The IASTED International Conference on Networks, Parallel and Distributed Processing, and Applications, Tsukuba, Japan, October 2002, pp.1–6

HAMILTONIAN CYCLE EMBEDDING FOR FAULT TOLERANCE IN DUAL-CUBE

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

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

ABSTRACT
The hypercube has been widely used as the interconnection network (IN) in parallel computers. However, the major drawback of the hypercube is the increase in the number of communication links for each node with the increase in the total number of nodes in the system. A dual-cube DC(m) has $m+1$ links per node where m is the degree of a cluster (m-cube), one more link is used for connecting to a node in another cluster. The dual-cube mitigates the problem of increasing number of links in the large-scale hypercube network while keeps most of the topological properties of the hypercube network. Embedding a linear array or a ring into interconnection networks even when faulty-links exit is an important issue for the design of INs. In this paper, we show that a hamiltonian cycle exists in a DC(m) with up to $m-1$ faulty links. This is optimal because the degree of a DC(m) is $m+1$. We also give efficient algorithms for constructing hamiltonian cycles in DC(m).

KEY WORDS
Interconnection networks, hypercube, hamiltonian cycle, Gray code, fault-tolerant embedding

1. Introduction

The binary hypercube has been widely used as the interconnection network in a wide variety of parallel systems such as Intel iPSC, the nCUBE [4], the Connection Machine CM-2 [12], and SGI Origin 2000 [9]. A hypercube network of dimension n, or n-cube, contains up to 2^n nodes and has n edges per node. If unique n-bit binary addresses are assigned to the nodes of the hypercube, then an edge connects two nodes if and only if their binary addresses differ in a single bit. Because of its elegant topological properties and the ability to emulate a wide variety of other frequently used networks, the hypercube has been one of the most popular interconnection networks for parallel computer/communication systems.

However, the conventional hypercube has a major shortage, that is, the number of edges per node in a system increases logarithmically as the total number of nodes in the system increases. Since the number of links is limited to eight per node with current IC technology, the total number of nodes in a hypercube parallel computer is restricted to several hundreds. Therefore, it is interesting to develop an interconnection network which keeps most of topological properties of the hypercube, and has more nodes in the system than the hypercube with the same number of edges per node.

Several variations of the hypercube have been proposed in the literature. Some variations focused on the reduction of diameter of the hypercube, such as folded hypercube [1] and crossed cube [2]; some focused on the reduction of the number of edges of the hypercube, such as cube-connected cycles [8] and reduced hypercube [14]; and some focused on the both, like hierarchical cubic network [3]. Generally, the variations of the hypercube that reduce the diameter, e.g. crossed cube and hierarchical cubic network, will not satisfy the following key property in the hypercube: each node can be represented by a unique binary number such that two nodes are connected by an edge only if the two binary numbers differ in one bit. This key property is at the core of many algorithmic designs for efficient routing and communication.

A new interconnection network for large parallel systems called *dual-cube* (DC) has been introduced recently [6] [7]. The dual-cube shares the desired properties of the hypercube (e.g., the key property of the hypercube mentioned above), and increases tremendously the total number of nodes in the system compared with the hypercube of the same node degree. The size of the dual-cube can be as large as thirty thousands with up to eight links per node. It is practically important to refine the hypercube networks such that the size of the network can be increased while the number of the links per node is limited by the technology.

A *hamiltonian cycle* of an undirected graph G is a simple cycle that contains every node in G exactly once. A *hamiltonian path* in a graph is a simple path that visits every node exactly once. A hamiltonian path can be obtained from a hamiltonian cycle by removing any one link from that cycle. A graph that contains a hamiltonian cycle is said to be *hamiltonian*. G is *k-link hamiltonian* if it remains hamiltonian after removing any k links. It is clear that if graph G is k-connected then G can be at most $(k-2)$-link hamiltonian.

Constructing hamiltonian cycles is important for linear array or ring embedding. Previous results about k-link hamiltonian networks are as follows. The n-cube is $(n-2)$-link hamiltonian [5]. The n-dimensional folded hypercube is $(n-1)$-link hamiltonian [13]. The n-dimensional star graph is $(n-3)$-link hamiltonian [11]. A k-ary undirected de Bruijn graph is $(k-1)$-link hamil-

tonian [10]. In this paper, we show that the $(m+1)$-connected DC(m) is $(m-1)$-link hamiltonian. Therefore, a DC(m) can tolerate a maximal number of link faults while embedding a longest fault-free ring.

