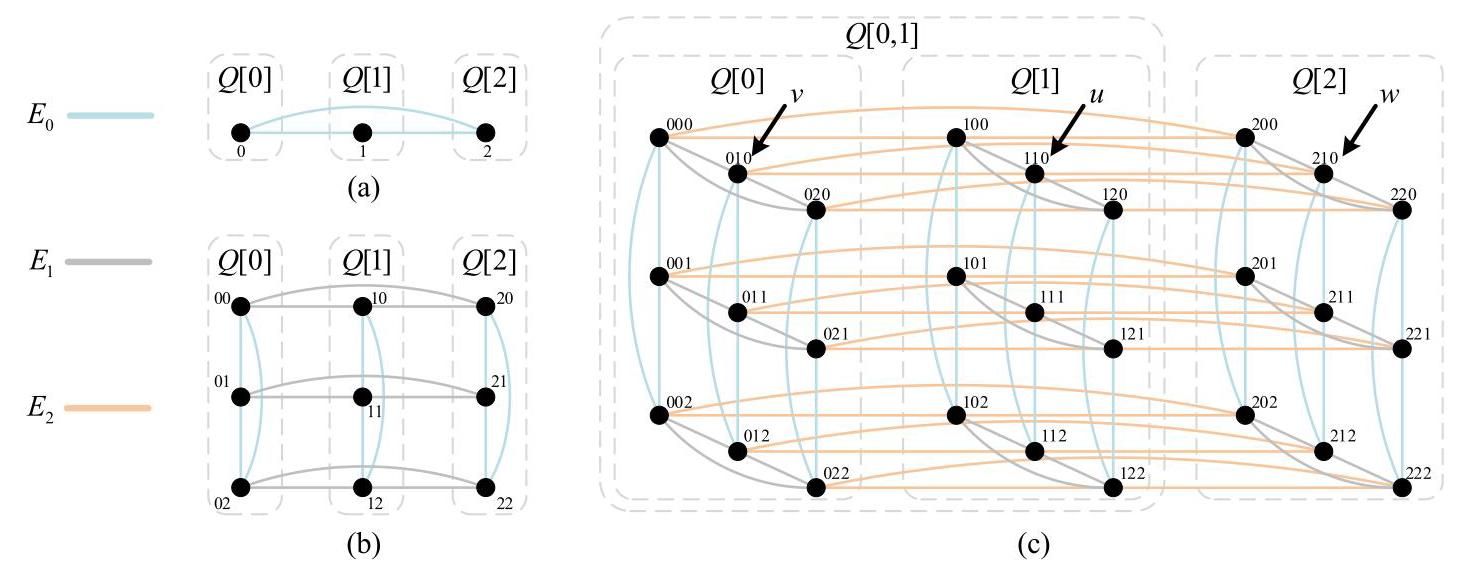


Fig. 1. The structures of (a) ; (b) ; (c) .

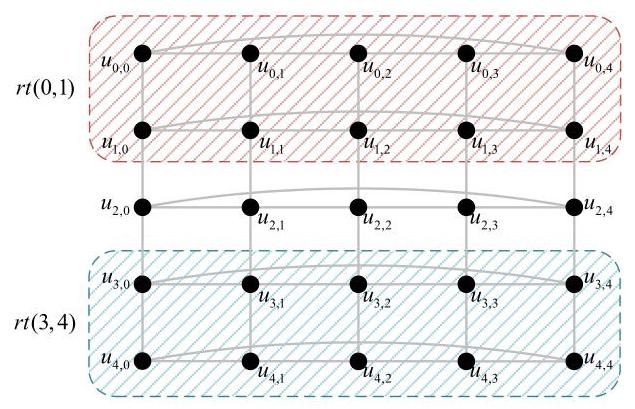


Fig. 2. The structure of .

For the row torus with , we define the four types of paths in as follows. The notations of these paths are derived from the shape of their pictorial representations, where .

In particular, if , then

# III. THEORETICAL BASIS FOR EMBEDDING THE HAMILTONIAN PATH INTO -ARY -CUBES

In this section, we establish the theoretical basis for embedding the H-path into -ary -cubes under the PEF model. That is, we will prove that an -path can be found in a -ary -cube in the presence of a partitioned edge fault set, described below.

Let be a faulty edge set in , and let with . We set such that . The faulty edge set is a partitioned edge fault set (PEF set for short) if and only if for each , where is a function of or a fixed value.

For a PEF set , since is recursively constructed with edge symmetry, we can utilize the inductive method to analyze the Hamiltonian property of by partitioning it into subgraphs along a dimension we expected. Consequently, provided that is Hamiltonian-connected, the exact values of can be derived by analyzing the number of faulty edges that can be tolerated when the H-path passes through two consecutive subgraphs.

The following result is provided for dealing with the base case of our forthcoming inductive proof.

Lemma III.1. (See [42]) For odd , let with . Then is Hamiltonian-connected.

Theorem III.2. For and odd , let be a PEF set satisfying the following conditions:

1) ;

2) for each ;

3) and .

Then, is Hamiltonian-connected.

Proof. The proof is by induction on . When , by Lemma III.1, the theorem holds obviously. For , assume this theorem holds for all ’s with . Therefore, what we need to prove is that this theorem holds for .

Since is edge symmetric, without loss of generality, let . That is, . Along the(n - 1)-dimension, we divide into disjoint subgraphs , all of which are isomorphic to . Let and be arbitrary two vertices of with and . By the arbitrariness of and , suppose that .

Let with and . Moreover, for each , let such that . According to the recursive nature and conditions 2) and 3), when , we have , and . When , we have

In addition, , and for each . Therefore, every with is Hamiltonian-connected. That is, when , there exists an -path in between and .

Without loss of generality, suppose that Since and is odd, for all .

Claim 1: Suppose that there exists an -path in between any two distinct nodes in . When

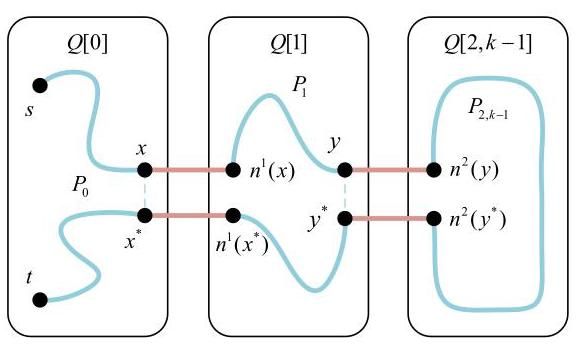


Fig. 3. The constructions in Case 1.1 of Theorem III.2.

2, there exists at least one edge such that . And when 1, there exists at least one edge such that .

