Disjoint-Paths and Fault-Tolerant Routing on Recursive Dual-Net

Yamin Li and Shietung Peng
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan
{*yamin, speng*}*@k.hosei.ac.jp*

Wanming Chu
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan
w-chu@u-aizu.ac.jp

***Abstract*—The recursive dual-net is a newly proposed interconnection network for of massive parallel computers. The recursive dual-net is based on a recursive dual-construction of a base network. A k-level dual-construction for $k > 0$ creates a network containing $(2n_0)^{2^k}/2$ nodes with node-degree d_0+k, where n_0 and d_0 are the number of nodes and the node-degree of the base network, respectively. The recursive dual-net is node and edge symmetric and can contain huge number of nodes with small node-degree and short diameter. Disjoint-paths routing and fault-tolerant routing are fundamental and critical issues for the performance of an interconnection network. In this paper, we propose efficient algorithms for disjoint-paths and fault-tolerant routings on the recursive dual-net.**

***Keywords*-interconnection network; disjoint paths; fault-tolerant routing;**

I. Introduction

In massively parallel processor (MPP), the interconnection network plays a crucial role on the issues such as communication performance, hardware cost, computational complexity, fault-tolerance, etc. Much research has been reported in the literatures for interconnection networks that can be used to connect parallel computers of large scale (see [1], [2], [3] for the review of the early work). The following two categories have attracted a great research attention. One is the hypercube-like family that has the advantage of short diameters for high-performance computing and efficient communication [4], [5], [6], [7], [8]. The other is 2D/3D mesh or torus that has the advantage of small and fixed node-degrees and easy implementations. Traditionally, most MPPs in the history including those built by NASA, CRAY, FGPS, IBM, etc., use 2D/3D mesh or torus or their variations with extra diagonal links. The recursive networks also have been proposed as effective interconnection networks for parallel computers of large scale. For example, the WK-recursive network [9], [10] is a class of recursive scalable networks. It offers a high-degree of regularity, scalability, and symmetry and has a compact VLSI implementation.

Recently, due to the advance in computer technologies, the community of supercomputers rises competition to construct high-performance supercomputers containing millions of nodes [11]. For such a very-large-scale parallel computer, the traditional interconnection networks such as hypercube or mesh networks will have either large node-degree or long diameter. New interconnection networks that have the merits of traditional networks such as node and edge symmetry and recursive structure etc., and also have small node-degree and short diameter are in great demand.

In this paper, we first present a new interconnection network, called *Recursive Dual-Net* (RDN) [12]. A recursive dual-net is based on a recursive dual-construction of a base network. The dual-construction extends a network with n nodes and node-degree d to a network with $2n^2$ nodes and node-degree $d + 1$. The k-level RDN is obtained by recursively applying dual-construction k time starting from a symmetric base-network B. The recursive dual-net has the all the merits mentioned above and can connect a huge number of nodes with just a small number of links per node. It is not difficult to construct a recursive dual-net with 50 millions of nodes that has 5 links per node and its diameter equals to 30. We show some topological properties of the recursive dual-net. Then we give the basic routing algorithm for the recursive dual-net.

The main contributions of this paper are the disjoint-paths and fault-tolerant routing algorithms in recursive dual-net. Let d_0 be the node-degree of the symmetric base-network B. Given two nodes s and t in a recursive dual-net $RDN^k(B)$ with a base network B such that, for any two nodes in B, there are d_0 disjoint paths connecting them in $O(d_0^2)$ time, we propose an $O((d_0+k)^2)$ time algorithm for finding d_0+k disjoint paths connecting s and t. Given two nonfaulty nodes s and t and up to $d_0 + k - 1$ faulty nodes in $RDN^k(B)$ with a base network B such that, for any two nonfaulty nodes and up to $d_0 - 1$ faulty nodes in B, there exists a fault-free path connecting them in $O(d_0)$ time, we propose a fault-tolerant routing algorithm that connecting s and t by a fault-free path in $O(d_0 + k)$ time. Finally, we propose a heuristic fault-tolerant routing algorithm in $RDN^k(B)$ for arbitrary number of faulty nodes. The algorithm can find a fault-free algorithm between two nonfaulty nodes with very high probability.

The rest of this paper is organized as follows. Section 2 describes the recursive dual-net in details. Sections 3 gives the disjoint-paths and fault-tolerant routing algorithms on a recursive dual-net. In Section 4, we propose an efficient and

practical fault-tolerant algorithm that can finds a fault-free path for arbitrary number of faulty nodes on a recursive dual-net. Simulation results show that the proposed algorithm can find a fault-free path on any two nonfaulty nodes on an $RDN^2(B(3))$ containing up to 150 faulty nodes with very high probability. Section 5 concludes the paper.

II. Recursive Dual-Net

Let G be an undirected graph. The size of G, denoted as $|G|$, is the number of vertices. A path from node s to node t in G is denoted by $s \rightarrow t$. The length of the path is the number of edges in the path. For any two nodes s and t in G, we denote $D(s,t)$ as the length of a shortest path connecting s and t. The diameter of G is defined as $D(G) = \max\{D(s,t)|s,t \in G\}$. For any two nodes s and t in G, if there is a path connecting s and t, we say G is a connected graph.

Suppose we have a symmetric connected graph B and there are n_0 nodes in B and the node degree is d_0. A Recursive Dual-Net $RDN(B)$, also denoted as $RDN^k(B(n_0))$, can be recursively defined as follows:

1) $RDN^0(B) = B$ is a symmetric connected graph with n_0 nodes, called *base network*;
2) For $k > 0$, an $RDN^k(B)$ is constructed from $RDN^{k-1}(B)$ by a dual-construction as explained below (also see Figure 1).

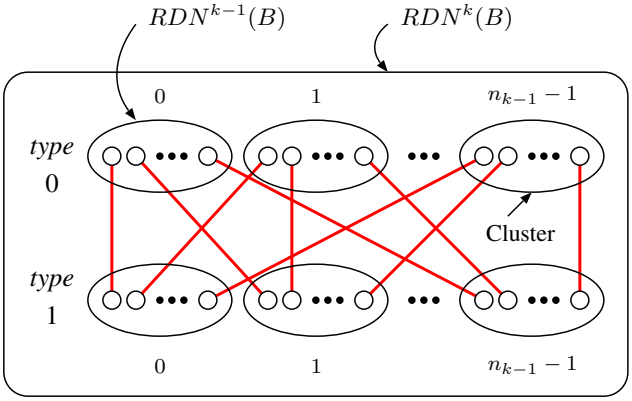

Figure 1. Build an $RDN^k(B)$ from $RDN^{k-1}(B)$

Dual-construction: Let $RDN^{k-1}(B)$ be referred to as a *cluster* of level k and $n_{k-1} = |RDN^{k-1}(B)|$. An $RDN^k(B)$ is a graph that contains $2n_{k-1}$ clusters of level k as subgraphs. These clusters are divided into two sets with each set containing n_{k-1} clusters. Each cluster in one set is said to be of *type* 0, denoted as C_i^0, where $0 \le i \le n_{k-1}-1$ is the cluster ID. Each cluster in the other set is of *type* 1, denoted as C_j^1, where $0 \le j \le n_{k-1}-1$ is the cluster ID. At level k, each node in a cluster has a new link to a node in a distinct cluster of the other type. We call this link *cross-edge* of level k. By following this rule, for each pair of clusters C_i^0 and C_j^1, there is a unique edge connecting a node in C_i^0 and a node in C_j^1, $0 \le i,j \le n_{k-1}-1$. In Figure 1, there are n_{k-1} nodes within each cluster $RDN^{k-1}(B)$.