The rest of this paper is organized as follows. Section 2 describes the dual-cube architecture. Section 3 shows how to construct a hamiltonian cycle in a DC(m). Section 4 extends the result to a faulty DC(m) with up to $m-1$ faulty links. Section 5 concludes the paper and presents some future research directions.

2. Preliminaries

A dual-cube uses hypercubes as basic components. Each hypercube component is referred to as a *cluster*. Assume that the number of nodes in a cluster is 2^m. In a dual-cube, there are two *classes* with each class consisting of 2^m clusters. The total number of nodes is $2^m \times 2^m \times 2$, or 2^{2m+1}. Each node in a dual-cube has $m+1$ links: m links are used within cluster to construct an m-cube and a single link is used to connect a node in a cluster of the other class. There is no link between the clusters of the same class. If two nodes are in one cluster, or in two clusters of distinct classes, the distance between the two nodes is equal to its *Hamming distance* (the number of bits where the addresses of the two nodes have different values). Otherwise, it is equal to the Hamming distance plus two: one for entering a cluster of the other class and one for leaving.

An $(m+1)$-connected dual-cube DC(m) is an undirected graph on the node set $\{0,1\}^{2m+1}$ and there is an edge between two nodes $u = (u_{2m} \dots u_0)$ and $v = (v_{2m} \dots v_0)$ if and only if the following conditions are satisfied:

1. u and v differ exactly in one bit position i,
2. if $0 \le i \le m-1$ then $u_{2m} = v_{2m} = 0$ and
3. if $m \le i \le 2m-1$ then $u_{2m} = v_{2m} = 1$.

Intuitively, the set of nodes u of form $(0u_{2m-1} \dots u_m * \dots *)$, where $*$ means "don't care", constitutes an m-dimensional hypercube. We call these hypercubes clusters of class 0. Similarly, the set of nodes u of form $(1 * \dots * u_{m-1} \dots u_0)$ constitutes an m-dimensional hypercube and we call them clusters of class 1. The edge connecting two nodes in two clusters of distinct classes is called *cross-edge*. In the other word, $e = (u : v)$ is a cross-edge if and only if u and v differ in the leftmost bit.

Each node in a DC(m) is identified by a unique $(2m+1)$-bit number, an *id*. Each *id* contains three parts: *class_id*, *cluster_id* and *node_id*. In the following discussion, we use *id* = (*class_id*, *cluster_id*, *node_id*) to denote the node address where *class_id* is a 1-bit number, *cluster_id* and *node_id* are m-bit numbers. The bit-position of *cluster_id* and *node_id* depends on the value of *class_id*. If *class_id* $= 0$ (*class_id* $= 1$), then *node_id* (*cluster_id*) is the rightmost m bits and *cluster_id* (*node_id*) is the next (to the left) m bits. The cluster containing node u is denoted as C_u. For any two nodes u and v in a DC(m), $C_u = C_v$ if and only if u and v are in the same cluster.

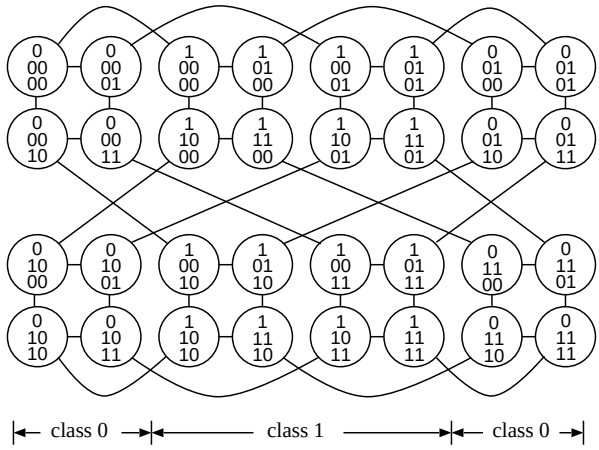

Figure 1. A dual-cube DC(2)

Figure 1 depicts a DC(2) network. In each node, *class_id* is shown at the top position. For the nodes of class 0 (class 1), *node_id* (*cluster_id*) is shown at the bottom and *cluster_id* (*node_id*) is shown at the middle. Figure 2 shows a DC(3). Notice that only those cross-edges connecting to cluster 0 of class 1 are shown, the other cross-edges are omitted for simplicity.

