International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 5, Number 2, pages 329–346, July 2015

# Fault-Tolerant Routing Algorithms for Hierarchical Dual-Nets with Limited and Arbitrary Number of Faulty Nodes

Jun Arai
Graduate School of CIS
Hosei University
Tokyo 184-8584 Japan

Yamin Li
Faculty of Computer and Information Sciences
Hosei University
Tokyo 184-8584 Japan

Received: February 14, 2015
Revised: May 4, 2015
Accepted: June 1, 2015
Communicated by Yoshiaki Kakuda

**Abstract**

The hierarchical dual-net (HDN) is an interconnection network for building ultra-scale parallel systems. The HDN is constructed based on a symmetric product graph (called base network), such as three-dimensional torus and $n$-dimensional hypercubes. A $k$-level hierarchical dual-net, HDN$(B, k, S)$, is obtained by applying $k$-time dual constructions on the base network $B$. $S$ defines a super-node set that adjusts the scale of the system. The node degree of an HDN$(B, k, S)$ is $d_0 + k$ where $d_0$ is the node degree of $B$. The HDN is node and edge symmetric and can contain huge number of nodes with small node degree and short diameter. In this paper, we propose two efficient algorithms for finding a fault-free path on HDN. The first algorithm can always find a fault-free path in $O(2^k F(B))$ time if the number of faulty nodes on HDN is less than $d_0 + k$, where $F(B)$ is the time complexity of fault-tolerant routing in the base network. The second algorithm, more practical one, can find a fault-free path on HDN with arbitrary number of faulty nodes. The simulation results show that the second algorithm can find a fault-free path at high probability.

*Keywords:* Interconnection network, fault-tolerant routing

## 1 Introduction

Because of the advances in computer and networking technologies, the recent high-performance supercomputers containing hundreds of thousands of processors (not cores) have been built [12]. It was predicted that top supercomputers of the next decade will contain 10 to 100 millions of processors, or billions of cores. The interconnection network plays an important role for achieving high-performance in such ultra-scale parallel systems. The system performance depends largely on the time complexities of communication schemes, and in turn depends on the diameter of the network.

An interconnection network consists of switches with multiple communication ports and cables connecting ports by following certain topology. For an ultra-scale parallel computer, the traditional interconnection networks may no longer satisfy the requirements for the high-performance computations or efficient communications because they have a large node degree or a long diameter. The node degree and diameter are the critical measures for the efficiency of the interconnection networks. The node degree is limited by the hardware technologies and the diameter affects all kinds of communication schemes directly. The number of communication ports (node degree) in the network-on-chip (NoC) is typically 4 to 8 in current implementations. The off-chip interconnect switches can have tens of ports, but the cost becomes expensive as the number of ports increases. Other important measures for the efficiency of the interconnection networks include symmetricity, scalability, and efficient routing algorithms.

The following two categories of interconnection networks have attracted a great research attention and been used in many supercomputers' implementations. One is the hypercube-like family that has the advantage of short diameters for high-performance computing and efficient communications. The other is the 2D/3D mesh or torus family that has the advantage of small and fixed node degrees and easy implementations [1]. Traditionally, most supercomputers including those built by CRAY, IBM, SGI, Intel, and Fujitsu use 3D torus or hypercubes.

However, the node degree of the hypercube increases logarithmically as the number of nodes in the systems increases; the diameter of the 2D/3D torus becomes large in ultra-scale parallel systems. To solve these problems, the hierarchical (cluster-based) architectures are proposed in literature [4, 7, 11]. The Roadrunner supercomputer built by IBM adopts a new approach for the interconnection network [6]. It is a cluster-based architecture: the connection among clusters is fully connected, and the fat-tree is used for the connection inside a cluster.

A switch in the higher-level of the Roadrunner does not connect the compute nodes; it connects the switches of the lower-level. This as well as the fat-tree makes the system asymmetric. In this paper, we first introduce a flexible interconnection network, called *Hierarchical Dual-Net* (HDN) [10]. A $k$-level hierarchical dual-net, HDN$(B, k, S)$, is obtained by applying $k$-time dual constructions on a base network $B$. $S$ defines a super-node set that adjusts the scale of the system. A small scale hypercube or torus can be used as the base network. As the level $k$ increases, the number of nodes in the system increases super-exponentially and the number of links increases linearly. In a real parallel system design, $k = 1$, 2, or 3 is enough. The HDN is symmetric and can connect a large number of nodes with a small node degree, meanwhile keeping the diameter short. The HDN also adapts the cluster-based architecture. Compared to the Roadrunner, the HDN is symmetric, uses small number of links and keeps the diameter short.

Fault-tolerant routing is the ability for nodes to communicate each other in a system in which some nodes and/or links are faulty. As the number of nodes in supercomputers grows so does the probability for failure, the fault-tolerant routing becomes a more important issue in the parallel systems that require the high-reliability. Gu and Peng proposed an optimal fault-tolerant routing algorithm for hypercube [5]. Li et al. proposed fault-tolerant routing algorithms for RDN [8]. Ebrahimi et al. proposed a fault-tolerant routing approach for 3D meshes [3]. Arai and Li proposed a disjoint path routing algorithm for HDN [2].

The main contribution of this paper is to propose fault-tolerant routing algorithms for hierarchical dual-net. Let $d_0$ be the node degree of the symmetric base network $B$. For any two nodes and up to $d_0 - 1$ faulty nodes in $B$, a fault-free path can be found in $O(F(B))$ time. Given two nodes $u$ and $v$ and up to $d_0 + k - 1$ faulty nodes in a hierarchical dual-net HDN$(B, k, S)$, we propose an $O(2^k F(B))$ time algorithm for finding a fault-free path connecting $u$ and $v$. We also propose a heuristic fault-tolerant routing algorithm for an HDN$(B, k, S)$ with arbitrary number of faulty nodes. The simulation results show that this algorithm can find a fault-free path between two non-faulty nodes with high probability.

The rest of the paper is organized as follows. Section 2 introduces the hierarchical dual-net. Section 3 describes a fault-tolerant routing algorithm for a hierarchical dual-net with limited faulty nodes. Section 4 proposes an efficient and practical fault-tolerant algorithm that can finds a fault-free path on a hierarchical dual-net with arbitrary number of faulty nodes, and gives the performance evaluation result. Section 5 concludes the paper.

## 2 The Hierarchical Dual-Net

The HDN was motivated by recursive dual-net (RDN) [9]. The RDN can be viewed as a special case of HDN. We begin with a brief introduction to the RDN. An RDN is constructed recursively by a dual-construction. The dual-construction is a way to expand a given symmetric graph $G$ of size $n$ to a new symmetric graph $G^*$ of size $2n^2$. It generates $2n$ copies of $G$ as subgraphs (denoted as clusters) of $G^*$. Half of them, $n$ clusters, are of class 0 and the others are of class 1. Each cluster contains $n$ nodes. If $G$ is symmetric then the expanded graph $G^*$ is unique and symmetric. Therefore, the dual-construction can be applied recursively from a symmetric network (the base network). RDN$(m, k)$ denotes an RDN generated from a base network of size $m$ by applying dual-construction $k$ times. The number of nodes in an RDN$(m, k)$ is $(2m)^{2^k}/2$ and the node degree is $d_0 + k$ where $d_0$ is the node degree of the base network.

The problem about an RDN is that its growth rate is super-exponential. There is very little space for selection of the size of an RDN. For example, let the base network be a 3-cube, then the sizes of RDN$(8, k)$ will be $2^7$, $2^{15}$, and $2^{31}$ for $k = 1, 2$, and 3, respectively. In HDN, we provide a mechanism (super-node) to control the growth rate through its expansion from a base network. This new interconnection network has a very flexible way for adjusting its size.

The hierarchical dual-net [10], HDN$(B, k, S)$, contains three sets of parameters: $B$ is a *symmetric product graph*, we call it *base network*; $k$ is an integer that indicates the *level* of the HDN (the number of *dual-constructions* applied); and $S = \{G'_1, G'_2, \ldots, G'_k\}$, where $G'_i$ is a *sub-graph* of HDN$(B, k-1, S)$ and $s_i = |G'_i|$ is the number of nodes in a *super-node* at the level $i$ for $1 \leq i \leq k$. If $s_i = 1$ for $1 \leq i \leq k$, the HDN becomes an RDN.

Given $r$ graphs $G_i = (V_i, E_i)$, $1 \leq i \leq r$, their product graph $G = G_1 \times G_2 \times \ldots \times G_r$ is defined as the graph $G = (V, E)$, where $V = \{(v_{j1}, \ldots, v_{ji}, \ldots, v_{jr}) | v_{ji} \in V_i,\ 1 \leq i \leq r\}$ and $E = \{[(v_{j1}, \ldots, v_{ji}, \ldots, v_{jr}), (v_{k1}, \ldots, v_{ki}, \ldots, v_{kr})] | v_{ji} \neq v_{ki},\ (v_{ji}, v_{ki}) \in E_i$, and $v_{jl} = v_{kl}$ for $l \neq i, 1 \leq i \leq r\}$. In other words, the nodes of the product graph $G$ are labeled with $r$-tuples, where the $i$th element of the $r$-tuples is chosen from the node set of the $i$th component graph. The edges of the product graph connect pairs of nodes whose labels are identical in all but the $j$th element, and the two nodes corresponding to the $j$th elements in the $j$th component graph are connected by an edge. Meshes/tori or hypercubes are typical examples of product graphs.

Given a product graph $G = G_1 \times G_2 \times \ldots \times G_r$, we define a *quotient graph* $Q$ as $Q = G/G'$ where $G'$ is a sub-product graph of $G$ such that $G = G' \times Q$. A node in a product graph $G = G_1 \times \ldots \times G_i \times \ldots \times G_r$ can be represented by $(a_1, \ldots, a_i, \ldots, a_r)$ with $0 \leq a_i \leq |G_i| - 1$. We define a sub-graph $G'$ as $G' = G''_1 \times \ldots \times G''_j \times \ldots \times G''_q$ with $G''_j = G_i$ for $1 \leq j \leq q \leq r$ and $1 \leq i \leq r$, $G''_j \neq G''_k$ if $j \neq k$ for $1 \leq j, k \leq q$. Then a node in the sub-graph $G'$ can be represented by $(b_1, \ldots, b_i, \ldots, b_q)$ with $0 \leq b_i \leq |G''_i| - 1$. We can consider a quotient graph $Q$ as a reduced graph of $G$ with $G'$ being mapped into a single node (a super-node).

