An Efficient Fault-Tolerant Routing Methodology for Direct Interconnection Networks

M.E. Gómez, J. Flich, A. Robles, P. López, and J. Duato
Dept. of Computer Engineering, Universidad Polit´ecnica de Valencia
Camino de Vera, 14, 46071–Valencia, Spain
E-mail: megomez@gap.upv.es

N.A. Nordbotten, O. Lysne, and T. Skeie
Simula Research Laboratory
P.O. Box 134, N-1325 Lysaker, Norway
E-mail: nilsno@simula.no

***Abstract*— Nowadays, massively parallel computing systems are being built with thousands of nodes. This huge number of nodes significantly affects the probability of failure. Thus, it is critical to keep these systems running even in the presence of failures. The interconnection network plays a key role in the performance achieved by these systems, since failures in the interconnection network may isolate a large fraction of the machine containing many healthy nodes.**

In this paper we present a methodology to design fault-tolerant routing algorithms for regular direct interconnection networks. It supports fully adaptive routing, does not degrade performance in the absence of faults, and supports a reasonably large number of faults without significantly degrading performance. The methodology is mainly based on the selection of an intermediate node (if needed) for each source-destination pair. Packets are adaptively routed to the intermediate node and, at this node, without being ejected, they are adaptively forwarded to their destination. In order to allow deadlock-free minimal adaptive routing, the methodology requires only one additional virtual channel (for a total of three), even for tori.

Evaluation results for a $4 \times 4 \times 4$ torus network show that the methodology is 5-fault tolerant. Indeed, for up to 14 link failures, the percentage of fault combinations supported is higher than 91.3%. Additionally, network throughput degrades by less than 10% when injecting three random link faults without disabling any node. In contrast, a mechanism similar to the one proposed in the BlueGene/L, that disables some network planes, would strongly degrade network throughput by 79%.

Keywords: fault-tolerance, direct networks, adaptive routing, bubble flow control.

I. Introduction

MANY compute-intensive applications require continued research and technology development, only achieved with massively parallel processor systems (MPPs) like the Earth Simulator [8], the ASCI Red [1], and the BlueGene/L [11].

The huge number of processors and associated devices of these machines significantly affects the probability of failure. In particular, failures in the interconnection network may isolate a large fraction of the machine containing many healthy processors that could otherwise be used. Although network components, like switches and links, are robust, they are working close to their technological limits and are therefore prone to failures. Increasing clock frequencies leads to a higher power dissipation, which again could lead to premature failures. Therefore, fault-tolerant mechanisms for interconnection networks are becoming a critical design issue for large massively parallel computers.

This work was supported by the Spanish MCYT under Grant TIC2003-08154-C06-01.

Faults can be classified as transient or permanent. Transient faults are usually handled by communication protocols, using CRCs to detect faults and retransmitting packets. In order to deal with permanent faults in a system, two fault models can be used: static or dynamic. In a static fault model, it is assumed that all the faults are known in advance when the machine is (re)booted. This fault model needs to be combined with checkpointing techniques in order to be effective. In a dynamic fault model, once a new fault is found, actions are taken in order to appropriately handle the faulty component.

Many solutions have been proposed in the literature to address reliability problems in interconnection networks. The most frequently used technique in commercial systems consists of replicating components. However, the main drawbacks of this approach are the high extra cost of spare components and the non-negligible probability of failure of the circuits required to switch spare components into the system or to bypass faulty components. Another powerful technique is based on reconfiguring the routing tables in case of failure, adapting them to the new topology after the failure [2]. This technique is extremely flexible but this flexibility may also kill performance.

However, most of the solutions proposed in the literature are based on designing fault-tolerant routing algorithms able to find an alternative path when a packet meets a fault along the path to its destination. Most of the proposed fault-tolerant routing strategies require a significant amount of extra hardware resources (e.g., virtual channels) to route packets around faulty components depending on either the number of tolerated faults [5] or the number of dimensions of the topology [12]. Alternatively, there exist some fault-tolerant routing strategies that use none or a very small number of extra resources to handle failures at the expense of providing a lower fault-tolerance degree [5], [9], disabling a certain number of healthy nodes (either in blocks (fault regions) [4], [3] or individually [6], [7]), preventing packets from being routed adaptively [10], or drastically increasing the latencies for some packets [14]. Moreover, when faults occur, link utilization may become significantly unbalanced when using those fault-tolerant routing strategies, thus leading to premature network saturation and consequently degrading network performance even more.

