A Fully Adaptive Fault-Tolerant Routing Methodology Based on Intermediate Nodes*

N.A. Nordbotten[1], M.E. Gómez[2], J. Flich[2], P. López[2], A. Robles[2], T. Skeie[1], O. Lysne[1], and J. Duato[2]

[1] Simula Research Laboratory, P.O. Box 134, N-1325 Lysaker, Norway
nilsno@simula.no
[2] Dept. of Computer Engineering, Universidad Politécnica de Valencia, Camino de Vera, 14, 46071-Valencia, Spain
megomez@gap.upv.es

Abstract. Massively parallel computing systems are being built with thousands of nodes. Because of the high number of components, it is critical to keep these systems running even in the presence of failures. Interconnection networks play a key-role in these systems, and this paper proposes a fault-tolerant routing methodology for use in such networks. The methodology supports any minimal routing function (including fully adaptive routing), does not degrade performance in the absence of faults, does not disable any healthy node, and is easy to implement both in meshes and tori. In order to avoid network failures, the methodology uses a simple mechanism: for some source-destination pairs, packets are forwarded to the destination node through a set of intermediate nodes (without being ejected from the network). The methodology is shown to tolerate a large number of faults (e.g., five/nine faults when using two/three intermediate nodes in a 3D torus). Furthermore, the methodology offers a gracious performance degradation: in an $8 \times 8 \times 8$ torus network with 14 faults the throughput is only decreased by 6.49%.

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

1 Introduction

There exist many compute-intensive applications that require a huge amount of processing power, and this required computing power can only be provided by massively parallel computers. Examples of these systems are the Earth Simulator [12], the ASCI Red [1], and the BlueGene/L [2].

The huge number of processors and associated devices (memories, switches, links, etc.) significantly increases the probability of failure. It is therefore critical to keep such systems running even in the presence of failures. Much work deal

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

H. Jin et al. (Eds.): NPC 2004, LNCS 3222, pp. 341–356, 2004.

with failures of processors and memories. In this paper, we consider failures in the interconnection network. These failures may isolate a large fraction of the machine, wasting many healthy processors that otherwise could have been used. Therefore, fault-tolerant mechanisms for interconnection networks are becoming a critical design issue for large massively parallel computers.

There exist several approaches to tolerate failures in the interconnection network. The most prominent technique in commercial systems consists of replicating components. The spare components are switched on in the case of failure while switching off (or bypassing) the faulty components. The main drawback of this approach is the high extra cost of the spare components. Another powerful technique is based on reconfiguring the routing tables in the case of failure, adapting them to the new topology after the failure [5]. 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 these 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 [9] or the number of dimensions in the topology [17]. 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 [9,14], disabling a certain number of healthy nodes (either in blocks (fault regions) [6,7] or individually [10,11]), preventing packets from being routed adaptively [15], or drastically increasing the latencies for some packets [19]. 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 [13] we proposed a fault-tolerant routing methodology for n-dimensional meshes and tori, and that only requires one extra virtual channel. In order to avoid network failures, an intermediate node is used for some source-destination pairs.[3] This node is selected in such a way that the faults are avoided when the packets are routed first to the intermediate node and then from this node to the destination node. However, in order to tolerate an acceptable number of faults, an additional mechanism is used, that is, disabling adaptive routing for some paths (i.e., routing packets deterministically).

Disabling adaptivity has two main drawbacks: The first is that it has a negative impact on network performance, because it prevents packets from being adaptively routed. The second is that it needs additional complexity at the routers in order to enable turning off adaptivity on a per packet basis. For these reasons it would be beneficial to use a single mechanism only. In this paper we propose a methodology solely based on intermediate nodes, but instead of using only one intermediate node, we propose to use several ones in order to cir-

[3] Intermediate nodes were introduced by Valiant [21] for other purposes, such as traffic balance.

cumvent faulty components. This way, regardless of the number of intermediate nodes being used, the way packets are being routed does not need to be modified, allowing the same router design as in the absence of intermediate nodes to be used. Furthermore, this methodology allows all packets to be adaptively routed, which again contributes to a good network performance.

On the other hand, this approach requires using additional virtual channels as long as more intermediate nodes are used. However, virtual channels are nowadays inexpensive. Current interconnects are able to provide several virtual channels. This is the case for the Cray T3E [20] with five virtual channels, the BlueGene/L [2] with four virtual channels, and InfiniBand switches [16] with 16 virtual channels.