A graph $G$ is symmetric (node-symmetric) if all its nodes looks alike. A product graph is symmetric if all its component graphs are symmetric. We use the symmetric product graph as the base network for generating a hierarchical dual-net through dual-constructions. We denote the base network as $B = B_1 \times B_2 \times \ldots \times B_r$ where all the $B_i$, $1 \leq i \leq r$, are symmetric. We define a super-node of $B$, denoted as $SN$ as a sub-product graph of $B$. That is, $SN = B_{i_1} \times B_{i_2} \times \ldots \times B_{i_q}$, where $i_j, 1 \leq j \leq q$, are distinct and $q \leq r$.

Let $|B_i| = b_i$ be the number of nodes in $B_i$ for $1 \leq i \leq r$. The HDN$(B, 0, S) = B$ is the base network. For $i > 0$, the HDN$(B, i, S)$ is generated from HDN$(B, i-1, S)$ by a construction to be explained below. Note that $S = \{G'_1, G'_2, \ldots, G'_k\}$, where $G'_i$ is a sub-graph of HDN$(B, k-1, S)$ and $s_i = |G'_i|$ is the number of nodes in a super-node at the level $i$ for $1 \leq i \leq k$. First, we define a super-node of level $i$, denoted as $SN^i$, to be a sub-product graph $G'_i$ of size $s_i$ in $B$. Then, we define graph $Q^i$ as the quotient graph HDN$(B, i-1, S)/SN^i$. Suppose that there are $N_{i-1}$ nodes in the HDN$(B, i-1, S)$, then the number of nodes $n_i$ in $Q^i$ is $N_{i-1}/s_i$. The $s_i$ can be 1 or $\prod_{j=1}^{q} |B_{i_j}|$, where $1 \leq i_j \leq r$ and $q \leq r$. That is, $s_i$ can be a product of any number of integers in $\{b_1, b_2 \ldots, b_r\}$. For example, if $r = 3$, $b_1 = 2$, $b_2 = 3$, and $b_3 = 5$, the possible $s_i$ can be 1, 2, 3, 5, $2 \times 3$, $2 \times 5$, $3 \times 5$, or $2 \times 3 \times 5$.

The construction of HDN$(B, i, S)$, $1 \leq i \leq k$, can be defined by a two-step process: First, we

perform a dual-construction on the quotient graph $Q^{i-1} = \text{HDN}(B, i-1, S)/SN^i$ with $\text{HDN}(B, 0, S) = B$. Let the graph generated by the dual-construction be $Q^i$, and the subgraph of two nodes that is connected by a cross-edge of level $i$ be $K_2$. Second, to get the $\text{HDN}(B, i, S)$, we replace every $K_2$ in $Q^i$ by a product graph $K_2 \times SN$. We call $\text{HDN}(B, i-1, S)$ *cluster* of $\text{HDN}(B, i, S)$.

An $\text{HDN}(B, i, S)$ consists of $2n_i$ clusters which are divided into two *classes*: class 0 and class 1 with each class containing $n_i$ clusters. That is, the number of clusters in each class is equal to the number of super-nodes in a cluster. At level $i$, each super-node in a cluster has $s_i$ new links to a super-node in a distinct cluster of the other class. Because there are $s_i$ nodes in a super-node, one node contributes a new link. The dual-construction of an RDN is a special case of the construction of an HDN with $s_i = 1$ for $1 \leq i \leq k$.

The indexes of the nodes in $\text{HDN}(B, k, S)$ can be defined as follows. Let $SN^k$ be a *super-node_id* in a cluster of $\text{HDN}(B, k, S)$ and $N^k$ be a *node_id* in a super-node, then a node in the $\text{HDN}(B, k, S)$ can be represented by $(C^k, U^k, SN^k, N^k)$ where $C^k$ is the *class_id* (0 or 1) and $U^k$ is the *cluster_id*. A cross-edge at level $k$ connects node $(C^k, U^k, SN^k, N^k)$ and node $(\overline{C^k}, SN^k, U^k, N^k)$.

Suppose that the node degree of the base network $B$ is $d_0$, the node degree of the $\text{HDN}(B, k, S)$ is $d_0 + k$. Let $N_{i-1}$ be the number of nodes in the $\text{HDN}(B, i-1, S)$. There are $N_i = 2 \times (N_{i-1}/s_i) \times N_{i-1} = 2N_{i-1}^2/s_i$ nodes in the $\text{HDN}(B, i, S)$ for $1 \leq i \leq k$, where $N_{i-1}/s_i$ is the number of clusters in each class. That is, the number of nodes in the $\text{HDN}(B, k, S)$ is $(2N_0)^{2^k}/(2 \times \prod_{i=1}^{k} s_i)$, where $N_0$ is the number of nodes in the base network.

Let the diameter of the $\text{HDN}(B, i-1, S)$ be $D_{i-1}$ and the diameter of the super-node (SN) be $D(SN_i)$. Then, if we map a super-node into a single node, the diameter of the quotient graph $Q^{i-1}$ is $D(Q^{i-1}) = D_{i-1} - D(SN^i)$.

To route a node $u$ in a cluster of class 0 (or 1) to a node $v$ in a different cluster of the same class, we can route $u$ along with a direct link of level $i$ to a node $u'$ in a cluster of class 1 (or 0). This takes one step. Then, we route $u'$ inside the cluster to a node $w'$ that can reach a node $w$ in the same cluster of node $v$ along with direct link of level $i$. The longest distance between nodes $u'$ and $w'$ is $D(Q^{i-1})$.

Similarly, we can route node $w'$ to a node $w$ (by one step) and then to a node $v'$ which is in the same super-node of $v$ (by $D(Q^{i-1})$ steps). Finally, we route $v'$ to node $v$, this takes $D(SN^i)$ steps. Therefore, we have the following recurrence:

$$\begin{aligned} D_i &= 2(1 + D(Q^{i-1})) + D(SN^i) \\ &= 2(1 + D_{i-1} - D(SN^i)) + D(SN^i) \\ &= 2D_{i-1} - D(SN^i) + 2 \end{aligned}$$

Solving the above recurrence, we get the diameter $D_k$ of $\text{HDN}(B, k, S)$ as below:

$$D_k = 2^k D(B) - \sum_{j=0}^{k-1} 2^j D(SN^{k-j}) + 2^{k+1} - 2$$

where $D(B)$ and $D(SN^i), 1 \leq i \leq k$, are the diameters of the base network and the super-nodes, respectively. The results of the analysis in this section are summarized in the following theorem.

**Theorem 1** *Assume that the base network $B$ is a symmetric, product graph and $SN^i, 1 \leq i \leq k$, are sub-product graphs of $B$ with $|SN^i| = s_i$. Let the number of nodes, the node degree, and the diameter of $B$ be $N_0$, $d_0$, and $D_0$, respectively. Let the diameters of $SN^i, 1 \leq i \leq k$, be $D(SN^i)$. Let $S = \{G'_1, G'_2, \ldots, G'_k\}$, where $G'_i$ is a sub-graph of HDN$(B, k-1, S)$ and $s_i = |G'_i|$ is the number of nodes in a super-node at the level $i$ for $1 \leq i \leq k$. Then, the number of nodes of HDN$(B, k, S)$ is $(2N_0)^{2^k}/(2\prod_{i=1}^{k} s_i)$, the node degree is $d_0 + k$, and the diameter is $D_k = 2^k D(B) - \sum_{j=0}^{k-1} 2^j D(SN^{k-j}) + 2^{k+1} - 2$, where $N$ is the number of nodes in HDN$(B, k, S)$.*

# 3 Fault-Tolerant Routing on HDN

The problem of finding a path from a source $u$ to a destination $v$ and forwarding a message along the path is known as the routing problem. Finding a fault-free path from $u$ to $v$ on a network

with a set of faulty nodes is called fault-tolerant routing. The solutions for these routing problems are fundamental and critical for the performance of an interconnection network. In this section, we introduce an efficient fault-tolerant routing algorithm that finds a fault-free path on hierarchical dual-net with limited faulty nodes.

We use a model of the global failure information (off-line). That is, each non-faulty node in the system knows all the IDs of the faulty nodes. In contrast, there is a model of the local failure information (in-line) in which each node knows the failure information of its neighbors or a limited number of the intermediate hops. Also, there are two types of the component failures: node failure and link failure. Node failure means that the faulty node (processor and switch) and the links connected to it can not work anymore; while the link failure means that only the links (usually some ports of the switch) can not be used - the processor and switch (the other ports of the switch) is still alive. In this paper, we only investigate the fault-tolerant routing on node failure.

The proposed algorithm can always find a fault-free path if the number of faulty node is less than the node degree of the hierarchical dual-net. We introduce some notations that will be used later. Let $\mathrm{SN}(\mathrm{HDN}(B,k,S),n_0,n_1,n_2)$ be a super-node of level $k$ where $n_0$, $n_1$, and $n_2$ are *class-id*, *cluster-id*, and *supernode-id*, respectively. In an $\mathrm{HDN}(B,k,S)$, there are $s_k$ nodes in every super-node of level $k$, and the *node-id* is 0, 1, ..., or $s_k-1$ inside the super-node. All the nodes having a same node-id form an RDN, and there are $s_k$ RDNs. Let $\mathrm{RDN}(\mathrm{HDN}(B,k,S),n)$ be an RDN that consists of the nodes whose node-id is $n$. Thus, an HDN can be considered as a product graph of RDN and $S_k$. Suppose that a fault-tolerant routing algorithm exists for the base network $B$ and we name it Supernode_FaultTolerantRouting$(u,v)$.

Figure 1 shows the super-nodes in an $\mathrm{HDN}(B,k,S)$ with $k=1$ and $s_1=4$. The base network is a 3-cube (8 nodes). Both the number of clusters of each class and the number of super-nodes in each cluster are $8/4=2$. There are 8 super-nodes and each of them is identified with supernode(class-id,cluster-id,supernode-id).