In our opinion, what is really needed is a fault-tolerant strategy for the interconnection network that does not degrade performance at all in the absence of faults and supports a reasonably large number of faults without significantly degrading performance, without disabling any healthy node, and without

requiring too many extra hardware resources.

In this paper, we take on this challenge and propose a fault-tolerant routing mechanism that satisfies the properties mentioned above. The design methodology we propose is generic and can be applied to different topologies, such as tori and meshes. A static faul model based on checkpointing techniques is assumed. In particular, we propose in this paper a methodology that will allow the use of fully adaptive routing in the absence of failures, will not sacrifice any healthy node, and only requires the use of one additional virtual channel. Note that two virtual channels are already required to provide fully adaptive routing [13]. Therefore, this mechanism will require the use of at least three virtual channels. Basically, the methodology avoids faults by using intermediate nodes for routing. For some source-destination pairs, packets are first forwarded to an intermediate node, and then from this node to the destination node, without being ejected. Minimal adaptive routing is used in both subpaths. However, in some special situations, it is also required to disable adaptive routing to avoid all the faults. Intermediate nodes were introduced by Valiant [15] for other purposes, such as traffic balance.

Indeed, the main goal of this paper is to minimize the number of extra resources required to providing a reasonable degree of fault-tolerance. However, once we achieved this, we pursue as a secondary goal the idea of maximizing the number of tolerated faults without increasing the number of extra resources. To the achievement of the second objective will contribute the fact of using a static fault model based on checkpointing techniques, which allows for significanly higher degrees of fault-tolerance for the same number of resources.

It is important to highlight the main differences between our proposal and other approaches in the literature that also use a small number of extra resources. In particular, unlike [14], in no case the proposed mechanism requires ejecting/reinjecting the packet at the intermediate node, thus reducing latency drastically. Moreover, unlike [10], our fault-tolerant mechanism does not need to deactivate any ”lamb” nodes to achieve good fault-tolerance degrees and allows packest to be adaptivelly routed instead of making it in a deterministic way, thus increasing the overall network throughput.

So, the main contributions of this paper are the proposal of a new fault-tolerant methodology for regular direct interconnection networks, a formal description of the procedures followed by the methodology to select intermediate nodes under different scenerios, and a detailed evaluation showing the performance benefits that can be obtained in the presence of failures when using the proposed routing algorithm. This algorithm could be easily incorporated into the IBM BlueGene/L supercomputer. For this, a performance comparison will be performed with respect to the fault tolerance mechanism used in the IBM BlueGene/L supercomputer.

The rest of the paper is organized as follows. In Section II, the methodology is described. In Section III, some examples are presented. In Section IV, evaluation results are shown. Finally, in Section V, some conclusions are drawn.

II. The Methodology

In what follows, we will denote the source node as S, the destination node as D, and the intermediate node as I. Faulty

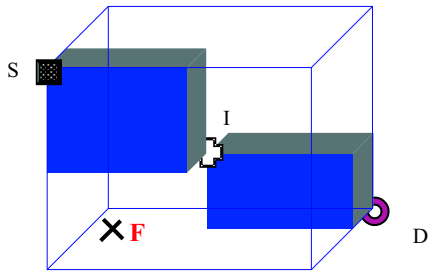

Fig. 1. A source node sending packets to a destination via an intermediate node.

links are denoted as F_i. A node failure can be modeled as the failure of all of its links. We will assume a k-ary n-cube (or torus) network with minimal adaptive routing with three virtual channels per physical channel (one adaptive and two escape channels). Deadlock freedom is ensured by having a separate escape channel for each phase, that is, one escape channel is used (if needed) from S to I and another one from I to D. Therefore, we define two virtual networks. Each one will rely on a different escape channel, but both will use the same adaptive channel(s). The transition between both virtual networks will be done at the intermediate node.

The escape channels can use any deadlock-free deterministic routing method. In this paper we use DOR with bubble flow control [13]. With this mechanism, packets that are injected into the network or move to another network dimension require two free buffers (one for the packet plus an additional one) to guarantee deadlock freedom. Hence, in order to avoid deadlocks, packets changing virtual network at intermediate nodes should be considered as packets that are moving to another dimension, and thus, two free buffers are required.