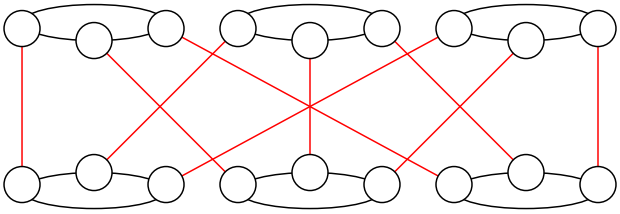

Figure 2. A Recursive Dual-Net $RDN^1(B(3))$

We give two simple examples of recursive dual-nets with $k = 1$ and 2, in which the base network is a ring with 3 nodes, in Figure 2 and Figure 3, respectively. Figure 2 depicts an $RDN^1(B(3))$ network. There are 3 nodes in the base network, therefore the number of nodes in $RDN^1(B(3))$ is 2×3^2, or 18. Figure 3 shows the $RDN^2(B(3))$ constructed from the $RDN^1(B(3))$ in Figure 2. We did not show all the nodes in the figure. The number of nodes in $RDN^2(B(3))$ is 2×18^2, or 648.

Similarly, we can construct an $RDN^3(B(3))$ containing 2×648^2, or 839,808 nodes with node-degree of 5 and diameter of 22. In contrast, the 839,808-node 3D torus machine (adopt by IBM Blue Gene/L [13]) configured as $108 \times 108 \times 72$ nodes, the diameter is equal to $54+54+36 = 144$ with a node degree of 6.

We can see from the recursive dual-construction described above that an $RDN^k(B)$ is a symmetric connected network with node-degree $d_0 + k$, where d_0 is the node-degree of the base network B. The number of nodes n_k in $RDN^k(B)$ satisfies the recurrence $n_k = 2n_{k-1}^2$ for $k > 0$. Solving the recurrence, we get $n_k = (2n_0)^{2^k}/2$.

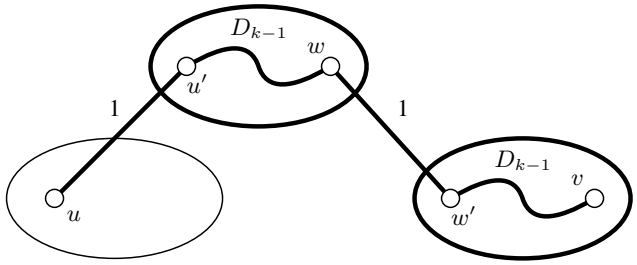

Figure 4. The diameter of the Recursive Dual-Net

Concerning the diameter D_k of $RDN^k(B)$, we know that the worst-case (the longest one) for the shortest path $P(u,v)$ connecting any two nodes u and v in $RDN^k(B)$ is as follow: u and v are of the same type and path $P = u \rightarrow u' \rightarrow w \rightarrow w' \rightarrow v$, where $u \rightarrow u'$ and $w \rightarrow w'$ are cross-edges of level k, and $|u' \rightarrow w| = |w' \rightarrow v| = D_{k-1}$, as shown as in Figure 4. Therefore, the diameter of $RDN^k(B)$

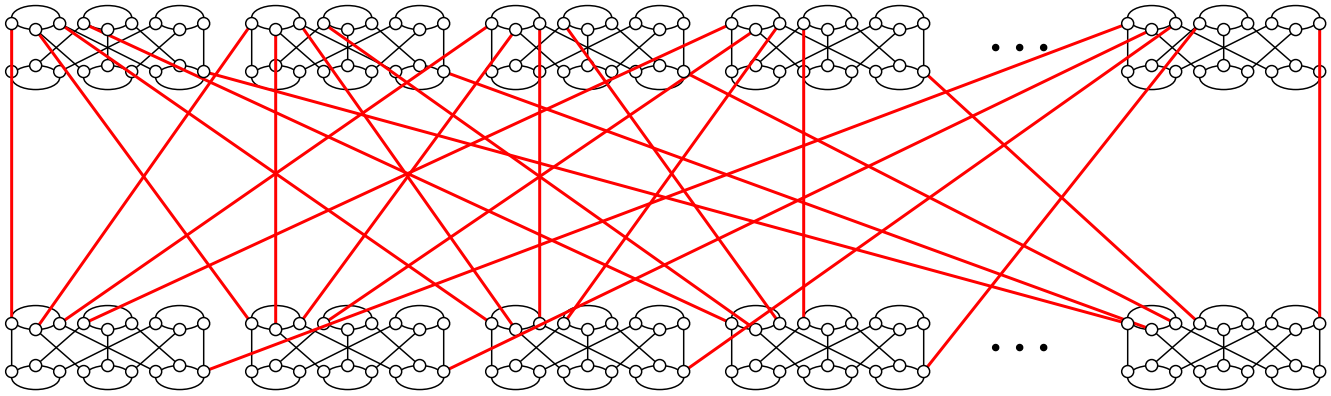

Figure 3. A Recursive Dual-Net $RDN^2(B(3))$

satisfies the recurrence $D_k = 2D_{k-1} + 2$ for $k > 0$. Solving the recurrence, we get $D_k = 2^k D_0 + 2^{k+1} - 2$, where D_0 is the diameter of the base network.

The bisection bandwidth is important for fault-tolerance. Next, we investigate the bisection bandwidth of the $RDN^k(B)$ for $k \geq 1$.

From the dual-construction, we know that there is no link between the clusters of level k that are of the same type. Therefore, the minimum number of links those removal will disconnect two halves occurs when both halves contain equal numbers of clusters of type 0 or 1. That is, the minimum number of links those removal will disconnect two halves equals to half of the total number of cross-edges of level k which is $\lceil (2n_0)^{2^k}/8 \rceil$.

Notice that if n_0 is odd and $k = 1$ we should divide the RDN into two halves such that one half contains $\lfloor n_0/2 \rfloor$ (or $\lceil n_0/2 \rceil$) type 0 clusters and $\lceil n_0/2 \rceil$ (or $\lfloor n_0/2 \rfloor$) type 1 clusters. For example, the bisection bandwidth of $RDN^1(B(3))$ is $\lceil 6^2/8 \rceil = \lceil 9/2 \rceil = 5$.

We summarize the discussion above about the fundamental properties of the Recursive Dual-Net in the following theorem.

