A Fault-tolerant Routing Strategy for Gaussian Cube Using Gaussian Tree

Peter, K K, Loh [1] and Zhang, Xinhua [2]
[1] *School of Computer Engineering, Nanyang Technological University, Singapore*
[2] *Dept. of Computer Science and Engineering, Shanghai Jiao Tong Univ., P.R.China*
E-mail: askkloh@ntu.edu.sg

Abstract

Gaussian Cubes (GCs) are a family of interconnection topologies in which the interconnection density and algorithmic efficiency are linked by a common parameter, the variation of which can scale routing performance according to traffic loads without changing the routing algorithm. However, there is no existing fault-tolerant routing strategy for GCs as well as node/link diluted cubes. In this paper, the void is filled for GC with an algorithm based on a new topology: Gaussian Tree (GT). With a many-to-one mapping, the original problem is converted into routing in GT, which is found to be more definite and predictable. A new approach to categorizing faulty components is presented to overcome the problem of low node availability and the maximum number of faults tolerable is given. The algorithm is livelock free and generates deadlock-free routes, which are at most 2F hops longer than the optimal route in a fault-free setting, if F faults are encountered. Finally, simulation is done to show the algorithm's performance, demonstrating its contribution to making GC a more fault-tolerant topology.

1. Introduction

Gaussian Cubes (*GC*s) are a family of interconnection networks parameterized by a modulus *M* and a dimension *n* [1]. Their desirable scalability makes possible generalized analysis of interconnection cost, routing performance, and reliability. Besides, communication primitives such as unicasting, multicasting, broadcasting /gathering [7] can also be done rather efficiently in all *GC*s [1]. However, although research achievements abound in routing in binary hypercubes, there are no existing fault-tolerant routing strategies for *GC*s or for node/link dilution cubes. One of the difficulties lies in the low network node availability (maximum number of faulty neighbors of a node that can be tolerated without disconnecting the node from the network). Therefore, if the topology is fixed, a new method has to be employed to tackle this intrinsic problem.

In this paper, we present a new routing algorithm based on a new topology called Gaussian Tree (*GT*). In $GC(n,2^{\alpha})$, *GT* is dependent only on α and divides all the nodes in $GC(n,2^{\alpha})$ into 2^{α} classes according to their least significant α bits. So the original problem is converted into first routing in *GT* (i.e. between different classes) and then routing in one such class. The former is facilitated by the definite and predictable routing in trees while the latter is actually routing in ordinary binary hypercube. Faults encountered in different stages of this divide-and-conquer strategy lead to a new categorization of faulty components, enabling analysis of routing strategy in the presence of far more faults than the network node availability. Methodologically speaking, this approach also opens window to a brand-new way of analyzing network reliability, which is especially valuable for incomplete networks.

The characteristics of our routing strategy for $GC(n,2^{\alpha})$ encompass:

1) Incurs message overhead of only $O(n)$.
2) The computation complexity for intermediate nodes is at most $O(\alpha(n-\alpha)\log\alpha)$.
3) Guarantees a message path length not exceeding $2F$ longer than the optimal path found in a fault-free setting, provided the distribution of faulty components in the network satisfies the precondition of *Theorem* 3 and 5.
4) Each node requires at most $\left\lceil \frac{n-1}{2^{\alpha}} \right\rceil + 1$ rounds of fault status exchange with its neighbors.
5) Each node maintains and updates at most *F* *n*-bit node addresses, where *F* is the number of faults related to nodes whose least significant α bits are same as the current node.
6) Generates deadlock-free and livelock-free routes.
7) The number of faulty components tolerable is presented in *Figure* 4 and *Theorem* 5.

This paper is organized as follows. Preliminaries are given in Section 2 to provide an equivalent definition of

GC that facilitates the following discussion. Section 3 defines *GT*. The routing algorithm for the fault-free *GC* is described in Section 4 separately to make the subsequent section clearer. In Section 5, the fault-tolerant routing strategy that deals with all categories of faults is studied. Then in Section 6, simulation results are presented to demonstrate the performance of our algorithm. The whole paper is concluded by Section 7 where some suggestions for further work are given.

2. Preliminary

The binary *Gaussian Cube* [1] is denoted by $GC(n,M)$, where n (network dimension) ≥ 0 and M (modulus) ≥ 1. It has 2^n nodes that are identified with unique n-bit labels. A link connects two nodes p and q if the following conditions are both true:

1) The labels of p and q differ in the c^{th} bit for some c, $0 \leq c \leq (n-1)$.
2) p and q are in the congruence class $[c]_{M'}$, where $M' = \min\{2^c, M\}$.

The congruence class of c modulo M, $[c]_M$, is the set $\{kM + c | k \in Z\}$, where Z represents the set of integer.

According to the definition above, if node $p = a_{n-1}a_{n-2}\cdots a_c \cdots a_1 a_0$ ($a_i \in \{0, 1\}$ for $i \in [0, n-1]$) is connected with $q = a_{n-1}a_{n-2}\cdots \overline{a_c} \cdots a_1 a_0$, then there must exist k_1 and $k_2 \in Z$, such that:

$$\overline{a_{n-1}a_{n-2}\cdots a_c \cdots a_1 a_0} = k_1 M' + c \quad (1)$$

$$\overline{a_{n-1}a_{n-2}\cdots \overline{a_c} \cdots a_1 a_0} = k_2 M' + c \quad (2)$$

(2) – (1) and take absolute value on both sides, we get:

$2^c = \quad |k_1 - k_2| M' \quad (3)$

Therefore, M' must be the power of 2. Since $M' = \min\{2^c, M\}$, M must also be the power of 2 if $M < 2^c$. Otherwise, as $M' = \min\{2^c, M\}$ we can assume $M < 2^{n-1}$ without loss of generality. Consequently, there will be no link spanning in any dimension $c > \lfloor \log M \rfloor$. Effectively, the network is separated into $2^{n-1-\lfloor \log M \rfloor}$ disconnected subnetworks, with each combination of the first $n-1-\lfloor \log M \rfloor$ bits representing one such subnetwork. Formally, $GC(n, M) = \bigcup_{i=0}^{2^{n-1-\lfloor \log M \rfloor}-1} G_i$. Each subnetwork G_i is composed of $<V_i, E_i>$, where

$V_i = \{a_{n-\lfloor \log M \rfloor-2}\cdots a_i \cdots a_0 b_{\lfloor \log M \rfloor}\cdots b_j \cdots b_0 \mid$
$b_j \in \{0,1\},\ 0 \leq j \leq \lfloor \log M \rfloor,\ \overline{a_{n-\lfloor \log M \rfloor-2}\cdots a_i \cdots a_0} = i\}$

$E_i = \{(v_1, v_2) \in E \mid v_1 \in V_i, v_2 \in V_i\}$, where E is the set of edges in the original network.