The dual-cube has a binary presentation of nodes, similar to a hypercube, in which two nodes are connected by an edge only if their addresses differ in one bit. This feature is the key for designing efficient routing and communication algorithms in dual-cube. Another important feature of a dual-cube is that, within the given bound to the number of links per node, say $m+1$, the network can have up to 2^{2m+1} nodes, more than the hypercube or the hierarchical cubic network can have.

The DC(m) topological properties are given in [6] and the collective communication schemes in DC(m) can be found in [7].

3. Hamiltonian Cycle in Dual-Cube

The key for constructing a hamiltonian cycle in a DC(m) is to construct a *virtual hamiltonian cycle* that connects all 2^{m+1} clusters in DC(m). The virtual hamiltonian cycle in a DC(m) contains equal numbers of cube-edges and cross-edges; the cube-edges and the cross-edges are interleaved. To construct a fault-free hamiltonian cycle in a DC(m) with up to $m-1$ faulty links, we need to put some constraints on the cube-edges in the virtual hamiltonian cycle since a hamiltonian path inside a cluster with faulty links might have fixed end nodes.

We use $0^{\{i\}}$ to denote a bit pattern $0 \dots 0$ of i bits. The hamiltonian cycle in an n-cube can be constructed by the *binary reflected Gray code*. A *Gray code* for binary numbers is a listing of all n-bit numbers so that successive numbers, including the first and last, differ in exactly one bit position. The best known example

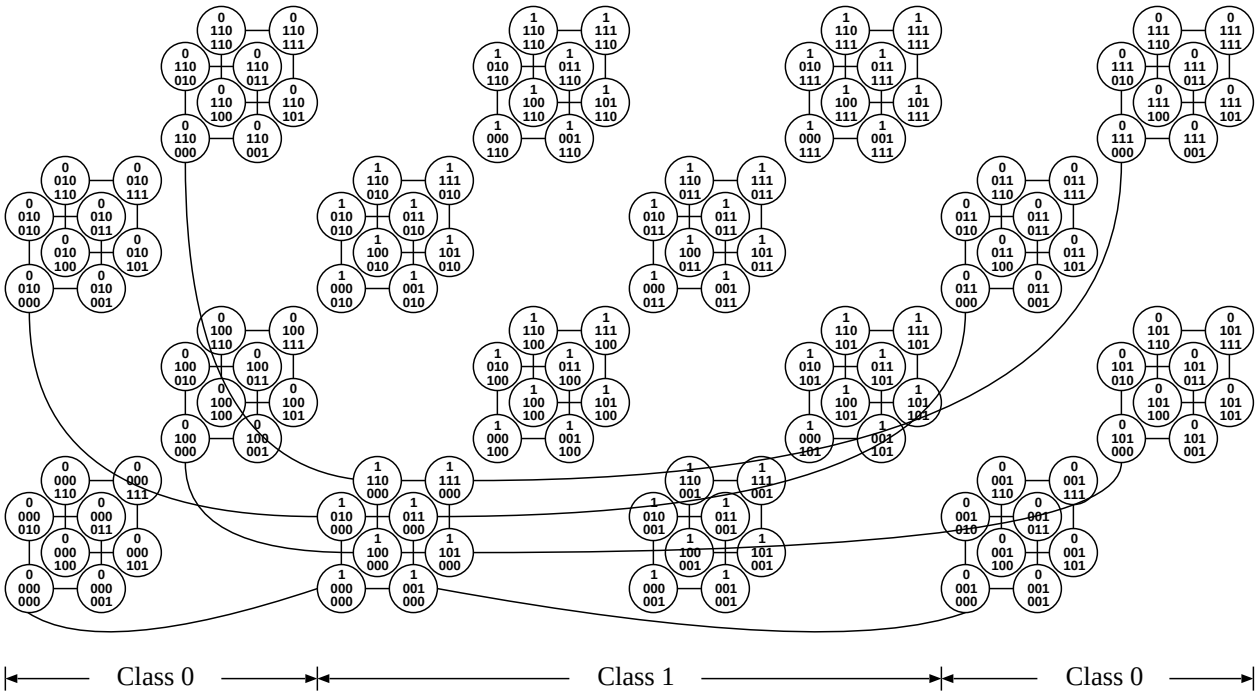

Figure 2. A dual-cube DC(3)

