Proceedings of the Intl. Symposium on CIS. Shanghai, China, Dec. 16-18, 2004. pp.51-56

Binomial-Tree Fault Tolerant Routing in Dual-Cubes with Large Number of Faulty Nodes

Yamin Li[1], Shietung Peng[1], and Wanming Chu[2]

[1] Department of Computer Science, Hosei University, Tokyo 184-8584 Japan
[2] Department of Computer Hardware, University of Aizu, Fukushima 965-8580 Japan

Abstract. A dual-cube DC(m) has $m+1$ links per node where m is the degree of a cluster (m-cube), and one extra link is used for connection between clusters. 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. In this paper, we propose efficient algorithms for finding a nonfaulty routing path between any two nonfaulty nodes in the dual-cube with a large number of faulty nodes. A node $v \in$ DC(m) is called k-safe if v has at least k nonfaulty neighbors. The DC(m) is called k-safe if every node in DC(m) is k-safe. The first algorithm presented in this paper is an off-line algorithm that uses global information of faulty status. It finds a nonfaulty path of length at most $d(s,t) + O(k^2)$ in $O(|F| + m)$ time for any two nonfaulty nodes s and t in the k-safe DC(m) with number of faulty nodes $|F| < 2^k(m+1-k)$, where $0 \leq k \leq m/2$. The second algorithm is an online algorithm that uses local information only. It can find a fault-free path with high probability in an arbitrarily faulty dual-cube with unbounded number of faulty nodes.

1 Introduction

As the size of computer networks increases continuously, the node failures are inevitable. Routing in computer networks with faults has been more important and has attracted considerable attention in the last decade. Hypercube is a popular network studied by researchers and adopted in many implementations of parallel computer systems, such as Intel iPSC, the nCUBE, the Connection Machine CM-2, and the SGI's Origin 2000 and Origin 3000. Previous research has shown that a hypercube can tolerate a constant fraction of faulty nodes. For example, Najjar et al[1] demonstrated that for the 10-cube, 33 percent of nodes can fail and the network can still remain connected with a probability of 99 percent. Gu and Peng[2] proposed off-line routing algorithms for a k-safe n-cube with up to $2^k(n-k)-1$ faulty nodes, where a node u is k-safe if u has at least k nonfaulty neighbor nodes, and an n-cube is k-safe if all nodes in the cube are k-safe. Chen et al[3] also proposed a distributed routing algorithm in hypercube with large amount of faulty nodes based on local subcube-connectivity.

A dual-cube DC(m)[4, 5] is an undirected graph on the node set $\{0,1\}^{2m+1}$ and there is a link between two nodes $u = (u_0u_1 \dots u_m u_{m+1} \dots u_{2m})$ and $v = (v_0v_1 \dots v_m v_{m+1} \dots v_{2m})$ if and only if the following conditions are satisfied: 1) u and v differ exactly in one bit position i, 2) if $1 \le i \le m$ then $u_0 = v_0 = 1$ and 3) if $m + 1 \le i \le 2m$ then $u_0 = v_0 = 0$. We use $(u : v)$ to denote a link connecting nodes u and v, and $(u \to v)$ or $(v_1 : v_2 : \dots : v_r)$ to denote a path or a cycle. For a node $u = (u_1 \dots u_n)$, $u^{(i)}$ denotes the node $(u_1 \dots u_{i-1}\overline{u_i}u_{i+1} \dots u_n)$, where $\overline{u_i}$ is the logical negation of u_i. The set of neighbors of a subgraph T in G is denoted as $N(T) = \{v | (w : v) \in E(G), w \in V(T), v \notin V(T)\}$.

