Adaptive Box-Based Efficient Fault-Tolerant Routing in 3D Torus

Yamin Li, Shietung Peng
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan

Wanming Chu
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan

Abstract

In this paper, we propose efficient fault-tolerant routing algorithms for 3D torus with possible large number of faulty nodes. There is no any presumption on the number and the distribution of faulty nodes. The proposed algorithms find a fault-free path between any two nonfaulty nodes with high probability in linear time by using only the local faulty information of the network. The results of our empirical analysis through simulations show that the algorithms can find a fault-free path between any two nonfaulty nodes with a probability higher than 90% in a 3D torus with the number of faulty nodes up to 30%.

1. Introduction

The three dimensional (3D) mesh/torus has constant node degree, recursive structure, simple communication algorithms, and good scalability. Due to these attractive properties, the mesh/torus has been the common interconnection network for several commercially available parallel computers, such as Cray XT3 [9] and IBM Blue Gene/L (BG/L)[2].

It is foreseeable that 3D mesh or torus architectures will become viable alternatives for parallel computer design because to implement a 3D mesh or torus in the 3D physical space is becoming possible as the IC technology advances. In IBM BG/L's case, the network is integrated onto the same chip that does computing, that is, no separate switch is required.

A 3D mesh can be laid out on a 3D chip in an area that increases linearly with the number of processors. Since the implementation of 3D mesh uses short, local links only, it is possible to perform communication at very high speed. A 3D torus has wraparound links. However,the method of folding can be used to lay out a 3D torus in such a way that it uses only short, local links too.

The 3D torus is an interconnection network of great potential due to its high bandwidth nearest neighbor connectivity for efficient computation and fast communication in many scientific applications. In this paper, we focus our designs on 3D torus. However, the design is easily to extend to kD torus for any $k > 3$.

Fault-tolerant routing is a dominant issue facing the design of interconnection networks for large-scale parallel computers. There are many fault models used for designing fault-tolerant routing algorithms [1, 3, 4, 7, 8, 10, 11, 12]. Some set conditions on the number of faulty nodes or the shape of faulty components. Others use global fault information (off-line) or partially global information. Chen et. al [5, 6] introduced the concept of local-subcube connectivity for hypercubes. In this paper, we develop fault-tolerant routing algorithms on 3D torus using local information only (on-line), and allow arbitrary number of faulty nodes with no restriction on the shape of the faulty nodes (blocks).

The rest of this paper is organized as follows: In the next section, we give necessary notation and definitions used throughout the paper. We also show a theorem that is a theoretical ground of the proposed routing algorithms. In Sections 3, 4, and 5, respectively, three fault-tolerant routing algorithms are proposed on 3D torus with possible large number of faulty nodes. In Section 6, simulations are performed and the results are analyzed and discussed. Finally, in the last section, we conclude this paper with some remarks.

2. Locally-Safe Torus

A kD n-torus T_n^k has k dimensions, n nodes per dimension, and $N = n^k$ nodes. Each node is uniquely indexed by a radix-n k-tuple. Each node is connected via communication links to two other nodes in each dimension. The neighbors of node $s = (s_0, \ldots, s_{k-1})$ in dimension i are $(s_0, \ldots, s_{i-1}, s_i \pm 1, s_{i+1}, \ldots, s_{k-1})$, where addition and subtraction are performed modulo n. For simplicity, throughout this paper, all arithmetics on the indices of nodes in a given torus should be modulo n implicitly. The distance between two nodes s and t in T_n^k is $d(s,t) = \sum_{i=0}^{k-1} \min(|s_i - t_i|, n - |s_i - t_i|)$. In this paper, we work on 3D torus only. For simplicity, we use term

T, instead of T_n^3, to denote a 3D n-torus if no confusion arises.

For a given node $s = (s_0, s_1, s_2)$ in T, we denote its two neighbors in dimension i by s^{i+} and s^{i-}, respectively. For example, $s^{0+} = (s_0 + 1, s_1, s_2)$ and $s^{0-} = (s_0 - 1, s_1, s_2)$. A 3D m-mesh M_m^3, or simply M, in a 3D torus T is a subgraph of T, and M is a 3D mesh of size m (m nodes in each of the three dimensions).

We say that T is *locally-m-safe* if the following conditions are satisfied.

1. for every 3D m-mesh M in T, the subgraph formed by all nonfaulty nodes in M is connected, and
2. every boundary B of M (m-square) contains at least one nonfaulty node.

We say that T is *locally-safe* if there exists an integer $m,\ 2 \leq m \leq n$, such that T is locally-m-safe. The following theorem shows that local-safety implies connectedness of T.

