Fault-Tolerant Routing in Meshes/Tori Using Planarly Constructed Fault Blocks

Dong Xiang, Jia-Guang Sun, Jie Wu, and Krishnaiyan Thulasiraman

Abstract

A few faulty nodes can make an n-dimensional mesh or torus network unsafe for fault-tolerant routing methods based on the block fault model, where the whole system ($n-$dimensional space) forms a fault block. A new concept, called extended local safety information in meshes or tori, is proposed to guide fault-tolerant routing, and classifies fault-free nodes inside 2$-$dimensional planes. Many nodes globally marked as unsafe become locally enabled inside 2$-$dimensional planes. A fault-tolerant routing algorithm based on extended local safety information is proposed for k$-$ary n$-$dimensional meshes/tori. Our method does not need to disable any fault-free nodes, unlike many previous methods, and this enhances the computational power of the system and improves performance of the routing algorithm greatly. All fault blocks are constructed inside 2-dimensional planes rather than in the whole system. Extensive simulation results are presented and compared with the previous methods.

Index Terms: Computational power, fault-tolerant, routing, extended local safety, unsafe systems, mesh/torus.

1 Introduction

Torus and mesh-connected networks have been widely used in recent experimental or commercial multicomputers [1]. The performance of such multicomputers is highly dependent on the node-to-node communication cost. It is necessary to present an effective fault-tolerant routing algorithm in a mesh/torus.

Dong Xiang and Jia-Guang Sun are with the School of Software, Tsinghua University, Beijing 100084, P. R. China. Jie Wu is with the Department of Computer Science and Engineering, Florida Atlantic University, Baca Raton, FL 33431, USA. Krishnaiyan Thulasiraman is with the School of Computer Science, the University of Oklahoma, Norman, OK 73019, USA. Work in this paper is supported in part by the NSF of China under grants 60373009 and 60425203 and the NSF of USA under grants CNS0422762 and CNS0434533.

The block fault model is the most popular fault model. Some fault-free nodes must be disabled when faults are arbitrarily shaped to form fault blocks. A fault-free node is marked unsafe according to previous methods [2,4,6,15,16] if it has two faulty or unsafe neighbors along different dimensions. Since an unsafe node is disabled, a few faulty nodes can disable a large number of fault-free nodes or even disable all fault-free nodes based on the block fault model, especially for higher dimensional meshes/tori. As shown in Fig. 1, only 7 faulty nodes make all fault-free nodes disabled in the 5x5x5 mesh. All previous methods [2,4,6,15,16] can handle only the case in which both the source and destination are outside of a fault block.

The proposed method constructs fault blocks inside 2D planes, where many unsafe nodes become safe in separate planes. All resources related to the disabled nodes can still be used to route a message. Each fault-free node keeps its status in separate planes based on the safety measure to be introduced in this paper. Safety information kept in each fault-free node is about three times as that of the extended safety levels [16] in a 3D mesh/torus and n times in an n-dimensional mesh/torus because each fault-free node needs to keep its safety information inside $((n-1)\cdot n/2)$ 2-dimensional planes and the proposed safety measure *in* the whole system.

Note that fault blocks may be conjointed. Connected fault blocks may not maintain the convexity property. Therefore, a message may have to be routed around a non-convex fault block that may result in substantial backtracks for the whole message in a wormhole-routed network. Backtracking the whole message in a network can greatly influence performance of a fault-tolerant routing algorithm, especially when the network contains enough faulty nodes or the load of the system is large enough. This also indicates the necessity to construct fault blocks planarly and establish a path before sending *a* message, as in pipelined-circuit-switching [9].

The main work of this paper is: (1) A new limited-global safety-based measure called the extended local safety information is proposed to guide fault-tolerant

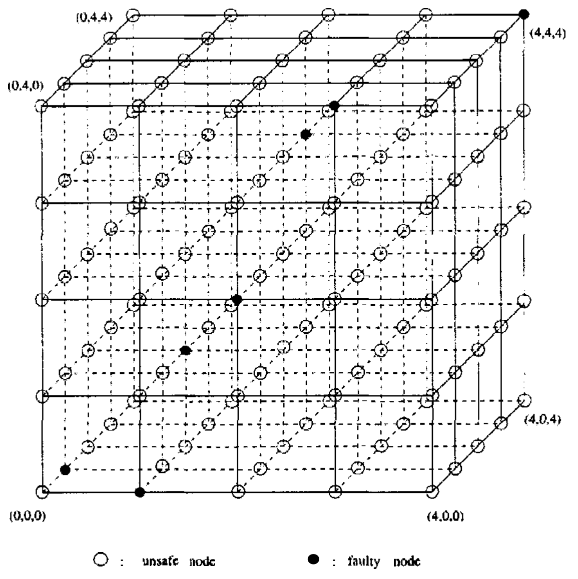

Figure 1: An unsafe 5−ary **3D** mesh.