An intermediate node I is used only if there exists at least one fault that can be encountered when routing packets from S to D using adaptive routing. By appropriate selection of the intermediate node I, packets will be routed from S to I and then from I to D without encountering the fault(s), and adaptive routing can be used in most cases.

The intermediate node I is selected inside the minimal adaptive cube defined by S and D. Thus, both subcubes defined by S and I, and by I and D, will be inside this cube, but they will be smaller and will avoid the failure. Figure 1 shows these subcubes for a given source, destination and intermediate node.

Packets sent through intermediate nodes contain two destinations in their header. The first one corresponds to the intermediate node, and is removed there. The second one is the true destination of the packet. Every source node should also maintain a table that contains the intermediate node that should be used to reach a given destination and two bits that indicate if adaptive routing must be disabled in any of the subpaths. This table does not require a large amount of memory. For instance, a system with 65,536 nodes, only requires 144KB (65,536 entries $\times$(16+2) bits). This table can be easily compacted by storing information about faulty paths only.

Furthermore, the computational cost of the proposed methodology is not high, especially if we take into account that routing info is computed off-line in a static fault model. For

each source–destination pair, it must be analyzed if adaptive paths are affected by faults. If so, the intermediate node must be computed. As the cost of computing each intermediate node is $O(1)$ for meshes and tori, the computational cost is $O(n^2)$, where n represents the number of nodes.

Next, a methodology for identifying intermediate nodes will be presented. First, the case where adaptive shortest path routing can be used will be discussed. We then show how the power of the methodology can be increased by using deterministic routing and non-minimal paths. Finally, it is shown how non-minimal paths can also be used with adaptive routing, and combinations of adaptive and deterministic routing are discussed. We assume that adaptive routing can be turned off on a per packet basis.

A. Intermediate Nodes for Adaptive Routing Giving Minimal Paths

When minimal adaptive routing is used, the intermediate node I should have the following properties so that the fault(s) F_i is(are) avoided when routing from S via I to D:

1) For all i, F_i is not on any shortest path from S to I.
2) For all i, F_i is not on any shortest path from I to D.
3) I is on at least one shortest path from S to D.

The first property guarantees that packets can be routed from S to I without being affected by the faulty link(s). Likewise, the second requirement guarantees that packets are not affected by the faulty link(s) when routed from I to D. The third property guarantees that routing through the intermediate node I gives minimal routing from S to D.

Let $\mathcal{T}_0$ be the set of nodes that can be traversed on any shortest path from S to D. Furthermore, let $\mathcal{T}_{SF}$ be the set of nodes that can be traversed on any shortest path from S to F_i (for all i) and let $\mathcal{T}_{FD}$ be the set of nodes that can be traversed on any shortest path from F_i (for all i) to D.

Teorema 1: A node N fulfills all three requirements if and only if it is in the set $\mathcal{T}_0 \setminus (\mathcal{T}_{SF} \cup \mathcal{T}_{FD})$.[1]

Thus, when the set $\mathcal{T}_0 \setminus (\mathcal{T}_{SF} \cup \mathcal{T}_{FD})$ is non-empty, a suitable intermediate node can be selected among its members. By selecting the intermediate node this way, it is guaranteed that the path S-I-D yields minimal path routing from S to D, and that adaptive routing can be used on each subpath without encountering a fault. If the set contains more than one node, the intermediate node can be selected randomly or based on some other criteria such as traffic balancing or routing flexibility.

B. Intermediate Nodes for Deterministic Routing Giving Minimal and Non-Minimal Paths

A deterministic minimal routing function uses a subset of the paths returned by an adaptive minimal routing function. Therefore, a node from the set $\mathcal{T}_0 \setminus (\mathcal{T}_{SF} \cup \mathcal{T}_{FD})$ can also be used as intermediate node when deterministic routing is used. However, there are scenarios where the set $\mathcal{T}_0 \setminus (\mathcal{T}_{SF} \cup \mathcal{T}_{FD})$ is empty but where it is still possible to find a suitable intermediate node if routing is restricted to a deterministic route, and, if required, by the use of non-minimal paths from S to D. This way, nodes that could not be used as intermediate nodes with adaptive routing may be used as intermediate nodes with deterministic routing.