of the Gray codes is the *binary reflected Gray code* which can be described as follows. If $P(n)$ denotes the listing for n-bit numbers, then $P(1) = (0,1)$. For n greater than 1, $P(n)$ is formed by taking the list for $P(n-1)$ with each number prefixed by 0 then following that list by the reverse of $P(n-1)$ with each number prefixed by 1. For example, $P(2) = (00,01,11,10)$, $P(3) = (000,001,011,010,110,111,101,100)$, and so on. Since the first and last numbers of $P(n)$ also differ in one bit position, the code is in fact a cycle. In an n-cube, there is a link connecting two nodes if their numbers differ in one bit position: connecting the adjacent nodes, also the first and last nodes, in the binary reflected Gray code list with links, a hamiltonian cycle is formed.

Let $D(n)$ denote the listing for the dimensions which changed in the number sequence in the reflected Gray code list. Then, $D(1) = 0$. For n greater than 1, $D(n)$ is formed by taking the list $D(n-1)$ two times and inserting a number $n-1$ into between the two lists. For example, $D(2) = (0,1,0)$, $D(3) = (0,1,0,2,0,1,0)$, $D(4) = (0,1,0,2,0,1,0,3,0,1,0,2,0,1,0)$, and so on. That is, $D(n)$ can be constructed recursively as follows: $D(n) = (D(n-1), n-1, D(n-1))$ if $n > 1$, and $D(1) = (0)$. Note that reversing the node numbers performed in the generation of the reflected Gray code does not affect the dimensions which change in the sequence of the reflected Gray code: we just copy $D(n-1)$ to the second half part of $D(n)$. We call $D(n)$ a *reflected dimension list*.

In the what follows, we use $(u \rightarrow v)$ to denote a path or a cycle, and $(u : v)$ to denote a link connecting nodes u and v. Also, a format like $(00 : 01 : 11 : 10)$ denotes a path or a cycle. The following algorithm is for generating a hamiltonian cycle P in an n-cube with the reflected dimension list. The $\oplus$ does bit-wise exclusive OR operation.

Algorithm 1 (cubeHC(n))
begin /* build a hamiltonian cycle P in an n-cube */
  $D(n) =$ DL(n); /* $D(n)$: reflected dimension list */
  $w = 0$; /* starting from node 0 */
  $P = w$; /* P is the hamiltonian cycle */
  for each dimension number i in $D(n)$ **do**
    $w = w \oplus 2^i$; /* find the next node */
    $P = P : w$; /* add the node into P */
  endfor
end
Procedure DL(n)
begin /* build a reflected dimension list for an n-cube */
  if ($n == 1$) **return** (0);
  else return (DL($n-1$), $n-1$, DL($n-1$));
end

Note that the reflected Gray code or reflected dimension list is just one solution of the Gray codes. By renumbering the node numbers (exchanging bit positions of the all node numbers), we can have $n!$ different Gray code sequences. Furthermore, since there are 2^n links in the cycle, breaking a different link will get a different path: there are $2^n n!$ hamiltonian paths with different patterns in an n-cube.

Next, we add a condition to let a hamiltonian cycle contain a given link. This is needed for constructing a fault-free hamiltonian cycle in a dual-cube with faulty links.

Lemma 1. *Given any link $e = (u : v)$ in an n-cube where u and v are two distinct nodes and $d(u,v) = 1$, there is a hamiltonian cycle going through e.*

Proof: The lemma can be proved by renumbering every node in the cube with a mapping function $f(x)$ so that $u' = f(u) = 0^{\{n-1\}}0$ and $v' = f(v) = 0^{\{n-1\}}1$. Then a

hamiltonian cycle is built by Algorithm 1 with the new numbers. Finally, the hamiltonian cycle denoted with the original node numbers is obtained by applying $f^{-1}(x)$ to every node number in the built cycle with the new numbers, where $f^{-1}(x)$ is the reverse of function $f(x)$: $u = f^{-1}(u')$ and $v = f^{-1}(v')$. One possible $f(x)$ does exclusive OR operation with u on every node number so that node u will have a new number $0^{\{n-1\}}0$, and then exchanges bit positions so that the node v will have a new number $0^{\{n-1\}}1$. □

By removing $e = (u : v)$ from the hamiltonian cycle constructed in Lemma 1, we get a hamiltonian path from node u to node v, $(u \rightarrow v)$. We name the procedure that generates such a path as cubeHP(m,u,v). This procedure will be used in constructing a hamiltonian cycle in a DC(m).