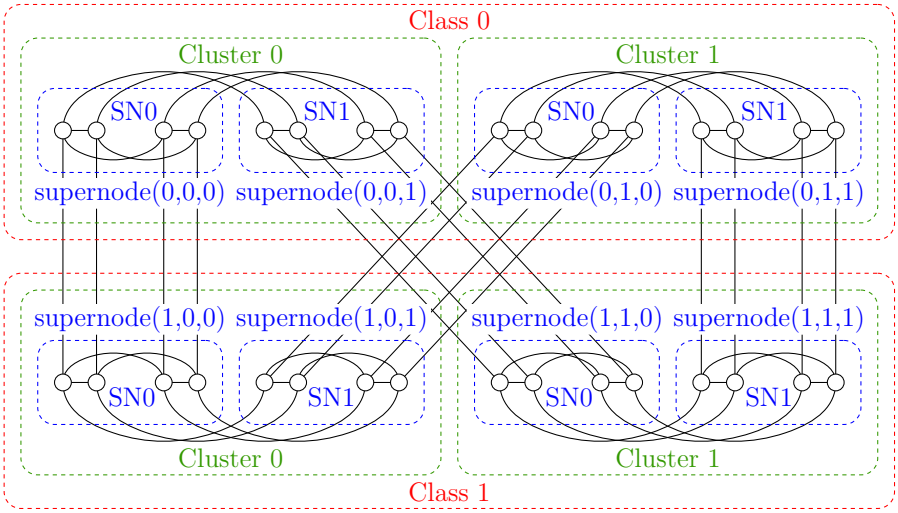


Figure 1: $\mathrm{SN}(\mathrm{HDN}(B,k,S),n_0,n_1,n_2)$ in an $\mathrm{HDN}(B,k,S)$

Figure 2 shows the RDNs of the HDN of Figure 1. There are four RDNs ($s_1=4$) which are marked with different colors and line widths.

Figure 3 shows the super-nodes and Figure 4 shows the RDNs of the same HDN but at a different view-point. From these two figures, we can understand easily that an HDN is a product graph of the RDN and $S_1$ (a 2-cube in this example).

Consider an $\mathrm{HDN}(B,k,S)$ with at most $d_0+k-1$ faulty nodes. A node will not be isolated because the node degree of the $\mathrm{HDN}(B,k,S)$ is $d_0+k$. To find a fault-free path between node $u$ and node $v$ on HDN, our basic idea is to find two paths from $u$ and $v$, respectively, terminating at a same super-node, and then to connect the two paths inside the super-node. The algorithm is

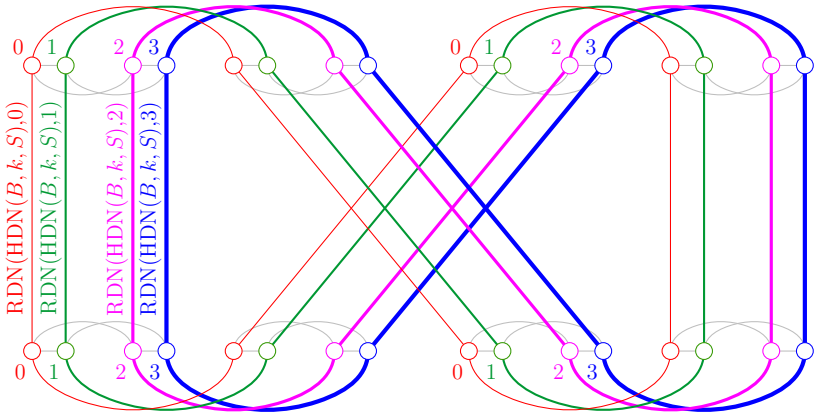


Figure 2: RDN(HDN$(B, k, S), n)$ in an HDN$(B, k, S)$

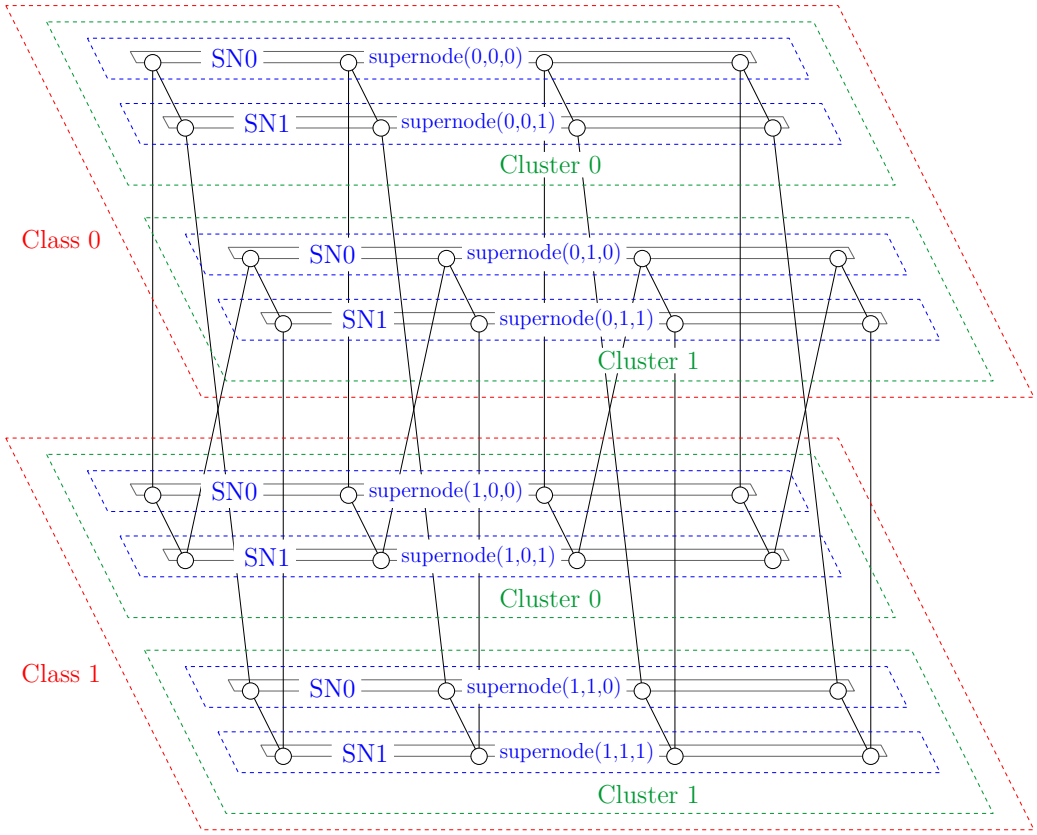


Figure 3: SN(HDN$(B, k, S), n_0, n_1, n_2)$ in an HDN$(B, k, S)$

divided into three steps: Step-1 and Step-2 find two paths, and Step-3 connects them.

The algorithm is formally given in Algorithm 1. Step-1 finds two RDNs whose numbers of faulty nodes are less than the node degree of RDN. The RDN_FaultTolerantRouting algorithm requires this constraint. The paths can be found inside the super-nodes of $u$ and $v$. Then we get two paths $[u \to u_s]$ and $[v \to v_s]$. Step-2 finds two paths starting from $u_s$ and $v_s$ and ending at a same super-node by using RDN_FaultTolerantRouting algorithm. At this step, we must choose the super-node whose number of faulty node is less than the node degree of the super-node. We get two paths

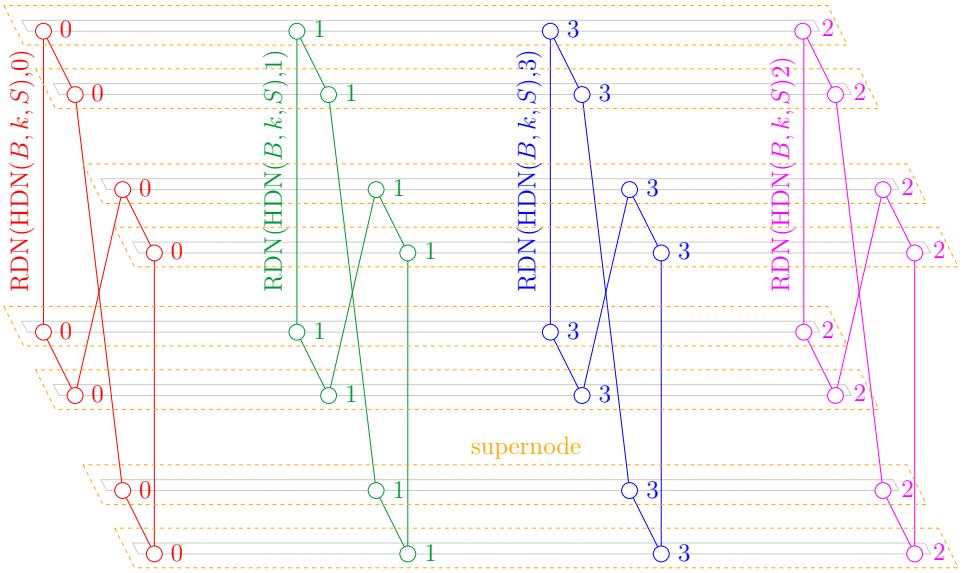


Figure 4: RDN(HDN$(B,k,S),n)$ in an HDN$(B,k,S)$

$[u_s \rightarrow u_r]$ and $[v_s \rightarrow v_r]$. In Step-3, a path $[u_r \rightarrow v_r]$ is built inside the super-node. Finally, by connecting the five paths, a fault-free path $[u \rightarrow u_s \rightarrow u_r \rightarrow v_r \rightarrow v_s \rightarrow v]$ can be found.

Figures 5 - 8 show a path example found by the algorithm. The base network is a 3-cube and $S_1$ is a 2-cube with $k = 1$. The node degree is $3 + 1 = 4$. Figure 5 gives all the node ids in the format of (class-id, cluster-id, supernode-id, node-id). Two nodes $u(u_0, u_1, u_2, u_3)$ and $v(v_0, v_1, v_2, v_3)$ are $(0, 0, 0, 0)$ (red circle) and $(1, 1, 1, 3)$ (blue circle), respectively. Three faulty nodes, $(0, 0, 0, 1)$, $(0, 1, 1, 3)$, and $(1, 1, 0, 3)$, are marked with black solid circles.