Obviously, for $\forall i, j \in [0, 2^{n-1-\lfloor \log M \rfloor})$ and $i \neq j$, $V_i \cap V_j = \Phi,\ E_i \cap E_j = \Phi$. So routing can be done within the subnetwork G_i if the source and destination both belong to G_i, or fails otherwise. Furthermore, as G_i is isomorphic to $GC(\lfloor \log M \rfloor + 1,\ 2^{\lfloor \log M \rfloor})$, this situation is covered in the following case, where M is power of 2. So henceforth, we assume M is power of 2.

(*Theorem* 1) In $GC(n, 2^\alpha)$, node $p = a_{n-1}a_{n-2}\cdots a_c \cdots a_0$ ($a_i \in \{0, 1\}$ for $i \in [0, n-1]$) has a link in dimension c ($c \in [1, n-1]$) if and only if:

$$\begin{cases} \overline{a_{\alpha-1}a_{\alpha-2}\cdots a_0} = c \ \%\ 2^\alpha & \text{if } c \in (\alpha, n) \\ \overline{a_{c-1}a_{c-2}\cdots a_0} = c & \text{if } c \in [1, \alpha] \end{cases}$$

where 'x % y' represents the modulus of x divided by y, like in *C/C++*. And each node has a link in dimension 0.

Proof: For dimension 0, since for any $M \geq 1$, $M' = \min\{2^0, M\} = 1$. Any integer p and q must be in the congruence class $[0]_{M'} = [0]_1$. So each node has a link in dimension 0. Two other cases should be considered to prove *Theorem* 1.

(Case I) $c \in (\alpha, n)$.

(Necessary) According to Equation (1), $\overline{a_{n-1}a_{n-2}\cdots a_\alpha \cdots a_1 a_0} = k_1 M' + c$. Thus, $\overline{a_{n-1}a_{n-2}\cdots a_{\alpha+1}a_\alpha} \cdot 2^\alpha + \overline{a_{\alpha-1}a_{\alpha-2}\cdots a_0} = k_1 \cdot 2^\alpha + c$. Take the modulus of 2^α on both sides and due to the fact that $\overline{a_{\alpha-1}a_{\alpha-2}\cdots a_0} < 2^\alpha$, we obtain $\overline{a_{\alpha-1}a_{\alpha-2}\cdots a_0} = c\%2^\alpha$.

(Sufficient) If $\overline{a_{\alpha-1}a_{\alpha-2}\cdots a_0} = c\%2^\alpha$, then $\overline{a_{n-1}a_{n-2}\cdots a_\alpha a_{\alpha-1}\cdots a_1 a_0} - c$ can be wholly divided by 2^α. Define $k_1 = \dfrac{\overline{a_{n-1}a_{n-2}\cdots a_c \cdots a_1 a_0} - c}{2^\alpha} \in Z$ and if $a_c = 0$, $k_2 = k_1 + 2^{c-\alpha}$ otherwise $k_2 = k_1 - 2^{c-\alpha}$. Then, $\overline{a_{n-1}a_{n-2}\cdots a_c \cdots a_1 a_0} = k_1 \cdot 2^\alpha + c = k_1 \cdot \min(2^c, 2^\alpha)$ $= k_1 M' + c$ and $\overline{a_{n-1}a_{n-2}\cdots \overline{a_c} \cdots a_1 a_0} = k_2 \cdot 2^\alpha + c$ $= k_2 \cdot \min(2^c, 2^\alpha) + c = k_2 M' + c$.

In other words, according to the original definition, $a_{n-1}a_{n-2}\cdots a_c \cdots a_1 a_0$ has a link in dimension c.

(Case II) $c \in [1, \alpha]$. The proof is similar to case I. ■

3. Gaussian Tree

According to *Theorem* 1, we can see that whether a packet can be forwarded through dimension c at node p is entirely irrelevant to $a_{n-1}a_{n-2}\cdots a_\alpha$, regardless of whether $c > \alpha$ or not. So the last α bits in nodes' address are of more importance. We define a *Gaussian Graph* based on these α bits.

(*Definition* 1) **:** Gaussian Graph

We call the undirected graph G_n ($n \geq 2$) *Gaussian Graph* if it is composed of <V$_{\boldsymbol{n}}$, E$_{\boldsymbol{n}}$>, where:

V$_{\boldsymbol{n}}$ = $\{ a_{n-1}a_{n-2}\cdots a_1a_0 \mid a_i \in \{0,1\}, for\ \ i \in [0,\ \ n-1] \}$

E$_{\boldsymbol{n}}$ = $\{(a_{n-1}a_{n-2}\cdots a_c \cdots a_1a_0\ ,\ a_{n-1}a_{n-2}\cdots \overline{a_c}\cdots a_1a_0) \mid$ $c = 0$ or $c \in [1, n\text{-}1]$ and $\overline{a_{c-1}a_{c-2}\cdots a_1a_0} = c\}$.

Figure 1 demonstrates the topology of G_2, G_3, and G_4. They can be generated easily by adding edges, according to the definition of E_n, to the original graph which is composed only of nodes.