The length of is . Then there exist mutually disjoint edges on . When , since , there exists at least one edge such that . Analogously, when , we can also find at least one edge such that . Then the claim holds.

Next, we discuss the following cases separately.

Case 1: .

Case 1.1: .

Since is Hamiltonian-connected, there exists an H-path in between and . By Claim 1, there exists at least one edge of such that . Similarly, since is Hamiltonian-connected, there exists an -path in between and . By Claim 1, there exists at least one edge of such that . By constantly iterating in this way, we can obtain an H-path in between and . Thus, , forms the required H-path between and in (see Fig. 3).

Case 1.2: .

Since with and odd , there exists one node such that , and . Since is Hamiltonian-connected, there exists an -path in between and . If , since is Hamiltonian-connected, there exists an -path in between and ; otherwise, if , proceeding iteratively in this manner can construct an -path in between and . If , then forms the required H-path between and in ; otherwise, by Claim 1, there exists one edge such that . Similar to Case 1.1, we can construct an H-path in between and . Therefore, forms the required -path between and in (see Fig. 4).

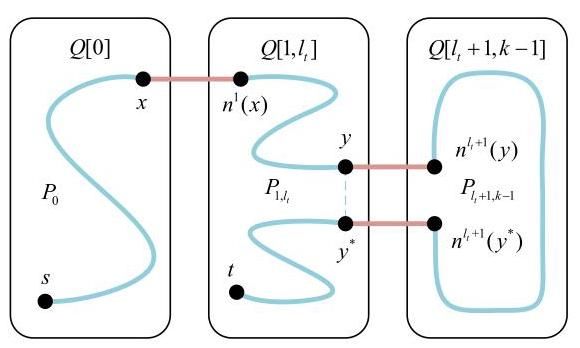


Fig. 4. The constructions in Case 1.2 of Theorem III.2.

Case 2: .

When , we can construct the required -path similar to Case 1. Then we discuss the case of .

Similar to Case 1, we can construct an H-path in between and . If the part of in is constructed by the method similar to Case 1.1, by the proof of Case 1.1, let be the edge in satisfying . By Claim 1, there exists one edge (or if exists) and . Similar to Case 1.1, we can construct an H-path in between and . If then forms the required -path between and in .

Otherwise, we have and thus . In this situation, the -path must be constructed by the manner in Case 1.1. It implies that and there exists an H-path in between and , which passes through the edge . If , then

Since with and odd 3, there exists one edge such that and . Otherwise, if , since , then there exists at least one edge such that and . Thus, there exists at least one edge such that and or .

Note that the four edges , . If , , by Case 1.1, we can construct an H-path in between and . Similar to Case 1.1, we can construct an H-path in between and . Then forms the required -path between and in . If , by Case 1.1, we can construct an -path in between and . Similar to Case 1.1, we construct an H-path in between and . Then forms the required -path between and in (see Fig. 5).

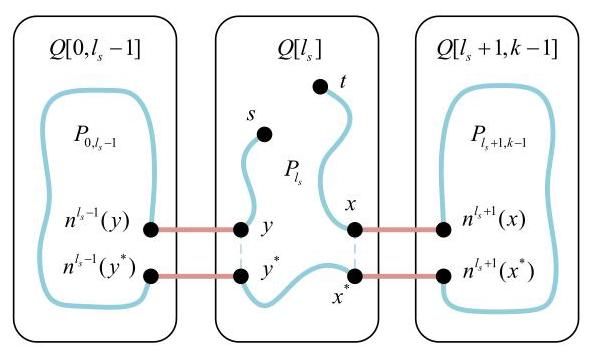


Fig. 5. The constructions in Case 2 of Theorem III.2.

# IV. FAULT-TOLERANT HAMILTONIAN PATH EMBEDDING ALGORITHM OF -ARY -CUBES

In this section, we present the fault-tolerant -path embedding algorithm for under the PEF model.

First, we design Algorithm 1 costing time to construct the -path in according to the theoretical proof in [42], where Procedure HP-rtFree is utilized to find the H-path in a fault-free . In Algorithm 1, we let and be two arbitrary distinct nodes in . In addition, without loss of generality, suppose that . Given an edge fault set with , by the symmetry of , suppose that is the faulty edge if it exists.

Based on Algorithm 1, we design the Algorithm HP-PEF under the PEF model. Note that the Algorithm HP-PEF is essentially based on the theoretical basis in Section III, where Procedures HP-Round and HP-Direct correspond to the constructive approaches of Case 1.1 and Case 1.2 in Theorem III.2, respectively. In addition, the fault tolerance of Algorithm HP-PEF has been determined by Theorem III.2 (i.e., the three conditions shown in Theorem III.2).

Theorem IV.1. For and odd , the algorithm HP-PEF can embed an H-path between arbitrary two nodes and into , where is a PEF set satisfying (1) ; (2) for each ; (3) and .

Proof. We prove this theorem by induction on . When , HP-PEF calls Algorithm 1 to embed the required H-path into , where is a PEF set satisfying and . Assume this theorem holds for all ’s with . Then we need to prove that this theorem holds for .

Algorithm 1: Embed an H-path into .

Input: A -ary 2-cube with odd , an edge fault

set with , two distinct nodes

and .

Output: An H-path in between and .

;

if then Implement an exhaustive search;

else

if there exists no faulty edge then

Call HP-rtFree ;

else

if then

Construct two disjoint paths avoiding the

edge from to and

from to in such that

;

Let (resp. ) be the neighbor of (resp.

in ;

Call HP-rtFree ;

else if then

Call HP-rtFree ;

Merge into through an edge of on

row 2, which isn’t ;

else

Let be the -path avoiding the edge

in which begins at and

ends at that isn’t adjacent to ;

Let be the neighbor of in ;

Call HP-rtFree ;

return ;

In Lines 4-5, HP-PEF divides into disjoint subgraphs along the -dimension such that . Given an integer , let such that . Then similar to the proof of Theorem III.2, for each is a PEF set satisfying (1) ; (2) for each and . By the induction hypothesis, HP-PEF can embed an H-path between arbitrary two nodes into . Next, in Line 6, let denote the position of the subgraph which connects to the largest number of -dimensional faulty edges. The subsequent constructive process prohibits the use of the edges between and (see the cross sign in Fig. 6). In Lines 7-10, if the node (resp. is closer to along the clockwise (see the green lines in Fig. 6) than (resp. ), HP-PEF takes (resp. ) as the source node . Correspondingly, the node (resp. ) is deemed as the destination node .