Theorem 1. *If a 3D torus T is locally-safe then T is a connected graph.*

Proof Let $s = (s_0, s_1, s_2)$ and $t = (t_0, t_1, t_2)$ are two nonfaulty nodes in T. Without loss of generality, we assume that $s_i < t_i$ for $0 \leq i \leq 2$. We also assume that s^{i+} and s^{i-} be the neighbors of s on dimension i such that s^{i+} is the neighbor of s closer to t ($d(s^{i+}, t) = d(s, t) - 1$). Since T is locally-safe there exists an integer $m,\ 2 \leq m \leq n$, such that T is locally-m-safe. Let $P = (s \rightarrow t)$ be the shortest path from s to t constructed by the dimension-order routing (routing along dimension 0, then dimension 1, and then dimension 2). We construct a tube with 2 turns (3 pipes) such that path P is enclosed in the tube as shown as in Figure 1.

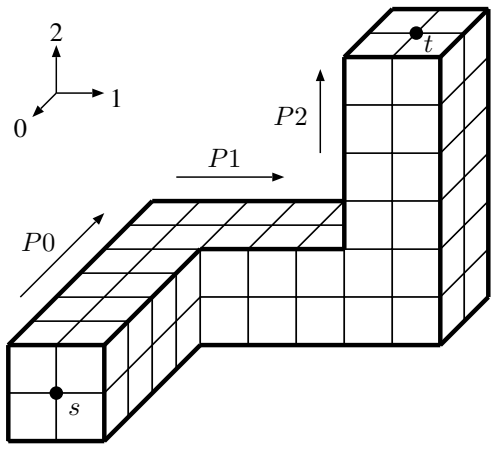

Figure 1. Routing in tubes

Let $P = P_0 \cup P_1 \cup P_2$, where $P_0,\ P_1$, and P_2 are the segments $s \rightarrow u$, $u \rightarrow w$, and $w \rightarrow t$ along 0, 1, and 2 dimensions, respectively, where $u = (u_0, u_1, u_2) = (t_0, s_1, s_2)$ and $w = (w_0, w_1, w_2) = (t_0, t_1, s_2)$. The tube has three parts $pipe_0,\ pipe_1$, and $pipe_2$ defined as follows:

- $pipe_0 = \{v \in T \mid s_0 \leq v_0 \leq t_0$, and $s_i - \lfloor m/2 \rfloor \leq v_i \leq s_i + \lceil m/2 \rceil - 1$, for $i = 1$ and $2\}$
- $pipe_1 = \{v \in T \mid u_1 - \lfloor m/2 \rfloor \leq v_1 \leq w_1$ and $u_i - \lfloor m/2 \rfloor \leq v_i \leq u_i + \lceil m/2 \rceil - 1$, for $i = 0$ and $2\}$
- $pipe_2 = \{v \in T \mid w_2 - \lfloor m/2 \rfloor \leq v_2 \leq t_2$ and $w_i - \lfloor m/2 \rfloor \leq v_i \leq w_i + \lceil m/2 \rceil - 1$, for $i = 0$ and $1\}$

Consider $pipe_0$ as a sequence of 3D m-meshes $M_i, 0 \leq i \leq p$, such that

1. $pipe_0 \subset \cup_{i=0}^{p} M_i$ and $B_i = M_{i-1} \cap M_i,\ 1 \leq i \leq p$, are 2D m-meshes;
2. $s \in M_0$ and $u \in M_p$.

Let $B_u = \{v \in pipe_0 | v_0 = u_0\}$. Obviously, we have $B_u \subset M_p$. Since T is locally-m-safe, there exist nonfaulty nodes $v^i \in B_i,\ 1 \leq i \leq p$, nonfaulty node $u' \in B_u$, and fault-free paths: $(s \rightarrow v^1) \subset M_0, (v^1 \rightarrow v^2) \subset M_1, \ldots, (v^p \rightarrow u') \subset M_p$. Then, the path $(s \rightarrow v^1 \rightarrow v^2 \ldots \rightarrow v^p \rightarrow u')$ is the fault-free path from s to u'. From the definition of $pipe_1$, we have $u' \in pipe_1$. Let $w' \in pipe_1$ be a nonfaulty node such that $w'_1 = w_1 = t_1$. By the similar argument, we can find nonfaulty node $w' \in B_w = \{v \in pipe_1 | v_1 = w_1\}$ and a fault-free path in $pipe_1$ from u' to w'; and a fault-free path in $pipe_2$ from w' to t. Therefore, s and t can be connected through the fault-free path $s \rightarrow u' \rightarrow w' \rightarrow t$. We conclude that T is connected. □

Corollary 1. *If a 3D torus T is locally-m-safe then T is also locally-l-safe for all $l \geq m$.*

The proof of the theorem is a constructive one. It provides the ground for our first fault-tolerant routing algorithm to be presented at next section.