A hamiltonian cycle in a DC(m) can be constructed as follows. First, we can build a *virtual* hamiltonian cycle, $V(m)$, which connects all the clusters with only two neighboring nodes, u and v for instance, from each cluster (Figure 3). It is called *virtual* since the cube-edge $e = (u : v)$ in the cycle will be replaced with a hamiltonian path $(u \rightarrow v)$ in that cluster to form a "real" hamiltonian cycle in DC(m).

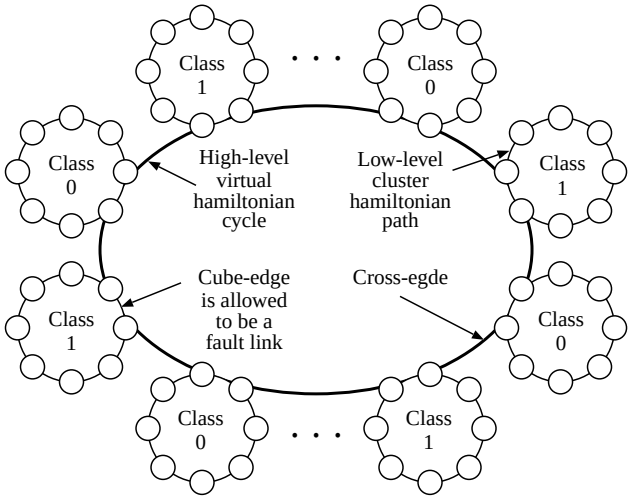

Figure 3. A virtual hamiltonian cycle in a dual-cube

The construction of the virtual hamiltonian cycle can be done by using an *extended double-dimension list*, or $EDD(m)$, defined as follows. Let the reflected double-dimension list be $DD(m) = (DD(m-1), m-1, m-1, DD(m-1))$ if $m > 1$, and $DD(1) = (0,0)$. Then the extended double-dimensional list $EDD(m) = (DD(m), m-1, m-1)$. Since there are two classes in a dual-cube, $EDD(m)$ doubles each dimension number in an extended list which consists of $D(m)$ plus the highest dimension $m-1$. For example, $EDD(2) = (0,0,1,1,0,0,1,1)$, $EDD(3) = (0,0,1,1,0,0,2,2,0,0,1,1,0,0,2,2)$, and so on. Then the virtual hamiltonian cycle can be constructed with $EDD(m)$. For example, $V(2) =$ (00000, 00001, 10001, 10101, 00101, 00111, 10111, 11111, 01111, 01110, 11110, 11010, 01010, 01000, 11000, 10000). Second, in each cluster we replace the edge $e = (u : v)$ with a hamiltonian path $(u \rightarrow v)$ to connect all the nodes in the cluster to form a hamiltonian cycle in DC(m).

This virtual hamiltonian cycle could be considered as a high-level cycle which connects all the clusters. Note that only two neighboring nodes in each cluster are contained in the virtual hamiltonian cycle, and cube-edges and cross-edges are interleaved. Because there are two classes in a DC(m) and each class has 2^m clusters, the virtual hamiltonian cycle contains $2^m \times 2 \times 2$, or $2^m \times 4$ nodes. If we group 4 nodes, whose right m bits of the addresses are the same (e.g., 00001, 10001, 10101, 00101), into a *big* node, the virtual hamiltonian cycle contains 2^m big nodes. Therefore, the algorithm to construct the virtual hamiltonian cycle is similar to that of the hypercube. The difference is that once the next node is chosen based on the reflected dimension list in a cluster of a class, we need to go through the cross-edge to a cluster of the other class and do the same work in that cluster. This is the reason why $DD(m)$ doubles each dimension number of $D(m)$. Argorithm 2 shows how to build a hamiltonian cycle in a DC(m) and hence we have