In Lines 11-20, HP-PEF embeds the required H-path into by discussing three situations similar to Theorem III.2. In the constructive process, we let denote the destination subgraph, and denote the constructive direction. More specifically, (resp. ) means the constructive process will proceed along the clockwise (resp. counterclockwise). Theorem III. 2 provides a detailed proof related to the correctness of Lines 11-20 when , and . Though , and may change in each iteration, their relative positions remain unchanged, and the construction process is similar in different cases. Thus, the theorem holds.

Procedure: .

begin

if then

if and is odd then

;

else if and is even then

;

else if and is odd then

else if and is even then

if then

else

if and then

Call HP-rtFree ;

;

else

Without loss of generality, suppose that

;

Call HP-rtFree ;

Choose an edge of on

row ;

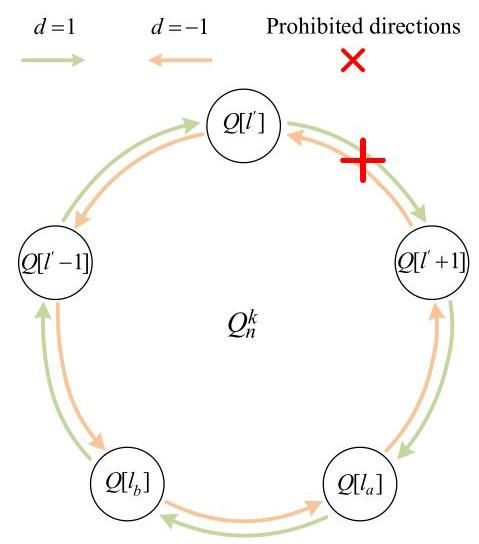


Fig. 6. The relative positions of and nodes and .

Algorithm 2: Embed an H-path into where is a PEF set (HP-PEF).

Input: A -ary -cube with and odd ,

two distinct nodes and , a PEF set satisfying

(1) ; (2) for each

; (3) and .

Output: An H-path in between and .

;

if then ;

else

Divide into disjoint subgraphs ,

along the -dimension;

// Take (resp. ) as the source node (resp.

destination node).

if then

else ;

if then // Situation 1

if then Call HP-Round ;

else Call HP-Direct ;

else if then // Situation 2

Call HP-Direct ;

else // Situation 3

Call HP-Direct ;

Select an edge such

that ;

Call HP-Round ;

;

return ;

Procedure: HP-Round(l, d, s, t).

begin

;

// Take as the destination subgraph, and as the

constructive direction.

if then

for 1 to do

Algorithm ;

Select an edge such that

;

Algorithm ;

The time complexity of HP-PEF is showed as follows.

Theorem IV.2. The time complexity of HP-PEF is , where .

Proof. When , HP-PEF calls Algorithm 1, which costs time. Next, we discuss the case of in the following.

Since HP-PEF calls HP-Round and HP-Direct frequently, we first analyse the time complexity of HP-Round and HP-Direct.

Procedure: .

begin

;

// Take as the destination subgraph, and as the

constructive direction.

if then

for 1 to do

Select a node with ,

, and ;

Algorithm ;

Algorithm ;

if then

Select an edge such

that ;

Call HP-Round ;

;

The main time costs of HP-Round and HP-Direct are on the for-loops. The for-loops execute at most iterations. Each iteration calls HP-PEF and selects a required edge or node. Calling HP-PEF to construct the H-path in every subgraph of requires . The H-path obtained contains mutually disjoint edges, and every subgraph of contains nodes. Thus, selecting a required edge or node costs time. Therefore, the time complexity of both HP-Round and HP-Direct is .

Line 4 costs time. Line 5 classifies all nodes of , costing time. Line 6 costs time. Lines 7-10 cost time. Lines 11-13 cost time. Lines 14-15 cost time. Line 17 costs time. Line 18 costs time since there exist mutually disjoint edges of in each subgraph. Line 19 costs time. Line 20 costs time.

Therefore, HP-PEF needs time.

In the following, we give a detailed example to explain how Algorithm HP-PEF embeds the H-path into under the PEF model. In the example, we set , and . Moreover, let . Obviously, we have , and . Since , the algorithm HP-PEF divides into 3 disjoint subgraphs , and along the 1-dimension. Since , we have , which implies that the subsequent constructive process doesn’t involve the edges between and . Next, the algorithm HP-PEF will take the node as the source node since is closer to along the clockwise than . Correspondingly, the node is deemed as the destination node . Since and , the algorithm HP-PEF calls the procedure HP-Direct to construct the H-path along the clockwise (i.e., ) by the approaches of Case 1.2 in Theorem III.2, where the H-path in (isomorphic to ) with is constructed by Algorithm 1. Fig. 7 shows the H-path constructed by Algorithm HP-PEF in , where the dashed lines represent the faulty edges.

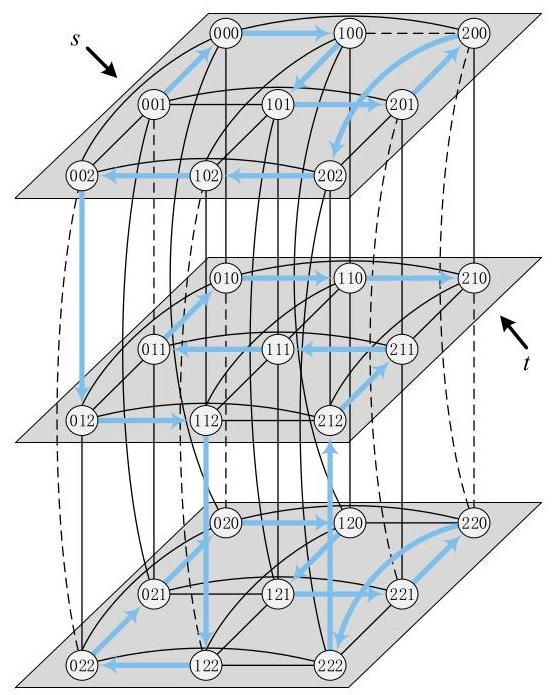


Fig. 7. The H-path constructed by Algorithm HP-PEF in , where the solid lines represent the fault-free edges and the dashed lines represent the faulty edges.

# V. Performance Analysis

In this section, we implement the algorithm HP-PEF by using Python programs and evaluate the performance of HP-PEF. We carry out the simulation by using a Intel i9-10900 CPU and 127 GB RAM under the Linux operating system.

# A. Average Path Length

Along the H-path constructed by HP-PEF, we aim to compute the path length which is the number of edges between the source and destination nodes. We are interested in computing the path length since it’s beneficial to evaluate the performance of HP-PEF for future applications to NoCs. Moreover, based on the following concerns, we choose the adjacent node pair in as the source and the destination to compute the path length:

1) When there exists no faulty edge in networks, the adjacent node pair possesses the shortest path length 1.