routing, based on which a new path set-up scheme is proposed. **(2)** Fault blocks are constructed inside separate planes, where many unsafe nodes can be activated. This can significantly improve the computational power of the system and enhance the performance of the fault-tolerant routing algorithm greatly. **(3)** A new fault-tolerant routing algorithm based on the extended local safety information is presented with a new virtual subnetwork partitioning scheme to avoid deadlocks.

The number of virtual channels required by the proposed method is linearly proportional to the number of dimensions of the network. This number is acceptable for a practical mesh/torus network according to [3,7], where the number of virtual channels is not the main factor that contributes to *the* cost of a router in **a** network. Unlike PCS [9,10], the proposed path set-up scheme does not need to reserve any resource of the network, which returns only the path. This scheme can save a lot of bandwidth compared to PCS while still presenting the same reliable path establishment. Certainly, *a* new deadlock avoidance technique is necessary. The proposed method works better when the number of faults in the network and the load of the network are not very low.

2 Related Work

Bruck *et al.* [5] proposed a method to partition a hypercube into small subcubes, each of which has a small number of faults. It was guaranteed that most of the fault-free nodes form a fault-free connected component. These kinds of connected components can be used to implement various fault-tolerant algorithms in hypercubes. Xiang [17] routed a message by localizing safety inside some safe subcubes even though the whole hypercube is unsafe. Safety information inside subcubes called local safety information was used to guide fault-tolerant routing effectively. This idea is also used in this paper to construct fault blocks inside separate **2D** submeshes (called planes).

Fault-tolerant routing in direct networks has been studied extensively. Xiang and Chen in [18] proposed a fault-tolerant routing scheme for **2D** meshes/tori based on local safety. Local safety is an improved safety measure of extended safety levels [15,16]. The method does not need to disable any fault-free node to form fault blocks. Gomez *et al.* proposed techniques for fault-tolerant routing in **3D** meshes/tori without disabling any fault-free nodes in [11].

Linder and Harden [12] extended the concept of virtual channel to multiple virtual interconnection networks that provide adaptivity, deadlock-freedom, and fault-tolerance. Chien and Kim [6] proposed a planar-adaptive routing algorithm that limits routing freedom and makes it possible to prevent deadlocks with only a fixed number of virtual channels (three) independent of network dimension. Judicious extension of the proposed algorithm can efficiently handle routing inside faulty n**D** meshes. Boppana and Chalasani [2] developed fault-tolerant routing algorithms for mesh-connected networks based on the e−cube routing algorithm and the block fault model. At most four virtual channels are sufficient to make fully-adaptive algorithms tolerant to multiple fault blocks in n−dimensional meshes. A deadlock-free fault-tolerant routing algorithm for n−dimensional meshes was proposed in Boura and Das [4] using three virtual channels per physical channel. Fault regions were converted into rectangular regions by a node labeling scheme. However, the above methods [2,4,6] must disable some fault-free nodes to construct the fault blocks, which can result in a great loss of computational power for 3D or higher dimensional networks. Recently, Wang [14] proposed a rectilinear-monotone polygonal fault block model to do fault-tolerant routing in **2D** meshes by disabling fewer fault-free nodes. Most recently, Puente. Gregorio, Vallejo, and Beivide [13] proposed a fault-tolerant routing mechanism for the **2D** torus, which can handle any number of faults if the network is connected. The method [13] must build multiple routing tables.

Path set-up was used first by circuit switching that needs to reserve a physical path before routing a message without any further deadlock avoidance technique. but can waste bandwidth and increase message latency. The pipelined-circuit-switching (PCS) [9,10] establishes a path by reserving a virtual channel path before sending a message, which can tolerate dynamic faults and simplify deadlock-free design. Wu [15,16] proposed an adaptive and deadlock-free fault-tolerant routing method based on extended safety levels. The method needs to establish a region of minimal paths before sending a message, which is the first known fault-tolerant routing scheme based on a limited-global safety measure.