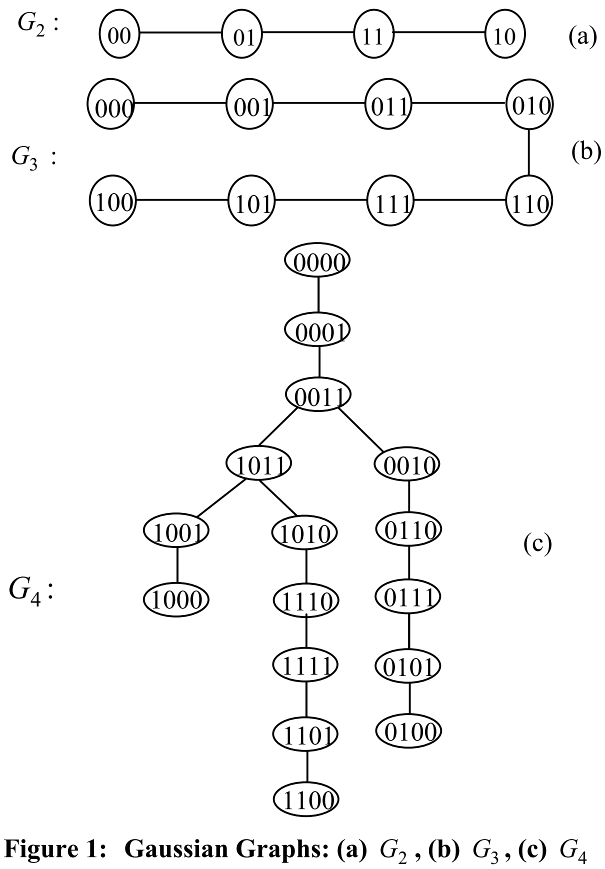

Figure 1: Gaussian Graphs: (a) G_2 **, (b)** G_3 **, (c)** G_4

(*Lemma* 1): Tree's Equivalent Definition

Suppose graph G has n vertices and e edges. G is a tree if and only if G is connected and $e = n-1$. [2]

(*Theorem* 2) G_n is a tree. ($n \geq 2$)

Proof:

Step 1. G_n is connected.

This is evident from the following algorithm *PC*. It can find a route from s to d in G_n, when n, s and d are given.

(*Algorithm* 1) *Path Construction Algorithm* (*PC*)

```
path $PC(n,\ s = s_{n-1}s_{n-2}\cdots s_1s_0\ , d = d_{n-1}d_{n-2}\cdots d_1d_0)$
{
    Let $c$ be the dimension corresponding to the leftmost
    '1' in $R = s \oplus d$; // '$\oplus$' means bitwise exclusive OR
    if ($c$==0)
        return ($s$, $d$);          // $s$ and $d$ are neighbors,
    suppose $\overline{a_{c-1}a_{c-2}\cdots a_0} = c$ ( $a_i \in \{0,1\},\ i \in [0, c-1]$ )
    path0 = ( $s_{n-1}\cdots s_c a_{c-1}\cdots a_0$ , $s_{n-1}\cdots \overline{s_c} a_{c-1}\cdots a_0$ );
    path1 = $PC$( $s_{n-1}s_{n-2}\cdots s_1s_0$ , $s_{n-1}\cdots s_c a_{c-1}\cdots a_0$ );
    path2 = $PC$( $s_{n-1}\cdots \overline{s_c} a_{c-1}\cdots a_0$ , $d_{n-1}d_{n-2}\cdots d_0$ );
    return path1|| path0 || path2);
    // Here, '||' stands for the concatenation operation.
}
```

E.g., *PC* (010110,011110) = *PC* (010110, 010011) || (010011,011011) || *PC* (011011, 011110).

The recursion must be able to terminate within depth n because the leftmost '1' moves at least one bit rightward after one recursion, until it reaches dimension 0 when the source and destination will be neighbors. As links are not found step by step from source to destination, it requires a sort to re-order the final link set.

Step 2. There are 2^n nodes in G_n .(Obvious)

Step 3. There are only $2^n - 1$ edges in G_n .

We denote the number of links spanning in dimension i as E$_{\boldsymbol{n}}$(i) ($i \in [0,\ \ n-1]$). E$_{\boldsymbol{n}}$(0) = 2^{n-1}. A node has a link on dimension 1 if and only if its rightmost bit is 1. Such links only connect nodes in the form of ($a_{n-1}a_{n-2}\cdots x\ 1$, $a_{n-1}a_{n-2}\cdots \overline{x}\ 1$). So E$_{\boldsymbol{n}}$(1) = 2^{n-2}. A link spanning in dimension 2 can only connect node pairs in the form of: ($a_{n-1}a_{n-2}\cdots x10$, $a_{n-1}a_{n-2}\cdots \overline{x}10$). So E$_{\boldsymbol{n}}$(2) = 2^{n-3}. Likewise, it is easy to prove that E$_{\boldsymbol{n}}$(i) = 2^{n-i-1}. Thus $|$ E$_{\boldsymbol{n}}$ $|$ = $\sum_{i=0}^{n-1} E_n(i) = \sum_{i=0}^{n-1} 2^{n-i-1} = 2^n - 1$.

Combine 1-3 and apply *Lemma* 1, it can be concluded that G_n is a tree. ■

From now on, we denote G_n as T_n and name it as Gaussian Tree (GT) to emphasize this property. We denote the node k in T_n as $T_n(k)$.

In *PC* Algorithm, since the path will not go to one node more than once and we are routing in a tree, the resultant route must be optimal. Besides, as the algorithm finds the path link by link, the spatial and computational complexities are dependent on the diameter of T_n (maximum distance between node pairs), denoted as $D(T_n)$. Figure 2 shows that $D(T_n)$ is $O(n)$. Thus, the time and space complexity for running Path Construction Algorithm is $O(D(T_n) + D(T_n)\log D(T_n))$ $= O(n \log n)$. The second term is for the sorting.

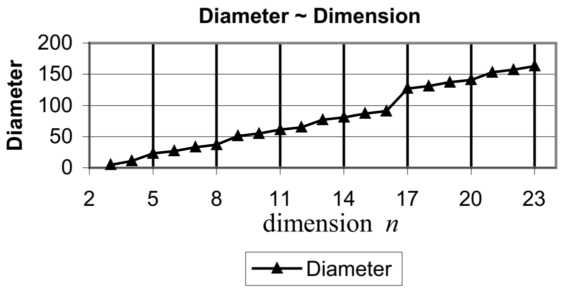

Figure 2 Diameter of T_n versus n (Dimension)

The existence of Gaussian Tree is crucial for our algorithm because, for each source and destination pair in a tree, there is a set of nodes, which the packet must cover in its journey, and which can be calculated at the source. This makes routing much more definite and predictable.