2) The algorithm HP-PEF can address large-scale edge faults. When the edge fault occurs, the path length of adjacent node pairs is naturally more sensitive to edge fault than other kinds of node pairs.

3) Our method focuses on the distribution pattern of edge faults in each dimension. The path length of adjacent node pairs may be followed by the distribution pattern of edge faults. It’s interesting to explore the relationship between the path length of adjacent node pairs and the number of edge faults in each dimension.

TABLE I

EXPERIMENTAL SETTINGS

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| Adjacent node pairs |  |  |  |  |  |  |  |
| Faulty edges | 8 | 24 | 48 | 80 | 33 | 112 | 353 |
| H-paths | 351 | 7750 | 58653 | 265356 | 3240 | 29403 | 265356 |

One edge corresponds to exactly one adjacent node pair. Thus, we define the dimension of an adjacent node pair as the dimension of the edge connecting it. By the definition of , there exist -dimensional edges, which implies that there exist -dimensional adjacent node pairs for each . Then has adjacent node pairs in total. As for the faulty edges, we directly set to prevent the occurrence of in the experimental process. Moreover, to make comparisons more obvious, we assume . Then we set with , and . Given a node pair(s, t) and a PEF set, the algorithm HP-PEF can generate an H-path between and . There exist different node pairs (s, t). Thus, given a PEF set, the algorithm HP-PEF can generate different H-paths. Our experimental settings are shown in Table I.

In the simulation, we first randomly generate a PEF set with for , and . Next, under the edge fault set , we utilize the algorithm HP-PEF to generate all possible -paths, that is, different -paths. And then, according to the dimension of adjacent node pairs, we compute the average path length (APL for short) of them in all -paths. More specifically, we denote the APL of each adjacent node pair in all -paths as , where is the dimension of the adjacent node pair and is the unique identification of the adjacent node pair in the -dimension. Then we compute the for each dimension as follows:

Simultaneously, we compute the standard deviation (SD for short) to evaluate the quantity of difference in path length as follows:

Fig. 8 shows the simulation results about the of with different and . We observe that is positively related to . For example, the values of and differ by only 1, and is slightly higher than . The value of with is a power function with base , and is much higher than and . In , the distance value between and exhibits a sharper growth trend with larger . Moreover, with increasing and , we observe from the results that the growth rate of is more sensitive to than . For example, in with increasing from 3 to 9, the grows rapidly from 11.1 to 154.3. However, in with increasing from 3 to 6, the grows slowly from 11.1 to 17.7. Although increment by 2 and increment by 1 on the -axis, the growth rate of in can still indicate the modest influence of on . This phenomenon may be caused by the fact that the value of is only influenced by , independent of . Fig. 9 shows the of with different and . The growth trends for and are roughly the same. However, the parameter can significantly affect the growth rate of , which is obviously different from that of . It implies that the larger and , the more dispersed the values of path length.

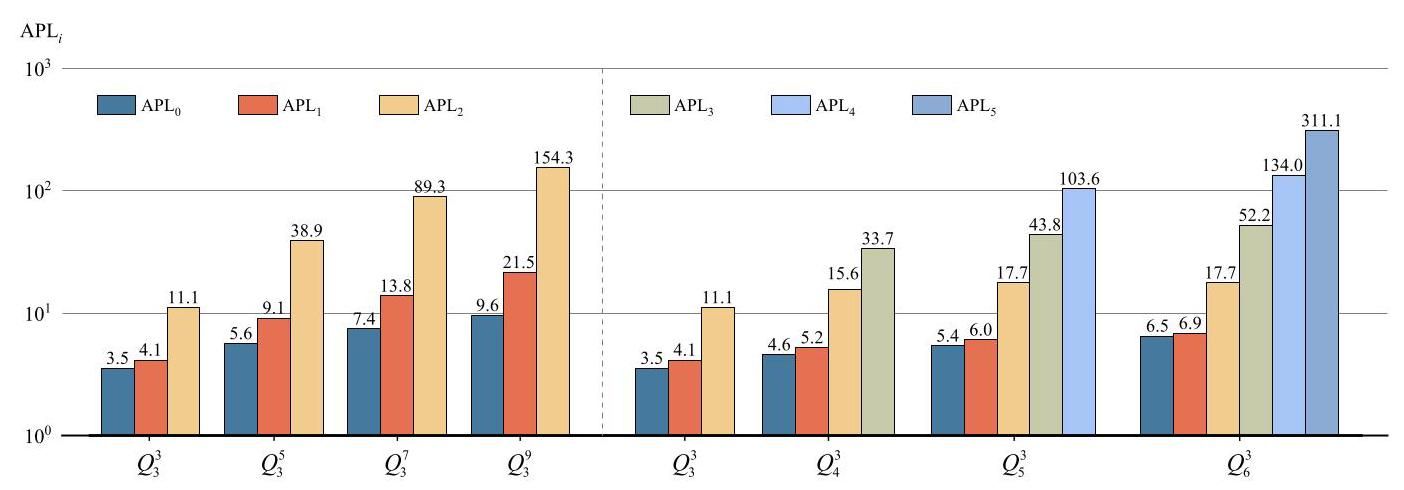


Fig. 8. The of adjacent node pairs in with different and .

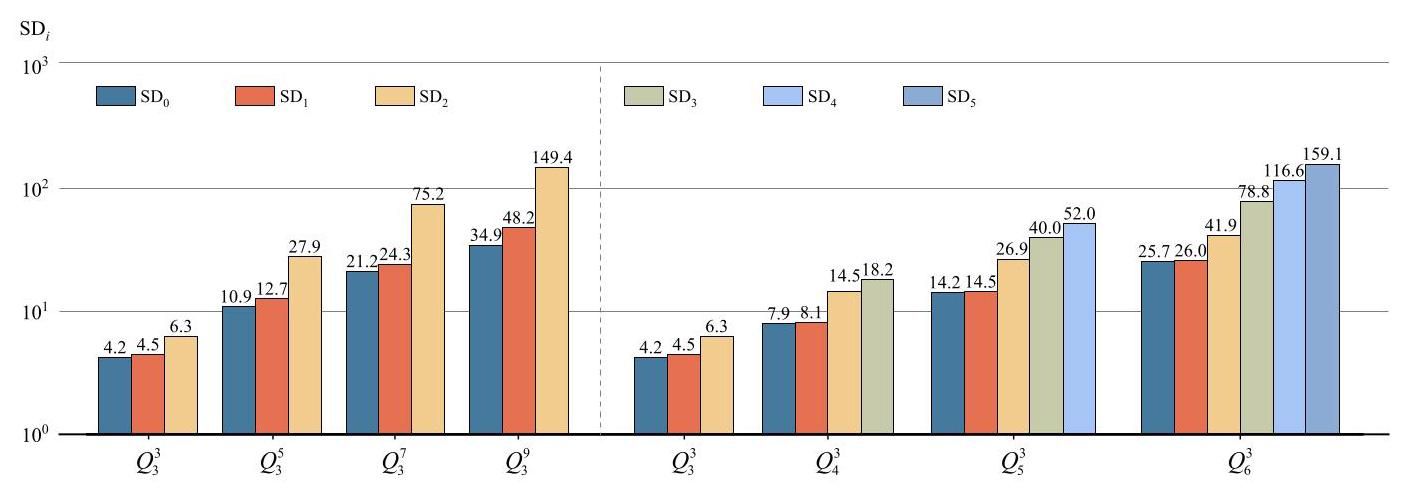


Fig. 9. The of adjacent node pairs in with different and .