3 Preliminaries

A mesh has k^n nodes, in which each dimension has k nodes. Two nodes $(a_n a_{n-1} \ldots a_2 a_1)$ and $(b_n b_{n-1} \ldots b_2 b_1)$ in a $k-$ary n−dimensional torus network are connected if they differ at exactly one bit i with $a_i = (b_i + 1) \bmod k$. Two nodes in a k−ary n−dimensional mesh $(a_n a_{n-1} \ldots a_2 a_1)$ and $(b_n b_{n-1} \ldots b_2\ b_1)$ are connected if they differ at exactly one bit i $(a_i \neq b_i)$, where $|a_i - b_i| = 1$, and $a_i, b_i \in \{0, 1, 2, \ldots, k-1\}$.

A rectangular fault block contains all the connected faulty and unsafe nodes. Some unsafe nodes can still be locally enabled in one or more planes. In this paper, we say a message is sent from a source s to a destination d along a *minimum feasible path* if the length of the path equals the number of hops that s and d differ, where all nodes in the path are fault-free.

Faults and unsafe nodes in **2D** meshes can form rectangular shapes. **A** set of faults or unsafe nodes F in a *2D* mesh (or torus) are block faults if there is one or more rectangles such that: (1) There are no faults on the boundary of each rectangle. **(2)** The interior of the rectangle includes all faults and unsafe nodes in F. **(3)** The interior of the rectangle contains no node or link that is not presented in F. Fault blocks in a **3D** mesh/torus or n-dimensional mesh/torus can be formed like [2,6]. The only difference is that no fault-free node is disabled. Local safety was introduced to guide fault-tolerant routing in *2 0* torus and mesh-connected networks [18]. Assume the system contains rectangular fault blocks, in which the distance between any two fault blocks is at least two. For each fault-free node v in a mesh/torus network, the local safety of v is defined as follows:

Definition 1 *The local safety of a safe node v in an nD mesh (or torus) network* **is** *defined as a 2n− element tuple* $(v_1, v_2, \ldots, v_{2n})$*, where* $v_1, v_2, \ldots, v_{2n}$ *are each defined as the length of the longest feasible path (not entering a fault block) from v along all 2n different directions, respectively.*

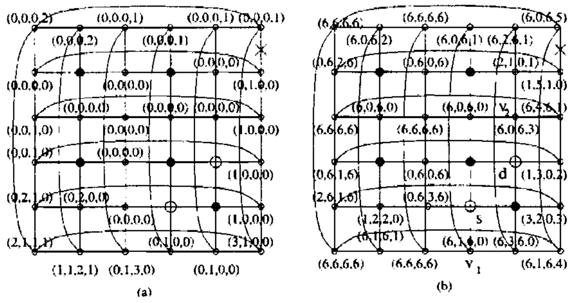

Figure 2: Extended safety level and the local safety: (a)extended safety level [15], and (b) local safety [18].

An element of the extended safety levels is defined as the distance from the node along the corresponding direction to a fault block [15,16]. Figs. 2(a) and 2(b) show the extended safety level and the local safety of all enabled fault-free nodes [18] in the 6−ary 2−cube. Let *(E,S,W.N)* and (v_e, v_s, v_w, v_n) be the extended safety level and the local safety of a node along directions east, south, west, and north, respectively. It is clear that the local safety of a fault-free node is never less than the extended safety level of that node.

Consider node (0.0) at the southwestern corner of the 6x6 torus as shown in Fig. *2*. The eastward path from the node touches a fault block after *2* hops, therefore, $E = 2$. The westward, northward, and southward paths from the node reach the boundaries of three fault blocks after one hop. So, $W = S = N = 1$. As for the local safety of the same node, the length of the longest feasible paths from the node to east, south, west, and north are all 6.

Intermediate nodes must be found like [18] when either the source or the destination is inside a fault block. **As** shown in Fig. 2(b), consider sending a message from s to *d*. It is necessary to find two enabled nodes near to s and *d*. As shown in Fig. 2(b), v_1 and v_2 are selected as intermediate nodes of s and *d*. respectively.