2 Fault Tolerant Routing Algorithm for *k*-Safe Dual-Cube

We first briefly introduce the algorithm for the fault-tolerant routing in k-safe hypercubes[2]. Given a k-safe n-cube H_n, a set of faulty nodes F with $|F| < 2^k(n-k)$, and two nonfaulty nodes s and t, the idea of finding a fault-free path $s \to t$ is as follows. First, partition H_n along dimension i into two $(n-1)$-cubes, H^0_{n-1} and H^1_{n-1}, such that s and t are separated, say $s \in H^0_{n-1}$ and $t \in H^1_{n-1}$. Assume that $|F \cap H^1_{n-1}| \le |F|/2$. Then, we want to route s to H^1_{n-1} by a fault-free path of length at most $k+2$. This can be done by first constructing a fault-free k-binomial tree with root s $T_k(s)$ in H_n. Since H_n is k-safe the $T_k(s)$ can be found. If $T_k(s) \cap H^1_{n-1} \neq \emptyset$ or $u^{(i)}$ is nonfaulty, where $u \in T_k(s)$, then s is routed to H^1_{n-1}. Otherwise, since $|F| < 2^k(n-k)$ there exists a $u \in N(T_k(s))$ such that u and $u^{(i)}$ are nonfaulty. Therefore, we can route s to $s' \in H^1_{n-1}$. The fault-free path $s' \to t$ in H^1_{n-1} can be found recursively since H^1_{n-1} is $(k-1)$-safe and $|F \cap H^1_{n-1}| < 2^{k-1}((n-1)-(k-1))$. The recursion halts when $k = 0$. In this case, $|F| < n$ and a fault-free path $s \to t$ of length at most $d(s,t)+2$ can be found in $O(n)$ time[6]. The fault-free path $s \to s'$ of length at most $k+2$ can be found in $O(|F|)$ time. Since at most half of the faulty nodes are involved in the next recursion. The time complexity of the algorithm $T(n) = \sum_{i=0}^{k-1} O(|F|/2^i) + O(n) = O(|F| + n)$. The length of the path, $L(k)$, satisfies the equation $L(k) \le L(k-1) + (k+2)$ if $k > 0$, and $L(0) \le d(s,t) + 2$. From this, $L(k) = d(s,t) + O(k^2)$. The algorithm described above is denoted as Hypercube_Routing(H_n, s, t, k, F, P), where P is the fault-free path $s \to t$ in H_n.

A dual-cube DC(m) is k-safe if every node in DC(m) is k-safe. We present an algorithm for finding a fault-free path $s \to t$ in a k-safe DC(m) with number of faulty nodes $|F| < 2^k(m-k+1)$. First, we describe two key techniques for the design of the algorithm. The first one is called *virtual cube*. Given two distinct clusters of the same class in DC(m), say C_s and C_t are of class 0, the virtual $(m+1)$-cube $VH_{m+1} = C_s \cup C_t \cup \{(u : v) | u \in C_s, v \in C_t$, and $u_i = v_i$ for all $i, m+1 \le i \le 2m\}$.

We call the edge $(u : v)$ virtual edge. A virtual edge in VH_{m+1} corresponds to a path $(u \to v)$ in DC(m), and $(s \to t) = (s \to u : u' = u^{(0)} \to v' = v^{(0)} : v \to t)$, where $(u' \to v')$ is a path of length at most m in $C_{u'}$, a cluster of class 1. The 2^m paths in DC(m) corresponding to the 2^m virtual edges in VH_{m+1} are disjoint.

The virtual edge $(u : v)$ is nonfaulty if nodes u, v, u', and v' are nonfaulty and $|F \cap C_{u'}| < 2^{k-1}(m - k + 1)$. If the virtual edge is nonfaulty then since $C_{u'}$ is a $(k-1)$-safe m-cube, a fault-free path $u \rightarrow v$ in DC(m) can be found. Finding all faulty virtual edge takes at most $O(|F|)$ time.