3. Tube-Routing Algorithm

For practice, we do not presume that T is locally-safe. The number of faulty nodes or its distribution is arbitrary. Our routing algorithms are local-information-based: no global information about the situation of the network is needed. If T is locally-m-safe then from theorem 1, the algorithm will generate a fault-free path. Otherwise, it will either generate a fault-free path or report a failure.

The algorithm follows the constructive proof of theorem 1. A tube that encloses the shortest path P is used for the fault-tolerant routing. While routing from source s to destination t, the path is allowed to move inside the tube

through a sequence of boxes as specified in the proof of theorem 1. However, for higher successful routing rate, we construct the $pipe_0$ such that, for any $v \in pipe_0$, $d(v_i, t_i) \leq d(s_i, t_i)+1,\ i = 1, 2$. This is done by shifting $pipe_0$ toward the destination node t along dimensions 1 and 2 such that one of the farthest corner nodes b on the boundary B_s will be $(s^{1-})^{2-}$, where s^{i-} is the neighbor of s along diemnsion i that is farther away from t. Notice that P_0 is still enclosed insde the shifted $pipe_0$.

The $pipe_1$ and $pipe_2$ are constructed similarly. The details are specified in Algorithm 1. We call this algorithm Tube_Routing (see Algorithm 1). The rate of successful routing of the algorithm will be analyzed empirically and the simulation results will be used to compare with that of the other routing algorithms proposed in this paper.

Algorithm 1 (Tube_Routing(T_n, m, s, t))
Input: 3D n-torus T_n, size of local mesh m, source node $s = (s_0, s_1, s_2)$, and destination node $t = (t_0, t_1, t_2)$
Output: a fault-free path $P = (s \rightarrow t)$ or report failure
begin
  $P = \phi$;
  $r = s$;
  $dir_0 = dir_1 = dir_2 = 1$; /* determine routing direction */
  if $(0 \leq r_0 - t_0 \leq n/2)$ OR $(0 \leq t_0 - r_0 > n/2)$ $dir_0 = -1$;
  if $(0 \leq r_1 - t_1 \leq n/2)$ OR $(0 \leq t_1 - r_1 > n/2)$ $dir_1 = -1$;
  if $(0 \leq r_2 - t_2 \leq n/2)$ OR $(0 \leq t_2 - r_2 > n/2)$ $dir_2 = -1$;
  for $i = 0, 1, 2$ **do** /* for each dimension i */
    /* determine *mesh* boundaries of i and j dimensions */
    $j = (i + 1)$ mod 3; /* dimension j */
    $k = (j + 1)$ mod 3; /* dimension k */
    $b_j = (r_j - dir_j + n)$ mod n; /* $[b_j, B_j]$ in dimension j */
    $b_k = (r_k - dir_k + n)$ mod n; /* $[b_k, B_k]$ in dimension k */
    $B_j = (r_j + (m - 2) \times dir_j + n)$ mod n;
    $B_k = (r_k + (m - 2) \times dir_k + n)$ mod n;
    while $r_i \neq t_i$ **do**
      /* determine *mesh* boundaries of i dimension */
      $b_i = r_i$; /* $[b_i, B_i]$ in dimension i */
      $B_i = (r_i + (m - 1) \times dir_i + n)$ mod n;
      if t is in *mesh*
        if there is a fault-free path $P' = (r \rightarrow t)$ in *mesh*
          $P = P \cup P'$;
          return P; /* path constructed */
        else return failure; /* failed */
      else
        if there is a fault-free path $P' = (r \rightarrow r')$ in *mesh* such that $(r'_i = B_i)$ OR $(r'_i = t_i)$
          $P = P \cup P'$;
          $r = r'$; /* continue */
        else return failure; /* failed */
    endwhile
  endfor
end

In Algorithm 1, we route source node $s = (s_0, s_1, s_2)$ to destination node $t = (t_0, t_1, t_2)$ through a tube of 3 pipes. A pipe consists of a sequence of boxes (3D m-meshes). A box is uniquely determined with two nodes: b and B.

If routing in the first pipe succeeds, a node $r = (t_0, r_1, r_2)$ will be reached where r_1 and r_2 are in the m-square bounded by b_0, b_1, B_0, and B_1. The path $(s \rightarrow r)$ may goes through several boxes. To route in a box, we can use any search algorithm, depth-first search (DFS) or breadth-first search (BFS) algorithm for instance. If all three dimensions are routed successfully, a fault-free path $(s \rightarrow t)$ is found. Whenever the local routing in a box fails, the algorithm reports a failure and terminates.

The running time of the algorithm is $O(n)$, assuming that the local routing inside a small local 3D m-mesh takes constant time. We summarize this result into the following theorem.