When fault-tolerance through intermediate nodes is applied in combination with deterministic routing we are looking for an intermediate node I such that:

1) For all i, F_i is not on the $S - I$ deterministic path.
2) For all i, F_i is not on the $I - D$ deterministic path.
3) There is no I' giving a shorter path than I.

The first requirement guarantees that packets can be routed from S to I, and the second requirement guarantees that packets can be routed from I to D. Because a deterministic path is used, this is sufficient to ensure that packets avoid the fault(s). The third requirement guarantees that the final path is the shortest possible.

To identify the possible intermediate nodes, let $\mathcal{T}_{RS}^d$ be the set of nodes reachable through deterministic routing from S and $\mathcal{T}_D^d$ the set of nodes that have a valid deterministic route to D. Furthermore, let $l(x, y)$ be the length of the minimal path, in the fault free case, from x to y. We then generalize the definition of $\mathcal{T}_0$, to $\mathcal{T}_j$ (for $j \geq 0$), in the following way: A node N is in $\mathcal{T}_j$ if, and only if, $l(S, N)+l(N, D) = l(S, D)+ j$. This way, $\mathcal{T}_j$ defines non-overlapping sets of nodes that can easily be identified by starting with $\mathcal{T}_0$ and continuing outwards.

Teorema 2: Let j' be the smallest j for which $\mathcal{T}_j \cap \mathcal{T}_{RS}^d \cap \mathcal{T}_D^d$ is non-empty. A node N fulfills all three requirements if and only if $N \in \mathcal{T}_{j'} \cap \mathcal{T}_{RS}^d \cap \mathcal{T}_D^d$.[1]

This way, we start considering the shortest paths ($j = 0$), and then if necessary non-minimal paths ($j > 0$), to avoid the faulty link(s).

C. Intermediate Nodes for Adaptive Routing and Combinations of Adaptive and Deterministic Routing Giving Minimal and Non-Minimal Paths

Even when it is not possible to use minimal adaptive routing all the way from S via I to D (i.e. when the set $\mathcal{T}_0 \setminus (\mathcal{T}_{SF} \cup \mathcal{T}_{FD})$ is empty), it may still be possible to use adaptive routing from S to I or from I to D. In addition, when the intermediate node is selected outside $\mathcal{T}_0$, it may be possible to use adaptive routing both from S to I and from I to D.

To identify these cases, let $\mathcal{T}_{RS} \subseteq \mathcal{T}_{RS}^d$ be the set of nodes reachable through any shortest path from S (i.e. without possibly encountering any F_i) and $\mathcal{T}_D \subseteq \mathcal{T}_D^d$ the set of nodes that reach D through any shortest path.

If the intermediate node is selected from $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}_D^d$, adaptive routing can be used from S to I, whereas deterministic routing must be used from I to D. Similarly, if the intermediate node is selected from $\mathcal{T}_j \cap \mathcal{T}_{RS}^d \cap \mathcal{T}_D$, deterministic routing must be used from S to I, whereas adaptive routing can be used from I to D. If the intermediate node is selected from the set $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}_D$, adaptive routing can be used both from S to I and from I to D. Notice that when $j = 0$, this case amounts to the case shown in Section II-A.

III. Example Scenarios

We will now consider some particular fault situations to illustrate the proposed methodology. A 2-D mesh is used

[1]The proofs are omitted due to space considerations.

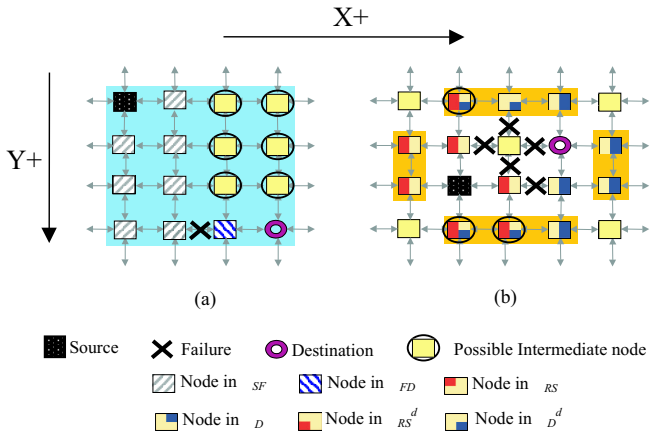

Fig. 2. (a). Scenario with a single link fault. $\mathcal{T}_0$ corresponds to the shaded area. (b). Scenario where all the shortest paths are blocked by faults. The shaded areas identify nodes within $\mathcal{T}_2$.

for this purpose, but the mechanism also applies to other topologies such as a 3-D torus or mesh. Let us first consider a simple scenario with only one fault. We will later consider a more complex scenario with several faulty links.