Theorem 1: Assume that the base network B is a symmetric graph with size n_0, node-degree d_0, and the diameter D_0. Then, the size, the node-degree, the diameter and the bisection bandwidth of $RDN^k(B)$ are $(2n_0)^{2^k}/2$, $d_0 + k$, $2^k D_0 + 2^{k+1} - 2$, and $\lceil (2n_0)^{2^k}/8 \rceil$, respectively.

It is desirable for the interconnection network to be symmetric and to have a low node degree and a low diameter. In general, the requirements of a low node degree and a low diameter are conflicting. We introduce *cost ratio* $CR(G)$ as an important measure for the combined effects of the hardware cost and the software efficiency of an interconnection network presented as graph G. Let $|(G)|$, $d(G)$, and $D(G)$ be the number of nodes, the node-degree, and the diameter of G, respectively. We define $CR(G)$ as

$$CR(G) = (d(G) + D(G))/\lg|(G)|.$$

The motivation here is that the node-degree and diameter should not increase faster than the logarithm of the size of of the graph. It should be considered as a basic rule for high-performance MPPs. The design of interconnection network should make effort to reduce the cost ratio, especially for an MPP with very large scale. One of the reasons that hypercube has been and will be still popular as an interconnection network of MPPs is that its node-degree and diameter grow logarithmically with its size. Therefore, the cost ratio of hypercube is a constant 2 for any size. However, for an MPP with more than a million of nodes, the logarithmic growth rate is still too big for the hardware technologies or software efficiency.

Table I summarizes the number of nodes, the node-degree, the diameter, and the cost ratio for 3D torus, hypercube, CCC, dual-cube, WK-recursive network and recursive dual-net. The *torus*, also called *wrap-around mesh* or a *toroidal mesh*, was adopt by IBM Blue Gene/L. This topology includes the p-ary, q-cube which is a q-dimensional torus with the restriction that each dimension is of the same size p. In a $CCC(n)$, each node in an n-cube is replaced with an n-node ring [7]. A dual-cube $DC(n)$ contains 2^n $(n-1)$-cubes called *clusters* [5]. Half of the clusters are of type 0 and the other half are of type 1. There is a unique link (cross-edge) connecting each pair of clusters of distinct types. $DC(n)$ is equal to $RDN(2^{n-1}, 1)$, where the base network is an $(n-1)$-cube. A WK-recursive network of level t denoted as $WK\ (n,t)$ can be constructed recursively as follows [10]. $WK(n,1)$ is an n-node complete graph augmented with n open links each at a node. Each node of $WK(n,t)$ is incident with $n-1$ substituting links and one flipping link (or open link). The substituting links are those within basic building blocks, and the j-flipping links are those connecting two embedded $WK(n,j)$.

The RDN is a potential candidate for the interconnection network of an MPP. For example, $RDN^2(B(27))$ (where $B(27)$ is a 3-ary, 3-cube) has 4,251,528 nodes and its node-degree, diameter, and cost ratio are 8, 18, and 1.18,

Table I
CR OF RECURSIVE DUAL-NET AND THE OTHER NETWORKS

Network	Number of nodes	Node-degree	Diameter	CR
p-ary, 3-cube	p^3	6	$3p/2$	$(6+3p/2)/3\lg p$
n-cube	2^n	n	n	2
$CCC(n)$	$n*2^n$	3	$2n+\lfloor n/2 \rfloor -2$	$(2n+\lfloor n/2 \rfloor +1)/(n+\lg n)$
$DC(n)$	2^{2n-1}	n	$2n$	$3n/(2n-1)$
$WK(n,t)$	n^t	n	2^t-1	$(n+2^t-1)/\lg n^t$
$RDN^k(B)$	$n_k=(2n_0)^{2^k}/2$	d_0+k	$2^k*D_0+2^{k+1}-2$	$(d_0+k+D_k)/\lg n_k$

respectively. As far as the CR is concerned, the value 1.18 is the lowest value among all known topologies.

III. DISJOINT-PATH AND FAULT-TOLERANT ROUTING ON RDN

The problem of finding a path from a source s to destination t and forwarding a message along the path is known as the routing problem. Finding multiple, disjoint paths for routing from s to t is called disjoint-path routing. Finding a fault-free path from s to t 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 will propose efficient algorithms for these routing problems on the recursive dual-net.

Given two nodes u and v in $RDN^k(B)$, we first present a simple routing algorithm that finds a shortest path from u to v. Assume that a routing algorithm RDN_routing (B,u,v) for the base network B is available. The routing algorithm that routes node u to node v in $RDN^k(B)$ is a recursive one for $k>0$. If u and v are in the same cluster of level k then just call itself for $k-1$. Otherwise, we assume that u and v has distinct typeID (for the case $u_0=v_0$, we simply route u to w via a cross-edge of level k then we treat w as u). We route u to u' with $u'_2=v_1$ and v to v' with $v'_2=u_1$ inside the clusters of level k where u and v belong to. This can be done by recursive calls for $k-1$. Then we can route u' to v' in 1 hop since there is a cross-edge of level k from u' to v'. The routing algorithm is described formally as Algorithm 1.

The following lemma follows directly from Algorithm 1.

Lemma 1: Assume that a path connecting two nodes in the base-network B can be found in $O(f(n_0))$ time, where n_0 is the number of nodes in B. Then, in $RDN^k(B)$, a path from source s to destination t can be found in $O(2^k f(n_0))$ time and the length of the path is at most $2^k*D_0+2^{k+1}-2$, where D_0 is the diameter of the base network.

A. Algorithm for Disjoint-paths Routing on RDN

We introduce the some notation to be used. Let the $d+k$ neighbors of node u be $u^{(i)}, 1\le i\le d+k$, where $u^{(i)}, 1\le i\le d$, are the nodes belong to $RDN_u^k(B)$, and edge $(u,u^{(i)}), d+1\le i\le d+k$, is the cross-edge of level $i-d$. Let $u^{(i,j)}=(u^{(i)})^{(j)}$ for $1\le i,j\le d+k$, and so on. The algorithm for finding $d+k$ disjoint paths from u to v on $RDN^k(B)$ can be divided into three cases. In Case 1, u and v are in the same cluster of level k. In this case, $d+k-1$ disjoint paths can be found by a recursive call and a path connecting $u^{(d+k)}$ and $v^{(d+k)}$ outside the cluster $RDN_u^{k-1}(B)$ can be found. In Case 2, the clusters containing u and v are of distinct types. In this case, we first find $d+k$ disjoint paths of length at most 2 from u such that each path includes a cross-edge of level k.

Similarly, we find $d+k$ disjoint paths of length at most 2 from v such that each path includes a cross-edge of level k. Then the required $d+k$ disjoint paths can be found by calling RDN_Routing$(RDN^k(B),u^i,v^i)$, $1\le i\le d+k$. In Case 3, the clusters contain u or v are distinct but of the same types. We first route u and v to u^i and $v^i, 1\le i\le d+k$, as in Case 2. Then $u^i, 1\le i\le d+k$, is routed to w^i by disjoint paths of length at most 2 such that each path includes a cross-edge of level k and all clusters that contain $w^i, 1\le i\le d+k$, are distinct. Finally, w^i and v^i can be connected by disjoint paths by calling RDN_Routing$(RDN^k(B),w^i,v^i)$, $1\le i\le d+k$. The proposed algorithm is formally presented as Algo. 2.