Definition 2 *Nodes in a mesh (or torus) network can be classified as faulty, unsafe, and enabled. A fault-free node* **as** *called* **a71** *unsafe node if it has* **two** *faulty or unsafe neighbors along different dimensions; otherwise, it* **as** *an enabled node. A system* **is** *called unsafe if fault-free nodes in the system are all unsafe.*

Sote that most of the previous methods based on the block fault model disabled all unsafe nodes, and the unsafe nodes cannot be a source or a destination. The unsafe nodes in this paper can be either a source or a destination. This can improve the performance of the system significantly.

4 Extended Local Safety

Only a few faults can make a 3D or n-dimensional mesh/torus unsafe. However, reliable message routing can still be conducted inside many submeshes in a mesh/torus in many cases although the system is unsafe. Let two dimensions form a 2D submesh called a plane. Let us check the 5−ary 3 0 mesh as shown in Fig. 1 again. All fault-free nodes are enabled in each plane except (0,0,0) and (1,0,1) in the plane (*,0,*). Actually, a message between any pair of fault-free nodes can be completed reliably. The following definition needs to be presented first.

Definition 3 *Fault-free nodes inside a faulty nD mesh (or torus) system are classified with respect to a specific 2D plane: a fault-free node is a locally unsafe node with respect to the submesh if it has at least two faulty or locally unsafe neighbors inside the submesh; otherwise, a fault-free node is a locally enabled node with respect to the plane.*

We consider safety of all fault-free nodes inside various 2D submeshes (planes) in this paper, which can also be extended to any other submeshes. Many nodes become safe inside different planes although they are unsafe in the whole system. Consider node $(0,0,0)$ in the faulty 5−ary 3D mesh. All fault-free nodes are unsafe in the whole system, but all of them except (0,0,0) and (1,0,1) are locally enabled in all planes. Sodes (0,0,0) and (1,0,1) are locally unsafe in the plane (*,0,*) because they have two faulty neighbors inside the submesh. Influences of faults can be limited inside the corresponding subnetworks when the local safety information is considered.

Definition 4 *The extended local safety of a fault-frer node v an a $k-ary$ nD mesh can be defined as a $2n-tuple$ $(v_1, v_2, \ldots, v_{2n})$; each v_i $(1 \leq i \leq 2n)$ is obtained by*

$$v_i = min_j\{v_{ij}\}, \quad (1)$$

where v_{ij} is the corresponding value of the local safety of the node along direction i inside the plane formed by dimensions i and j, and v is locally enabled inside the plane.

```
Procedure extended-local-safety()
repeat
For each fault-free node $v$ in the system, parallel do
local-class($v$);
parallel end.
until stable states have been obtained.
```

local-class(i)

1. Node i gets its states in different planes that contain it;
2. if i has at least two faulty or locally unsafe neighbors along different dimensions in a plane, set state of v in that plane as locally unsafe.

The extended local safety information of a system can be updated if necessary. It should also be noted that the extended local safety information is not prepared to route one message. It is kept by each fault-free node to guide fault-tolerant routing until new faults occur. The effort to capture the extended local safety information should be comparable to that required to form the rectangular fault blocks [2,4,6,15,16,18], or to obtain the extended safety levels [15,16] and the local safety [19]. The amount of safety information stored in each node should be at most as n times as that in [16], which should be acceptable for practical networks. The extra information stored in each node should include: (1) safety information of the node in $n\,.(n-1)/2$ different planes that contain it, and (2) the extended local safety information of the node, which is a 2n-tuple. Consider a 3D mesh/torus: the amount of the extended local safety information stored in each fault-free node is at most 3 times as much as that of the extended safety levels [16], which is acceptable.

The method in [18] presents a fault-tolerant routing algorithm only in 2D meshes or tori. Compared with the safety measure in [18], the proposed method can improve the adaptivity and performance of the algorithm because all fault-free nodes inside a fault-block can be the intermediate node of a message from a source inside or outside of the fault block, The resources with respect to globally unsafe nodes can thus be used by the messages.

Let the extended local safety of a fault-free node in a $k-$ary 3D mesh be represented as (E, S, F, W, N, B), where E, W, F, S, N, and B represent the extended local safety values of the node along directions east, west, front. south, north. and behind, respectively. As shown in Fig. 1. the extended local safety of nodes (0,1,0) and (3,2,2) is $(1,2,-,-,2,4)$

and $(1,2,2,3,2,2)$, respectively, where "-" stands for don't care. We consider safety of fault-free nodes inside each plane.