The second one is a technique to find a fault-free path $s \rightarrow u : u' = u^{(0)}$ of length at most $k + 2$, where path $s \rightarrow u$ is a path in C_s under the condition that DC(m) is k-safe and $|F| < 2^k(m - k + 1)$. The path $s \rightarrow u$ can be found by constructing a fault-free k-binomial-tree $T_k(s)$ in C_s, and then considering the nodes in $N(T_k(s))$.

```
Algorithm 1 (Binomial_Tree_Routing(DC($m$), $s, F, P$))
Input: DC($m$), a nonfaulty node $s$, and a set of faulty nodes $F$
    with $|F| < 2^k(m-k+1)$
Output: a fault-free path $P = (s \rightarrow u : u')$ of length at most $k+2$
begin
    $P = \emptyset$;
    find a fault-free $(k-1)$-binomial tree $T_{k-1}(s)$ in $C_s$;
    if there exists a nonfaulty $u'$ for $u \in T_{k-1}(s)$
    then $P = P \cup (s \rightarrow u : u')$;
    else find a fault-free $k$-binomial tree
        $T_k(s) = T_{k-1}(s) \cup \{(u : u^{(i)}) | u \in T_{k-1}(s)\}$,
        where $u^{(i)}$ is nonfaulty, and $i \neq i_j, 1 \leq j \leq r$, the dimensions
        used for the path $s \rightarrow u$ in $T_{k-1}$;
        find a node $u \in N(T_k(s)) \cap C_s$ such that $u$ and $u'$ are nonfaulty;
        $P = P \cup (s \rightarrow w : u : u')$, where $s \rightarrow w$ is a path in $T_k(s)$;
end
```

The details is depicted by Algorithm 1. The next lemma shows that Binomial_Tree_Routing algorithm is correct.

Lemma 1. *For $0 \leq k \leq m/2$, and a nonfaulty node s in a k-safe DC(m) with number of faulty nodes $|F| < 2^k(m - k + 1)$, the fault-free path $(s \rightarrow u : u')$ of length at most $k + 2$ can be found in $O(|F| + m)$ time.*

Proof: From Binomial_Tree_Routing algorithm, since DC(m) is k-safe we know that either a fault-free $T_k(s)$ in C_s is found or there exists a nonfaulty node $w \in T_{k-1}(s)$ such that w' is nonfaulty. In the letter case, let $w = u$ and it is done. So, we assume that $T_k(s)$ is found and for every w in $T_k(s)$, w' is faulty. It was known that for $T_k(s)$ in an m-cube, we have $|N(T_k(s))| \geq 2^k(m - k)$ ([6]). Since C_s is an m-cube and there are at most $|F| - 2^k < 2^k(m^k + 1) - 2^k = 2^k m^k$ faulty nodes in C_s, there exists a node $u \in N(T_k(s))$ such that u and u' are nonfaulty.

The main idea of the proposed algorithm is to route s to C_t if s and t are in different clusters and $|F \cap C_s| \geq |F \cap C_t|$. This can be done using the similar idea of Algorithm 1 and is shown in Algorithm 2, Cluster_Routing. The next lemma shows that the algorithm is correct.

```
Algorithm 2 (Cluster_Routing(DC($m$), $s, t, F, P$))
Input: a $k$-safe DC($m$), nodes $s$ and $t$, $C_s$ and $C_t$ are of the same class,
    and a set of faulty nodes $F$ with $|F| < 2^k(m-k+1)$ and $|F \cap C_s| \geq |F \cap C_t|$
Output: a fault-free path $P = (s \to u \to v)$, where $v \in C_t$
    and $u \to v$ is the path corresponding to virtual edge $(u : v)$
begin
    $P = \emptyset$;
    find a fault-free $(k-1)$-binomial tree $T_{k-1}(s)$ in $C_s$
    if there is a $u \in T_{k-1}(s)$ such that $u'$ is nonfaulty
    then if virtual edge $(u : v)$ is nonfaulty
            then $P = (s \to u \to v)$, $(u \to v)$ is a fault-free path in DC($m$)
            else find a fault-free $(k-1)$-binomial tree $T_{k-1}(u')$ in $C_{u'}$;
                    find a nonfaulty $w \in N(T_{k-1}(u')) \cap C_{u'}$
                    such that virtual edge $(u = w' : v)$ is nonfaulty;
                    $P = (s \to w : u \to v)$, $u \to v$ is a fault-free path in DC($m$)
    else find a fault-free $k$-binomial tree $T_k(s)$ by extending $T_{k-1}(s)$;
            find a nonfaulty $u \in N(T_k(s)) \cap C_s$
            such that virtual edge $(u : v)$ is nonfaulty;
            $P = (s \to u \to v)$, $u \to v$ is a fault-free path in DC($m$);
end
```