Algorithm 1: RDN_routing$(RDN^k(B),u,v)$
begin
  if $k=0$ **then** RDN_routing(B,u,v)
  else
    Case 1:$u_0=v_0$ and $u_1=v_1$
      RDN_routing$(RDN^{k-1}_{u_0,u_1}(B),u_2,v_2)$;
      /* $RDN^{k-1}_{u_0,u_1}(B)$ is the cluster with typeID = u_0 and clusterID = u_1. */
    Case 2: $u_0\neq v_0$
      RDN_routing$(RDN^{k-1}_u(B),u_2,v_1)$;
      /* let $u'=(u_0,u_1,v_1)$. */
      RDN_routing$(RDN^{k-1}_v(B),v_2,u_1)$;
      /* let $v'=(v_0,v_1,u_1)$. */
      connect u' and v' via a cross-edge of level k;
    Case 3: $u_0=v_0$ and $u_1\neq v_1$
      route u to w via the cross-edge of level k;
      route node w to node v as in Case 2;
  endif
end

Example 1 (also see Fig. 5):
$k=2$:
  $u=(u_0,u_1,u_2)=(0,(0,0,0),(0,0,0))$;

Algorithm 2: RDN_disjoint_paths($RDN^k(B), u, v$)
Input: Nodes u and v in $RDN^k(B)$
Output: $d+k$ disjoint paths connecting nodes u and v
begin
Case 1: $u_0 = v_0$ and $u_1 = v_1$
RDN_disjoint_paths($RDN_u^{k-1}(B), u_2, v_2$);
find path $P_{d+k}(u) = u \rightarrow u^{(d+k)} \rightarrow u^{(d+k,j)} \rightarrow u^{(d+k,j,d+k)} = w$, where $1 \leq j \leq d+k-1$;
RDN_routing($RDN^k(B), w, v^{(d+k)}$); /* w and $v^{(d+k)}$ are in clusters of distinct types. */
Case 2: $u_0 \neq v_0$
find $d+k$ disjoint paths, $P_i(u) = u \rightarrow u^{(i)} \rightarrow u^{(i,d+k)} = u^i, 1 \leq i \leq d+k-1$, and $P_{d+k}(u) = u \rightarrow u^{(d+k)} = u^{d+k}$, of length at most 2;
find $d+k$ disjoint paths, $P_i(v) = v \rightarrow v^{(i)} \rightarrow v^{(i,d+k)} = v^i, 1 \leq i \leq d+k-1$, and $P_{d+k}(v) = v \rightarrow v^{(d+k)} = v^{d+k}$, of length at most 2;
if $\exists u^p \in RDN_v^{k-1}(B)$ **then** RDN_routing($RDN_v^{k-1}(B), u^p, v$); /* Assume $v^{(j)} \in P(u^p, v)$. */
remove path $P_j(v)$;
if $\exists v^q \in RDN_u^{k-1}(B)$ **then** RDN_routing($RDN_u^{k-1}(B), v^q, u$); /* Assume $u^{(j)} \in P(v^q, u)$. */
remove path $P_j(u)$;
/* Without loss of generality, assume that $u^i \notin RDN_v^{k-1}(B)$ and $v^i \notin RDN_u^{k-1}(B)$ for all i. */
for i = 1 to $d+k$ **do**
RDN_routing($RDN_{u^i}^{k-1}(B), u^i, s^i$), where $s_2^i = v_1^i$;
RDN_routing($RDN_{v^i}^{k-1}(B), v^i, t^i$); where $t_2^i = u_1^i$
connect s^i to t^i by a cross-edge of level k;
endfor
Case 3: $u_0 = v_0$ and $u_1 \neq v_1$
find $d+k$ disjoint paths, $P_i(u) = u \rightarrow u^{(i)} \rightarrow u^{(i,d+k)} = u^i, 1 \leq i \leq d+k-1$, and $P_{d+k}(u) = u \rightarrow u^{(d+k)} = u^{d+k}$, of length at most 2;
find $d+k$ disjoint paths, $P_i(v) = v \rightarrow v^{(i)} \rightarrow v^{(i,d+k)} = v^i, 1 \leq i \leq d+k-1$, and $P_{d+k}(v) = v \rightarrow v^{(d+k)} = v^{d+k}$, of length at most 2;
/* If $RDN_{u^i}^{k-1}(B) = RDN_{v^j}^{k-1}(B)$ for some u^i and v^j then u^i and v^j can be connected inside the cluster of level k. For simplicity, we assume that $RDN_{u^i}^{k-1}(B) \neq RDN_{v^j}^{k-1}(B)$ for all i and j. */
find $d+k$ disjoint paths $P_i(u^i) = u^i \rightarrow (u^i)^{(j_i)} \rightarrow (u^i)^{(j_i,d+k)} = w^i, 1 \leq j_i \leq k-1$, of length 2 such that $RDN_{w^i}^{k-1}(B)$ are distinct for all $i, 1 \leq i \leq d+k$;
for i = 1 to $d+k$ **do**
RDN_routing($RDN_{w^i}^{k-1}(B), w^i, s^i$), where $s_2^i = v_1^i$;
RDN_routing($RDN_{v^i}^{k-1}(B), v^i, t^i$); where $t_2^i = w_1^i$;
connect s^i and t^i by a cross-edge of level k
endfor
end

$v = (v_0, v_1, v_2) = (1, (1,2,2), (0,2,2))$;
$u_0 = 0$, $u_1 = (0,0,0)$, $u_2 = (0,0,0)$;
$v_0 = 1$, $v_1 = (1,2,2)$, $v_2 = (0,2,2)$.
$u_0 \neq v_0$ (Case 2):

The four disjoint paths from u of length at most 2 are

$u \rightarrow (0,(0,0,0),(0,0,1)) \rightarrow (1,(0,0,1),(0,0,0)) = u^1$
$u \rightarrow (0,(0,0,0),(0,0,2)) \rightarrow (1,(0,0,2),(0,0,0)) = u^2$
$u \rightarrow (0,(0,0,0),(1,0,0)) \rightarrow (1,(1,0,0),(0,0,0)) = u^3$
$u \rightarrow (1,(0,0,0),(0,0,0)) = u^4$

The four disjoint paths from v of length at most 2 are

$v \rightarrow (1,(1,2,2),(0,2,0)) \rightarrow (0,(0,2,0),(1,2,2)) = v^1$
$v \rightarrow (1,(1,2,2),(0,2,1)) \rightarrow (0,(0,2,1),(1,2,2)) = v^2$
$v \rightarrow (1,(1,2,2),(1,2,2)) \rightarrow (0,(1,2,2),(1,2,2)) = v^3$
$v \rightarrow (0,(0,2,2),(1,2,2)) = v^4$

Then, u^i is routed to s^i in the clusters of level 2, where

$s^1 = (1,(0,0,1),(0,2,0))$,
$s^2 = (1,(0,0,2),(0,2,1))$,
$s^3 = (1,(1,0,0),(1,2,2))$, and
$s^4 = (1,(0,0,0),(0,2,2))$.