The size of the safe node set based on the planarly constructed fault blocks should be no less than that of the one based on the conventional fault block model. The extended local safety information can always present more information for fault-tolerant routing. As shown in Fig. 1, the extended local safety of nodes (0,3,0), (4,3,4), and (2,0,2) are (4,3,-,-,1,4), (-,3,4,4,0,-), and (2,-,2,2,4,2), respectively, where "-" is don't care. We consider safety inside planes; therefore, our method is presented based on the extended local safety information with respect to each plane. However, the extended safety level [16], the planar adaptive routing method *[6]*and Boppana's routing protocol [1] can do nothing on the **3D** mesh because the network is unsafe.

5 Virtual Subnetwork Partitioning for Deadlock Avoidance

A **3D** mesh can be partitioned into eight different virtual subnetworks: x+y+z+, x+y+z-, x+y-z+, x+y-z-, x-y+z+, x-y+z-, x-y-z+, and x-y-z-. All eight virtual subnetworks can be combined into 4 different virtual subnetworks: x+y+z* (c_1+,c_1+,c_1), x-y*z+ (c_2-,c_2,c_2+), x-y*z- (c_1-,c_3,c_2-) and x+y-z* (c_2+,c_1-,c_3). Labels in the brackets show virtual channel assignments of all virtual subnetworks. Messages are classified based on the relative locations of the source and destination. For example, a message to be sent from (0,1,0) to (3,0,3) falls into the 4th class of messages.

Let both the source and destination be safe in any plane. Only one additional virtual channel is enough to support non-minimal routing and avoid deadlocks because no turn is generated by the additional virtual channels in these cases. Let at least one of source and destination be unsafe inside at least one plane. Turns may be formed by the extra virtual channel. Two additional virtual channels are required in this case. A deroute message should use c_4 first. A turn from a lower label dimension to a higher label dimension should use c_4, and a turn from a higher label dimension to a lower label dimension should use extra virtual channel c_5. So cyclic dependency among the extra virtual channels forms.

As for **3D** torus networks, possible cyclic dependency generated by the wraparound links must be eliminated. Linder and Harden [12] used another virtual network after traversing a wraparound link, which needs $O(n \cdot 2^n)$ virtual channels for each physical channel. In our method, techniques similar to those for meshes are adopted. All messages are partitioned into **4** different classes in a **3D** torus. Virtual network partitioning is still utilized like **3D** meshes. **A** message uses the regular virtual channels assigned to the corresponding virtual network along a specific dimension with "+") if the label of the current node with respect to the dimension is less than that of the destination. And the message uses an extra virtual channel c_4 or c_5 if the label of the current node with respect to the dimension is greater than that of the destination. A message uses the regular virtual channels assigned to the corresponding virtual network along a specific dimension with "$-$" if the label of the current node with respect to the dimension is greater than that of the destination. And the message uses an extra virtual channel c_4 or c_5 if the label of the current node with respect to the dimension is less than that of the destination. When a message is routed along a dimension with "*", virtual channels are utilized like a dimension with "+" if the label of the destination along the dimension is greater than that of the source; otherwise, virtual channels are assigned like a dimension with "$-$".

The potential cyclic dependency among the wraparound links must also be avoided. As mentioned earlier, a message, when it uses a regular virtual channel for the first wraparound link to a higher label dimension wraparound link, uses virtual channel c_4, while a turn from a higher dimension label wraparound link to a lower dimension wraparound link utilizes extra virtual channel c_5. This scheme can successfully eliminate the cyclic dependency among the wraparound links. It is clear that no cycle exists in any virtual subnetwork and among virtual subnetworks [12,8,16]. Also, the derouted messages cannot form any rycles if the network is not disconnected. Virtual channels c_4 and c_5 should be idle in most cases. Our method utilizes the idle virtual channels and assigns them as any virtual channels when necessary. The label of the virtual channel can be relabeled as the previous number after the message has been sent [7]. The technique can improve performance of the system in most cases. Unlike [16], the proposed method does not partition each virtual subnetwork into a sequence of planes when sending a message. A message can be routed *to* any minimum hop via the assigned virtual channel inside the specified virtual subnetwork, which can enhance adaptivity and performance of the algorithm compared with [16].