Lemma 2. *For $0 \leq k \leq m/2$, and nonfaulty nodes s and t in a k-safe DC(m) with number of faulty nodes $|F| < 2^k(m-k+1)$, the fault-free path $(s \to u \to v)$ can be found in $O(|F|+m)$ time, where $(s \to u)$ is a path in C_s and $(u \to v)$ is the path corresponding to virtual edge $(u : v)$.*

Proof: We divide the proof into two cases. Case 1: $T_k(s)$ is in C_s. Since $|N(T_K(s))| \geq 2^k(m-k+1)$ we can find a nonfaulty virtual edge $(u : v)$, $u \in N(T_k(s))$. Then we are done. Case 2: there exists $u \in T_{k-1}(s)$ such that u' is nonfaulty and virtual edge is faulty. In this case, we should try to route u' to C_t using a fault-free $(k-1)$-binomial tree in $C_{u'}$. From Cluster_Routing algorithm, since the paths that route $u \in N(T_{k-1}(s))$ to C_t and the path that route $w \in N(T_{k-1}(u'))$ to C_t are disjoint and there are totally $2^k(m-k+1)$ disjoint paths, we claim that a fault-free path that route s to C_t does exist.

The algorithm that constructs $s \to t$ is shown in Algorithm 3.

Theorem 1. *For $0 \leq k \leq m/2$, and two nonfaulty nodes s and t in in a k-safe DC(m) with number of faulty nodes $|F| < 2^k(m-k+1)$, the fault-free path $s \to t$ of length at most $d(s,t) + O(k^2)$ can be found in $O(|F|+m)$ time.*

Proof: The correctness of the DualCube_Routing algorithm follows easily from Binomial_Tree_Routing, Cluster_Routing, and Hypercube_Routing algorithms. The Length of the path $L(s,t)$ and time complexity of the algorithm $T(m)$ are shown below for Case 1. The other cases follows easily from the algorithm. $L(s,t) \leq (k+2) + (d(u',v')+1) + (d(v,t) + O(k^2))$. Since $d(u',v') + d(v,t) \leq d(s,t) + d(s,u) = d(s,t) + k + 1$, we have $L(s,t) = d(s,t) + O(k^2)$. The time is $O(|F|)$ for Binomial_Tree_Routing; $O(m)$ for finding path $u \to v$; and $O(|F|+m)$ for Hypercube_Routing. Therefore, the running time $T(m) = O(|F|+m)$.

```
Algorithm 3 (DualCube_Routing(DC($m$), $s, t, k, F, P$))
Input: DC($m$), nonfaulty nodes $s$ and $t$, and
    a set of faulty nodes $F$ with $|F| < 2^k(m-k+1)$
Output: a fault-free path $P = (s \rightarrow t)$
begin
    $P = \emptyset$;
    Case 1: $C_s \neq C_t$ and $class\_id(s) = class\_id(t)$
        Cluster_Routing(DC($m$), $s, F, P$);
        Hypercube_Routing($C_t, v, t, k-1, F \cap C_t, P'$);
        $P = P \cup P'$;
    Case 2: $class\_id(s) \neq class\_id(t)$
        Binomial_Tree_Routing(DC($m$), $s, F, P$);
        if $C_{u'} = C_t$
        then Hypercube_Routing($C_t, u', t, k-1, P$);
        else find a fault-free path $P' = (u' \rightarrow t)$ as in Case 1;
            $P = P \cup P'$;
    Case 3: $C_s = C_t$
        if $|F \cap C_s| < 2^{k-1}(m-(k-1))$
        then Hypercube_Routing($C_s, s, t, k-1, P$);
        else Binomial_Tree_Routing(DC($m$), $s, F, P$);
            Binomial_Tree_Routing(DC($m$), $t, F, P'$);
            find a fault-free path $P' = (u'_s \rightarrow u'_t)$ as in Case 1;
            $P = P \cup P' \cup (u'_s \rightarrow u'_t)$;
end
```

3 A Practical Fault-tolerant Routing Algorithm

The proposed algorithm in the previous section requires that the dual-cube is k-safe. In reality, the chance that the dual-cube will not be k-safe increases when the number of faulty nodes grows. We propose an efficient routing algorithm for the fault-tolerant routing in dual-cube containing a large number of faulty nodes and dual-cube may not be k-safe. In general, without k-safe property, the fault-free k-binomial tree $T_k(s)$ might not exist and the routing algorithm might fail to find the fault-free path $s \rightarrow t$ although the fault-free path does exist. However, for practical reasons, it is interesting to design an efficient algorithm that find a fault-free path $s \rightarrow t$ using only local information of fault status in the dual-cube with a large number of faulty nodes.