Next, v^i is routed to t^i in the clusters of level 2, where

$t^1 = (0,(0,2,0),(0,0,1))$,
$t^2 = (0,(0,2,1),(0,0,2))$,
$t^3 = (0,(1,2,2),(1,0,0))$, and
$t^4 = (0,(0,2,2),(0,0,0))$.

Finally, s^i is connected to t^i via a cross-edge of level k.

Theorem 2: In $RDN^k(B)$, assume that, for any two nodes in the base-network $B(n_0, d_0$ disjoint path can be found in $O(f(n_0))$ time. Then $d_0 + k$ disjoint paths connecting any two nodes in $RDN^k(B), k > 0$, can be found in $O(2^k(d_0+k)f(n_0))$ time. The maximum length of the paths is at most $2^k * D_0 + 2^{k+1} - 2$, where D_0 is the diameter of the base network.

Proof: The theorem follows directly from algorithm 2. Let the time complexity for finding $d_0 + k$ disjoint paths in $RDN^k(B)$ be $T(k)$. From Algorithm 2 and Lemma 1,

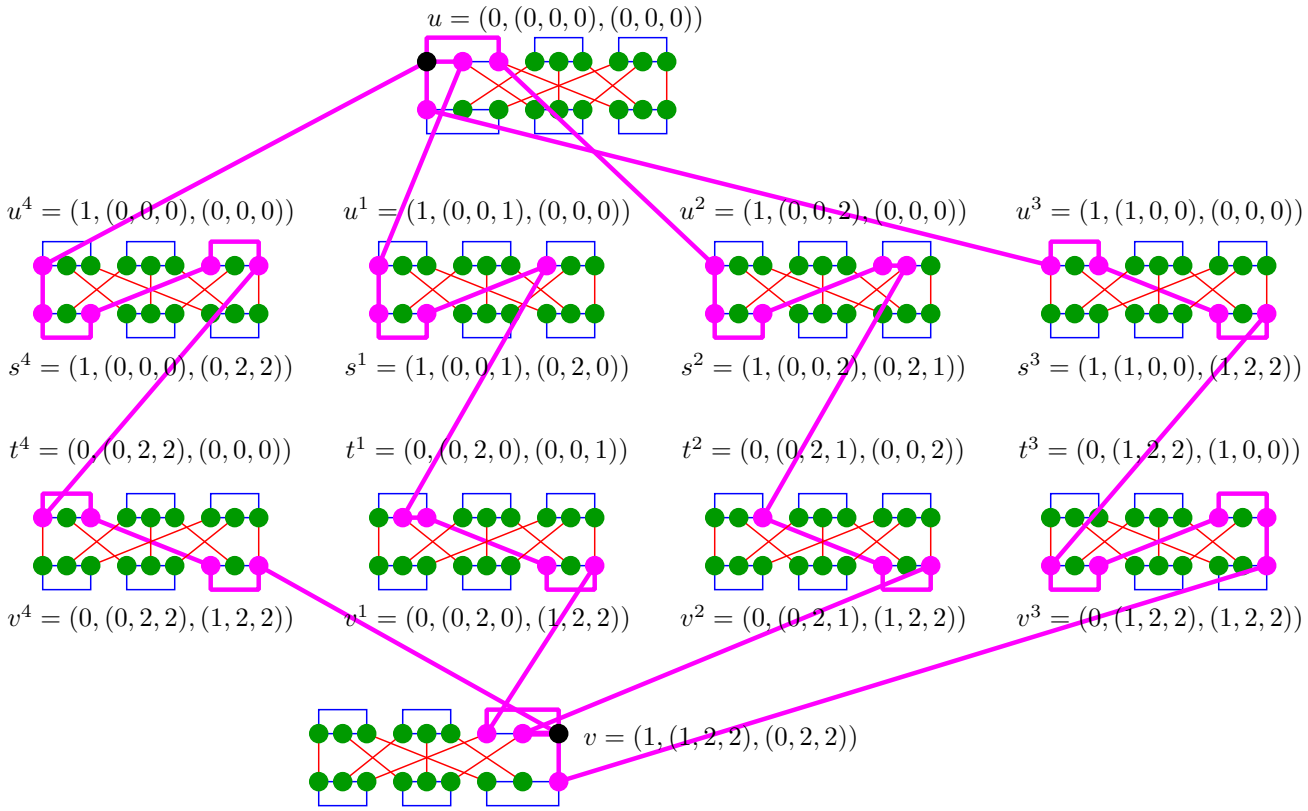

Figure 5. Disjoint-paths routing in $RDN^2(B(3))$

we have $T(k) = O(d_0 + k) \times 2^k f(n_0)) + O(d_0 + k) = O(2^k(d_0 + k)f(n_0))$. The worst-case for the length of the longest path occurs in Case 3 of the algorithm. The length of the longest path equals to $4 + 2 * D_{k-1} + 2$ for $k > 0$. Therefore, the maximum length is $2^k * D_0 + 2^{k+1} - 2$, where D_0 is the diameter for the base network. ❑

B. Algorithm for Fault-tolerant Routing on RDN

Given two non-faulty nodes and a set F of d_0+k-1 faulty nodes in an $RDN^k(B)$, we propose an efficient algorithm for finding a fault-free path connecting u and v. Assume that the clusters that contain faulty nodes can be identified in $O(d_0 + k)$ time. Then, our algorithm can find a fault-free path between u and v in $O(d_0 + k)$ optimal time. The proposed algorithm for fault-tolerant routing is similar to that for disjoint-paths routing.

First assume that u and v are in the same cluster (Case 1). If $|RDN_u^{k-1}(B) \cap F| < d_0 + k$ then just call RDN_Routing($RDN_u^{k-1}(B), u, v$) and we are done. If $|RDN_u^{k-1}(B) \cap F| = d_0 + k$ then we find a fault-free path $P(u, u') : u \rightarrow u^{(d_0+k)} \rightarrow u^{(d_0+k,i)} \rightarrow u^{(d+k,i,d_9+k)} = w$ of length 3, where $i < d_0 + k$. Then, we call RDN_Routing($RDN^k(B), w, v$) and we are done.