The first scenario is illustrated in Figure 2.(a). As shown, there is a faulty link in some of the shortest paths between the source and the destination. Because there is a risk that a packet routed from S to D may encounter this faulty link, an intermediate node is necessary. Using the proposed methodology we will now demonstrate how to identify such a node.

In order to use adaptive routing, both from S to I and from I to D, we want to select a node from the set $\mathcal{T}_0 \setminus (\mathcal{T}_{SF} \cup \mathcal{T}_{FD})$. The shaded area in the figure marks the nodes in $\mathcal{T}_0$, i.e. all the nodes traversed on some shortest path from S to D. As shown in Figure 2.(a), there are several nodes in the set $\mathcal{T}_{SF}$, whereas the set $\mathcal{T}_{FD}$ only contains a single node in this scenario. The remaining nodes within the shaded area belong to the set $\mathcal{T}_0 \setminus (\mathcal{T}_{SF} \cup \mathcal{T}_{FD})$ and are thus possible intermediate nodes. This ensures that a packet routed using minimal adaptive routing cannot encounter the faulty link, when first routed from S to I and then from I to D.

Figure 2.(b) shows a more complex scenario where multiple faults are present. All the shortest paths between the source and the destination are here blocked by faults, resulting in the set $\mathcal{T}_0 \setminus (\mathcal{T}_{SF} \cup \mathcal{T}_{FD})$ being empty. It is therefore necessary to apply the mechanism given by theorem two, to identify an intermediate node. To get a more restricted routing function we assume that dimension order routing (XY-routing) can be used when needed. As all the shortest paths between S and D are blocked, the set $\mathcal{T}_j \cap \mathcal{T}_{RS}^d \cap \mathcal{T}_D^d$ is empty for j=0. Because this is a mesh network, $\mathcal{T}_j$ is empty for all odd values of j (it is impossible to add only one hop, when there are no wraparound links, as adding one node to a path equals adding two hops), and we must therefore try with j=2. The set $\mathcal{T}_2$ contains the nodes within the shaded areas in the figure. Among the nodes within $\mathcal{T}_2$ that are reachable from S, the ones with a valid route to D are possible intermediate nodes (i.e. $\mathcal{T}_2 \cap \mathcal{T}_{RS}^d \cap \mathcal{T}_D^d$).

Finally, notice that in this scenario it is only necessary to use deterministic routing from I to D, as adaptive routing can be used from S to I without risk of encountering a fault (i.e. the intermediate nodes belong to $\mathcal{T}_2 \cap \mathcal{T}_{RS} \cap \mathcal{T}_D^d$).

IV. Evaluation of the Methodology

We will now evaluate the proposed methodology, first in terms of fault-tolerance, then in terms of performance.

A. Fault-Tolerance

We will now first present the fault analysis models that have been used to determine the degree of fault-tolerance exhibited by the proposed methodology. Then, the results from the fault-tolerance analysis are presented, individually for each mechanism, intermediate nodes (I) and disabling adaptivity (D), and the combination of both mechanisms (I+D).

For a limited number of faults in the network, all the possible combinations of faults can be explored. However, as the number of faults increases, the number of possible fault combinations increases exponentially. Therefore, from a particular number of faults, it is impossible to explore all the fault combinations in a reasonable amount of time. We use two approaches to deal with this problem. The first approach focuses on faults bounded into a limited region of the network. Notice that, the worst combinations of faults to be solved are those where the faults are closely located. This is because the number of fault-free paths among the nodes in a region with many faults will be reduced. As the number of possible fault combinations is much lower for such a region than for the entire network, all the fault combinations can be evaluated. Although the results obtained cannot be directly extended to the generic case, where faults are located over the entire network, it will give us an approximation of the effectiveness of the methodology in the worst case.