The proposed algorithm for fault tolerant routing in an arbitrary faulty dual-cube is distributed and local-information-based. The algorithm is similar to algorithm 3. However, the new algorithm doesn't calculate F while route s to cluster C_t, and it uses binomial trees of increasing size starting with 0-binomial tree, a tree of single node. If the algorithm fails to find a fault-free path $s \rightarrow u \rightarrow v$ then it tries 1-binomial tree and so on until either a fault-free path is found or the k-binomial tree cannot be constructed. Due to the page limitation, we will not present the details of our algorithm in this draft. The simulations for this algorithm have been conducted with uniformly distribution of faulty nodes in DC(m) for $m = 5, 6, 7, 8$ and 9.

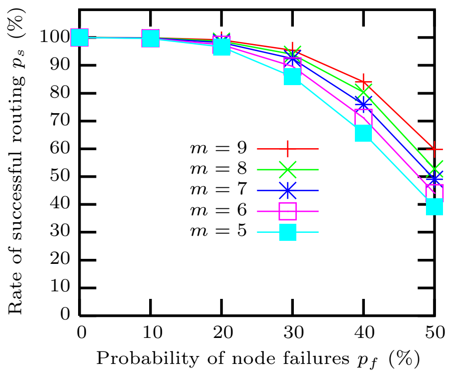

Fig. 1. Successful routing rate

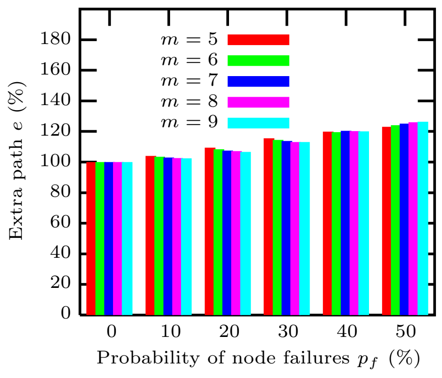

Fig. 2. Path length (%)

The results for successful routing v.s. the node failure rate are shown in Figure 1. It can be seen from the figure that the successful routing rate is very high ($> 90\%$) if the node failure rate is less than 30%. The successful routing rate drop more deeply when the node failure rate is beyond 30%. However, we can say that in most cases, the successful routing rates are still larger than 50% with the node failure rates up to 50%. As for the length of the routing path, we show the results in Figure 2. From the figure, we can say that the fault-free paths found by our algorithm are very close to the minimum paths in most of the cases. The experimental data show that the proposed algorithm performs well in an arbitrarily faulty dual-cubes with possible very large set of faulty nodes.

References

1. Najjar, W., Gaudiot, J.L.: Network resilience: A measure of network fault tolerance. IEEE Transactions on Computers **39** (1990) 174–181
2. Gu, Q.P., Peng, S.: Unicast in hypercubes with large number of faulty nodes. IEEE Transactions on Parallel and Distributed Systems **10** (1999) 964–975
3. Chen, J., Wang, G., Chen, S.: Locally subcube-connected hypercube networks: Theoretical analysis and experimental results. IEEE Transactions on Computers **51** (2002) 530–540
4. Li, Y., Peng, S.: 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 (2000) 51–57
5. Li, Y., Peng, S., Chu, W.: Efficient collective communications in dual-cube. The Journal of Supercomputing **28** (2004) 71–90
6. Gu, Q.P., Peng, S.: Optimal algorithms for node-to-node fault tolerant routing in hypercubes. The Computer Journal **39** (1996) 626–629