Next, we assume that types of the clusters containing u and v are distinct (Case 2). If $|RDN\ k-1_u(B) \cap F| = |RDN_v^{k-1}(B) \cap F| = 0$ then just call RDN_Routing($RDN^k(B), u, v$) and we are done. Otherwise, among the $d_0 + k$ disjoint paths of length at most 2 that route u to other clusters of level k, we find a fault-free path $P(u, u')$ such that $|RDN_{u'}^{k-1}(B) \cap F| = 0$. Similarly, among the $d_0 + k$ disjoint paths of length at most 2 that route v to other clusters of level k, we find a fault-free path $P(v, v')$ such that $|RDN_{v'}^{k-1}(B) \cap F| = 0$. By calling RDN_Routing($RDN^k(B), u', v'$) we are done.

Finally, we consider the case that the clusters containing u and v are distinct but of the same type (Case 3). If at least one of the two clusters is fault-free, say it is the cluster contains v, then we route u to u' by a fault-free path of length at most 2 such that $|RDN_{u'}^{k-1}(B) \cap F| = 0$. By calling RDN_Routing($RDN^k(B), u', v$) we are done. Otherwise, we route u to w by a fault-free path of length at most 2 such that $|RDN_w^{k-1}(B) \cap F| = 0$ and then since the clusters containing w and v are of distinct types, following the Case 2, we can route w to v by a fault-free path. The proposed algorithm is formally presented as Algorithm 3.

Example 2:
$k = 2$:
$u = (u_0, u_1, u_2) = (0,(0,0,0),(0,0,0))$;
$v = (v_0, v_1, v_2) = (1,(1,2,2),(0,2,2))$;
$f_1 = (0,(0,0,0),(0,0,1))$;

Algorithm 3: RDN_ft_routing($RDN^k(B), u, v, F$)
Input: A set of faulty nodes F with $|F| < d_0 + k$ and non-faulty nodes u and v in $RDN^k(B)$
Output: A fault-free path connecting nodes u and v
begin
Case 1: $u_0 = v_0$ and $u_1 = v_1$
if $|RDN_u^{k-1}(B) \cap F| < d + k - 1$
then RDN_ft_routing($RDN_u^{k-1}(B), u_2, v_2$)
else find path $u \rightarrow u^{(d_0+k)} \rightarrow u^{(d_0+k,j)} \rightarrow u^{(d_0+k,j,d_0+k)} = w$, where $j \neq d_0 + k$;
RDN_routing($RDN^k(B), w, v^{(d_0+k)}$);
Case 2: $u_0 \neq v_0$
if $|RDN_u^{k-1}(B) \cap F| = |RDN_v^{k-1}(B) \cap F| = 0$
then RDN_routing($RDN^k(B), u, v$)
else find a fault-free path $P_p(u)$ among the $d_0 + k$ disjoint paths $P_i(u) = u \rightarrow u^{(i)} \rightarrow u^{(i,d_0+k)} = u^i, 1 \leq i \leq d_0 + k - 1$,
or $P_{d_0+k}(u) = u \rightarrow u^{(d_0+k)} = u^{d_0+k}$, of length at most 2 such that $|RDN_{u_p}^{k-1}(B) \cap F| = 0$;
/* Since $|F| < d_0 + k$, $P_p(u)$ does exist. If multiple $P_p(u)$ exist, we choose arbitrarily one of them. */
if $u^p \in RDN_v^{k-1}(B)$ **then**
if $|RDN_v^{k-1}(B) \cap F| < d_0 + k - 1$
then RDN_ft_routing($RDN_v^{k-1}(B), u_2^p, v_2$); exit
else /*$|RDN_v^{k-1}(B) \cap F| = d_0 + k - 1$. */
find path $v \rightarrow v^{(d+k)} \rightarrow v^{(d_0+k,j)} \rightarrow v^{(d_0+k,j,d_0+k)} = w$, where $j \neq d_0 + k$;
RDN_routing($RDN^k(B), w, v$); exit;
find a fault-free path $P_q(v)$ among the $d_0 + k$ disjoint paths $P_i(v) = v \rightarrow v^{(i)} \rightarrow v^{(i,d_0+k)} = v^i, 1 \leq i \leq d_0 + k - 1$,
or $P_{d_0+k}(v) = v \rightarrow v^{(d_0+k)} = v^{d_0+k}$, of length at most 2 such that $|RDN_{v^q}^{k-1}(B) \cap F| = 0$;
if $v^q \in RDN_u^{k-1}(B)$ **then**
if $|RDN_u^{k-1}(B) \cap F| < d_0 + k - 1$
then RDN_ft_routing($RDN_u^{k-1}(B), v_2^q, u_2$); exit
else /*$|RDN_u^{k-1}(B) \cap F| = d_0 + k - 1$. */
find path $u \rightarrow u^{(d_0+k)} \rightarrow u^{(d_0+k,j)} \rightarrow u^{(d_0+k,j,d_0+k)} = w$, where $j \neq d_0 + k$;
RDN_routing($RDN^k(B), w, u$); exit;
RDN_routing($RDN^k(B), u^p, v^q$);
Case 3: $u_0 = v_0$
if $|RDN_u^{k-1}(B) \cap F| = 0$ or $|RDN_v^{k-1}(B) \cap F| = 0$
/* Without loss of generality, we assume $|RDN_v^{k-1}(B) \cap F| = 0$. */
then find a fault-free path $P_p(u)$ among the $d_0 + k$ disjoint paths $P_i(u) = u \rightarrow u^{(i)} \rightarrow u^{(i,d_0+k)} = u^i, 1 \leq i \leq d_0 + k - 1$,
or $P_{d_0+k}(u) = u \rightarrow u^{(d_0+k)} = u^{d_0+k}$, of length at most 2 such that $|RDN_{u^p}^{k-1}(B) \cap F| = 0$;
RDN_routing($RDN^k(B), u^p, v$)
else find a fault-free path $P_p(u)$ among the $d_0 + k$ disjoint paths $P_i(u) = u \rightarrow u^{(i)} \rightarrow u^{(i,d_0+k)} = u^i, 1 \leq i \leq d_0 + k - 1$,
or $P_{d_0+k}(u) = u \rightarrow u^{(d_0+k)} = u^{d_0+k}$, of length at most 2 such that $|RDN_{u_p}^{k-1}(B) \cap F| = 0$;
find a fault-free path $P_q(v)$ among the $d_0 + k$ disjoint paths $P_i(v) = v \rightarrow v^{(i)} \rightarrow v^{(i,d_0+k)} = v^i, 1 \leq i \leq d_0 + k - 1$,
or $P_{d_0+k}(v) = v \rightarrow v^{(d_0+k)} = v^{d_0+k}$, of length at most 2 such that $|RDN_{v^q}^{k-1}(B) \cap F| = 0$;
if $RDN_{u^p}^{k-1}(B) = RDN_{v^q}^{k-1}(B)$
then RDN_routing($RDN_{u^p}^{k-1}(B), u^p, v^q$)
else find a fault-free path $P_r(u^p)$ among the $d_0 + k$ disjoint paths $P_i(u^p) = u^p \rightarrow (u^p)^{(i)} \rightarrow (u^p)^{(i,d_0+k)} = (u^p)^i$,
$1 \leq i \leq d_0 + k - 1$, or $P_{d+k}(u^p) = u^p \rightarrow (u^p)^{(d_0+k)} = (u^p)^{d_0+k}$, of length at most 2 such that
$|RDN_{(u^p)^r}^{k-1}(B) \cap F| = 0$;
RDN_routing($RDN^k(B), (u^p)^r, v^q$);
end