Theorem 2. *The Tube_Routing algorithm will terminate in $O(n)$ time. When the algorithm terminates, it either generates a fault-free path from s to t or reports that the path cannot be found.*

4. Adaptive Box Routing Algorithm

In this section, we describe another local-information-based, fault-tolerant routing algorithm, called Adaptive_Box Routing. The idea is as follows. Instead of arranging the sequence of m-meshes as a tube that encloses the shortest path generated by the dimension-order routing, a sequence of boxes is found recursively such that the routing direction of each box should be along the dimension i such that the distance $d(r_i, t_i)$ is a maximum, where r is the new source node after a local routing. Intuitively, the sequence of boxes is arranged to enclose a ladder-shaped shortest path, instead of a dimension-order shortest path. Each box should also adapt itself such that the new source node r and the path segment along dimension i should be enclosed inside the box.

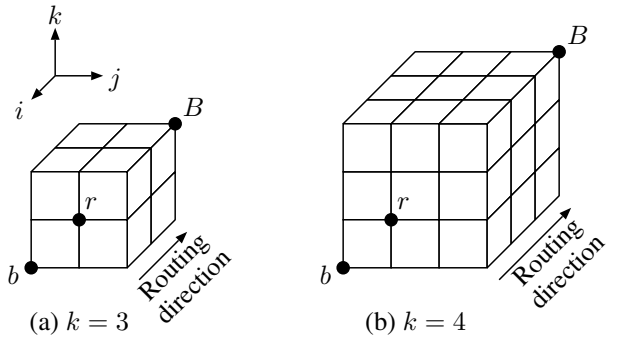

Figure 2. Routing with boxes

To describe the algorithm, we need a notation to show the position of an m-mesh inside T. Referring to Figure 2, we associate each node r in T with a unique 3D m-mesh M_r

to be used by the algorithm. The M_r is determined by two nodes b and B, representing the lower-corner node b and an upper-corner node B of the two 2D meshes along the dimension of the current pipe, respectively.

More precisely, let i be the dimension of the current pipe, and dir_j and dir_k be the unit directions (+1 or -1) of the shortest path along dimensions j and k. Then, we have $b_i = r_i,\ b_j = r_j - dir_j$, and $b_k = r_k - dir_k$, and $B_i = r_i - (m-1) \times dir_i, B_j = r_j - (m-2) \times dir_j$, and $B_k = r_k - (m-2) \times dir_k$ (all arithmetics are modulo n).

The proposed recursive algorithm is similar to that of the tube approach. The difference is that the box is adaptable in all dimensions instead of just in the dimension of the current pipe. Let $r = s$. The algorithm first determines the routing dimension i and the local 3D m-mesh M_r, and then performs local routing in M_r that routes node r to a nonfaulty node r', a node located at the opposite boundary of M_r from r along the ith dimension. If the local routing $r \rightarrow r'$ successes then we consider r' as a new source r and route from r recursively. If the local routing fails, the algorithm terminates unsuccessfully and reports a failure. The algorithm is formally specified as in Algorithm 2.

Next, we show that the algorithm will terminate in $O(n)$ time, and either finds a fault-free path from s to t or reports a failure. When $m > 3$, we show in the following theorem that the algorithm will terminate in $O(n)$ time. However, for $m = 3$, the node r becomes the center of the 2D m-mesh it locates. It is possible that the local routing within the box will not make progress. For example, if $r' = (r_i + 2 \times dir_i, r_j - dir_j, r_k - dir_k)$ then $d(r,t) = d(r',t)$ and the algorithm does not make any progress. Therefore, the proposed routing algorithm should provide a checking for such kind of local routing. We allow at most three consecutive non-progress steps in the algorithm for $m = 3$.

Theorem 3. *Adaptive_Box_Routing algorithm terminates in $O(n)$ time, and either outputs a fault-free path from source s to destination t or reports a failure.*

Proof We first show that, for $m > 3$, the local routing always makes progress toward destination t. Since the box M_r for node r is constructed in the way that the farthest corner node B is toward t. That is, $B = (r_i + (m-1) \times dir_i, r_j + (m-2) \times dir_j, r_k + (m-2) \times dir_k)$. The worst case is that r is routed to $r' = (r_i + (m-1) \times dir_i, r_j - dir_j, r_k - dir_k)$. Since $d(r,t) - d(r',t) = (m-1) + 2 > 0$ for $m > 3$, the local routing always makes progress toward t. For $m = 3$, we restrict the consecutive local routings that do not make progress at most three times. Therefore, the algorithm will terminate with at most $3d(s,t) = O(n)$ local routings. For fixed m, the running time for the local routing is a constant. Therefore, the total running time of the algorithm is $O(n)$. □