For this, we must define the region where the faults are to be located. The region is formed by all the links of the nodes that are one hop away from a node (the center node). The center node is randomly selected[2]. We will refer to this region as a *distance 1* region. Figure 3 shows a *distance 1* region, formed by 36 links. However, in a $3 \times 3 \times 3$ Torus it only consists of 33 unique links, as three of the links then are shared by nodes within the region. Notice that for a high number of faults and for a large number of fault combinations, the center node is hardly accessible, as very few links then are non-faulty. So the *distance 1* region actually represents a real worst case to access the center node.

In the second approach, an statistical analysis is performed, analyzing a subset of fault combinations randomly located over the entire network. From the obtained results, statistical conclusions are extracted about the fault-tolerance degree of the proposed methodology.

Table I shows, for different number of faults, the percentages of fault combinations not tolerated by the two mechanism (intermediate nodes and disabling adaptive routing) individually, and the combination of both mechanism (I+D) in a $3 \times 3 \times 3$ Torus network. Results for the three models of fault analysis (exhaustive, *distance-1* and probabilistic) are shown. As can be observed the D mechanism is not able to tolerate any fault. This result was expected since the deterministic routing

[2]The selection of the target node does not affect the results due to the symmetry property of torus networks.

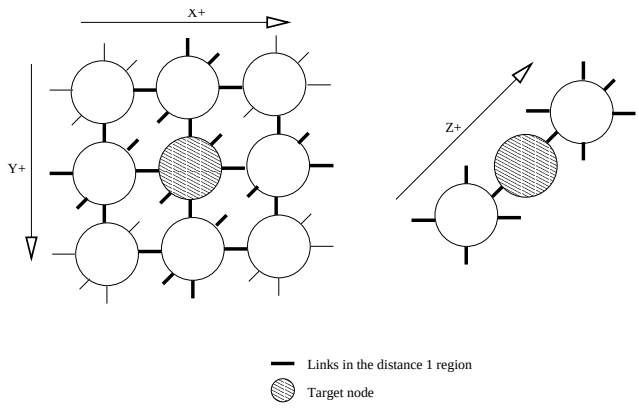

Fig. 3. Distance 1 region in a 3D Torus.

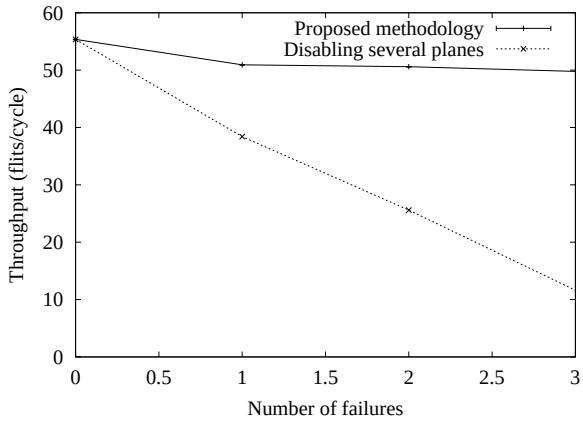

Fig. 4. Throughput (flits/cycle) degradation for the proposed methodology and for the mechanism used in the BlueGene/L supercomputer. $4 \times 4 \times 4$ Torus network.

used (DOR routing) is not even 1-fault tolerant. Regarding the I mechanism alone, we can observe that is 1-fault-tolerant. Moreover, the number of fault combinations not tolerated by the mechanism increases significantly as the number of the faults in the network increases.

Looking at the combinations of both mechanism, we can observe that it obtains a very good fault-tolerance degree. Results indicate that the methodology is 5-fault tolerant, as it tolerates all the possible combinations of up to 5 faults (exhaustive analysis). It is not 6-fault tolerant since there are some non-tolerated combinations in the distance-1 analysis for 6 fault, but the methodology tolerates quite well a large number of faults. We evaluated up to 14 faults in the network and found that the probability of having a combination that is not tolerated is smaller than 0.085. Notice that 14 link faults represent more than 7% of the total links. It is hard to imagine a machine that continues working with such a number of failures without being repaired.

B. Performance

We also analyzed the impact of the methodology on network performance. A detailed event-driven simulator has been developed to evaluate the performance behavior exhibited by the proposed methodology and to compare it with the mechanism used in the BlueGene/L supercomputer [3]. In both cases, each physical channel is split up into four virtual channels (two adaptive and two deterministic). A 4-ary 3-cube (64 nodes) network topology was used for the analysis.