$f_2 = (1, (0, 0, 0), (0, 0, 0)$;
$f_3 = (1, (1, 0, 0), (1, 2, 0)$.
$u_0 \neq v_0$ (Case 2):

Among the four disjoint paths of length at most 2 from u, only the path $u \rightarrow (0, (0, 0, 0), (0, 0, 2)) \rightarrow (1, (0, 0, 2), (0, 0, 0) = u^2$ is fault-free and $|RDN_{u^2}(3, 1) \cap F| = 0$. Notice that since $|RDN_{u^3}^1(B(3)) \cap F| = 0$, $u^3 = (1, (1, 0, 0), (0, 0, 0)$ cannot be chosen. Therefore, we have $p = 2$ and $u^p = (1, (1, 0, 0), (0, 0, 0))$. For the fault-free path of length at most 2 from v, we choose $v^q = (0, (0, 2, 2), (1, 2, 2)$ though any of the four paths is okey. Then, by calling RDN_routing($RDN^2(B(3)), u^p.v^q$), we find a fault-free path from u to v as shown in Fig. 6.

Theorem 3: Assume that, for any two nodes in the base-network $B(n_0)$, a path connecting the two nodes can be found in $O(f(n_0))$ time. Then for any two non-faulty nodes in $RDN^k(B), k > 0$, with at most $d_0 + k - 1$ faulty nodes, a fault-free path connecting the two nodes can be found in $O(d_0+k+2^k f(n_0))$ time. The maximum length of the paths is at most $2^k * D_0 + 2^{k+1} - 2$, where D_0 is the diameter of

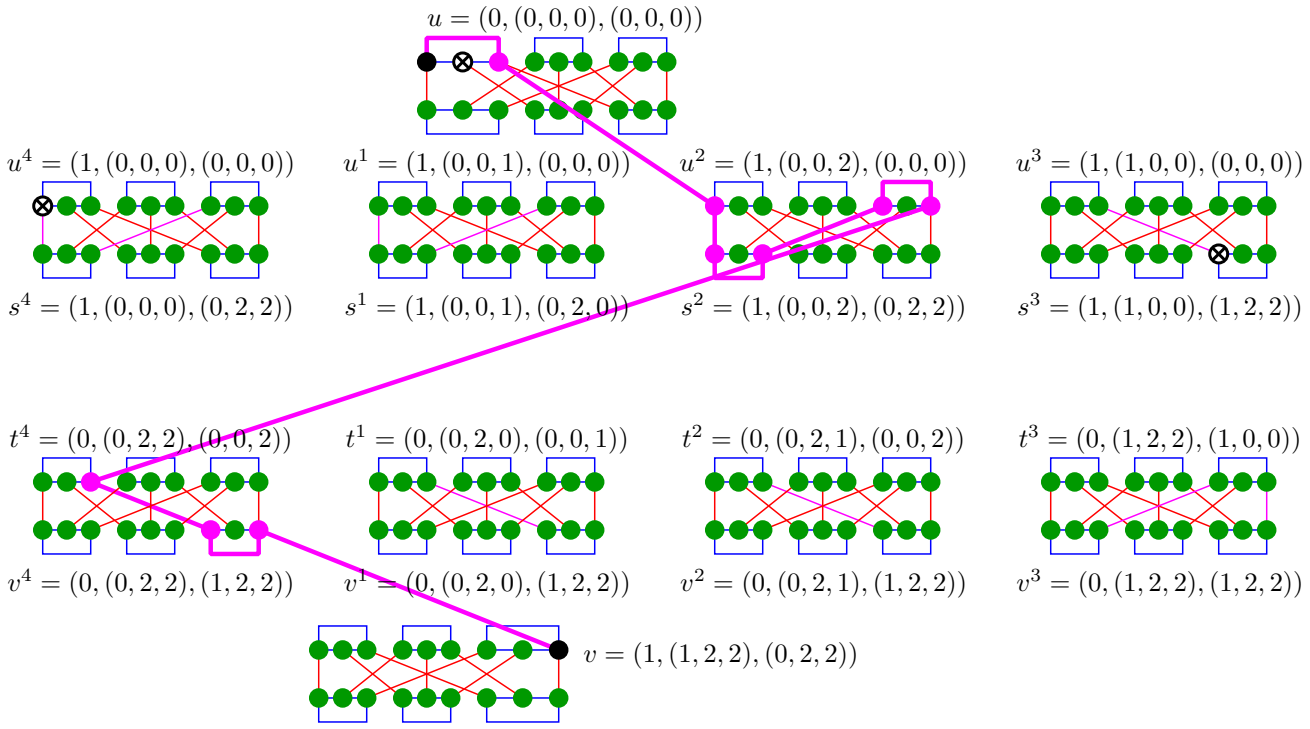

Figure 6. Fault-tolerant routing in $RDN^2(B(3))$

the base network.

Proof: The theorem follows directly from algorithm 3. Let the time complexity for finding a fault-free path in $RDN^k(B)$ be $T(k)$. From Algorithm 3 and Lemma 1, we have $T(k) = O(d_0 + k) + O(2^k f(n_0)) = O(d_0 + k + 2^k f(n_0))$. The worst-case for the length of the longest path occurs in Case 3 of the algorithm. The length of the longest path equals to $4 + 2 * D_{k-1} + 2$ for $k > 0$. Therefore, the maximum length is $2^k * D_0 + 2^{k+1} - 2$, where D_0 is the diameter for the base network. ❑

IV. Algorithm for Fault-tolerant routing on RDN with Arbitrary Number of Faulty Nodes

In this section, we propose an efficient practical algorithm for fault-tolerant routing in a RDN with arbitrary number of faulty nodes. Given a set of faulty nodes F and two nonfaulty nodes u and v in $RDN^k(B)$, $k > 0$. If u and v are in the same cluster of level k then it is done by a recursive call on the cluster where u and v reside. In case of $u_0 \neq v_0$ (Case 2), if the gateways $s = (u_0, u_1, v_1)$ and $t = (v_0, v_1, u_1)$ are nonfaulty then it is done by two recursive calls on the clusters where u and v reside. Otherwise, find fault-free paths of length at most 2 from u and v to u^p and v^q and then recursive call with u and v replaced by u^p and v^q, respectively. In case of $u_0 = v_0$ and $u_1 \neq v_1$, we route u or v to the cluster of distinct type by a fault-free path of length at most 2 and then treat it as Case 2. The algorithm will fail to find a fault-free path if we cannot find a fault-free path of length at most 2 to route u or v to a node in other cluster of distinct type. The proposed algorithm is formally presented as Algorithm 4.