4. Routing in fault-free Gaussian Cube

(*Definition* 2) *k-Ending Class*

In $GC(n,2^{\alpha})$, for $\forall k \in [0, 2^{\alpha}-1]$, we call the following set $EC(n,\alpha,k)$ *k*-ending class:

$$EC(n,\alpha,k) = \{a_{n-1}a_{n-2}\cdots a_{\alpha}a_{\alpha-1}\cdots a_1a_0 \mid a_i \in \{0,1\},$$
$$i \in [0, n-1],\ a_{\alpha-1}\cdots a_0 = k\ \}.$$

$EC(n,\alpha,k)$ is abbreviated as $EC(k)$ when the $GC(n,2^{\alpha})$ is given. According to *Theorem* 1, if link (v_1, v_2) spans in dimension $c \ge \alpha$, then $v_1, v_2 \in EC(c\%2^{\alpha})$. $EC(k)$ corresponds to $T_{\alpha}(k)$ in Gaussian Tree T_{α}. Let the dimensions in $[\alpha, n-1]$ on which each node of $EC(k)$ has a link comprise set *Dim*(*n, α, k*), then *Dim*(*n, α, k*) = $[\alpha, n-1] \cap [k]_{2^{\alpha}}$. When the $GC(n,2^{\alpha})$ is given, *Dim*(*n, α, k*) is also abbreviated as *Dim*(*k*).

Suppose the source is *s* and the destination is *d*. Denote *R* = *s* ⊕ *d* (Exclusive OR). If there is a '1' in *R* and its dimension *c* is no less than *α*, then the path from *s* to *d* must cover at least one node *x*, such that $x \in EC(c\%2^{\alpha})$. Viewed in T_{α}, that means the path must begin from $T_{\alpha}(s\%2^{\alpha})$, end at $T_{\alpha}(d\%2^{\alpha})$ and must pass all nodes in $S = \{\ T_{\alpha}(k\%2^{\alpha}) \mid k \ge \alpha,\ R\,\&\,2^k \ne 0\ \}$, where '&' stands for bitwise AND operation. Since the problem has now been mapped to a tree, with the starting and ending nodes as well as the intermediate nodes given, it is simpler to find an optimal route. We can use Algorithm 1 to find a route from $T_{\alpha}(s)$ to $T_{\alpha}(d)$ in T_{α}, when α, *s* and *d* are given.

Secondly, we introduce an algorithm for arranging multi-destination routing from a tree root. Several nodes belonging to the tree need to be visited and then the packet must go back to the root. It is easy to find that as long as the following principle is met, the path generated must be optimal: if the packet is currently at node *p*, it can never backtrack to the parent unless no destination still exists in the subtree of *p*.

(*Algorithm* 2) *Closed-Traverse Algorithm in tree* (*CT*)

Suppose we are at the root $r = r_{\alpha-1}r_{\alpha-2}\cdots r_1r_0$ where $r_i \in \{0,1\}$ for all $i \in [0, \alpha-1]$. We are to visit $D = \{\ d_1, d_2, \cdots, d_n\ \}$ whose members are all nodes in the tree and finally go back to *r*. The prototype of the algorithm is $CT(r, D)$. We first pick up randomly one $d \in D$ and use Algorithm 1 to find a route *L* from *r* to *d*. Then for each $d_i \in D$, if d_i is covered by *L*, we only need to record that fact. But if it is not covered, we must find a node in *L* at which the packet must branch away from *L*.

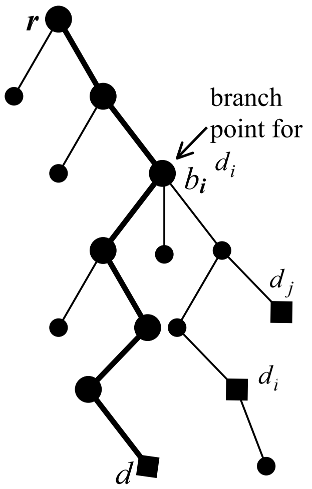

Figure 3 Example for *CT* algorithm

For example, in Figure 3, the bold line represents *L*, and to reach d_i, the route must branch at b_i. However, to calculate b_i, we need not find the complete path from *r* to d_i. The following function FindBP(*L*, *r*, *d_i*) is enough. Suppose CheckIn (*v*,*L*) returns whether *v* is covered by *L*.

```
Point FindBP(L, r, d_i) // FindBP(route, source, destination)
{
    Let c be the dimension corresponding to the leftmost
    '1' in R = s ⊕ d_i;
    if (c==0)        return r;   // $b_i = r$
    suppose $\overline{a_{c-1}a_{c-2}\cdots a_0} = c\ (a_i \in \{0,1\},\ i \in [0, c-1])$;
    $(v_1, v_2) = (\ r_{\alpha-1}\cdots r_c a_{c-1}\cdots a_0,\ r_{\alpha-1}\cdots \overline{r_c} a_{c-1}\cdots a_0\ )$;
    if (CheckIn($v_1$, L) && !CheckIn($v_2$, L)) return $v_1$;
    if (CheckIn($v_1$, L) && CheckIn($v_2$, L))
        return FindBP(L, $v_2$, d_i);
    if (!CheckIn($v_1$, L) && !CheckIn($v_2$, L))
        return FindBP(L, r, $v_2$);
    // !CheckIn($v_1$, L) && CheckIn($v_2$, L) is impossible
}
```

As a node in *L* might serve as branch point for more than one destination in *D*, we use a table to record it. We denote the mapping as $B(\cdot)$. For example, in Fig. 3, b_i is the branch point for d_i and d_j, so $B(b_i) = \{\ d_i,\ d_j\ \}$.