In order to make the results independent of the relative positions of the faults, a large number of simulations (500) was run for each value of the number of faults. The results show that network performance is not seriously affected by the presence of failures. In particular, the throughput decreased by 10% for 3 faults and by 30% for 14 faults. To match our fault-tolerance degree other schemes either disable several healthy nodes [3], [10] or use additional virtual channels (but save on routing table space at the source nodes): up to 6 virtual channels in [5] and 4 additional virtual channels in [3]. Other approaches would require less resources [9] but at the expense of tolerating only two faults.

Finally, we compared the performance degradation when using our methodology versus the performance degradation that would be obtained by the fault-tolerant mechanism used in the BlueGene/L supercomputer. In this system, once a failure is detected, all the nodes included in the four planes (4,096 or 8,192 nodes) that contain the faulty node/link are marked as faulty. A special hardware at each switch bypasses the four planes and, therefore, the topology remains the same and routing is not changed. As we used a small torus network, we modeled the mechanism of the BlueGene/L by only disabling one plane. Figure 4 shows the overall network throughput obtained with both methodologies when there are up to 3 faults in the network. For the BlueGene/L, Figure 4 shows the worst but most probable case (i.e. every fault is located on a different plane).

As can be seen in Figure 4, the proposed methodology obtains a higher network throughput even if only one plane is disconnected by the BlueGene/L mechanism. Network throughput degrades only by up to 10% with 3 random faults when using the proposed methodology, whereas when using the BlueGene/L mechanism, the network performance drops by 79% when disabling three planes. However, these results must be put in context, as they are obtained in a small network. For larger networks, in particular for the $32 \times 32 \times 64$ torus used in the BlueGene/L supercomputer, disabling planes will have a lower impact. In that case, a fault would disconnect four planes of at least 32×32 nodes (4,096 out of 65,536 nodes). So, with one fault (or several faults located in the same disconnected planes) a 6.25% decrease in throughput can be expected. On the other hand, we expect that the performance degradation of our mechanism on larger networks will be negligible, as it does not disconnect healthy nodes and supplies adaptive paths through intermediate nodes in most cases.

V. CONCLUSIONS

Fault-tolerant mechanisms for interconnection networks are becoming a critical design issue for large massively parallel

[3]In the BlueGene/L project [11], 65,536 nodes are connected in a $32 \times 32 \times 64$ torus. Minimal adaptive routing is used, with two adaptive virtual channels. There are also two deterministic virtual channels. The first one is the escape channel for the adaptive ones, which uses bubble flow control [13] and Dimension Order Routing (DOR). The second one is for forwarding control packets. A static fault model is used.

TABLE I

PERCENTAGES OF LINK FAULT COMBINATIONS NOT TOLERATED BY EACH OF BOTH MECHANISMS AND BY THEIR COMBINATION. EXHAUSTIVE, DISTANCE-1 REGION AND PROBABILISTIC RESULTS ARE SHOWN. IN THE PROBABILISTIC RESULTS, THE ERROR IS ALWAYS LESS THAN 1%.

Link failures	Analysis type	# of combinations	Not tolerated combinations		
			D	I	I+D
1	Exhaustive	81	100%	0%	0%
2		3,240	100%	2.50%	0%
3		85,320	100%	7.44%	0%
4		1,663,740	100%	14.67%	0%
5		25,621,596	100%	24.06%	0%
6	*dist. 1*	1,107,568	100%	54.52%	0.0574%
7		4,272,048	100%	70.31%	0.352%
8		13,884,156	100%	83.30%	1.25%
6	Probabilistic	$> 6,000,000$	100%	35.46%	0%
7		$> 4,500,000$	100%	48.72%	0%
8		$> 3,400,000$	100%	62.98%	0%
9		$> 2,700,000$	100%	76.51%	0.00008%
10		$> 2,200,000$	100%	87.40%	0.48%
11		$> 1,700,000$	100%	94.47%	1.063%
12		$> 1,500,000$	100%	98.05%	2.79%
13		$> 1,300,000$	100%	99.46%	3.16%
14		$> 1,000,000$	100%	99.88%	8.47%

computers. In this paper, we have proposed a fully adaptive fault-tolerant routing mechanism for n-dimensional mesh and Torus network topologies. It assumes a static fault model and it is able to tolerate a reasonably large number of faults without significantly degrading performance. Moreover, unlike other fault-tolerant approaches, the proposed strategy does not need to disable any healthy node, does not require too extra hardware resources, and does not degrade performance in the absence of failures. In particular, in order to allow minimal adaptive routing and guarantee deadlock freedom, this strategy only requires one additional virtual channel.