6 Routing Algorithm Based on the Extended Local Safety

The routing algorithm might be approximately described as follows: take a step in the direction most likely to meet fault blockage; repeat until the destination is reached. In the case of blockage at the final step, deroute along an unblocked path and continue. A heuristic based on the extended local safety information as presented in Equation (2) is used to avoid fault blocks. In Equation (2), $L_i(s,d)$ is the number of hops that s and d differ along dimension i, and d_i is the extended local safety information of the source with respect to dimension i as defined in Definition 4. Only the extended safety information of the source is necessary to compute the heuristic function.

$$h_i = \begin{cases} 0 & \text{if } d_i \geq L_i(s,d), \\ d_i - L_i(s,d) & \text{otherwise.} \end{cases} \qquad (2)$$

The PCS [9,10] and circuit switching need to reserve some resources of the network before sending a message, which may waste some bandwidth. We would like to propose a scheme without reserving any resource. However, path set-up is still used. Assume the source does not have knowledge of the extended local safety of the destination. Each message falls into a unique virtual subnetwork. A feasible path leading to the destination is established inside the selected virtual subnetwork until reaching the destination. The message is derouted along the additional virtual channels if necessary, and returns to the virtual subnetwork as soon as possible. The above process continues until a feasible path has been set up. The set-up path can be stored in the header.

In the rest of this section, procedure *route-message-in-nD-mesh*() sends a message from s to d in a $k-$ary n-dimensional mesh. Procedure *select-path*() completes the path set-up from the destination d to the source s, and the procedure *send-message()* sends the message from the source s to the destination d based on the extended local safety information of the destination and location of the current node if a minimum feasible path is not found in the first two stages.

route-message-in-nD-mesh()

1. Keep the extended local safety of the source s. Find a path from the source leading to the destination inside the assigned virtual subnetwork.

2. If a minimum feasible path from s to d has been set up at the destination d, send a signal from d to s along the setup path. The source s sends the message along the setup minimum feasible path. Otherwise (3).

3. Try to set up a minimum feasible path from d to s inside the corresponding virtual Subnetwork. If s and d differ in only one dimension and there exists a feasible path from the destination to the source: a minimum feasible path has been set up. Otherwise call *select-path*().

4. If a minimum feasible path has been found, send the message from s along the selected path. Otherwise, call *send-message()*.

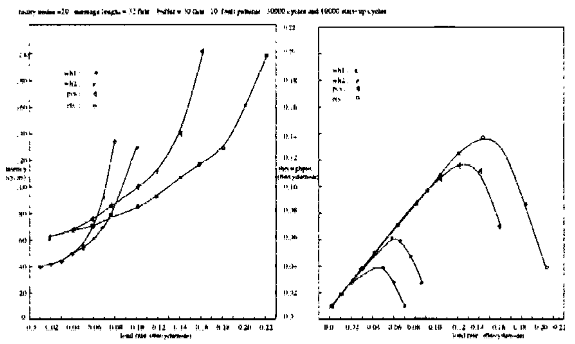

Figure 3: Performance comparison with PCS and the wormhole routing techniques in 8x8x8 meshes.

select-path()

1. Go along dimension t, where the source s has the least heuristic h_t corresponding to location of the current node v and the extended local safety information of the source if a possible path along dimension t is available. otherwise (2).

2. Go along another dimension among the remaining ones with the least heuristic based on Equation (2) if the selected next hop is inside a fault block in all planes that d and s differ.

3. Continue the above process until reaching a point where s and d differ only in one dimension. A feasible path has been set up if a feasible path from the node to s is available. Otherwise (4).

4. Deroute the signal to a node, where s has the most extended local safety dong that dimension. Go along a minimum hop if possible. Continue the above process until reaching the source s.

send-message()

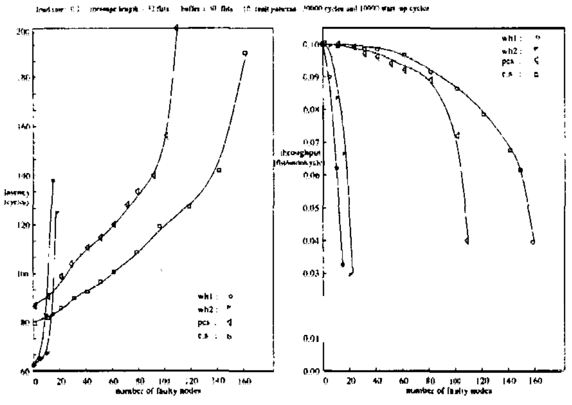
Figure 4: Performance comparison in 8x8x8 meshes with fixed load rate.