Algorithm 2 (Adaptive_Box_Routing(T_n, m, s, t))
Input: 3D n-torus T_n, size of local mesh m, source node $s = (s_0, s_1, s_2)$, and destination node $t = (t_0, t_1, t_2)$
Output: a fault-free path $P = (s \rightarrow t)$ or report failure
begin
  $P = \phi$;
  $r = s$;
  $count = 0$;
  while $r \neq t$ **do**
    $dir_0 = dir_1 = dir_2 = 1$; /* determine routing direction */
    if $(0 \leq r_0 - t_0 \leq n/2)$ OR $(0 \leq t_0 - r_0 > n/2)$ $dir_0 = -1$;
    if $(0 \leq r_1 - t_1 \leq n/2)$ OR $(0 \leq t_1 - r_1 > n/2)$ $dir_1 = -1$;
    if $(0 \leq r_2 - t_2 \leq n/2)$ OR $(0 \leq t_2 - r_2 > n/2)$ $dir_2 = -1$;
    Find the dimension i so that the distance between r_i and t_i $d(r_i, t_i) = \max(d(r_0, t_0), d(r_1, t_1), d(r_2, t_2))$;
    /* determine *mesh* boundary $[b, B]$, referring to Figure 2 */
    $j = (i+1) \bmod 3$; /* dimension j */
    $k = (j+1) \bmod 3$; /* dimension k */
    $b_i = r_i$; /* $[b_i, B_i]$ in dimension i */
    $b_j = (r_j - dir_j + n) \bmod n$; /* $[b_j, B_j]$ in dimension j */
    $b_k = (r_k - dir_k + n) \bmod n$; /* $[b_k, B_k]$ in dimension k */
    $B_i = (r_i + (m-1) \times dir_i + n) \bmod n$;
    $B_j = (r_j + (m-2) \times dir_j + n) \bmod n$;
    $B_k = (r_k + (m-2) \times dir_k + n) \bmod n$;
    if t is in *mesh* bounded by $[b, B]$
      if there is a fault-free path $P' = (r \rightarrow t)$ in *mesh*
        $P = P \cup P'$;
        return P; /* path constructed */
      else return failure; /* failed */
    else
      if ($count \leq 3$ and there is a fault-free path $P' = (r \rightarrow r')$ in *mesh* such that $(r'_i = B_i)$ OR $(r'_i = t_i)$)
        $P = P \cup P'$;
        if $(d(r,t) = d(r',t))$
          $count = count + 1$;
        else $count = 0$;
        $r = r'$; /* continue */
      else return failure; /* failed */
  endwhile
end

---

5. Heuristic Box Routing Algorithm

In this section, we make an effort to improve the performance of Adaptive_Box_Routing algorithm by allowing the routing to continue when the local routing along the dimension of the longest distance, say i, fails.

In the new algorithm, the routing continues by trying the local routings in the boxes set along the other two dimensions (the dimension with larger distance goes first) under the condition that the distance between r and t along that dimension is greater than 1. If the distance between r and t along the chosen dimension, say j, $d(r_j, t_j) < m - 1$ then the local routing will route r to r', where $d(r_j, r'_j) = d(r_j, t_j) > 1$.

Once the local routing along the dimension j that $d(r_j, t_j)$ is not an maximum successes, the box for the new source should be arranged along dimension i again. The algorithm that adds this heuristic strategy to the Adaptive_Box Routing is called Heuristic_Box routing. In the next theorem, we show that Heuristic_Box routing works.

Theorem 4. *Heuristic_Box_Routing algorithm terminates in $O(n)$ time, and either outputs a fault-free path from s to t or reports a failure.*

Proof Similar to that in theorem 3, we show that the number of the additional local routings is at most $O(n)$. Starting from $r = s$, there are two cases:

1. $d(r_j, t_j) > 2$: in this case, the algorithm always makes progress as it has been shown in theorem 3.
2. $d(r_j, t_j) = 2$: in this case, r' might not make any progress, that is, $d(r', t) = d(r, t)$ (the distance toward t along other two dimensions $\neq j$ goes one step backward within the box). However, routing along dimension j is done at most once in the algorithm for each source node. The $O(n)$ bound is still valid in this case.

In any case, the number of local routings is $O(n)$. For fixed m, the running time for the local routing is a constant. Therefore, the running time of the algorithm is $O(n)$. □

6. Simulation Results

We have performed a set of simulations on the performance of the proposed algorithms: Tube_Routing, Adaptive_Box_Routing, and Heuristic_Box_Routing. We focused on the successful routing rates of the algorithms for the 3D torus networks with different node faulty probabilities. The comparison of the successful routing rates of the algorithms are performed. We also presented the path length for the Adaptive_Box_Routing algorithm.