After all members in D are processed and table $B(\cdot)$ is obtained, we begin to go from r to d by following L. Once we arrive at a node p where $B(p) \neq \Phi$, run this algorithm again by calling $CT(p, B(p))$. After that, we proceed along L, until d is reached. Then go back to r in a reverse direction of L. Since this is a distributed algorithm, CT is not recursive as it appears here. ■

Finally, we present the complete routing algorithm for fault-free *Gaussian Cube*.

(*Algorithm* 3) Fault Free GC Routing (*FFGCR*)

The input of *FFGCR* is: n and α for $GC(n, 2^{\alpha})$, binary source $s = s_{n-1}s_{n-2}\cdots s_0$, destination $d = d_{n-1}d_{n-2}\cdots d_0$.

```
FFGCR (n, α, $s = s_{n-1}s_{n-2}\cdots s_1 s_0$, $d = d_{n-1}d_{n-2}\cdots d_1 d_0$)
{
    // map the problem from $GC(n,2^{\alpha})$ to $T_{\alpha}$
    $R = s \oplus d$;  $s' = T_{\alpha}(s \ \% \ 2^{\alpha})$, $d' = T_{\alpha}(d \ \% \ 2^{\alpha})$;
    $P = \{ i \in [\alpha, n-1] \mid R \,\&\, 2^{i} \neq 0 \}$, $D = \{ T_{\alpha}(x \% 2^{\alpha}) \mid x \in P \}$;
    $L = PC(s', d')$; // $L \subset T_{\alpha}$
    Build table $B(\cdot)$ for all nodes $n \in L$ with FindBP( ).
    $n = s'$;
    while ($n \neq$ Null)    // traverse using the least significant
    {                        // α dimensions in $GC(n, 2^{\alpha})$
        if ( $B(n) \neq \Phi$ )
            call CT(n, B(n)) to traverse all nodes in B(n)
            and go back to n ;
        if ( $n \in D$ )
            go through all preferred dimensions
            $c \in [\alpha, n-1] \cap [x]_{2^{\alpha}}$
        n = getNext(L, n); // get next node in L
    }                      // getNext(L, d') = Null.
}
```

It can be easily deduced that the message overhead is $O(n)$ and the computation complexity is $O(\alpha(n-\alpha)\log\alpha)$.

5. Fault-tolerant routing strategy in GC

To overcome the problem of low node availability, we categorize faulty components.

(*Definition* 3) A-category (link) fault

If a link error occurs at a dimension $c \geq \alpha$, it is called A-category (link) fault.

(*Definition* 4) B-category fault

If all link failures incurred by an error are in dimensions less than α, then the error is called B-category fault. B-category faults can be both link error and node error as long as that node has no incident link spanning in a dimension $c \geq \alpha$. A link error is either A or B-category.

(*Definition* 5) C-category (node) fault

If a node error implies break down of links in dimensions both smaller and no smaller than α, it is called C-category (node) fault. A node error is either B or C-category.

(*Definition* 6) *k*-Ending-*t*-Equivalent Class

In *k*-ending class $EC(n, \alpha, k)$, for $\forall\ t \in [0, 2^{n-\alpha-|Dim(k)|} - 1]$ (see *Definition* 2 for the meaning of *Dim*(k)), we call the set $EEC(n, \alpha, k, t)$ *k*-ending-*t*-equivalent class: $EEC(n, \alpha, k, t)$

$= \{ a_{n-1}\cdots a_{\alpha}a_{\alpha-1}\cdots a_0 \in EC(n, a, k) \mid$ bits in dimensions other than $[0, \alpha-1] \cup Dim(k)$ comprise value $t\}$.

k-Ending-t-Equivalent Graph $GEEC(n, a, k, t)$ is defined as a subgraph of $GC(n,\ 2^{\alpha})$ whose vertex set is $EEC(n, \alpha, k, t)$ with original edges connecting vertex in $EEC(n, \alpha, k, t)$.

(*Theorem* 3)

If only A-category faults exist in $GC(n,\ 2^{\alpha})$, and in all $GEEC(n, \alpha, k, t)$ ($k \in [0, 2^{\alpha} - 1]$, $t \in [0, 2^{n-\alpha-|Dim(k)|})$, the number of faulty component is less than $N(k) = \left\lfloor \frac{n-1-k}{2^{\alpha}} \right\rfloor + 1 - \delta(k, \alpha)$ ($\delta(k, \alpha) = k < \alpha \,?\, 1 : 0$), there is a fault-tolerant and cycle-free routing strategy for any source and destination pair.

Proof. Obviously, $GEEC(n, a, k, t)$ is a binary hypercube embedded in $GC(n, 2^{\alpha})$. Let source be s and destination be d. Let $p = s \oplus d$. Denote: $P = \{ i \in [\alpha, n-1] \mid p \ AND\ 2^{i} \neq 0 \}$, $D = \{ x \ \% \ 2^{\alpha} \mid x \in P \}$, $I = \{ EC(x) \mid x \in D \}$. As there are only A-category faults, traversing through links spanning in the least significant α dimensions is always successful. So it is guaranteed that for any member $EC(k)$ in I, a packet can reach at least one node in $EC(k)$. Suppose a packet reaches $EC(k) \in I$ by arriving at node x and $x \in EEC(n, \alpha, k, t)$. The k (*if* $k \geq \alpha$), $k + 2^{\alpha}, k + 2 \cdot 2^{\alpha}, k + 3 \cdot 2^{\alpha}, \cdots,$ $k + \max(0, \left\lfloor \frac{n-k-1}{2^{\alpha}} \right\rfloor) \cdot 2^{\alpha}$ bits of x and d are $x' = x_0 x_1 \cdots x_{|Dim(k)|-1}$ and $d' = d_0 d_1 \cdots d_{|Dim(k)|-1}$ respectively. Then we can focus on routing in binary hypercube $GEEC(n, \alpha, k, t)$ from x' to d', which is guaranteed by the precondition of the theorem and *FTCR* in [4] or strategies in [5][6] which ensure a packet to be sent from any non-faulty source to any non-faulty destination in a deadlock-free fashion, as long as the number of faulty links is less than the dimension of the binary hypercube. After all the bits in dimensions $[k]_{2^\alpha} \cap [\alpha, n-1]$ are set to be same as d, use links spanning in the last α dimensions to go to another member in I. Finally destination d is reached. ■

Suppose there are only A-category faults and F A-category faults are encountered, then the resultant route is at most $2F$ longer than the optimal route found in a fault free setting. We can also conclude that in $GC(n,2^\alpha)$, the maximum number of faulty links tolerable is:

$$T(GC(n,2^\alpha)) = \sum_{k=0}^{2^\alpha - 1} 2^{n-\alpha-t_k} \max(t_k - 1,\ 0)$$

where $t_k = \left\lfloor \frac{n-k-1}{2^\alpha} \right\rfloor + 1 - \delta(k,\alpha)$.