First the algorithm finds two nodes $u_s$ and $v_s$ inside the super-node of $u$ and super-node of $v$, respectively. There is no faulty node in RDN(HDN$(B,k,S),0)$, therefore $u_s$ is same as $u$. On the other hand, there are two faulty nodes in RDN(HDN$(B,k,S),3)$, nodes $(0, 1, 1, 3)$ and $(1, 1, 0, 3)$, not less than the node degree of the RDN. Therefore, we must find a suitable adjacent nodes of $v$, node $(1, 1, 1, 1)$ for example. There is one faulty node in RDN(HDN$(B,k,S),1)$, less than the node degree. Therefore, $v_s$ is $(1, 1, 1, 1)$. Two paths $[(0, 0, 0, 0)]$ and $[(1, 1, 1, 3) \rightarrow (1, 1, 1, 1)]$ are found in Step-1, as shown in Figure 6.

Step-2 finds two nodes $u_r$ and $v_r$ in a same super-node and connects $u_s$ and $v_s$ to $u_r$ and $v_r$, respectively. In this case, $u_r$ and $u_s$ must not be the same node. If $u_r$ and $u_s$ are the same node, $v_r$ will be $(0, 0, 0, 1)$ which is a faulty node. On the other hand, because there is no faulty node in the super-node of $v_s$, we select SN(HDN$(B,k,S),1,1,1)$ as the super-node in which the $u_r$ and $v_r$ must be found. In the case, the $v_r$ will be the same as $v_s$ because the number of faulty nodes in SN(HDN$(B,k,S),1,1,1)$ is less than the node degree of the super-node. Then the id of $u_r$ is determined. That is, $u_r$ is $(1, 1, 1, 0)$ and $v_r$ is $(1, 1, 1, 1)$. Then we route $u_s$ to $u_r$ in RDN(HDN$(B,k,S),0)$. As the result of Step-2, two paths, $[(0, 0, 0, 0) \rightarrow (0, 0, 1, 0) \rightarrow (1, 1, 0, 0) \rightarrow (1, 1, 1, 0)]$ and $[(1, 1, 1, 1)]$ $(v_r = v_s)$, are found, as shown in Figure 7.

Step-3 gets a path $[(1, 1, 1, 0) \rightarrow (1, 1, 1, 1)]$ by using Supernode_FaultTolerantRouting$(u_r, v_r)$ as shown in Figure 8. Note that in this case, we can use routing algorithm instead of the fault-tolerant routing algorithm, because there is no faulty node in the super-node (SN(HDN$(B,k,S),1,1,1)$). Finally, connecting the five paths, $[u \rightarrow u_s \rightarrow u_r \rightarrow v_r \rightarrow v_s \rightarrow v]$, a fault-free path $[(0, 0, 0, 0) \rightarrow (0, 0, 1, 0) \rightarrow (1, 1, 0, 0) \rightarrow (1, 1, 1, 0) \rightarrow (1, 1, 1, 1) \rightarrow (1, 1, 1, 3)]$ is found.

**Theorem 2** *For a given non-faulty node $n(n_0, n_1, n_2, n_3)$ in HDN$(B,k,S)$ with at most $d_0 + k - 1$ faulty nodes, a non-faulty node $n_s(n_{s_0}, n_{s_1}, n_{s_2}, n_{s_3})$ satisfied the following conditions always exists.*

1. *$n_s$ is same as $n$ or a neighbor of $n$.*

**Algorithm 1:** HDN_FaultTolerantRouting(HDN$(B,k,S),u,v,F$)
input: HDN$(B,k,S)$;
input: A non-faulty node $u(u_0,u_1,u_2,u_3)$ (the node representation of level $k$);
input: A non-faulty node $v(v_0,v_1,v_2,v_3)$ (the node representation of level $k$);
input: A set of faulty nodes $F$;
output: A fault-free path $u \Rightarrow v$;
**begin**
/* Step-1 */
**if** $|\text{RDN}(\text{HDN}(B,k,S),u_3) \cap F| < d_0 + k - d(S_k)$ /* $d(S_k)$ is degree of $S_k$ */
$u_s = u$;
**else**
find a non-faulty node $u_s = (u_0,u_1,u_2,u_{s_3})$;
/* $u_s$ is a neighbor of $u$ and $|\text{RDN}(\text{HDN}(B,k,S),u_{s_3}) \cap F| < d_0 + k - d(S_k)$ */
**endif**
**if** $|\text{RDN}(\text{HDN}(B,k,S),v_3) \cap F| < d_0 + k - d(S_k)$ /* $d(S_k)$ is degree of $S_k$ */
$v_s = v$;
**else**
find a non-faulty node $v_s = (v_0,v_1,v_2,v_{s_3})$;
/* $v_s$ is a neighbor of $v$ and $|\text{RDN}(\text{HDN}(B,k,S),v_{s_3}) \cap F| < d_0 + k - d(S_k)$ */
**endif**
/* Step-2 */
**if** $|\text{SN}(\text{HDN}(B,k,S),u_0,u_1,u_2) \cap F| < d(S_k)$
$u_r = u_s$;
**else**
find a non-faulty node $u_r = (u_{r_0},u_{r_1},u_{r_2},u_{s_3})$;
/* $u_r$ is a neighbor of $u_s$ and $|\text{SN}(\text{HDN}(B,k,S),u_{r_0},u_{r_1},u_{r_2}) \cap F| < d(S_k)$ */
**if** $u_r$ does not exist
**if** $|\text{SN}(\text{HDN}(B,k,S),v_0,v_1,v_2) \cap F| < d(S_k)$
$v_r = v_s$;
**else**
find a non-faulty node $v_r = (v_{r_0},v_{r_1},v_{r_2},v_{s_3})$;
/* $v_r$ is a neighbor of $v_s$ and $|\text{SN}(\text{HDN}(B,k,S),v_{r_0},v_{r_1},v_{r_2}) \cap F| < d(S_k)$ */
**endif**
RDN_FaultTolerantRouting(RDN(HDN$(B,k,S),u_{s_3}$),$u_s$,$u_r(v_{r_0},v_{r_1},v_{r_2},u_{s_3}),F$);
/* route node $u_s$ to node $u_r$ */
**else**
RDN_FaultTolerantRouting(RDN(HDN$(B,k,S),v_{s_3}$),$v_s$,$v_r(u_{r_0},u_{r_1},u_{r_2},v_{s_3}),F$);
/* route node $v_s$ to node $v_r$ */
**endif**
**endif**
/* Step-3 */
Supernode_FaultTolerantRouting$(u_r,v_r)$; /* route node $u_r$ to node $v_r$ */
**end**

2. *The number of faulty node in RDN(HDN$(B,k,S)$, $n_{s_3}$) is less than $d_0+k-d(S_k)$ where $d(S_k)$ is the node degree of $S_k$.*

*Proof:* Let $r = d_0 + k - d(S_k)$. We calculate how many faulty nodes will be needed if $n_s$ does not exists. Because if the number of faulty nodes on RDN(HDN$(B,k,S),n_3)$ is less than $r$, $n_s$ is same as $n$, the number of faulty nodes has to be at least $r$. There are $r + d(S_k)$ nodes connecting to node $n$. $d(S_k)$ nodes out of those nodes have different node-ids from node $n$. Assume that there are

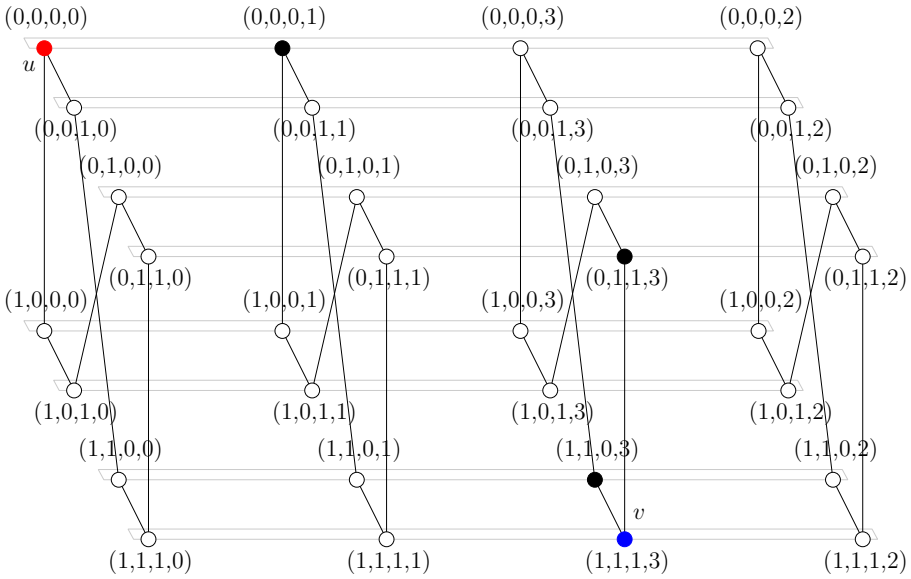


Figure 5: An example of fault-tolerant routing on HDN$(B, 1, S)$ with $s_1 = 4$

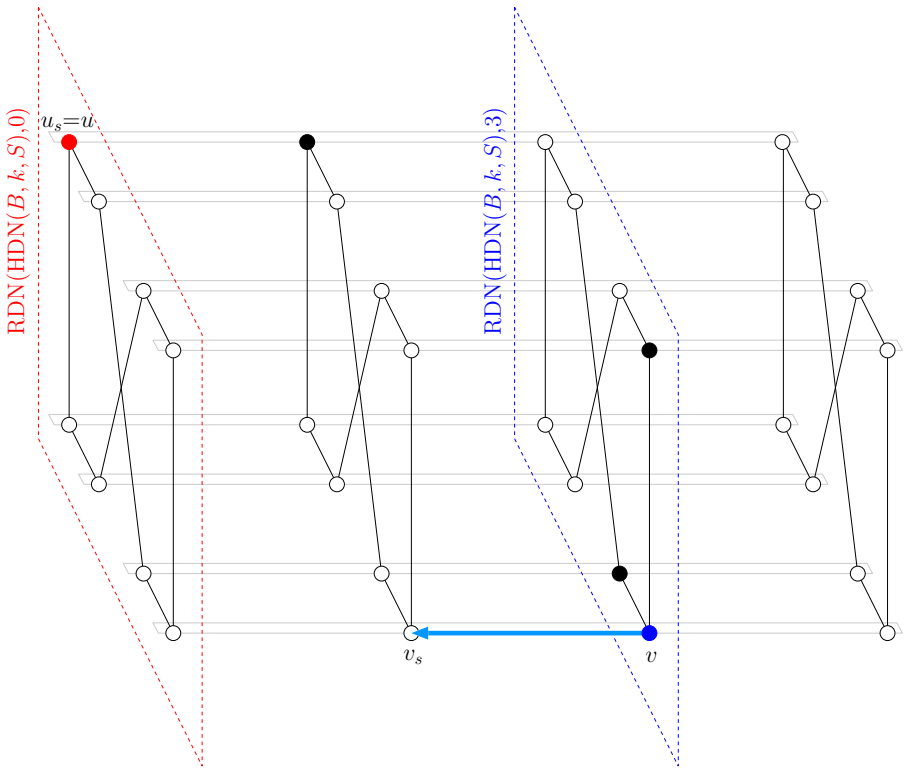


Figure 6: An example of fault-tolerant routing Step-1 on HDN$(B, 1, S)$ with $s_1 = 4$

$x$ faulty nodes in those nodes, $0 \leq x < d(S_k)$. Let $N = \{n_1, n_2, ..., n_{d(S_k)-x}\}$ be set of non-faulty adjacent nodes of $n$ whose node-ids are different from $n_3$. Because if the number of faulty nodes on RDN(HDN$(B, k, S), n_{i_3}$), for $1 \leq i \leq d(S_k) - x$, is less than $r$, $n_s$ is $n_i$, at least $(d(S_k) - x) \times r$ faulty nodes are needed. From the above, at least $r + x + (d(S_k) - x) \times r$ nodes are faulty on HDN$(B, k, S)$ if $n_s$ does not exists.