In order to avoid faults, the new strategy selects an intermediate node (if needed) for routing packets between each source-destination pair. Packets can be adaptively routed to the intermediate node and, then, they can be adaptively forwarded to their destinations. Along both subpaths, packets are routed through minimal paths. To deal with particular fault configurations, some source-destination pairs communicate through non minimal paths, with or without the use of an intermediate node. Indeed, sometimes adaptive routing is also disabled in the subpaths.

Most important, the methodology can be applied to large networks, as the one used in the BlueGene/L supercomputer. Evaluation results show that the proposed methodology is 5-fault tolerant for 4-ary 3-cube (64 nodes). Indeed, the percentage of tolerated fault combinations is higher than 91,3% when up to 14 failures are considered. Additionally, network throughput degrades by less than 10% when injecting three random failures. In contrast, a mechanism similar to the one proposed in the BlueGene/L, which disables some network planes, degrades network throughput by 79%. Even better results are expected for larger networks.

REFERENCES

[1] Asci Red Web Site, http://www.sandia.gov/ASCI/Red/.

[2] R. Casado et al., "A protocol for deadlock-free dynamic reconfi guration in high speed local area networks," *IEEE Transations on Parallel and Distributed Systems*, Vol. 12, No. 2, 2001, pp. 115-132.

[3] S.Chalasani and R.V. Boppana. *Communication in multicomputers with nonconvex faults*. *IEEE Transactions on Computers*, vol. 46, no. 5, pp. 616-622, May 1997.

[4] A. A. Chien and J. H. Kim. *Planar- adaptative routing: Low- cost adaptative networks for multiprocessors*. Proceedings of the 19th International Symposium on Computer Architecture, pp. 268-277, May 1992.

[5] W. J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputers Networks Using Virtual Channels," *IEEE Transations on Parallel and Distributed Systems*, Vol. 4, No. 4, 1993, pp. 466-475.

[6] W. J. Dally et al., "The Reliable Router: A Reliable and High-Performance Communication Substrate for Parallel Computers," *Proc. Parallel Computer Routing and Communication Workshop*, 1994.

[7] J. Duato. *A theory of fault-tolerant routing in wormhole networks*. Proceedings of the International Conference on Parrallel and Distributed Systems, pp. 600-607, December 1994.

[8] Earth Simulator Center, http://www.es.jamstec.go.jp/esc/eng/index.html.

[9] G.J. Glass, and L.M. Ni. Fault-Tolerant Wormhole Routing in Meshes without Virtual Channels. *IEEE Transactions Parallel and Distributed Systems*, vol. 7, no. 6, pp. 620-636, 1996.

[10] C.T. Ho and L. Stockmeyer, "A New Approach to Fault-Tolerant Wormhole Routing for Mesh-Connected Parallel Computers," *Proc. 16th International Parallel and Distributed Processing Symposium*, 2002.

[11] IBM BG/L Team, "An Overview of the BlueGene/L Supercomputer," *Proc. ACM Supercomputing Conference*, 2002.

[12] D.H. Linder and J.C. Harden. *An Adaptative and fault tolerant wormhole routing strategy for k-ary n-cubes*. IEEE Transactions on Computers, vol. C-40 no 1, pp.2-12, Jan. 1991.

[13] V. Puente, J.A. Gregorio, J.M. Prellezo, R. Beivide, J. Duato, and C. Izu, "Adaptive Bubble Router: A Design to Balance Latency and Throughput in Networks for Parallel Computers," *Proc. 22nd International Conference on Parallel Processing*, 1999.

[14] Y.J. Suh, B.V. Dao, J. Duato, and S.Yalamanchili, "Software-based rerouting for fault-tolerant pipelined communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 3, 2000, pp. 193-211.

[15] L.G. Valiant, *A Scheme for Fast Parallel Communication*. SIAM J. Comput. 11, pp. 350-361, 1982.