Figure 4 demonstrates the trend of $\log_2 T(GC(n,2^\alpha))$ versus n, when α < 5.

$\log_2(T(GC(n,2^\alpha)))\ \sim\ n$

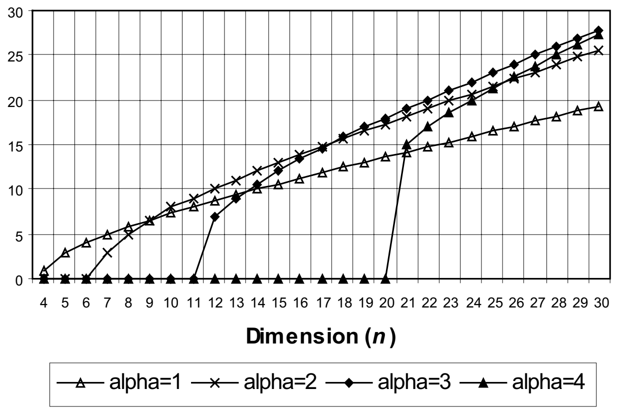

Figure 4 $\log_2(T(GC(n,2^\alpha)))\ \sim\ n$ **(dimension)**

(*Definition* 7) *Exchanged Hypercube*

The *Exchanged Hypercube* is defined as $EH(s,t) = (V,E)$ $(s \ge 1, t \ge 1)$, where

$V = \{a_{s-1} \cdots a_0 b_{t-1} \cdots b_0 c \mid a_i, b_j, c \in \{0,1\}$ for $i \in [0,s]$, $j \in [0,\ t]\}$

$E = \{(v_1, v_2) \in V \times V \mid v_1 \oplus v_2 = 1$ or $v_1[s+t : t+1] = v_2[s+t : t+1]$, $H(v_1[t:1], v_2[t:1]) = 1$, $v_1[0] = v_2[0] = 1$ or $v_1[t:1] = v_2[t:1]$, $H(v_1[s+t:t+1], v_2[s+t:t+1]) = 1$, $v_1[0] = v_2[0] = 0\ \}$

Here, $v[x:y]$ represents the bit pattern of v between dimension y and x inclusive. H (x, y) stands for the *Hamming* distance between x and y.

In $EH(s,t)$, the 0-ending nodes together with the links connecting in between comprise 2^t s-dimension binary hypercubes (denoted as $B_s(EH(s,t))$ collectively). For any $k \in [0, 2^t)$, we denote as $B_s(EH(s,t),k)$ the binary hypercube whose nodes are composed of the following set: $V_s(EH(s,t),k) = \{a_{s-1} \cdots a_0 b_{t-1} \cdots b_0 0 \mid \overline{b_{t-1} \cdots b_0} = k$, $a_i, b_j \in \{0,1\}, i \in [0,s), j \in [0,t)\}$. If $x \in V_s(EH(s,t),k)$ and $\overline{x[s+t:t+1]} = p$, we denote such nodes as $V_s(EH(s,t),k,p)$.

Likewise, the 1-ending nodes together with the links connecting in between comprise 2^s t-dimension binary hypercubes (denoted as $B_t(EH(s,t))$ collectively). For any $l \in [0, 2^s)$, we denote as $B_t(EH(s,t),l)$ the binary hypercube whose nodes are composed of the following set: $V_t(EH(s,t),l) = \{a_{s-1} \cdots a_0 b_{t-1} \cdots b_0 0 \mid \overline{a_{s-1} \cdots a_0} = l$, $a_i, b_j \in \{0,1\}\ \ i \in [0,s),\ \ j \in [0,t)\}$. If $x \in V_t(EH(s,t),l)$ and $\overline{x\ [t:1]} = q$, we denote x as $V_s(EH(s,t),l,q)$. So $V_s(EH(s,t),k,p)[s+t:1] = V_t(EH(s,t),p,k)[s+t:1]$.

Suppose there are F_s faulty components in $B_s(EH(s,t))$, and F_t faulty components in $B_t(EH(s,t))$. Let $E_0(EH(s,t)) = \{(v_1, v_2) \in EH(s,t) \mid v_1 \oplus v_2 = 1\ \}$. Suppose there are F_0 faulty links in $E_0(EH(s,t))$ $\backslash \{(v_1, v_2) \in EH(s,t) \mid v_1$ or v_2 is faulty$\}$. We have:

(*Theorem* 4)

If $F_s + F_0 < s$ and $F_t + F_0 < t$, there is a deadlock-free and livelock-free algorithm that can deliver messages from a nonfaulty source r to a nonfaulty destination d in no more than $H(r,d) + 2(F_s + F_t) + 2$ hops.

This theorem is evident from the following algorithm:

(*Algorithm* 4) **F**ault-tolerant **R**outing in $EH(s,t)$ (*FREH*)

(Case I) Suppose $r = B_s(EH(s,t),k_0,l_1)$ and $d = B_t(EH(s,t),l_0,k_1)$. Since $F_s + F_0 < s$, it is affordable to communicate within each $B_s(EH(s,t),k)$ in the initialization phase, so that each node in it knows and records the set of nodes in $B_s(EH(s,t),k)$ whose link in $E_0(EH(s,t))$ (i.e. in dimension 0) is faulty.