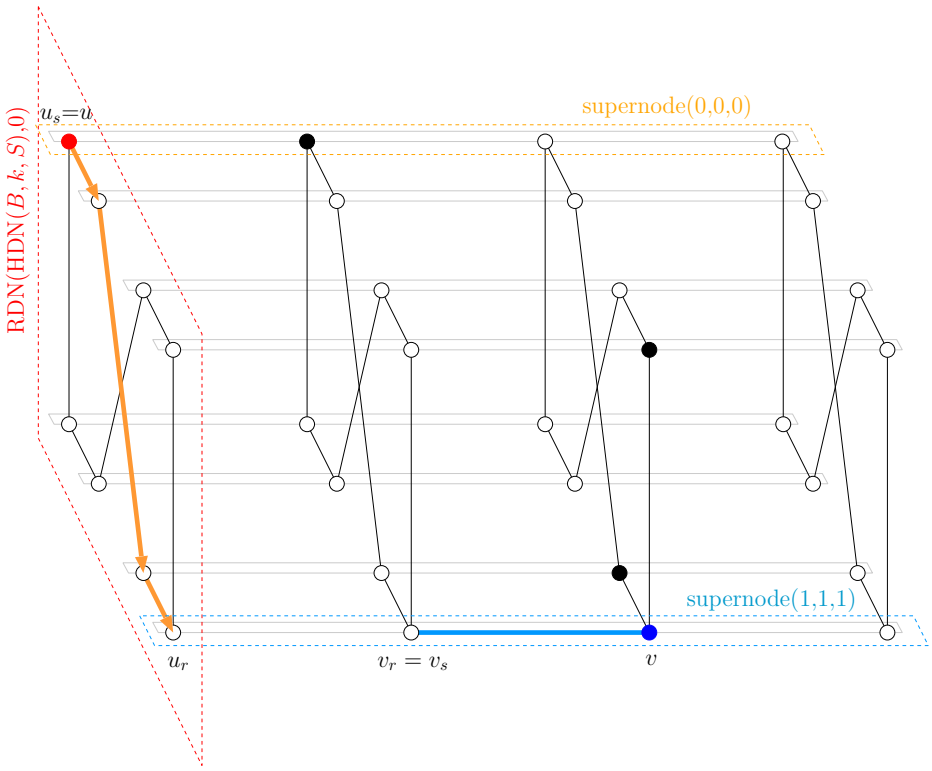


Figure 7: An example of fault-tolerant routing Step-2 on HDN$(B, 1, S)$ with $s_1 = 4$

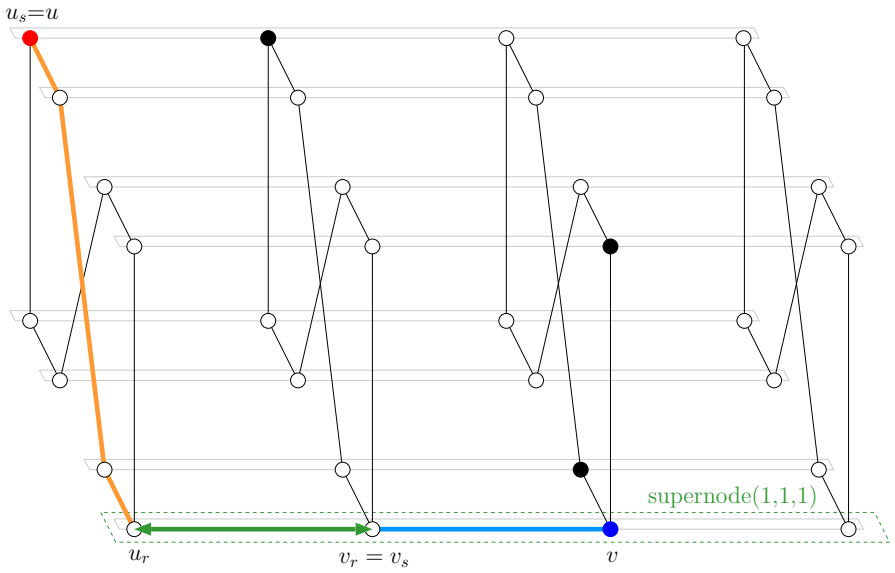


Figure 8: An example of fault-tolerant routing Step-3 on HDN$(B, 1, S)$ with $s_1 = 4$

This means that if the number of faulty nodes on HDN$(B, k, S)$ is less than $r+x+(d(S_k)-x)\times r$ then $n_s$ exists. We have $(d_0 + k - 1) - (r + x + (d(S_k) - x)\times r) = d(S_k) - 1 - x - (d(S_k) - x)\times r = (d(S_k) - x)(1 - r) - 1 < 0$. The expression shows that $r + x + (d(S_k) - x)\times r$ is always larger than $d_0 + k - 1$. This is against the assumption of at most $d_0 + k - 1$ faulty nodes, therefore, $n_s$ always exists. □

**Theorem 3** *For two non-faulty nodes $u(u_0, u_1, u_2, u_3)$ and $v(v_0, v_1, v_2, v_3)$ in HDN$(B, k, S)$ with at most $d_0 + k - 1$ faulty nodes, a non-faulty node $n(n_0, n_1, n_2, n_3)$ satisfied the following conditions always exists.*

1. *$n$ is same as $u$ or $v$, or $n$ is a neighbor of $u$ or $v$.*
2. *The number of faulty node in SN(HDN$(B, k, S)$, $n_0, n_1, n_2$) is less than $d(S_k)$ where $d(S_k)$ is node degree of $S_k$.*

*Proof:* $n$ and $u$ are same node if the number of faulty node in SN(HDN$(B, k, S), u_0, u_1, u_2$) is less than $d(S_k)$ and node $(n_0, n_1, n_2, v_3)$ is non-faulty node. Similarly, $n$ and $v$ are same node if the number of faulty node in SN(HDN$(B, k, S), v_0, v_1, v_2$) is less than $d(S_k)$ and node $(n_0, n_1, n_2, u_3)$ is non-faulty node. We assume that $n$ always exists if $n$ is neither $u$ nor $v$. Let $r = d_0 + k - d(S_k)$. Similar to the proof of Theorem 2, we calculate how many faulty nodes are needed if $n$ does not exists. There are $2(r + d(S_k))$ nodes connecting to node $u$ or $v$. $2r$ nodes out of those nodes have same node-ids as $u$ or $v$. Assume that there are $x$ faulty nodes in those nodes, $0 \leq x < 2r$. Let $\{n_1, n_2, ..., n_{2r-x}\}$ be set of non-faulty adjacent nodes of $u$ or $v$ whose node-ids are same as $u_3$ or $v_3$. If the number of faulty nodes on SN(HDN$(B, k, S), n_{i_0}, n_{i_1}, n_{i_2}$), for $1 \leq i \leq 2r - x$, is less than $d(S_k)$ then $n$ is $n_i$. Therefore, at least $(2r - x) \times d(S_k)$ faulty nodes are needed. From the above, at least $x + (2r - x) \times d(S_k)$ nodes are faulty on HDN$(B, k, S)$ if $n$ does not exists. This means that if the number of faulty nodes on HDN$(B, k, S)$ is less than $x + (2r - x) \times d(S_k)$ then $n$ exists. If $x \geq r$, $(d_0+k-1)-(x+(2r-x)d(S_k)) \leq (d_0+k-1)-(r+(2r-x)d(S_k)) = d(S_k)\times(1-(2r-x))-1 < 0$. If $x < r$, $(d_0+k-1)-(x+(2r-x)d(S_k)) < (d_0+k-1)-(x+r\times d(S_k)) = (r+d(S_k)-r\times d(S_k))-(x+1) \leq 0$. This means that $x + (2r - x)\times d(S_k)$ is always larger than $d_0 + k - 1$. Therefore, node $n$ exists. □

**Theorem 4** *Assume that a fault-free path connecting two nodes in base network can be found in $O(F(B))$ time. Then for any two non-faulty nodes in HDN$(B, k, S)$ with at most $d_0 + k - 1$ faulty nodes for $k > 0$, a fault-free path connecting the two nodes can be found in $O(2^k F(B))$ time. The maximum length of the paths is at most $D(f(S_k)) + 2(2^{k-1}D(B) - \sum_{j=0}^{k-2} 2^j D(SN^{k-1-j}) - D(S_k)) + 2^{k+1} + 3$, where $D(f(S_k))$ is the maximum length of the path found by supernode fault tolerant routing algorithm.*

*Proof:* Let the time complexity for executing Step-1, Step-2, and Step-3 of our algorithm be $T_1$, $T_2$, $T_3$, respectively. Step-1 finds two nodes $u_s$ and $v_s$ from adjacent nodes of $u$ and $v$, thus $T_1 = O(2(d_0 + k))$. Step-2 searches all of adjacent nodes of $u_s$ and $v_s$ and executes RDN_FaultTolerantRouting. Therefore, $T_2 = O(2(d_0 + k)) + O(2^k F(B))$. Finally, connecting the two nodes $u_r$ and $v_r$ on $S_k$ in Step-3 requires $T_3 = O(F(S_k))$. Therefore, the time complexity of finding fault-free path connecting the two nodes is $O(2^k F(B))$. According to Theorem 2, the length of the two paths, $[u \to u_s]$ and $[v \to v_s]$, are at most 1. According to Theorem 3, the length of either $[u_s \to u_r]$ or $[v_s \to v_r]$ is at most 1. The maximum length of another path is equal to the maximum length of the path found by RDN_FaultTolerantRouting algorithm. The maximum length of the path connecting a node $u_r$ and $v_r$ is $D(f(S_k))$. Therefore, the maximum length of fault-free path on HDN is $1 + 1 + 1 + (2(2^{k-1}D(B) - \sum_{j=0}^{k-2} 2^j D(SN^{k-1-j}) + 2^k - 2 - D(S_k)) + 7) + D(f(S_k)) = D(f(S_k)) + 2(2^{k-1}D(B) - \sum_{j=0}^{k-2} 2^j D(SN^{k-1-j}) - D(S_k)) + 2^{k+1} + 3$. □

# 4 Algorithm for Fault-Tolerant Routing on HDN with Arbitrary Number of Faulty Nodes and Experimental Results

In this section, we propose an efficient practical algorithm for fault-tolerant routing in HDN with arbitrary number of faulty nodes by giving a set of faulty nodes $F$ and two non-faulty nodes $u$ and $v$ in HDN$(B, k, S)$, $k > 0$.