We have tested for the dimensions $n = 12, 16$ and 20. For each dimension, we have tested the m-mesh-connectivity for $m = 3$ and 4. For the fault model, we use uniform distribution of node failures. That is, each node has an equal and independent failure probability p_f. We let p_f vary from 10% to 50% with a 10% increment. For each case, we simulate 10,000 times.

The simulation results of the successful routing rate and the length of the routing path for the set of parameters specified above are shown in the figures below. Because there are too many curves if we put all results in a single figure, we plotted them in separated figures.

Figure 3 plots the successful routing rate of the simplest algorithm, Tube_Routing, where the two numbers in brackets are n and m, respectively. We can see that, when the node failure probability is small, i.e., $p_f = 10\%$, the successful routing rates are almost 100% for both $m = 3$ and 4. For a given n, increasing m improves the successful routing. On the other hand, for a given m, routing in a small torus has higher successful routing rate than that in a large torus.

Figure 4 shows the successful routing rate of the Adaptive_Box_Routing algorithm. From the figure, we can see that, when the node failure probability $p_f \leq 20\%$, the

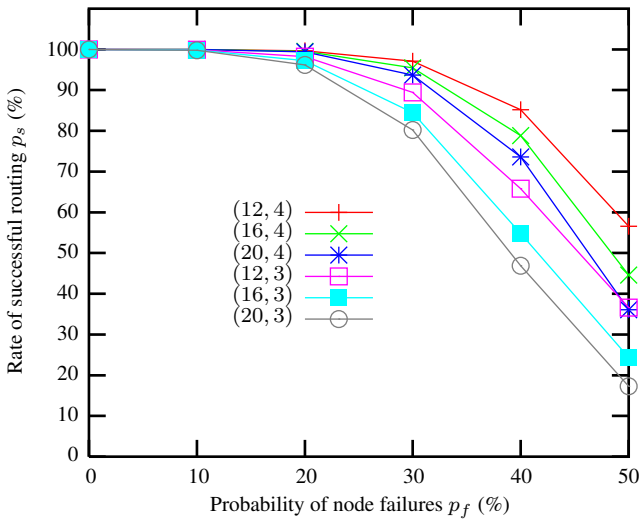

Figure 3. The successful routing rate of Tube_Routing

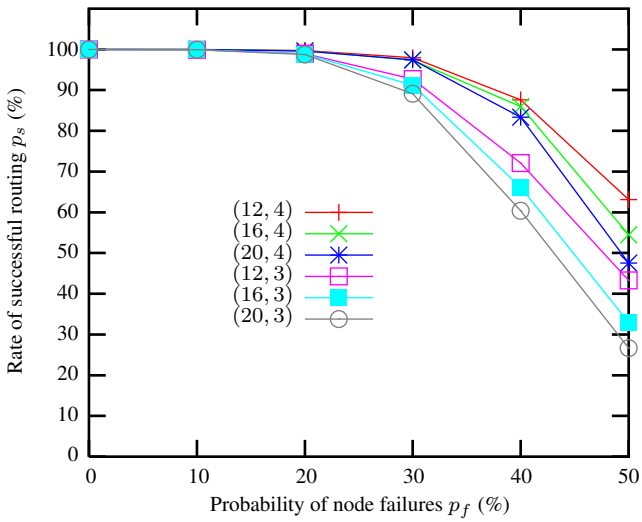

Figure 4. The successful routing rate of Adaptive_Box_Routing

successful routing rates are almost 100% for both $m = 3$ and 4. Figure 5 shows the performance improvement of the adaptive-box algorithm compared to that of the tube algorithm. From the figure, we conclude that the adaptive-box approach are better than the tube approach in most of the cases.

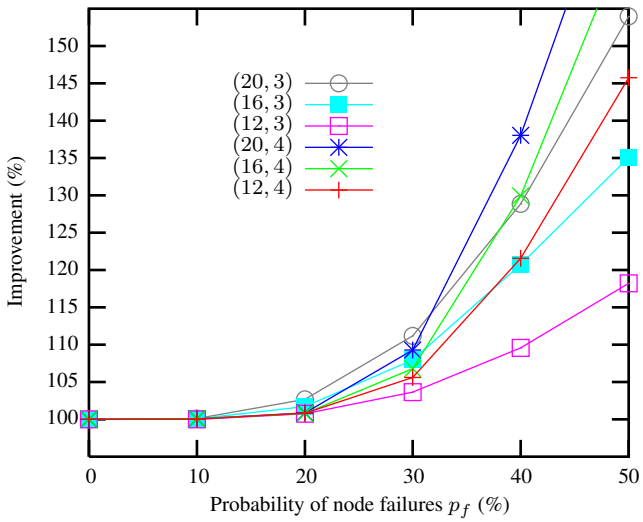

Figure 5. The successful improvement rate of Adaptive_Box_Routing v.s. Tube_Routing