Next, we compute the APL of all adjacent node pairs in as follows:

Correspondingly, the SD of all adjacent node pairs is also computed:

In Fig. 10, we show the APL and SD of . For , the SD is slightly smaller than APL. However, with increasing and , the SD grows faster than the APL. In addition, from the simulation, we are aware of the following attractive phenomena. (1) For , the ratio of APL to is about 1.3. With increases, the value of APL remains essentially the same. (2) For , the value of APL varies approximately linearly with . We list these results in Table II. As for what caused the positive correlation between and , there is currently no definite attribution. We roughly think that since the algorithm HP-PEF constructs the required -path according to the distribution pattern of edge faults in each dimension, the H-path obtained is endowed with the nature of traversing all dimensions, which results in an additional effect of on the APL.

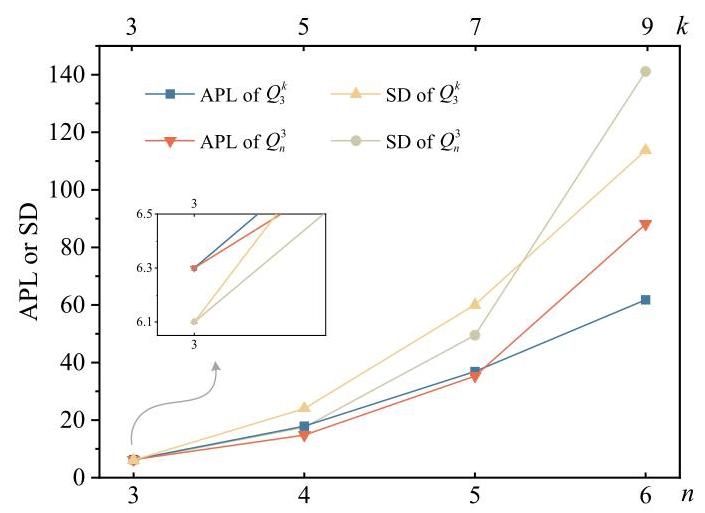


Fig. 10. The APL and SD of adjacent node pairs in with different and .

TABLE II

THE AND APL OF WITH DIFFERENT AND

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  |  |  |  |  |  |  |  |  |
|  | 8 | 24 | 48 | 80 | 8 | 33 | 112 | 353 |
| APL | 6.3 | 17.9 | 36.9 | 61.8 | 6.3 | 14.8 | 35.3 | 88.1 |
| APL | 1.3 | 1.3 | 1.3 | 1.3 | 1.3 | 2.2 | 3.2 | 4.0 |

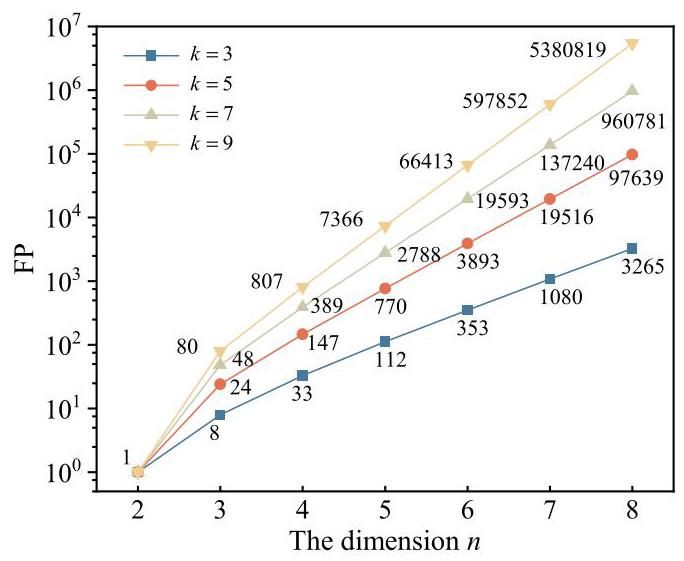


Fig. 11. The comparisons among FP with different .

# B. Fault Tolerance Analysis

1) Comparison Results: Improving the fault tolerance of when we embed an -path into the faulty is the original purpose of our work. In this subsection, we compare the fault tolerance of our method with the similar known result. First, let’s review the known results. Yang et al. [32] proved that with odd is(2n - 3)-edge fault-tolerant Hamiltonian-connected. We demonstrated that with odd is 5)-edge fault-tolerant Hamiltonian-connected under the PEF model. For convenience, let and .

Obviously, the value of FP is closely related to the parameters and . We first investigate how odd with different values affect FP in Fig. 11. Note that it takes on its -axis and takes the corresponding metrics with the exponential scale on its -axis. It can be seen that the values of FP with different are all the same when . That is because the base case of the induction in our proof is handled by the result in [42], which is irrelevant to the parameter . As increases, the distance between any two FP with different increases rapidly. It’s easy to see that FP is positively correlated with . When , FP with is ten times that with . When , FP with is more than times that with .

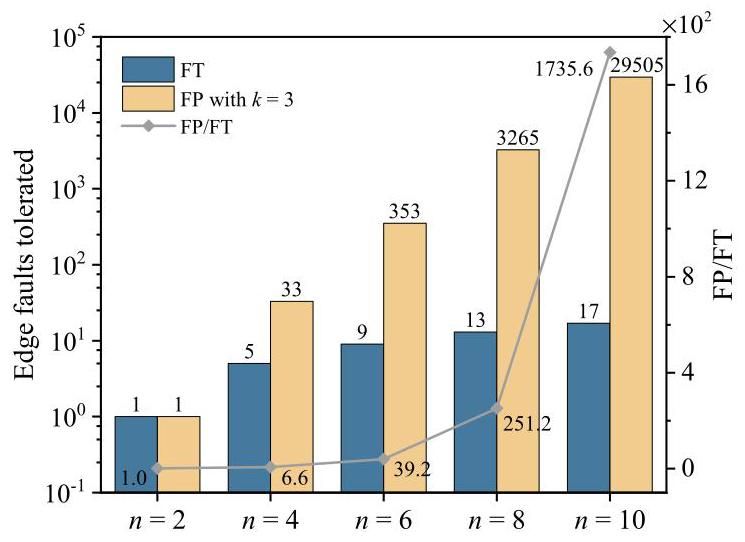


Fig. 12. The comparisons between FT and FP.