This algorithm also finds four non-faulty nodes $u_s$, $v_s$, $u_r$, and $v_r$. First, the algorithm finds fault-free paths of length at most 1 from $u$ and $v$ to $u_s$ and $v_s$. Next, the algorithm finds a non-faulty

node $u_r$ using RDN_FaultTolerantRouting algorithm inside RDN(HDN$(B,k,S),u_{s_3}$). If $u_{s_3} \neq v_{s_3}$, we use the RDN which contains information of faulty nodes of both the RDN(HDN$(B,k,S),u_{s_3}$) and RDN(HDN$(B,k,S),v_{s_3}$). And then, the algorithm finds a super-node which contains nodes in the given path and the number of faulty nodes is minimum. The node-id of the ending node is equal to $u_{s_3}$ in the super-node. Let $u_r$ be the ending node. $v_r$ is determined in the same way as $u_r$. If the algorithm succeeded to find two fault-free paths from $u_s$ and $v_s$ to $u_r$ and $v_r$ then the algorithm finds a path connecting $u_r$ and $v_r$ inside the super-node. If all steps succeed then five fault-free paths are found. Connecting these paths, a fault-free path $[u \to u_s \to u_r \to v_r \to v_s \to v]$ can be found. If the algorithm fail to find fault-free path then the algorithm retry to find a fault-free path to change the combination of $u_s$ and $v_s$. The algorithm is formally given in Algorithm 2.

---

**Algorithm 2:** HDN_FaultTolerantRouting_unlimited(HDN$(B,k,S),u,v,F$)
input: HDN$(B,k,S)$;
input: A non-faulty node $u(u_0,u_1,u_2,u_3)$ (the node representation of level $k$);
input: A non-faulty node $v(v_0,v_1,v_2,v_3)$ (the node representation of level $k$);
input: A set of faulty nodes $F$;
output: A fault-free path $u \Rightarrow v$;
**begin**
find fault-free paths $P_p(u)$, $0 \le p \le d_0+k, P_0(u)=u$, or $P_i(u)=u \to u^{(i)}, 1 \le i \le d_0+k$, of length at most 1;
/* $u^{(i)}$ is $i$th adjacent node of $u$ */
find fault-free paths $P_p(v)$, $0 \le p \le d_0+k, P_0(v)=v$, or $P_i(v)=v \to v^{(i)}, 1 \le i \le d_0+k$, of length at most 1;
/* $v^{(i)}$ is $i$th adjacent node of $v$ */
**for** $i_u = 1$ **to** number of path in $P_p(u)$
$u_s(u_{s_0},u_{s_1},u_{s_2},u_{s_3})$ = the end node of $P_{i_u}(u)$;
**for** $i_v = 1$ **to** number of path in $P_p(v)$
$v_s(v_{s_0},v_{s_1},v_{s_2},v_{s_3})$ = the end node of $P_{i_v}(v)$;
r_f = (RDN(HDN$(B,k,S),u_{s_3}) \cap F) \cup$ (RDN(HDN$(B,k,S),v_{s_3}) \cap F$);
/* r_f is set of faulty nodes in both of RDN(HDN$(B,k,S),u_{s_3}$) and RDN(HDN$(B,k,S),v_{s_3}$) */
$P$ = RDN_FaultTolerantRouting(RDN(HDN$(B,k,S),u_{s_3}$), $(u_{s_0},u_{s_1},u_{s_2})$, $(v_{s_0},v_{s_1},v_{s_2})$, r_f);
min_fn $= \infty$; /* min_fn is the minimum number of faulty nodes on super-node */
**for** $i_s = 1$ **to** number of nodes of path $P$
$n(n_0,n_1,n_2,n_3) = i_s$th node of $P$;
**if** min_fn $>$ |SN(HDN$(B,k,S),n_0,n_1,n_2) \cap F$|
min_fn = |SN(HDN$(B,k,S),n_0,n_1,n_2) \cap F$|;
/* update the number of faulty nodes on super-node */
$u_r = (n_0,n_1,n_2,u_{s_3})$;
$v_r = (n_0,n_1,n_2,v_{s_3})$;
**endif**
**endfor**
RDN_FaultTolerantRouting(RDN(HDN$(B,k,S),u_{s_3}),u_s,u_r,F$);
/* find a fault-free path connecting $u_s$ and $u_r$ */
RDN_FaultTolerantRouting(RDN(HDN$(B,k,S),v_{s_3}),v_s,v_r,F$);
/* find a fault-free path connecting $v_s$ and $v_r$ */
Supernode_FaultTolerantRouting$(u_r,v_r)$;
/* find a fault-free path connecting $u_r$ and $v_r$ */
**endfor**
**endfor**
**end**

---

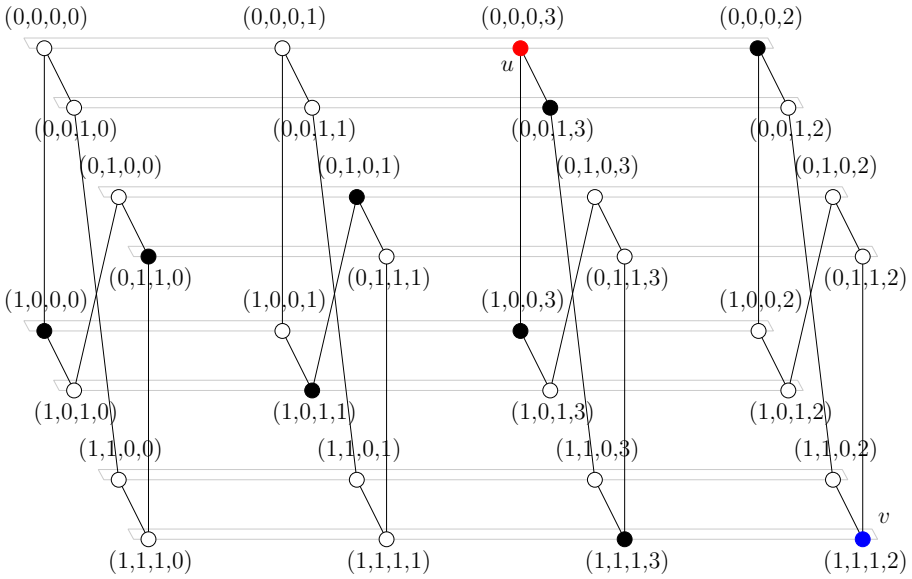


Figure 9: An example of fault-tolerant routing on HDN$(B, 1, S)$ with $s_1 = 4$ by using Algorithm 2

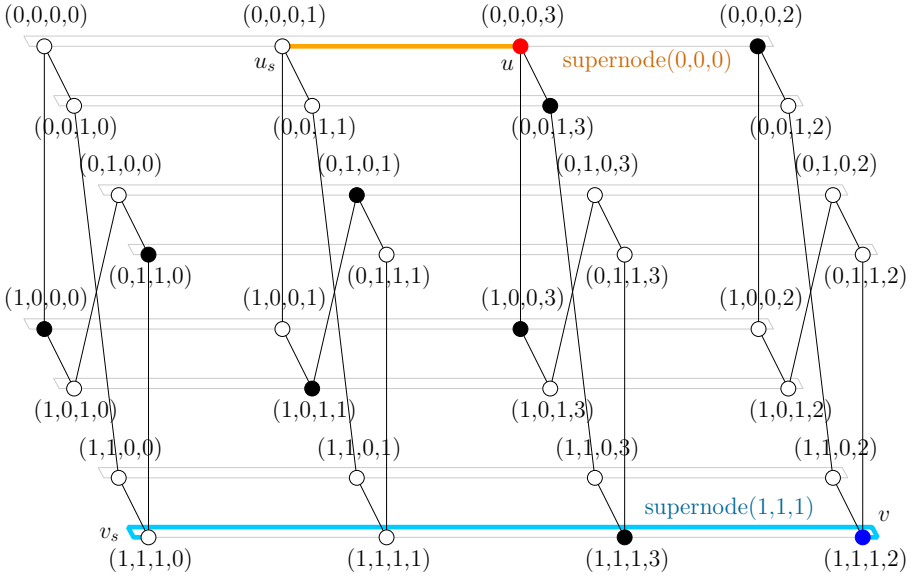


Figure 10: Finding fault-free paths of $u \to u_s$ and $v \to v_s$

Figures 9−12 show a path example on an HDN$(B, k, S)$ found by Algorithm 2. The base network $B$ is a 3-cube with $k = 1$ and $S = S_1$ is a 2-cube.

Referring to Figure 9, the two nodes $u(u_0, u_1, u_2, u_3)$ and $v(v_0, v_1, v_2, v_3)$ are (0,0,0,3) and (1,1,1,2), respectively. There are seven faulty nodes, larger than the node degree of the HDN. The faulty nodes ((0,0,0,2), (0,0,1,3), (0,1,0,1), (0,1,1,0), (1,0,0,0), (1,0,0,3), (1,0,1,1), and (1,1,1,3)) are marked with solid black circles. All the shortest paths between $u$ and $v$ contains faulty nodes (nodes (0,0,0,2), (0,0,1,3), and (1,0,0,3) are faulty).