Still, when designing a fault-tolerant routing scheme that requires extra virtual channels, it is desirable to use a bounded number of virtual channels. At the same time one should also tolerate a reasonably large number of faults, avoid disabling any healthy node, maintain a low router complexity, and guarantee routing through adaptive paths in order to provide high network performance both in the absence and in the presence of faults.

The rest of the paper is organized as follows. In Sect. 2, the methodology is presented. The methodology is then illustrated through some example scenarios in Sect. 3. In Sect. 4, the routing algorithm obtained by the methodology is analyzed in terms of performance and fault-tolerance. Finally, in Sect. 5, some conclusions are drawn.

2 The Methodology

The methodology for achieving fault-tolerance through the use of one or more intermediate nodes will now be presented. We will assume a k-ary n-cube (torus) or n-dimensional mesh network. The methodology is valid for any minimal routing function, although it is applied to minimal adaptive routing [18] in this paper. Minimal adaptive routing with v virtual channels allows the use of any minimal path through $v-1$ virtual (adaptive) channels, whereas the last channel (i.e., the escape channel) uses deterministic routing. Thus, at least two virtual channels per physical channel ($v = 2$) are required. In a torus the escape channel also uses the bubble flow control mechanism [4].

Furthermore, a static fault model is assumed. This means that when a fault is discovered all the processes are stopped, the network is emptied, and a management application is run in order to deal with the fault. Checkpointing techniques must also be used so that applications can be brought back to a consistent state prior to the fault occurred. Detection of faults, checkpointing, and distribution of routing info is assumed to be performed as part of the static fault model, and are therefore not further discussed in this paper.

A fault-free path is computed by the methodology for each source-destination pair. In the presence of faults, those paths that may use some faulty components are not valid. The methodology avoids these faults by using intermediate nodes. Packets are first forwarded to the first intermediate node, then from this node

to the second one, and so forth until the packet reaches its final destination. As shown in Fig. 1, the use of intermediate nodes reduces the number of possible paths, and therefore enables avoiding areas containing faults. The original routing algorithm (e.g., minimal adaptive routing) is used in all subpaths. Notice that the packets are not ejected from the network at each intermediate node.

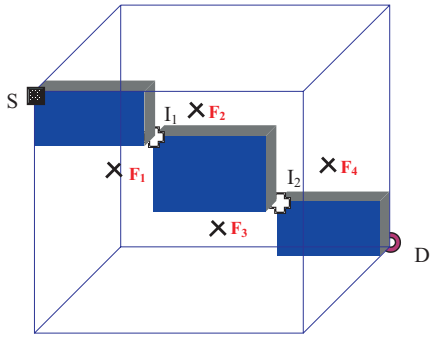

Fig. 1. The use of intermediate nodes (I) limits the number of possible paths, from the source (S) to the destination (D), enabling faults (F) to be avoided

Packets sent through intermediate nodes carry the address of each intermediate node, in addition to the address of the final destination. As the packet reaches each intermediate node, the address of that intermediate node is removed from the packet header, until the packet finally reaches its true destination. In addition, every source node must maintain a table specifying the intermediate node(s) to be used for each destination that requires such measures to be taken. When there are several candidates for the intermediate node(s), one of the alternatives can be selected randomly or more than one alternative could be listed in order to provide additional routing flexibility.

In what follows we will denote the source node as S and the destination node as D. The intermediate nodes are denoted I_x, where I_1 refers to the first intermediate node in a route. Faulty links are denoted as F_i. A node failure can easily be modelled as the failure of all the links of a node.

Deadlock freedom is ensured by having a separate escape channel for each phase. E.g., with two intermediate nodes, one escape channel is used (if required) from S to I_1, another from I_1 to I_2, and a third one from I_2 to D. This way, each phase defines a virtual network, and the packets change virtual network at each intermediate node. Although each virtual network relies on a different escape channel, they all share the same adaptive channel(s). If y is the allowed number of intermediate nodes for each source destination pair, and the minimal adaptive routing algorithm uses one adaptive and one escape channel per physical channel, the methodology requires a total of $y + 2$ virtual channels. Notice that one of them corresponds to the escape channel used in the minimal adaptive routing algorithm. So, for two intermediate nodes, four virtual channels are required.