Theorem 1. *There is a hamiltonian cycle in a dual-cube.*

Algorithm 2 (dualcubeHC(m))
begin /* build a hamiltonian cycle P in DC(m) */
  $DD(m) =$ DDL(m);
  $EDD(m) = (DD(m), m-1, m-1)$;
  $u = 0$;
  for each dimension number i in $EDD(m)$ **do**
    if (u is of class 0) $v = u \oplus 2^i$;
    else $v = u \oplus 2^{m+i}$;
    $P' =$ cubeHP(m,u,v);
    $P = P : P'$;
    $u = v \oplus 2^{2m}$;
  endfor
end
Procedure DDL(m)
begin /* build an double-dimension list for a DC(m) */
  if ($m == 1$) **return** $(0,0)$;
  else return (DDL($m-1$), $m-1$, $m-1$, DDL($m-1$));
end

Lemma 2. *Given any cube-edge $e = (u : v)$ in a DC(m), there is a hamiltonian cycle going through e.*

Proof: Similar to the proof of Lemma 1. □

Since there are $m2^{m-1}$ links in a cluster, by taking each of the links as edge $(u : v)$, we have $m2^{m-1}$ different virtual hamiltonian cycles. These cycles are different but not disjointed.

Theorem 2. *There are 2^{m-1} disjoint virtual hamiltonian cycles in a DC(m).*

Proof: We use induction to prove the theorem. For $m = 2$ (see Figure 4), two links in a cluster, for example $e_0 = (00000 : 00001)$ and $e_1 = (00011 : 00010)$ in the cluster 0 of class 0, connect four distinct nodes in the cluster. Therefore, there are $2^{2-1} = 2$ disjoint virtual hamiltonian cycles that contain e_0 and e_1, respectively.

These two cycles are constructed by $EDD(2)$ based on the reflected dimension list $D(2) = (0,1,0)$ with the starting nodes 00000 and 00011, respectively.

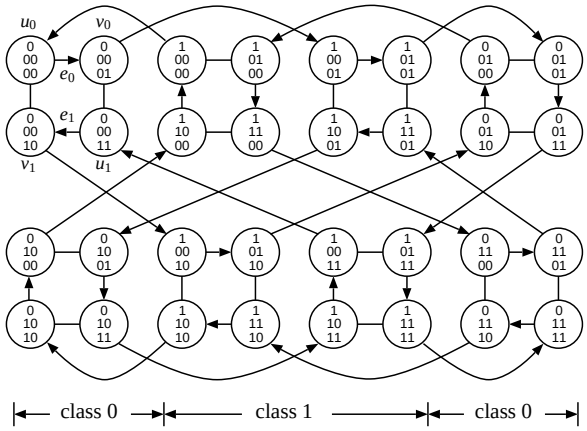

Figure 4. Disjoint virtual hamiltonian cycles in a DC(2)

Generally, there are $2^m/2$ such links in an m-dimensional cluster (m-cube): each link takes two nodes from the list of the reflected Gray codes. For $m > 2$, the $2^m/2$ virtual hamiltonian cycles that contain e_i, $0 \le i \le 2^{m-1}-1$, respectively, can be built based on the reflected dimension list $D(m)$. Because $D(m) = (D(m-1), m-1, D(m-1))$, by our induction hypothesis, the first half of all the cycles are disjointed. Then, all the paths that go through the $(m-1)$th dimension will still be disjoint. Similarly, the second half of all the cycles are also disjointed. Therefore, all 2^{m-1} cycles are disjoint. □

4. Fault-Free Hamiltonian Cycle in Dual-Cube

In this section, we consider the problem of finding fault-free hamiltonian cycle in dual-cube with faulty links.

Lemma 3. *Given any link $e = (u : v)$ in an n-cube with $n-2$ faulty links, there is a fault-free hamiltonian path $(u \to v)$.*

Proof: We use induction on n to prove the lemma. For $n = 2$, the lemma is true by applying Lemma 1 because the number of faulty links is 0.

Suppose the lemma is true for $n-1$. In an n-cube, there are n dimensions. Because the number of faulty links is $n-2$, there exists a dimension, say hth dimension, $0 \le h \le n-1$, such that the link e and all faulty links do not appear in this dimension. We cut the n-cube along the hth dimension into two $(n-1)$-cubes: *subcube0* and *subcube1*. Assume that e is in subcube0.