Next, we compare the values of FP and FT. Since the value of FT is irrelevant to the parameter , we fix in FP to ensure fairness when making the comparisons. Fig. 12 shows the trends of FT and FP with increasing . Note that it takes even on its -axis and takes the corresponding metrics with the exponential scale on its -axis. Since the base case of both our method and the method in [32] is handled by the result in [42], FT = FP when . However, as increases, FP and the ratio of FP to FT increase rapidly, while FT is linear on . When and , which implies . The great disparity between them comes from our complete consideration of the distribution pattern of edge faults in each dimension.

The fault tolerance of under the PEF model is positively correlated with and . This proposed model builds the positive relation between the fault tolerance with the parameter , which is its unique advantage. Even without considering the positive effect of , our method still outperforms the similar known result in terms of fault tolerance. Thus, our method can evaluate the fault tolerance of even when the edge faults are large-scale.

2) Average Success Rate: Next, we analyze the fault-tolerant capacity of the algorithm HP-PEF with increasing faulty edges. Given any PEF set satisfying the three conditions in Theorem III.2, the required H-path can be constructed by HP-PEF with probability, which has been proved in Theorem IV.1. In the following, provided that the PEF set exceeds the above conditions, by implementing computer programs, we evaluate the average success rate (ASR for short) of HP-PEF, which is the ratio of the number of successfully constructed -paths over generated instances.

Before the simulation, we randomly generate one thousand PEF sets of , denoted by with , which satisfy (1) ,(2) for each , and (3) and . Next, in each simulation, we randomly add exactly one faulty edge into such that grows by 1 with . We don’t add faulty edges to increase or since it will seriously affect the effectiveness of Algorithm 1, which is designed based on the method of [42]. Then, for each faulty edge set , we determine whether the required -path between the node pair(s, t)is constructed successfully (i.e., success 1) or not (i.e., success ), where is the unique identification of the node pair(s, t). Then we compute the of as follows:

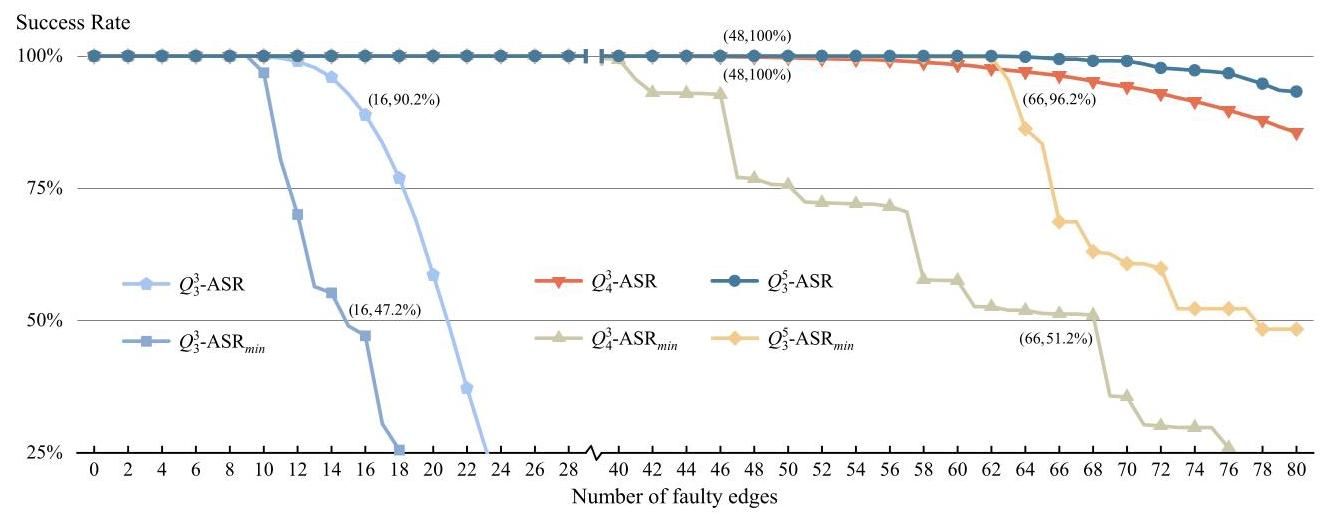


Fig. 13. The and ASR of , and with half dots drawn.

Based on , we then compute the and as follows:

For , we show the variation trend of and ASR of , and in Fig. 13, respectively. Note that for (resp. and ), FP (resp. 33 and 24). It can be observed that both and ASR support dynamic degradation. For (resp. ASR) maintains over 47.2% (resp. 90.2%) even when increasing faulty edges achieve 2FP (i.e., 16).

As or increases, the decline curve of ASR becomes moderate. The ASR of (resp. ) maintains (resp. over 100%) when increasing faulty edges achieve 2FP (i.e., 66 and 48, respectively). As for , the curve is much steeper than that of ASR though it still decreases in general. We can observe that there exist several intervals in the curve where remains unchanged. For instance, when , the of is always equal to . After the increasing faulty edges exceed 46, the of exhibits a cliff-like drop to . The reason behind this phenomenon can be easily explained. In the for-loops of Procedures HP-Round and HP-Direct, some expected fault-free edges connecting two consecutive subgraphs are selected to construct the H-path, which determines whether algorithm HP-PEF can successfully be implemented. Thus, if the faulty edges we add into aren’t concentrated between two consecutive subgraphs, then the success rate of HP-PEF will not be affected by the number of faulty edges. However, when the number of faulty edges exceeds a threshold value which makes that the procedures cannot successfully select the expected edges connecting two consecutive subgraphs, all the -paths that should pass through these two consecutive subgraphs cannot be successfully constructed.

It can be observed that the larger the values of and , the smaller the decline rate of both ASR and . Therefore, it can be predicted that the algorithm HP-PEF will exhibit a more excellent fault-tolerant capacity for systems with larger and .

# VI. Concluding Remarks

We introduce a new edge-fault model named PEF model, for the purpose of embedding -paths into the -ary -cube containing a large number of faulty edges. Using the PEF model, we design, as well as implement, an efficient fault-tolerant H-path embedding algorithm for . Due to the recursive structure of and its edge symmetry property, we believe similar algorithms can be designed for other recursive, edge symmetric networks, such as generalized hypercubes and star graphs. Another promising direction of future work is to investigate the embedding of the H-path in -ary -cubes under the PEF model, so that the H-path passes through some prescribed linear forests, exponentially improving the recent results in [34].