1. Send the message along dimension t, where d has the least heuristic h_t with respect to the extended local safety information of the destination and location of the current node if the node is fault-free and not in a fault block in one plane, otherwise (2).
2. Pass the message along another dimension, where the destination d has the least heuristic among the remaining dimensions and the next node is not in a fault block in at least one plane.
3. Continue the above process until reaching a node that differs from the destination in one dimension.
4. If there exists a minimum feasible path from the node to the destination, pass the message along the feasible path; otherwise, deroute the message to a fault-free neighbor until a fault-free neighbor in a minimum path from the current node to the destination along another dimension is available.
5. Continue the above process until the destination is reached.

7 Simulation Results

A flit-level simulator has been implemented to evaluate the proposed extended-local-safety-information-based fault-tolerant routing algorithm (*els*). Flit-level simulators on the planar adaptive routing algorithm (*wh1*) [6], the wormhole-routing-based algorithm proposed by Boppana and Chalasani (*wh2*) [2], and the pipelined-circuit-switching-based algorithm (*pcs*) [9] have also been implemented to compare with the proposed algorithm. All results in the following figures present the average of 10 different fault patterns.

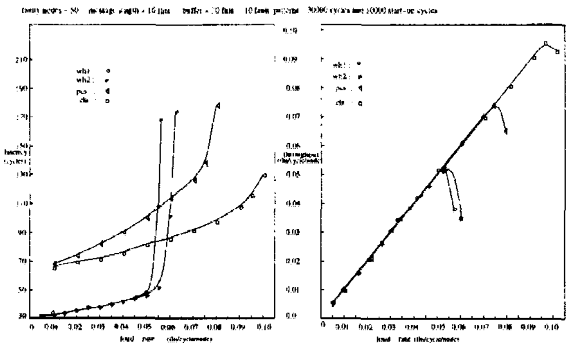
Figure 5: Performance comparison in 16x16x16 meshes.

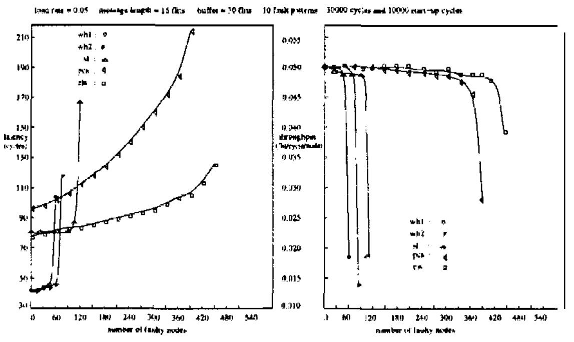
Figure 6: Performance comparison in 16x16x16 meshes with fixed load rate.

Faults are randomly inserted. The simulation results are presented only for static faulty nodes, which can be extended to link failure easily. The proposed method can also be extended to dynamic faults easily like PCS [9]. It is shown that *wh2* always works a little better than *wh1*. The most important reason can be that *wh1* always makes messages routed inside 2D planes and also *wh2* uses four virtual channels, but *wh1* uses only three virtual channels for each physical channel. Two important metrics, latency (the number of cycles required to deliver a message) and throughput (flit/node/cycle), are evaluated. The message length and buffer size of each node in 8x8x8 meshes are set as 32 flits and 30 flits, respectively.

Fig. 3 presents performance comparison of all four methods when the system has different load rates and the 8x8x8 mesh contains 20 faulty nodes. Fig. 4 presents performance comparison of the four methods when the load rate of the 8x8x8 mesh system is set as 0.10. Figs. 5 and 6 present performance comparisons of the proposed extended local safety information with the previous methods. Fig. 5 presents performance comparison of the four methods when the 16x16x16 mesh contain 50 faulty nodes. Latency and throughput comparisons of ***wh1, wh2, pcs,*** and *els* are presented when load rate of the system is set as 0.01 − 0.10. Fig. 6 demonstrates performance comparison of the proposed method with five previous methods on the 16x16x16 meshes when the load rate of the system is fixed as 0.05. The extended safety level (*sl*) [15,16] is implemented for 16x16x16 meshes.