In one case, if r finds that $B_s(EH(s,t),k_0,l_0)$'s link in dimension 0 is non-faulty, it sends the packet within $B_s(EH(s,t),k_0)$ to $B_s(EH(s,t),k_0,l_0)$. This is guaranteed to succeed as was proved in *Theorem* 3. After that, $B_s(EH(s,t),k_0,l_0)$ sends the packet to $B_t(EH(s,t),l_0,k_0)$ via the link in dimension 0. Finally, the packet is sent in $B_t(EH(s,t),l_0)$ to $B_t(EH(s,t),l_0,k_1)$, which is guaranteed by $F_t + F_0 < t$.

In the other case, if by looking up its local table, r finds that the 0-dimension link of $B_s(EH(s,t),k_0,l_0)$ is faulty, then there must be a nonfaulty neighbor of r whose 0-dimension link is also nonfaulty. This is guaranteed by

$F_s + F_0 < s$. Denote it as $B_s(EH(s,t),k_0,l_2)$. So the packet is sent to $B_s(EH(s,t),k_0,l_2)$, which in turn, sends the packet to $B_t(EH(s,t),l_2,k_0)$. Now there must be a nonfaulty neighbor of $B_t(EH(s,t),l_2,k_0)$ in $B_t(EH(s,t),l_2)$ whose 0-dimension link is also nonfaulty. If there is such a neighbor in preferred dimension, then use it. Otherwise, use the spare dimension and mask it so that it will not be used again. After going back to $B_s(EH(s,t))$, the process above repeats and finally the packet reaches *d*.

Due to the use of mask for dimensions in [1, t], the route is livelock free. Deadlock-freeness is still guaranteed. Since faulty components might cause the use of a spare dimension, which brings about for and pro between $B_t(EH(s,t))$ and $B_s(EH(s,t))$, the number of hops is bounded by $H(r,d)+2(F_s+F_t)$.

(Case II) If $r = B_t(EH(s,t),l_0,k_1)$ and $d = B_s(EH(s,t),k_0,l_1)$, as $EH(s,t)$ is isomorphic to $EH(t,s)$, the algorithm is the same as case I.

(Case III) Suppose $r = B_s(EH(s,t),k_0,l_0)$ and $d = B_s(EH(s,t),k_1,l_1)$. If $k_1 = k_0$, then it is routing in *s*-dimension binary hypercube. Otherwise, the packet is sent to $B_t(EH(s,t),k_0)$ via the 0-dimension link of *r* or one of its neighbors in $B_s(EH(s,t),k_0)$. Then the problem is the same as in case I. But now, the number of hops is bounded by $H(r,d)+2(F_s+F_t)+2$ because of the extra hops in dimension 0.

(Case IV) Suppose $s = B_t(EH(s,t),l_0,k_0)$ and $d = B_t(EH(s,t),l_1,k_1)$ ($l_1 \neq l_0$).

This case is handled in the same way as in case III. ■

Come back to $GC(n,2^{\alpha})$. Suppose $T_\alpha(p)$ and $T_\alpha(q)$ are neighbors in T_α. For each $k \in [0, 2^{n-\alpha-|Dim(p)|-|Dim(q)|}-1]$, define graph $G(n,\alpha,p,q,k) = <V(n,\alpha,p,q,k), E(n,\alpha,p,q,k)>$, where $V(n,\alpha,p,q,k)$ is the set of nodes in $GC(n,2^{\alpha})$ whose bits in dimensions other than $Dim\ (p) \bigcup Dim\ (q) \bigcup [0,\alpha-1]$ comprise *k* in value and whose rightmost α bits represent *p* or *q*. $E(n,\alpha,p,q,k)$ is the subset of links in $GC(n,2^{\alpha})$ which connect nodes in $V(n,\alpha,p,q,k)$. The links in $E(n,\alpha,p,q,k)$ that span between $B_t(G(n,\alpha,p,q,k))$ and $B_s(G(n,\alpha,p,q,k))$ comprise $E_0(G(n,\alpha,p,q,k))$. If the last α bits are viewed as dimension 0 that can take value only in {*p*, *q*}, then $G(n,\alpha,p,q,k)$ is effectively isomorphic to *Exchanged Cube* $EH(|Dim(p)|, |Dim(q)|)$ (or *EH* (|*Dim* (*q*)|, |*Dim* (*p*)|)). Suppose there are $e_t(n,\alpha,p,q,k)$ faulty components in $B_t(G(n,\alpha,p,q,k))$ and $e_s(n,\alpha,p,q,k)$ faulty components in $B_s(G(n,\alpha,p,q,k))$. The number of faulty links in $E_0(G(n,\alpha,p,q,k)) \setminus \{(v_1,v_2) \in EH(s,t) \mid v_1$ or v_2 is faulty} is denoted as $e_0(n,\alpha,p,q,k)$.

(*Theorem* 5)

In $GC(n,2^{\alpha})$, for all $T_\alpha(p)$ and $T_\alpha(q)$ which are neighbors in T_α, if $e_s(n,\alpha,p,q,k)$ + $e_0(n,\alpha,p,q,k)$ < $|Dim(p)|$ and $e_t(n,\alpha,p,q,k)$ + $e_0(n,\alpha,p,q,k)$ < $|Dim(q)|$, for all $k \in [0, 2^{n-\alpha-|Dim(p)|-|Dim(q)|}-1]$, then there is a fault-tolerant and cycle-free routing strategy for any source and destination pair.

Proof. (Outline)

The algorithm used in *Theorem* 3 fails only when links spanning in dimension [0, α − 1] are broken. With our discussion about the fault-tolerant routing in *Exchanged Cube*, such a problem is solved as long as the fault number satisfies the precondition of *Theorem* 5. ■

6. Simulation results

A software simulator is constructed to imitate the behavior of the real network, and thus test the performance of our algorithm. The assumptions are: (1) source and destination nodes must be nonfaulty, (2) Eager readership is employed where packet service rate is faster than packet arrival rate, (3) a faulty node makes all of its incident links faulty, (4) a node knows the status of its links to neighboring nodes and B or C category faults related to nodes which have the same least significant α bits as the node itself.