The escape channels use deterministic Dimension Order Routing (DOR) with the bubble flow control mechanism. With this mechanism, a packet that is injected into the network or cross a network dimension requires two free buffers (i.e., one for the packet itself and one additional free buffer) to guarantee deadlock freedom. Hence, in order to avoid deadlocks, a packet changing virtual network at an intermediate node should be considered as crossing a dimension, and therefore requires two free buffers.

The computational complexity for identifying one intermediate node is $O(1)$ in torus and mesh topologies. For all the paths in the network the computational complexity thus becomes $O(n^2)$. When using two intermediate nodes this increases to $O(n^3)$ in the worst case. However, as we will see in Sect. 4, the number of paths using more than one intermediate node is very low even when there are many faults (e.g., 0.000001% of the paths in a $3 \times 3 \times 3$ torus with six faults). Thus, the methodology has a low computational cost, especially when considering that a static fault model is used.

Next, a methodology for identifying the intermediate nodes is presented. First, the case where only one intermediate node is used is presented. We then show how the method can be extended to the use of multiple intermediate nodes.

2.1 One Intermediate Node

When at most one intermediate node is used for each source-destination pair, the intermediate node I_1 should have the following properties so that the fault(s) F_i are avoided when routing packets from S via I_1 to D:

1. I_1 is reachable from S.
2. D is reachable from I_1.
3. There is no I_1' giving a shorter path than I_1.

The first requirement guarantees that packets can be routed from S to I_1, and the second requirement guarantees that packets can be routed from I_1 to D. The third requirement guarantees that the final path is the shortest possible.

Also notice that, when minimal adaptive routing is used, a node N_2 is *reachable* from a node N_1 if and only if: For all i, F_i is not on any minimal path from N_1 to N_2.

To identify the possible intermediate nodes, let $\mathcal{T}_{RS}$ be the set of nodes reachable from S and $\mathcal{T}_D$ the set of nodes from which D is reachable. Furthermore, let $l(x, y)$ be the length of the minimal path, in the fault free case, from x to y. We then define $\mathcal{T}_j$ (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, as shown in figure 2. These sets can easily be identified by starting with the nodes that are reached (i.e., traversed) on any minimal path from S to D (i.e., $j = 0$), and continuing outwards. As can be seen in Figure 2, the sets $\mathcal{T}_j$ are non-empty only for even values of j. This is always the case for meshes, but not always for tori (due to the wraparound links).

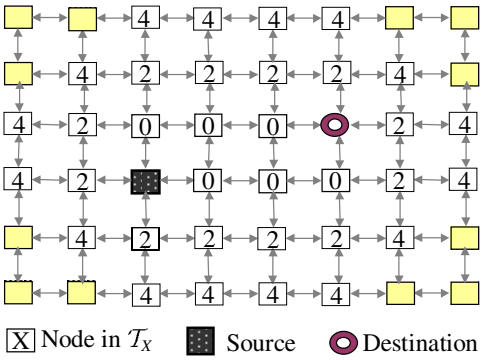

Fig. 2. The nodes in the sets $\mathcal{T}_j$, for $j \leq 4$ in a 2D mesh

Theorem 1. *Let j' be the smallest j for which $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}_D$ is non-empty. A node N fulfills all three requirements, if and only if, $N \in \mathcal{T}_{j'} \cap \mathcal{T}_{RS} \cap \mathcal{T}_D$.*

Proof. We prove the theorem by induction. The theorem is true for $j = 0$ (i.e., when a minimal route exists):

- Let us assume that there is one node N in the set that does not fulfill the requirements. Then N would either have to be unreachable from S, not have a valid route to D, or not be on a minimal path from S to D. If N is unreachable from S it is by definition not in $\mathcal{T}_{RS}$. If N does not have a valid route to D it is by definition not in $\mathcal{T}_D$. If N is not on a minimal path from S to D it is by definition not in $\mathcal{T}_0$. Because of the properties of set intersection, N must be in all the three sets $\mathcal{T}_{RS}$, $\mathcal{T}_D$, and $\mathcal{T}_0$ to be in the set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}_D$. Thus, we have a contradiction.
- Let us then assume that there is one node N, outside the set, which fulfills the requirements. N would then have to be outside at least one of the sets $\mathcal{T}_{RS}$, $\mathcal{T}_D$, or $\mathcal{T}_0$. If N is outside $\mathcal{T}_{RS}$ it is unreachable from S and therefore does not fulfill requirement one. If N is outside $\mathcal{T}_D$ it has no valid route to D and therefore does not fulfill requirement two. If N is outside $\mathcal{T}_0$ it violates our assumption that a minimal route exists (i.e., that $j = 0$). Thus, we have a contradiction in all three cases.