8 Conclusions

Extended local safety was utilized to guide fault-tolerant routing in an n-dimensional mesh/torus, which calculates safety information by forming fault blocks inside each plane. The extended local safety considers safety of **a** mesh/torus inside each plane instead of in the whole system. This technique **can** make numerous unsafe nodes in the whole system locally enabled in the 2D planes. The proposed method did not disable any fault-free nodes, and any fault-free nodes inside a fault block in a plane can still be **a** source or a destination. Extensive simulation results were presented by comparing with previous methods.

References

[1] F. Allen, *et al.*, "Blue gene: A vision for protein science using a petaflop supercomputer," *IBM Systems Journal,* vol. *40,* pp. 310-327: 2001.

[2] R. V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," *IEEE Trans. Computers,* vol. *44,* no. **7,** pp. 848-864, 1995.

[3] R. V. Bnppana and S. Chalasani, "A framework for designing deadlock-free wormhole routing algorithms,'' *IEEE Trnns. on Parallel and Distributed* Sys*tems,* vol. *7,* no. 2, pp. 169-183, 1996.

[4] Y. M. Boura and C. R. Das, "Fault-tolerant routing in mesh networks,?'Proc. of *IEEE Int. Conf. on Parallel Processing,* I106-I109, 1995.

[5] J. Bruck, R. Cypher, and D. Soroker, "Tolerating faults in hypercubes using subcube partitioning," *IEEE Trans. on Computers,* vol. 41, no. 5, pp. 599-605, 1992.

[6] **A.** A. Chien and J. H. Kim, "Planar adaptive routing: Low-cost adaptive networks for multiprocessors," *J.* of *ACM,* vol. 42, no. 1, pp. 91-123, 1995.

[7] W. J. Dally and Aoki, "Deadlock-free adaptive routing multicomputer networks using virtual channels," *IEEE Trans. on Parallel and Distributed Systems,* vol. 4, no. **4,** pp. 466-475, Apr. 1993.

[8] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach,* IEEE Press, 1997.

[9] P. T. Gaughan and S. Yalamanchili, "A family of fault-tolerant routing protocols for direct multiprocessor networks," *IEEE Trans. on Parallel and Distributed Systems,* vol. *6,* no. 5, pp. 482-497, 1995.

[10] P. T. Gaughan, B. V. Dao, S. Yalamanchili, and D. E. Schimmel, "Distributed, deadlock-free routing in faulty, pipelined, direct interconnection networks,': *IEEE **Trans.** Computers,* vol. **45,** no. 6, pp. 651-665, 1996.

[11] M. E. Gomez, J. Flich, P. Lopez, **A.** Robles, J. Duato, N. A. Nordbotten, O. Lysne, and T. Skeie, "An effective fault-tolerant routing methodology for direct networks," Proc. of *IEEE Int. Conference on Parallel Processing,* pp. 222-231, 2004.

[12] D. H. Linder and J. C. Harden, "An adaptive and fault-tolerant wormhole routing strategy for $k-$ary n−cube," *IEEE Trans. Computers,* Vol. **40**, no. 1, pp. 2-12, 1991.

[13] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide, "Immunet: A cheap and robust fault-tolerant packet routing mechanism," Proc. of *ACM/IEEE Int. Symp. on Computer Architecture,* pp. **198-209,** 2004.

[14] D. Wang, "A rectilinear-monotone polygonal fault block model for fault-tolerant minimal routing in mesh," *IEEE Runs. on Computers,* vol. 52, no. 3, pp. 310-320, 2003.

[15] J. Wu, "Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels," *IEEE Trans. Parallel and Distributed Systems,* vol. *11,* no. 2, pp. 149-159, 2000.

[16] J. Wu, "A fault-tolerant adaptive and minimal routing approach in nD meshes," Proc. of *IEEE Int. Conf. Parallel Processing,* Aug., **pp.** 431-438, 2000.

[17] D. Xiang, "Fault-tolerant routing in hypercube multicomputers using local safety information,'' *IEEE Trans.* on *Parallel and Distributed Systems,* vol. **12,** no. 9, pp. 942-951, 2001.

[18] D. Xiang and A. Chen, "Fault-tolerant routing in 2D tori or meshes using limited-global-safety information," Proc. of *IEEE Int.* **Conf.** *on Parallel Processing,* pp. 231-238, Vancouver, Aug., 2002.

[19] J. Zhou and F. C. M. Lau, "Adaptive fault-tolerant wormhole routing in 2D meshes,'' Proc. of 15-th *IEEE Int. Parallel and Distributed Processing Syrnp.,* pp. 249-256, 2001.