The performance of routing algorithm is measured by two metrics: average latency and throughput. Average latency is defined as *LP*/*DP*, where *LP* is the total latency of all packets that have reached destination while *DP* is the number of such packets. Throughput is defined as *DP*/*PT*, where *PT* is the total processing time taken by all nodes. We use its logarithm with base 2 for clearer comparison. Fig. 5 and Fig. 6 demonstrate the result of fault-free *GC* (*n*, *M*) (*n*∈[6, 14], *M*∈｛1, 2, 4｝).

From Figure 5, it can be observed that the average latency of Gaussian Cube increases as the networks dimension increases from 6 to 14. As the network size increases, the diameter of the hypercube also increases. A packet to be transmitted has to take a longer path to reach its destination, resulting in a higher average latency. Furthermore, as *M* increases, the network average latency also increases. This is due to the dilution of links with increasing *M*. The influence of *M* on the average latency is even more significant than network dimension.

In Fig. 6, it is demonstrated that the throughput of all networks is increasing as the dimension is increased from 6 to over 14. This is due to the parallelism of the networks and the increase in the number of nodes that can generate and route packets in the network, is faster than the time complexity of $O(\alpha(n-\alpha)\log\alpha)$ where $\alpha = \log M$. By increasing the network size, the number of link is also increasing at a higher rate than the node number. This in turn increases the total allowable packets in the network. With parallelism, more packets will reach destination in a given duration.

Figure 7 and 8 demonstrate the result for $GC(n,1)$ ($n \in [5,13]$), with the comparison between two situations: fault free and one faulty node in presence. It is clear that when the number of faults increases, the trend of average latency is to increase while the throughput is to decrease. This is because when more faults appear, the packet is more likely to use spare dimensions which makes the final route longer. Thus, the latency increases and throughput decreases.

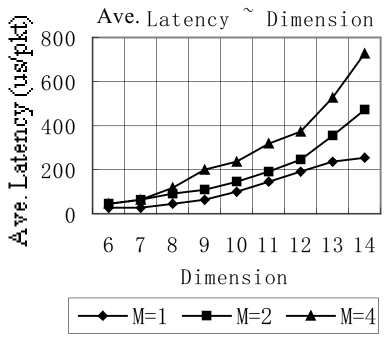

Fig. 5 Average. Latency versus Dimension

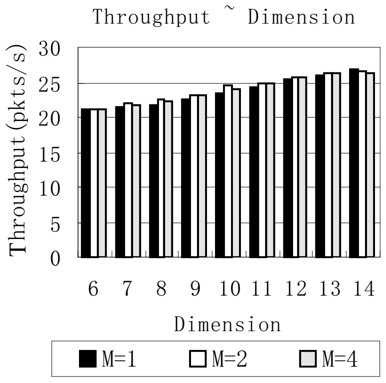

Fig 6. Throughput ~ Dimension

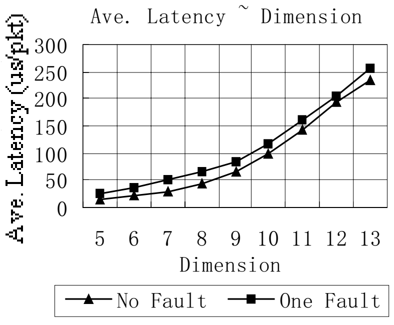

Fig. 7 Fault's influence on Ave. Latency

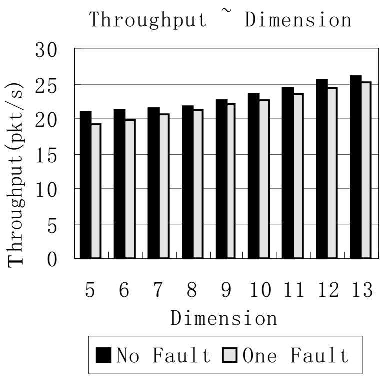

Fig. 8 Fault's influence on Throughput

7. Discussion and conclusion

In this paper, a new effective fault-tolerant routing strategy is proposed for Gaussian Cubes. Gaussian Tree is introduced to facilitate the algorithm and helps to make routing more definite. The routing strategy ensures livelock freeness and generates deadlock-free routes with the length no more than 2F longer than the optimal route found in the fault-free setting. The space and computation complexity as well as message overhead size are all reasonable. Although the Gaussian Cube is very sparsely connected, which partially caused the current non-existence of fault tolerant routing strategies for it, our algorithm can still tolerate a satisfactory number of faults, with careful analysis of their location and influence.

Some of our results can still be improved upon. For example, although the idea of categorizing the faulty components is useful in Gaussian Cube and possibly in other node/link diluted cubes, the exact way of categorization might vary due to different topologies. A new unified metric needs to be designed to measure the fault-tolerance ability of interconnection networks so that it is fair despite their different routing algorithms and different methods of fault categorization.

8. References

[1] Hsu, W. J., Chung, M. J., and Hu, Z., "Gaussian Networks For Scalable Distributed Systems", *The Computer Journal*, Vol. 39, No. 5, 1996 , pp 417-426.

[2] Hsu, W. J., Chung, M. J. and Hu, Z. "A New Gaussian networks and Their Applications" *Int'l Symp. Parallel and Distributed Supercomputing*, Japan, 1995.

[3] Douglas B. West, "*Introduction to Graph Theory* - Second edition" Chapter 2 N.J.: Prentice Hall, 2001.

[4] Peter K. K. Loh, H. Schröder, W. J. Hsu, "Fault-tolerant routing on complete Josephus Cubes". *Proc. 6th Australian Conf. Computer systems architecture*, IEEE Computer Society Press. Queensland, Australia, 2001, pp. 95-104.

[5] Wu, J., "Reliable Unicasting in Faulty Hypercubes Using Safety Levels", *IEEE Transactions on Computers*, Vol. 46, No. 2, February 1997 , pp 241-247.

[6] Lan, Youran, "An Adaptive Fault-Tolerant Routing Algorithm for Hypercube Multicomputers", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, No. 11, November 1995 , pp 1147-1152.

[7] D. P. Bertsekas and J. N. Tsitsiklis, "*Parallel and Distributed Computation: Numerical Methods*". Englewood Cliffs, NJ: Prentice-Hall, 1989, ch. 1, pp. 27–68.