If the theorem is true for $j = m$, then the theorem is also true for $j = m + 1$: Concerning requirements one and two, the arguments made for $j = 0$ also hold for $j = m+1$. Furthermore, when $j = m+1$, no route S-I_1-D exists for $j < m+1$. Indeed, as each increase of j adds one additional hop to the path S-I_1-D, all the intermediate nodes found when $j = m + 1$ yield paths S-I_1-D of equal lengths. Finally, for the same reason, no shorter path can be found for $j > m + 1$. The theorem, therefore, fulfills all three requirements. □

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

2.2 Multiple Intermediate Nodes

In cases where one intermediate node is insufficient, to avoid all the faults when routing from S to D, two or more intermediate nodes can be used. The use of multiple intermediate nodes may also enable shorter paths than those otherwise obtained when using fewer intermediate nodes.

We will now first present a methodology for using two intermediate nodes. We then generalize this methodology so that it can be used, in a recursive way, for any number of intermediate nodes.

Two Intermediate Nodes. When using two intermediate nodes, we are looking for intermediate nodes I_1 and I_2 so that:

- I_1 is reachable from S.
- I_2 is reachable from I_1.
- D is reachable from I_2.
- There are no I_1' and I_2' giving a shorter path than S-I_1-I_2-D.

However, it can be observed that if a suitable I_1 is identified, then the second intermediate node I_2 follows from Theorem 1. Thus, the problem can be reduced to identifying I_1.

In order to solve this problem, let us introduce a variation of $\mathcal{T}_D$, namely $\mathcal{T}_{D1}^k$. We define this new set as the set of nodes that can reach D through one intermediate node (i.e., the 1 in the subscript denotes that one intermediate node is used). This intermediate node is given by Theorem 1, and k here represents the j in the set $\mathcal{T}_j$ used, with Theorem 1, for identifying it. E.g., the set $\mathcal{T}_{D1}^0$ consists of the nodes that have minimal path, via one intermediate node, to D. The set $\mathcal{T}_{D1}^1$, on the other hand, consists of nodes which have a path length equal to the minimal path plus one, via one intermediate node, to D. As before, $\mathcal{T}_{RS}$ denotes the nodes reachable from S.

Theorem 2. *Let j' and k' be the smallest j and k (i.e., so that their sum is minimized) for which $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D1}^k$ is non-empty. A node N fulfills all four requirements if, and only if, $N \in \mathcal{T}_{j'} \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D1}^{k'}$.*

Proof. Let us define l as the sum of j and k, i.e., $l = j + k$. We then prove the theorem by induction. The theorem is true for $l = 0$ (i.e., when a minimal path exists):

- Let us assume that there is one node N in the set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D1}^0$ that gives a path S-N-I_2-D that does not fulfill the requirements. It follows from Theorem 1, and the definition of $\mathcal{T}_{D1}^k$, that I_2 is reachable from N and that D is reachable from I_2. Thus, N must be unreachable from S or the path S-N-I_2-D is not the shortest possible. If N is unreachable from S, N is by definition not in $\mathcal{T}_{RS}$. It also follows from Theorem 1 that the subpath N-I_2-D is the shortest possible. Thus, N can not be on a minimal path from S to D for the path S-N-I_2-D to be a non-minimal path. However, then N is by definition not in $\mathcal{T}_0$. Therefore, we have a contradiction.

– Let us then assume that there is one node N outside the set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}^0_{D1}$ that fulfills the requirements. N would then have to be outside at least one of the sets $\mathcal{T}_0$, $\mathcal{T}_{RS}$, or $\mathcal{T}^0_{D1}$. If N is outside $\mathcal{T}_0$ it violates our assumption that $l = 0$. If N is outside $\mathcal{T}_{RS}$, it is unreachable from S and therefore violates requirement one. If N is outside of the set $\mathcal{T}^0_{D1}$ it violates requirements two or three, or our assumption that $l = 0$.