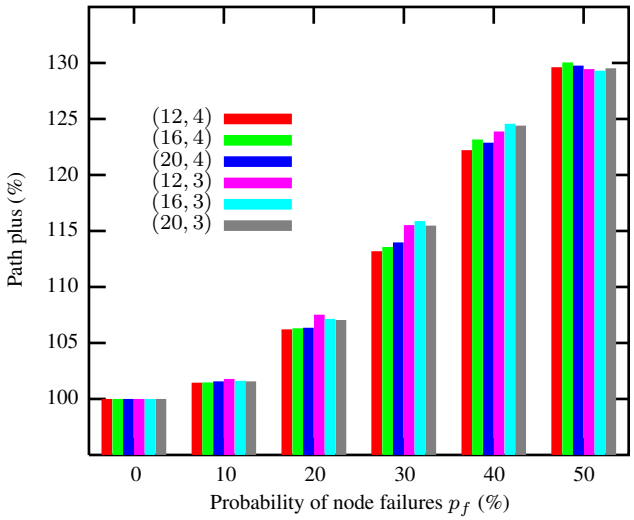

Figure 6. Path plus of Adaptive_Box_Routing

The successful improvement rate in Figure 5 is defined as

$$\frac{\text{Successful routing rate of box algorithm}}{\text{Successful routing rate of tube algorithm}}$$

When n is small and m is large, $n = 12$ and $m = 4$ for instance, the box algorithm is not much better than the tube algorithm. However, when m is smaller, especially as n increases, the successful improvement rate of the box algorithm grows obviously. As an example, when $m = 3$, $n = 20$, and there are half faulty nodes, the successful improvement rate is 1.54. For fixed n, using a larger box will have a better performance with a cost of increasing time complexity at a rate that is a cubic function of m.

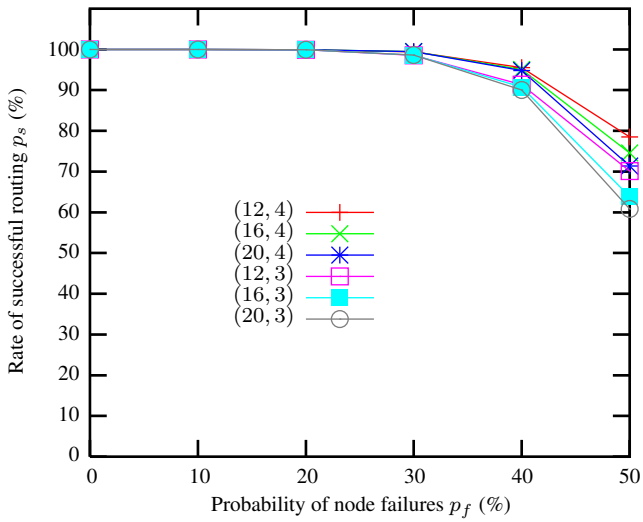

Figure 7. The successful routing rate of Heuristic_Box_Routing

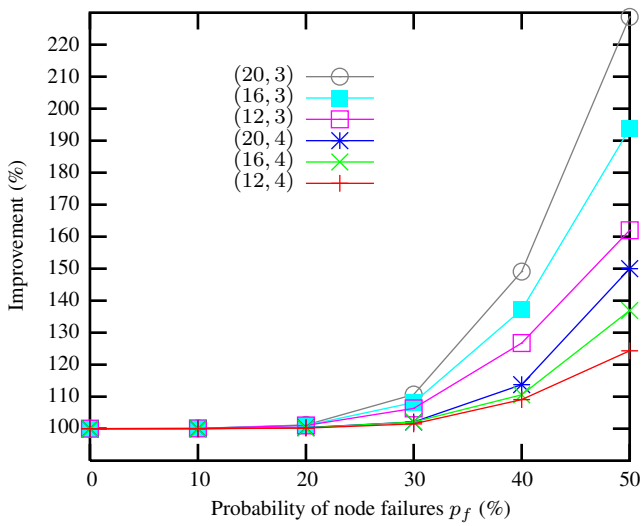

Figure 8. The successful improvement rate of Heuristic_Box_Routing v.s. Adaptive_Box_Routing