The proof of the lemma is divided into three cases. If all faulty links are in subcube1 and there is a faulty link $(u' : v')$ such that $d(u,u') = d(v,v') = 1$, then we should select h such that $(u : v)$ and $(u' : v')$ are in the same subcube, subcube0, as shown as in Figure 5. In this case, we can also ensure that there is no faulty link in the hth dimension because the faulty link $(u' : v')$ and the given link $(u : v)$ are in the same dimension. That is, the $(u : v)$'s corresponding edge $(u' : v')$ in subcube1 is not a fauly link. This condition is used in Case 2.

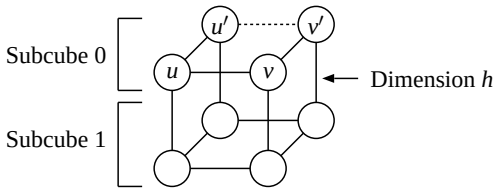

Figure 5. Subcube dividing

Case 1: All the $n-2$ faulty links are in subcube0. Let $(x : y)$ be a faulty link. By our induction hypothesis, there is hamiltonian path $(u \to v)$ in subcube0 that does not include any faulty link except $(x : y)$. If link $(x : y)$ appears in the path, then, the fault-free hamiltonian path $(u \to v)$ in the n-cube can be constructed as $(u \to x : x' \to y' : y \to v)$, where $(x' : y')$ is a link in subcube1 such that $x' \oplus x = y' \oplus y = 0^{\{n-1-h\}}10^{\{h\}}$, and $(x' \to y')$ is a hamiltonian path in subcube1 because there in no faulty link in subcube1. This is illustrated in Figure 6. If $(x : y)$ does not appear in the path, we can take any link other than e, (a,b) for instant, to construct the fault-free hamiltonian path $(u \to v)$ in the n-cube as $(u \to a : a' \to b' : b \to v)$ similarly.

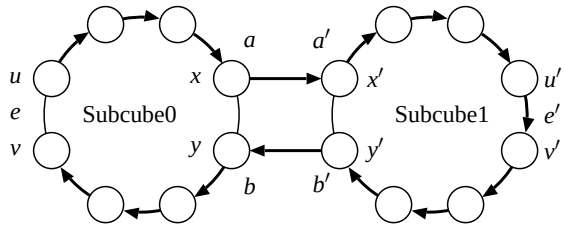

Figure 6. Fault-free hamiltonian path in hypercube

Case 2: All the $n-2$ faulty links are in subcube1. In subcube0, there is an hamiltonian path $(u \to v)$. Let $(x' : y')$ be a faulty link. Note that $(x' : y') \ne (u' : v')$. By our induction hypothesis, there is hamiltonian path $(u' \to v')$ in subcube1 that does not include any faulty link except $(x' : y')$. If link $(x' : y')$ appears in the hamiltonian path of subcube1, then, the fault-free hamiltonian path $(u \to v)$ in the n-cube can be constructed as $(u \to x : x' \to y' : y \to v)$. Similarly, if $(x' : y')$ does not appear in the path of subcube1, we can take any link other than $(u' : v')$, (a', b') for instance, to construct the fault-free hamiltonian path $(u \to v)$ in the n-cube as $(u \to a : a' \to b' : b \to v)$ similarly.

Case 3: There are at most $n-3$ faulty links in each subcube. By our induction hypothesis, there is a fault-free hamiltonian path $(u \to v)$ in subcube0. Choose any one link $(a : b)$ other than e from the path in subcube0. There is a corresponding link, $(a' : b')$, in subcube1. Also by our induction hypothesis, there is a fault-free hamiltonian path $(a' \to b')$ in subcube1. Then

$(u \to a : a' \to b' : b \to v)$ is a fault-free hamiltonian path $(u \to v)$ in the n-cube. □

Next, we show how to construct a fault-free hamiltonian cycle in a DC(m) which contains up to $m-1$ faulty links.

Theorem 3. *If a DC(m) contains $m-1$ faulty links, there is a hamiltonian cycle in the DC(m).*

Proof: We first prove that there is a high-level virtual hamiltonian cycle in a DC(m) with $m-1$ faulty links. We divide the proof into the following two cases.