Referring to Figure 10, first, the algorithm finds fault-free paths of length at most 1 from $u$ and $v$ to $u_s$ and $v_s$, in their super-nodes, respectively: $u_s = (0, 0, 0, 1)$ and $v_s = (1, 1, 1, 0)$. We get two paths: [(0,0,0,3) → (0,0,0,1)] and [(1,1,1,2) → (1,1,1,0)].

Referring to Figure 11, the algorithm then finds two nodes $u_r$ and $v_r$ by using the RDN fault-

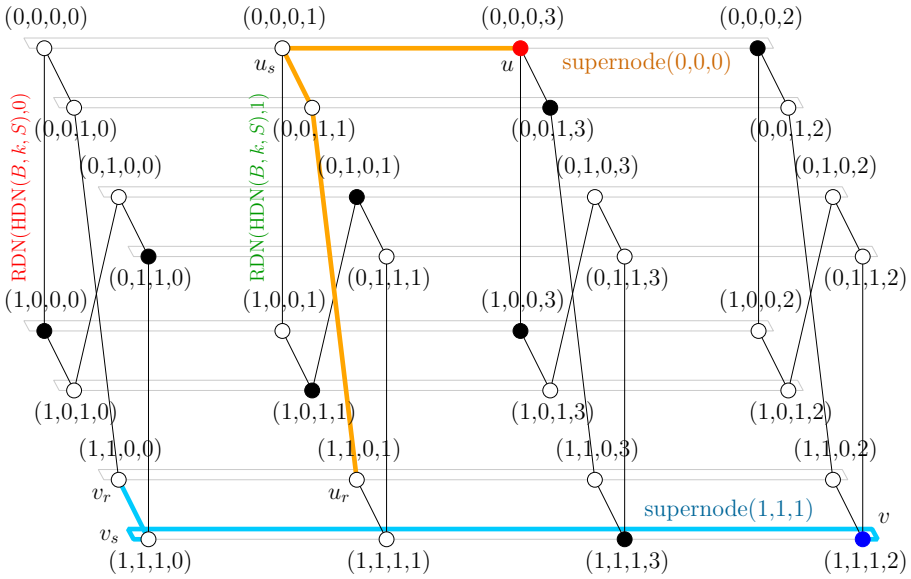


Figure 11: Finding fault-free paths of $u_s \to u_r$ and $v_s \to v_r$

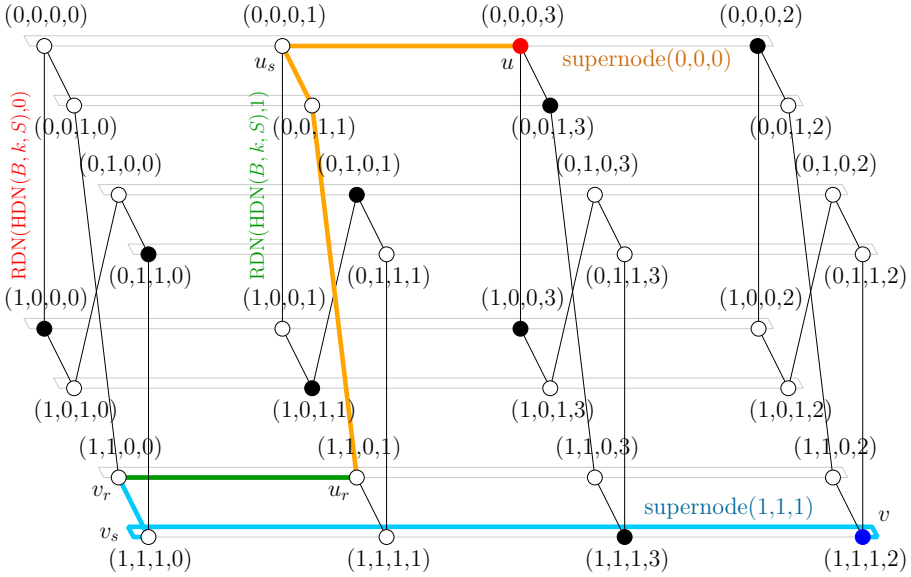


Figure 12: Finding a fault-free path of $u_r \to v_r$

tolerant routing algorithm so that nodes $u_r$ and $v_r$ are located in a same super-node which contains the smallest number of faulty nodes. We get $u_r$ = (1,1,0,1) and $v_r$ = (1,1,0,0).

Referring to Figure 12, the algorithm routes $u_r$ to $v_r$ in the super-node. The final path found by Algorithm 2 is [(0,0,0,3) → (0,0,0,1) → (0,0,1,1) → (1,1,0,1) → (1,1,0,0) → (1,1,1,0) → (1,1,1,2)].

We have performed a simulation to evaluate the algorithm. Our simulation focuses on the successful ratio of finding a fault-free path and the average path length of a fault-free path, for an HDN with different node faulty probabilities. We use two hierarchical dual-net (HDN($B_1, k_1, S_1$) and HDN($B_2, k_2, S_2$)). HDN($B_1, k_1, S_1$) uses a $3 \times 2 \times 5$ torus as the base network and let $k_1 = 2$ and $S_1 = \{15, 5\}$. Therefore, the numbers of nodes in HDN($B_1, 1, S_1$) and HDN($B_1, 2, S_1$) are $(30/15) \times (30/15) \times 2 \times 15 = 120$ and $(120/5) \times (120/5) \times 2 \times 5 = 5760$, receptively. The node degree of HDN($B_1, k_1, S_1$) is $5 + 2 = 7$. HDN($B_2, k_2, S_2$) uses a 3-cube as the base network and let

$k_2 = 2$ and $S_2 = \{4, 2\}$. Therefore, the numbers of nodes in HDN$(B_2, 1, S_2)$ and HDN$(B_2, 2, S_2)$ are $(8/4) \times (8/4) \times 2 \times 4 = 32$ and $(32/2) \times (32/2) \times 2 \times 2 = 1024$, receptively. The node degree of HDN$(B_2, k_2, S_2)$ is $3 + 2 = 5$. The simulation consists of the following four steps.

1. Mark fault nodes in HDN$(B, 2, S)$ randomly at a specified percentage of the faulty nodes.
2. Select two non-faulty nodes, $u$ and $v$, in HDN$(B, 2, S)$ randomly.
3. Find fault-free path for the two nodes by using the HDN_FaultTolerantRouting algorithm and HDN_FaultTolerantRouting_unlimited algorithm.
4. Record whether the path is found successfully. If the path is found successfully, then record the path length.

Figure 13 illustrates the successful ratio of finding a fault-free path on HDN$(B_1, 2, S_1)$ with the base network of a 3D Torus. The x-axis of the graph is the node faulty rate and the y-axis is the successful ratio of finding a fault-free path. The upper line of the graph is the successful ratio of finding a fault-free path using our proposed algorithm. The lower line of the graph is the successful ratio of finding a fault-free path using HDN_Routing algorithm [10] which does not consider the fault tolerance. Therefore, the successful ratio rapidly decreases as the faulty rate increases. The successful ratio is less than 50% if the faulty rate is 10% and is about 15% if the faulty rate is 25%. On the other hand, the successful ratio of finding a fault-free path using our proposed algorithm slowly decreases as the faulty rate increases. The ratio is always 100% if the number of faulty node is less than the node degree. The ratio of successful routing is over 99.9% if the faulty rate is 20% (1152 faulty nodes) and is over 98% even if the faulty rate is 25% (1440 faulty nodes).

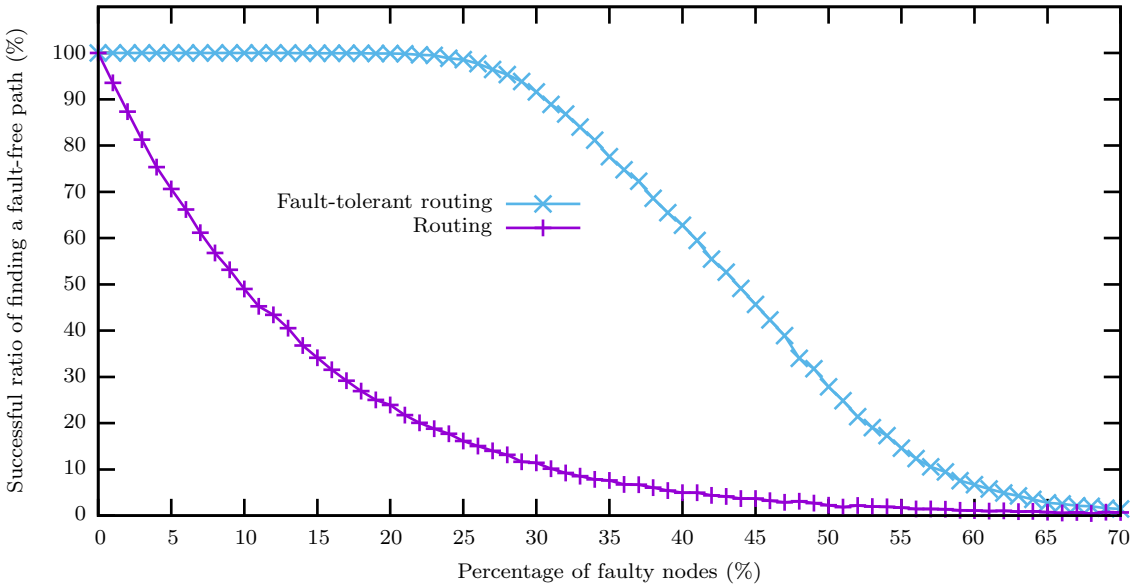


Figure 13: A successful ratio of finding a fault-free path with HDN$(B_1, 2, S_1)$, $B_1 = 3 \times 2 \times 5$ torus

Figure 14 illustrates the average path length of fault-free paths on HDN$(B_1, 2, S_1)$. The x-axis of the graph is the node faulty rate and the y-axis is the average path length. The path length found by HDN_Routing algorithm [10] become shorter as the faulty rate increases. This is because, as the faulty rate increases, the successful ratio decreases, especially for those paths which have longer lengths, and we just consider the paths which successfully connect the two nodes. On the other hand, the path length found by our proposed algorithm becomes longer as the faulty rate increases. This is because as the faulty rate increases, the number of roundabout routes also increases.