# REFERENCES

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in Proc. 38th Des. Automat. Conf., 2001, pp. 684-689.

[2] A. Jantsch and H. Tenhunen, Networks on Chip, Norwell, MA, USA: Kluwer, 2003.

[3] J. H. Lau, "Evolution, challenge, and outlook of TSV, 3D IC integration and 3D silicon integration," in Proc. IEEE Int. Symp. Adv. Packag. Mater., 2011, pp. 462-488.

[4] P. Bogdan, T. Dumitraş, and R. Marculescu, "Stochastic communication: A new paradigm for fault-tolerant networks-on-chip," VLSI Des., vol. 2007, p. 17, 2007.

[5] Y. Zhang et al., "A deterministic-path routing algorithm for tolerating many faults on very-large-scale network-on-chip," ACM Trans. Des. Automat. Electron. Syst., vol. 26, no. 1, pp. 1-26, Jan. 2021.

[6] E. Taheri, M. Isakov, A. Patooghy, and M. A. Kinsy, "Addressing a new class of reliability threats in 3-D network-on-chips," IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 39, no. 7, pp. 1358-1371, Jul. 2020.

[7] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "Fault-tolerant routing algorithm for 3D NoC using Hamiltonian path strategy," in Proc. Des. Autom. Test Eur. Conf. Exhib., 2013, pp. 1601-1604.

[8] C. Hu, C. M. Meyer, X. Jiang, and T. Watanabe, "A fault-tolerant Hamiltonian-based odd-even routing algorithm for network-on-chip," in Proc. IEEE 35th Int. Technol. Conf. Circuits/Syst. Circuits Commun., 2020, pp. 217-222.

[9] P. Bahrebar and D. Stroobandt, "Improving Hamiltonian-based routing methods for on-chip networks: A turn model approach," in Proc. Des. Autom. Test Eur. Conf. Exhib., 2014, pp. 1-4.

[10] P. Bahrebar and D. Stroobandt, "The Hamiltonian-based odd-even turn model for maximally adaptive routing in 2D mesh networks-on-chip," Comput. Electr. Eng., vol. 45, pp. 386-401, Jul. 2015.

[11] E. O. Amnah and W. L. Zuo, "Hamiltonian paths for designing deadlock-free multicasting wormhole-routing algorithms in 3-D meshes," J. Appl. Sci., vol. 7, no. 22, pp. 3410-3419, 2007.

[12] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "HAMUM - A novel routing protocol for unicast and multicast traffic in MPSoCs," in Proc. 18th Euromicro Int. Conf. Parallel Distrib. Netw. Based Process., 2010, pp. 525-532.

[13] X. Lin and L. M. Ni, "Multicast communication in multicomputer networks," IEEE Trans. Parallel Distrib. Syst., vol. 4, no. 10, pp. 1105-1117, Oct. 1993.

[14] W. J. Dally,"Performance analysis of -ary -cube interconnection networks," IEEE Trans. Comput., vol. 19, no. 6, pp. 775-785, Jun. 1990.

[15] W. J. Dally and C. L. Seitz, "The tours routing chip," Distrib. Comput., vol. 1, no. 3, pp. 187-196, Dec. 1986.

[16] D. H. Linder and J. C. Harden, "An adaptive and fault tolerant wormhole routing strategy for -ary -cubes," IEEE Trans. Comput., vol. 40, no. 1, pp. 2-12, Jan. 1991.

[17] W. Luo and D. Xiang, "An efficient adaptive deadlock-free routing algorithm for torus networks," IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 5, pp. 800-808, May 2012.

[18] P. Ren, X. Ren, S. Sane, M. A. Kinsy, and N. Zheng, "A deadlock-free and connectivity-guaranteed methodology for achieving fault-tolerance in on-chip networks," IEEE Trans. Comput., vol. 65, no. 2, pp. 353-366, Feb. 2016.

[19] H. Abu-Libdeh et al., "Symbiotic routing in future data centers," ACM SIGCOMM Comput. Commun. Rev., vol. 40, no. 4, pp. 51-62, Oct. 2010.

[20] T. Wang, Z. Su, Y. Xia, B. Qin, and M. Hamdi, "NovaCube: A low latency torus-based network architecture for data centers," in Proc. IEEE Glob. Commun. Conf., 2014, pp. 2252-2257.

[21] T. Wang, Z. Su, Y. Xia, and M. Hamdi, "CLOT: A cost-effective low-latency overlaid torus-based network architecture for data centers," in Proc. IEEE Conf. Commun., 2015, pp. 5479-5484.

[22] K. Chen et al., "WaveCube: A scalable, fault-tolerant, high-performance optical data center architecture," in Proc. IEEE Conf. Comput. Commun., 2015, pp. 1903-1911.

[23] S. Borkar et al., "iWarp: An integrated solution to high-speed parallel computing," in Proc. Int. Conf. Supercomput., 1988, pp. 330-339.

[24] M. D. Noakes, D. A. Wallach, and W. J. Dally, "The J-Machine multicomputer: An architectural evaluation," in Proc. 20th Ann. Int. Symp. Comput. Architecture, 1993, pp. 224-235.

[25] R. E. Kessler and J. L. Schwarzmeier, "Cray T3D: A new dimension for Cray research," in Proc. 38th IEEE Int. Comput. Conf., 1993, pp. 176-182.

[26] E. Anderson, J. Brooks, C. Grassl, and S. Scott, "Performance of the Cray T3E multiprocessor," in Proc. ACM/IEEE Conf. Supercomput., 1997, pp. 1-17.

[27] N. R. Adiga et al., "Blue Gene/L torus interconnection network," IBM J. Res. Dev., vol. 49, no. 2/3, pp. 265-276, Mar. 2005.

[28] Y. Lv, C.-K. Lin, J. Fan, and X. Jia, "Hamiltonian cycle and path embed-dings in 3-ary -cubes based on -structure faults," J. Parallel Distrib. Comput., vol. 120, pp. 148-158, 2018.

[29] S. Wang, S. Zhang, and Y. Yang, "Hamiltonian path embeddings in conditional faulty -ary -cubes," Inf. Sci., vol. 268, pp. 463-488, Jun. 2014.

[30] Y. Xiang and I. A. Stewart,"Bipancyclicity in -ary -cubes with faulty edges under a conditional fault assumption," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 9, pp. 1506-1513, Sep. 2011.

[31] S. Zhang and X. Zhang, "Fault-free Hamiltonian cycles passing through prescribed edges in -ary -cubes with faulty edges," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 2, pp. 434-443, Feb. 2015.