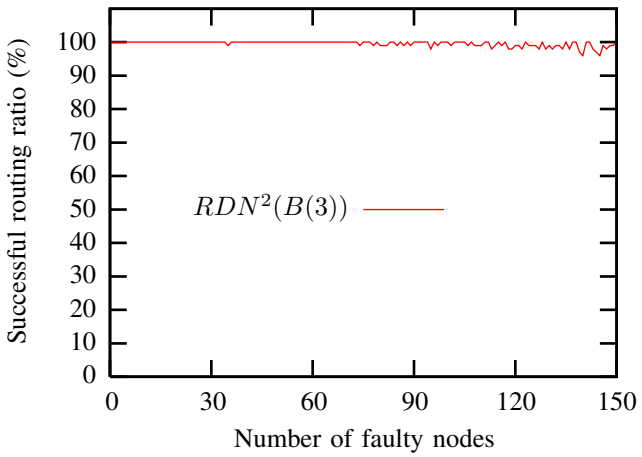

Figure 7. Successful routing rate of fault-tolerant routing in $RDN^2(B(3))$

We have simulated the performance of the proposed algorithm on an $RDN^2(B(3))$. The simulation result is shown in Fig. 7. In $RDN^2(B(3))$, there are 648 nodes in total. The number of faulty nodes is assigned from 0 to 150, stepped by 1. The faulty nodes are randomly distributed and 100 routings are simulated for a fixed number of faulty nodes. We calculate the successful routing rate by dividing the number of successful routings by 100.

Algorithm 4: RDN_unlimited_ftr($RDN^k(B), u, v, F$)
Input: A set of faulty nodes F and non-faulty nodes u and v in $RDN^k(B)$
Output: A fault-free path connecting nodes u and v
begin /* Assume that RDN_unlimited_ftr(B, u, v) is available. */
Case 1: $u_0 = v_0$ and $u_1 = v_1$
RDN_unlimited_ftr($RDN_u^{k-1}(B), u_2, v_2, F \cap RDN_u^{k-1}(B)$);
Case 2: $u_0 \neq v_0$
if $s = (u_0, u_1, v_1)$ and $t = (v_0, v_1, u_1)$ are nonfaulty
then RDN_unlimited_ftr($RDN_u^{k-1}(B), u_2, v_1$);
RDN_unlimited_ftr($RDN_v^{k-1}(B), v_2, u_1$)
else find a fault-free path $P_p(u)$ among the $d_0 + k$ disjoint paths $P_i(u) = u \rightarrow u^{(i)} \rightarrow u^{(i,d_0+k)} = u^i, 1 \leq i \leq d_0 + k - 1$, or $P_{d_0+k}(u) = u \rightarrow u^{(d_0+k)} = u^{d_0+k}$, of length at most 2 such that $|RDN_{u^p}^{k-1}(B) \cap F|$ is a minimum;
find a fault-free path $P_q(v)$ among the $d + k$ disjoint paths $P_i(v) = v \rightarrow v^{(i)} \rightarrow v^{(i,d_0+k)} = v^i, 1 \leq i \leq d_0 + k - 1$, or $P_{d_0+k}(v) = v \rightarrow v^{(d_0+k)} = v^{d_0+k}$, of length at most 2 such that $|RDN_{v^q}^{k-1}(B) \cap F|$ is a minimum;
RDN_unlimited_ftr($RDN^k(B), u^p, v^q$);
Case 3: $u_0 = v_0$ and $u_1 \neq v1$
if $|RDN_u^{k-1}(B) \cap F| < |RDN_v^{k-1}(B) \cap F|$
then find a fault-free path $P_p(u)$ among the $d_0 + k$ disjoint paths $P_i(u) = u \rightarrow u^{(i)} \rightarrow u^{(i,d_0+k)} = u^i, 1 \leq i \leq d_0 + k - 1$, or $P_{d+k}(u) = u \rightarrow u^{(d_0+k)} = u^{d_0+k}$, of length at most 2 such that $|RDN_{u_p}^{k-1}(B) \cap F|$ is a minimum;
RDN_unlimited_ftr($RDN^k(B), u^p, v$)
else find a fault-free path $P_q(v)$ among the $d_0 + k$ disjoint paths $P_i(v) = v \rightarrow v^{(i)} \rightarrow v^{(i,d_0+k)} = v^i, 1 \leq i \leq d_0 + k - 1$, or $P_{d_0+k}(v) = v \rightarrow v^{(d_0+k)} = v^{d_0+k}$, of length at most 2 such that $|RDN_{v_q}^{k-1}(B) \cap F|$ is a minimum;
RDN_unlimited_ftr($RDN^k(B), u, v^q$);
end

The simulation result is shown in Fig. 7. From the figure, we can see that the algorithm achieves high probability of the successful routings with a rather large number of faulty node. When the number of faulty nodes is less than 70, the probabilities of the successful routings are almost 100%. When the number of faulty nodes increases up to 150 that is about 1/4 of the total number of nodes, the probabilities of the successful routings are still larger than 97%.

V. Concluding Remarks

Recursive dual-net is a new interconnection network for MPP. It has great potential as a candidate for the network of the parallel computers of next generations. In this paper, we proposed efficient algorithms for disjoint-paths routing and fault-tolerant routing on the recursive dual-net. There are many issues concerning the recursive dual-net that are worth further research.

References

[1] S. G. Aki, *Parallel Computation: Models and Methods*. Prentice-Hall, 1997.

[2] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, 1992.

[3] A. Varma and C. S. Raghavendra, *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*. IEEE Computer Society Press, 1994.

[4] K. Ghose and K. R. Desai, "Hierarchical cubic networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 4, pp. 427–435, April 1995.

[5] Y. Li and S. Peng, "Dual-cubes: a new interconnection network for high-performance computer clusters," in *Proceedings of the 2000 International Computer Symposium, Workshop on Computer Architecture*, ChiaYi, Taiwan, December 2000, pp. 51–57.

[6] Y. Li, S. Peng, and W. Chu, "Efficient collective communications in dual-cube," *The Journal of Supercomputing*, vol. 28, no. 1, pp. 71–90, April 2004.

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

[8] Y. Saad and M. H. Schultz, "Topological properties of hypercubes," *IEEE Transactions on Computers*, vol. 37, no. 7, pp. 867–872, July 1988.

[9] G. H. Chen and D. R. Duh, "Topological properties, communication, and computation on wk-recursive networks," *Networks*, vol. 24, no. 6, pp. 303–317, 1994.

[10] G. Vicchia and C. Sanges, "A recursively scalable network vlsi implementation," *Future Generation Computer Systems*, vol. 4, no. 3, pp. 235–243, 1988.

[11] TOP500, *Supercomputer Sites*. http://top500.org/, Jun. 2009.

[12] 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 (ICA3PP'09)*. Taipei, Taiwan: Springer, Lecture Notes in Computer Science (LNCS), to be published, June 2009.

[13] 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, http://www.research.ibm.com/journal/rd/492/tocpdf.html*, vol. 49, no. 2/3, pp. 265–276, 2005.