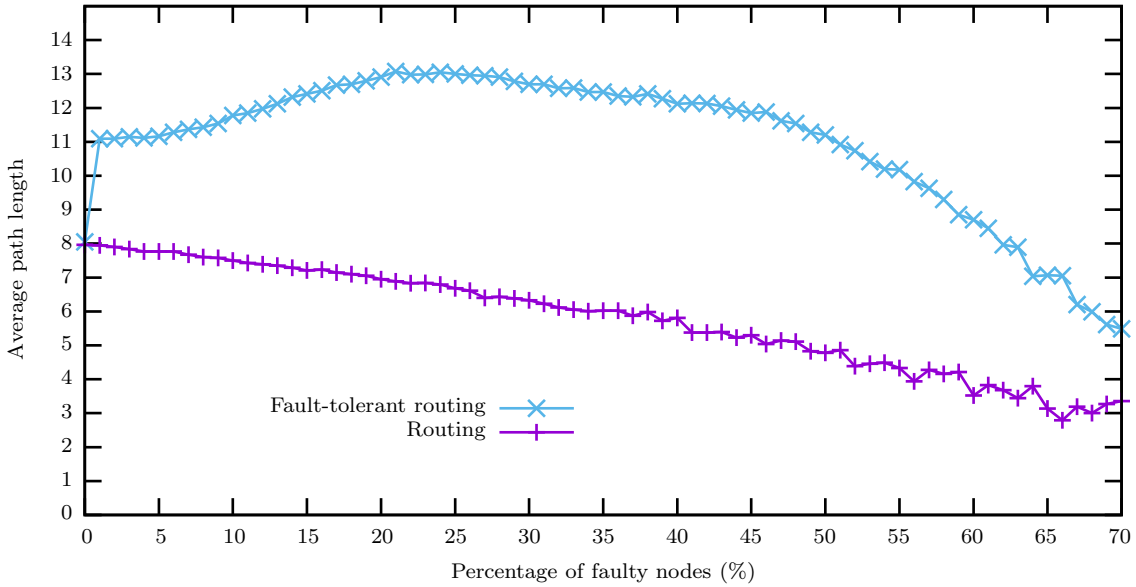


Figure 14: Average path length of fault-free path with HDN$(B_1, 2, S_1)$, $B_1 = 3 \times 2 \times 5$ torus

Figure 15 shows the successful ratio of finding a fault-free path on HDN$(B_2, 2, S_2)$ with the base network of a 3-cube. Again, the successful ratio of finding a fault-free path using our proposed algorithm slowly decreases as the faulty rate increases. The successful ratio of routing is over 96% even if the faulty rate is 25% (256 faulty nodes).

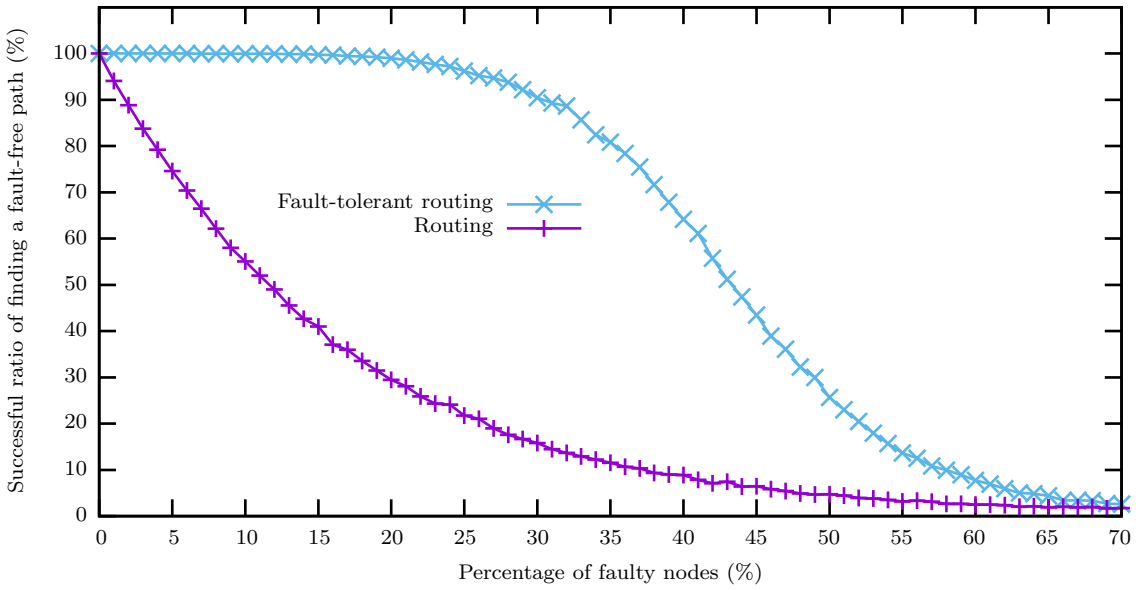


Figure 15: A successful ratio of finding a fault-free path with HDN$(B_2, 2, S_2)$, $B_2 = 3$-cube

Figure 16 shows the average path length of finding fault-free paths on HDN$(B_2, 2, S_2)$ with the base network of a 3-cube. Again, as the faulty rate increases, the path length found by HDN_Routing algorithm becomes shorter, and the path length found by our proposed algorithm becomes longer.

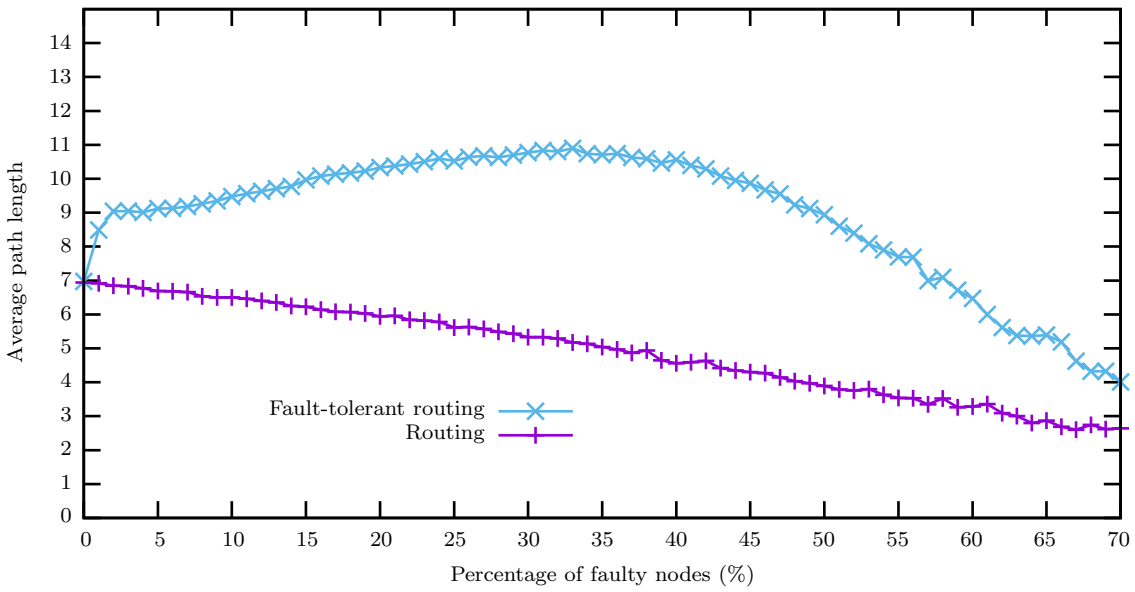


Figure 16: Average path length of fault-free path with HDN($B_2, 2, S_2$), $B_2$ = 3-cube

## 5 Concluding Remarks

In this paper we proposed two fault-tolerant routing algorithms on hierarchical dual-net. The first algorithm can always find a fault-free path on an HDN($B, k, S$) in $O(2^k F(B))$ time with at most $d_0 + k - 1$ faulty nodes. We also proposed an efficient algorithm for fault-tolerant routing on an HDN with arbitrary number of faulty nodes, and performed a simulation. The successful ratio of finding a fault-free path is much better than that of the traditional routing algorithm. The future work may include the node-to-set or set-to-set disjoint-path routing.

## References

[1] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. Blue gene/l torus interconnection network. *IBM Journal of Research and Development*, 49(2/3):265–276, 2005.

[2] J. Arai and Y. Li. Disjoint-path routing on hierarchical dual-nets. *International Journal of Networking and Computing*, 4(2):260–278, Jul. 2014.

[3] M. Ebrahimi, M. Daneshtalab, and J. Plosila. Fault-tolerant routing approach for 3d stacked meshes. In *Proceedings of the 2014 Workshop on Manufacturable and Dependable Multicore Architectures at Nanoscale*, pages 5–10, Dresden, Germany, Mar. 2014.

[4] K. Ghose and K. R. Desai. Hierarchical cubic networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(4):427–435, April 1995.

[5] Q-P Gu and S. Peng. Optimal algorithms for node-to-node fault tolerant routing in hypercubes. *The Computer Journal*, 39(7):626–629, 1996.

[6] IBM. *Roadrunner: Hardware and Software Overview*. IBM Corporation, http://www.redbooks.ibm.com/redpapers/pdfs/redp4477.pdf, 2009.

[7] Y. Li, S. Peng, and W. Chu. Efficient collective communications in dual-cube. *The Journal of Supercomputing*, 28(1):71–90, April 2004.

[8] Y. Li, S. Peng, and W. Chu. Disjoint-paths and fault-tolerant routing on recursive dual-net. In *Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 48–56, Hiroshima, Japan, Dec. 2009. IEEE Computer Society Press.

[9] Y. Li, S. Peng, and W. Chu. Recursive dual-net: A new universal network for supercomputers of the next generation. In *Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing*, pages 809–820, Taipei, Taiwan, June 2009.

[10] Y. Li, S. Peng, and W. Chu. Hierarchical dual-net: A flexible interconnection network and its routing algorithm. In *Proceedings of the Second International Conference on Networking and Computing*, pages 58–67, Osaka, Japan, Nov. 2011.

[11] F. P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Commun. ACM*, 24:300–309, May 1981.

[12] TOP500. *Supercomputer Sites*. http://top500.org/, Nov. 2014.