[32] M.-C. Yang, J. J. M. Tan, and L.-H. Hsu, "Hamiltonian circuit and linear array embeddings in faulty -ary -cubes," J. Parallel Distrib. Comput., vol. 67, pp. 362-368, Apr. 2007.

[33] I. A. Stewart and Y. Xiang,"Embedding long paths in -ary -cubes with faulty nodes and links," IEEE Trans. Parallel Distrib. Syst., vol. 19, no. 8, pp. 1071-1085, Aug. 2008.

[34] Y. Yang and L. Zhang,"Hamiltonian paths of -ary -cubes avoiding faulty links and passing through prescribed linear forests," IEEE Trans. Parallel Distrib. Syst., vol. 33, no. 7, pp. 1752-1760, Jul. 2022.

[35] J. Yuan et al.,"The -good-neighbor conditional diagnosability of -ary -cubes under the PMC model and MM model," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 4, pp. 1165-1177, Apr. 2015.

[36] L. Xu, L. Lin, S. Zhou, and S.-Y. Hsieh, "The extra connectivity, extra conditional diagnosability, and -diagnosability of arrangement graphs," IEEE Trans. Rel., vol. 65, no. 3, pp. 1248-1262, Sep. 2016.

[37] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh, "A resilient routing algorithm with formal reliability analysis for partially connected 3D-NoCs," IEEE Trans. Comput., vol. 65, no. 11, pp. 3265- 3279, Nov. 2016.

[38] A. Charif, A. Coelho, M. Ebrahimi, N. Bagherzadeh, and N.-E. Zergainoh, "First-Last: A cost-effective adaptive routing solution for TSV-based three-dimensional networks-on-chip," IEEE Trans. Comput., vol. 67, no. 10, pp. 1430-1444, Oct. 2018.

[39] H. Zhang, R.-X. Hao, X.-W. Qin, C.-K. Lin, and S.-Y. Hsieh, "The high faulty tolerant capability of the alternating group graphs," IEEE Trans. Parallel Distrib. Syst., vol. 34, no. 1, pp. 225-233, Jan. 2023.

[40] L.-H. Hsu and C.-K. Lin, Graph Theory and Interconnection Networks. Boca Raton, FL, USA: CRC Press, 2008.

[41] Y. A. Ashir and I. A. Stewart,"On embedding cycles in -ary -cubes," Parallel Process. Lett., vol. 7, no. 1, pp. 49-55, 1997.

[42] H.-C. Kim and J.-H. Park, "Fault Hamiltonicity of two-dimensional torus networks," in Proc. 5th Jpn. Korea Joint Workshop Algorithms Comput., 2000, pp. 110-117.



Hongbin Zhuang received the BEng degree from Huaqiao University, Xiamen, China, in 2019. He is currently working toward the doctoral degree with the College of Computer and Data Science, Fuzhou University, China. His research interests include design and analysis of algorithms, fault diagnosis, and fault-tolerant computing.



Xiao-Yan Li received the PhD degree in computer science from Soochow University, Suzhou, China, in 2019. She was a visiting scholar in the Department of Computer Science, the City University of Hong Kong, Hong Kong, from June 2018-June 2019. She is currently an associate professor with the College of Computer and Data Science, Fuzhou University, China. She has published more than 30 papers in research-related journals and conferences, such as IEEE Transactions on Parallel and Distributed Systems, IEEE/ACM Transactions on Networking, IEEE Transactions on Computers, Journal of Parallel and Distributed Computing, and Association for the Advancement of Artificial Intelligence. She has served some conferences as the Session Chair and Program Committee Member, including IEEE BIBM 2020, IEEE TrustCom 2020 Workshop, WWW 2021, AAAI 2022 Workshop. Her research interests include graph theory, data center networks, parallel and distributed systems, design and analysis of algorithms, and fault diagnosis.



Jou-Ming Chang received the BS degree in applied mathematics from Chinese Culture University, Taipei, Taiwan, in 1987, the MS degree in information management from National Chiao Tung University, Hsinchu, Taiwan, in 1992, and the PhD degree in computer science and information engineering from National Central University, Zhongli, Taiwan, in 2001. He served as the Dean of the College of Management, National Taipei University of Business (NTUB), Taipei, in 2014. He is currently a Distinguished Professor in the Institute of Information and Decision Sciences, NTUB. He has published more than 150 research papers in refereed journals and conferences, including IEEE Transactions on Parallel and Distributed Systems, IEEE/ACM Transactions on Networking, IEEE Transactions on Computers. His major research interests include algorithm analysis and design, graph theory, and parallel and distributed computing.



Dajin Wang received the BEng degree in computer engineering from Shanghai University of Science and Technology, Shanghai, China, in 1982, and the PhD degree in computer science from the Stevens Institute of Technology, Hoboken, USA, in 1990. He is currently a professor in the Department of Computer Science with Montclair State University. He received several university level awards for his scholarly accomplishments. He has held visiting positions in other universities, and has consulted in industry. He served as an associate editor of IEEE Transactions on Parallel and Distributed Systems from 2010 to 2014. He has published over one hundred papers in these areas. Many of his works appeared in premier journals including IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Systems, Man and Cybernetics, IEEE Transactions on Relibility, Journal of Parallel and Distributed Computing, and Parallel Computing. He has served on the program committees of influential conferences. His main research interests include interconnection networks, fault tolerant computing, algorithmic robotics, parallel processing, and wireless ad hoc and sensor networks.

1. Manuscript received 23 July 2022; revised 4 February 2023; accepted 1 April 2023. Date of publication 5 April 2023; date of current version 8 May 2023. This work was supported by the National Natural Science Foundation of China under Grant 62002062 (X.-Y. Li), in part by the Ministry of Science and Technology of Taiwan under Grant MOST-111-2221-E-141-006 (J.-M. Chang), and in part by the Natural Science Foundation of Fujian Province under Grant 2022J05029 (X.-Y. Li). Recommended for acceptance by D. Yang. (Corresponding author: Xiao-Yan Li.)

   Hongbin Zhuang and Xiao-Yan Li are with the College of Computer and Data Science, Fuzhou University, Fuzhou 350108, China (e-mail: hbzhuang476@gmail.com; xyli@fzu.edu.cn).

   Jou-Ming Chang is with the Institute of Information and Decision Sciences, National Taipei University of Business, Taipei 10051, Taiwan (e-mail: spade@ntub.edu.tw).

   Dajin Wang is with the Department of Computer Science, Montclair State University, Montclair, NJ 07043 USA (e-mail: wangd@montclair.edu).

   Digital Object Identifier 10.1109/TPDS.2023.3264698 [↑](#footnote-ref-22)