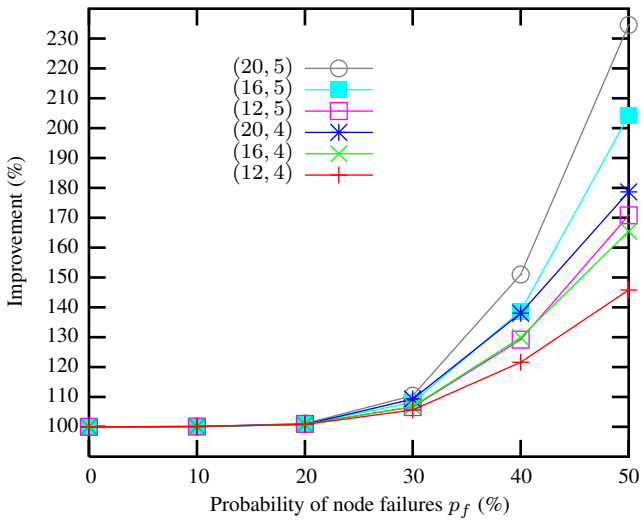

Figure 9. The successful improvement rates of *m* = 4 and 5 v.s. *m* = 3 for Adaptive_Box_Routing

Figure 6 displays the path plus of the box algorithm, which is calculated by

$$\text{Path plus} = \frac{\text{Path length of } P = (s \rightarrow t)}{\text{Distance between } s \text{ and } t}$$

Figure 7 plots the successful routing rate of the heuristic box routing algorithms. From the figure, we can see that, when the node failure probability $p_f \leq 30\%$, the successful routing rates are almost 100% for both $m = 3$ and 4. Figure 8 depicts the improvement of successful routing rate gained by using the heuristic box routing algorithms to that of box algorithm. The improvement is significant, especially when m is small. As an example, when $m = 3$, $n = 20$, and there are half faulty nodes, the successful improvement rate is 2.34. It is worth to adopt those routing algorithms when the probability of faulty nodes is high.

Figure 9 shows the successful improvement rate of $m = 4$ and 5 against $m = 3$, for Adaptive_Box_Routing. For practical reason, we consider small m only as we assume that the time of routing within an m-mesh is a constant.

7. Conclusions

In this paper, we first presented a concept of local-safety for a kD n-torus. Then, we proposed two different approaches for fault-tolerant routing in a 3D n-torus with possible large and arbitrarily faulty nodes. The algorithms are online (only local information is used) and efficient ($O(n)$ time assuming that the local routing is $O(1)$). The simulation results show that the rates of successful routing of the algorithms are quite high considering that there are only six links for each node in a 3D torus. The possible directions of the further research include

1. Analyze the performance of the proposed algorithms theoretically;
2. Apply the similar approaches to other practical interconnection networks; and
3. Investigate the important issues (e.g., deadlock-free) while implement the proposed routing algorithms on some switching models.

References

[1] L. D. Aronson. Homogeneous routing for homogeneous traffic patterns on meshes. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):781–793, August 2000.

[2] M. Blumrich, D. Chen, P. Coteus, A. Gara, M. Giampapa, P. Heidelberger, S. Singh, B. Steinmacher-Burow, T. Takken, and P. Vranas. Design and analysis of the bluegene/l torus interconnection network. *IBM Research Report, http://www.research.ibm.com/bluegene/*, December 2003.

[3] R. V. Boppana and S. Chalasani. Fault-tolerant wormhole routing algorithms for mesh networks. *IEEE Transactions on Computers*, 44(7):848–864, July 1995.

[4] S. Chalasani and R. V. Boppana. Fault-tolerant wormhole routing in tori. *IEE Proc.: Computers and Digital Techniques*, 142(11):386–394, Nov 1995.

[5] J. Chen, G. Wang, and S. Chen. Routing in hypercube networks with a constant fraction of faulty nodes. *Journal of Interconnection Networks*, 2(3):283–294, September 2001.

[6] J. Chen, G. Wang, and S. Chen. Locally subcube-connected hypercube networks: Theoretical analysis and experimental results. *IEEE Transactions on Computers*, 51(5):530–540, May 2002.

[7] Q.-P. Gu and S. Peng. Fault tolerant routing in toroidal networks. *IEICE Transactions on Information and Systems*, E-79D:1153–1159, August 1996.

[8] Q.-P. Gu and S. Peng. Unicast in hypercubes with large number of faulty nodes. *IEEE Transactions on Parallel and Distributed Systems*, 10:964–975, October 1999.

[9] C. Inc. Cray xt3 supercomputer. *http://www.cray.com/products/xt3/index.html*, 2004.

[10] M.-J. Tsai and S.-D. Wang. Adaptive and deadlock-free routing for irregular faulty patterns in mesh multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, 11(1):50–63, January 2000.

[11] J. Wu. Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety level. *IEEE Transactions on Parallel and Distributed Systems*, 11(2):149–159, February 2000.

[12] D. Xiang. Fault-tolerant routing in hypercube multicomputers using local safety information. *IEEE Transactions on Parallel and Distributed Systems*, 12(9):942–951, September 2001.