Case 1: All the $m-1$ faulty links appear in a same cluster. Because an m-cube is $(m-2)$-link hamiltonian, there is a fault-free hamiltonian path in that cluster. Let u and v be the first node and the last node of the path, respectively. Then a high-level virtual hamiltonian cycle containing edge $(u : v)$ can be built easily because there is no any faulty link outside the cluster.

Case 2: Each cluster contains at most $m-2$ faulty links. By Theorem 2, the number of disjoint virtual hamiltonian cycles in a cluster is 2^{m-1}, greater than the total number of faulty links which is $m-1$ for any m. Therefore, there is at least a virtual hamiltonian cycle which does not contain faulty cross-edge. By Lemma 3, a fault-free hamiltonian path $(u \to v)$ in each cluster can be built, where $(u : v)$ is a cube-edge in the virtual hamiltonian cycle.

Finally, a fault-free hamiltonian cycle in the DC(m) can be built by replacing each cube-edge $(u : v)$ in the virtual hamiltonian cycle with the hamiltonian path $(u \to v)$ in each cluster. □

A fault-free hamiltonian cycle can be found in a DC(m) with $m-1$ faulty links. That is, a DC(m) is $(m-1)$-link hamiltonian. This result is optimal because the degree of a DC(m) is $m+1$. Notice that a linear array can be embedded in the dual-cube with m faulty links.

5. Conclusion and Future Work

In this paper, we showed that a fault-free hamiltonian cycle can be constructed in a DC(m) with $m-1$ faulty links. Therefore, a ring and a linear array network can be embedded in a DC(m) when there are $m-1$ and m faulty links, respectively. Because a dual-cube can link much more nodes than other variations of hypercube, it could be used as an interconnection network for large scale parallel computers.

Recently, much of the community has moved on to lower-dimensional topologies such as meshes and tori. However, the SGI Origin2000, a fairly recent multiprocessor, does use a hypercube topology, so the dual-cube could be of use to industry. A lot of issues concerning the dual-cube require further research. Some of them are:

1. Evaluate the architecture complexity vs. performance of benchmarks vs. real cost.
2. Investigate the embedding of other frequently used topologies into a dual-cube.
3. Develop techniques for mapping application algorithms onto a dual-cube.
4. Develop fault-tolerant routing algorithms for a dual-cube with faulty nodes.

References

[1] A. E. Amawy and S. Latifi. Properties and performance of folded hypercubes. *IEEE Transactions on Parallel and Distributed Systems*, 2:31–42, 1991.

[2] Kemal Efe. The crossed cube architecture for parallel computation. *IEEE Transactions on Parallel and Distributed Systems*, 3(5):513–524, Sep. 1992.

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

[4] J. P. Hayes and T. N. Mudge. Hypercube supercomputers. *Proc. IEEE*, 17(12):1829–1841, Dec. 1989.

[5] S. Latifi, S.Q. Zheng and N. Bagherzadeh. Optimal ring embedding in hypercubes with faulty links. In *Proceedings of the IEEE Symposium on Fault-tolerant Computing*, pages 178–184, 1992.

[6] 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*, pages 51–57, ChiaYi, Taiwan, Dec. 2000.

[7] Y. Li, S. Peng, and W. Chu. Efficient collective communications in dual-cube. In *Proceedings of the Thirteen IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 266–271, Anaheim, USA, Aug. 2001.

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

[9] SGI, *Origin2000 Rackmount Owner's Guide*, 007-3456-003, http://techpubs.sgi.com/, 1997.

[10] R. A. Rowley and B. Bose. Fault-tolerant ring embedding in de Bruijn networks. *IEEE Transactions on Computers*, 46(2):187–190, 1994.

[11] Y. C. Tseng, S. H. Chang and J. P. Sheu. Fault-tolerant ring embedding in a star graph with both link and node failures. *IEEE Transactions on Parallel and Distributed Systems*, 8(12):1185–1195, 1997.

[12] L. W. Tucker and G. G. Robertson. Architecture and applications of the connection machine. *IEEE Computer*, 21:26–38, August 1988.

[13] D. J. Wang. Embedding hamiltonian cycles into folded hypercubes with link faults. *Journal of Parallel and Distributed Computing*, 61(4):545–564, 2001.

[14] S. G. Ziavras. RH: a versatile family of reduced hypercube interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(11):1210–1220, November 1994.