If the theorem is true for $l = m$, then it is also true for $l = m + 1$: As for reachability, the same arguments as for $l = 0$ are still valid. Thus, it only remains to be shown that the path S-N-I_2-D is the shortest possible. By definition, when $l = m+1$, no N exists for $l < m+1$. Each increase of l adds one hop to the path S-N-I_2-D. Thus, all paths where $l = m + 1$ are of equal length, and no shorter path can be found for $l > m + 1$. □

Thus, as before, we start considering the minimal paths (i.e., $j + k = 0$) and then consider non-minimal paths (i.e., $j + k > 0$), if necessary, to avoid all the faults.

Any Number of Intermediate Nodes. Let us now generalize the definition of $\mathcal{T}^k_{D1}$, in order to apply Theorem 2 for any number of intermediate nodes. We therefore define $\mathcal{T}^k_{Dz}$ in the following way:

– $\mathcal{T}^0_{D0}$: The set of nodes from which D is reachable without the use of any intermediate node (i.e., the set of nodes defined by the original set $\mathcal{T}_D$).
– $\mathcal{T}^k_{Dz}$, for $z > 0$ and $k \geq 0$: The set of nodes given by $\mathcal{T}_{j'} \cap \mathcal{T}_{RS} \cap \mathcal{T}^{k'}_{Dz'}$, where $z = z' + 1$ and $k = j' + k'$.

Thus, $\mathcal{T}^k_{Dz}$ is the set of nodes that reach D through z intermediate nodes, and where k is the number of additional hops, in the path to D, compared to the minimal path. When paths of equal length exist, preference should be given to paths with fewer intermediate nodes.

Notice that the set $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}^0_{D0}$ is actually the same as that in Theorem 1, and thus results in paths with one intermediate node. The set $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}^k_{D1}$ is that given by Theorem 2, resulting in paths with two intermediate nodes. Similarly, $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}^k_{D2}$ gives paths with three intermediate nodes. Continuing this way, an arbitrary number of intermediate nodes can be obtained.

3 Example Scenarios

We will now illustrate the methodology through two example scenarios. A 2D mesh is used for this purpose, although the methodology is also valid for other topologies such as a 3D mesh or torus. For both scenarios we assume that minimal adaptive routing is used, and that at most two intermediate nodes are allowed in each route.

Figure 3a shows a scenario with three faults. Because there are faults present in some of the minimal paths between S and D, an intermediate node is needed.

In order to find a minimal path, we look for an intermediate node within $\mathcal{T}_0$. As shown in Fig. 3a, there are several nodes within $\mathcal{T}_0$ that are either reachable from S, or able to reach D. However, we are only interested in nodes with both of these attributes, i.e., the nodes given by the set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}^0_{D0}$. In this scenario there is only one such node, i.e., the one identified as a possible intermediate node in the figure. By using this node as the intermediate node, it is guaranteed that the faults are not encountered when packets are routed first from S to I_1 and then from I_1 to D.

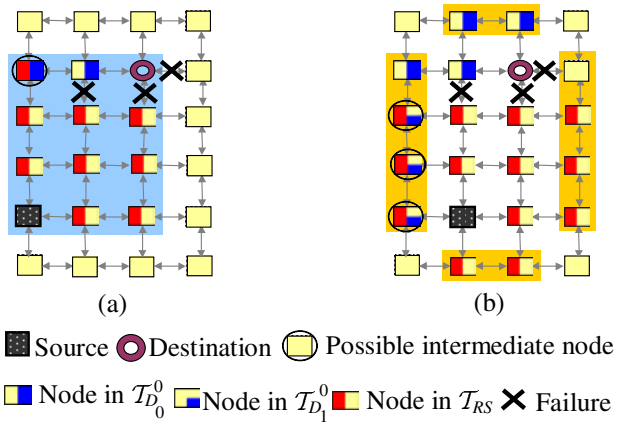

Fig. 3. (a) The faults are avoided by the use of one intermediate node. The shaded area identifies the nodes in $\mathcal{T}_0$. (b) Two intermediate nodes must be used in order to avoid the faults. The figure shows how the first of these intermediate nodes (i.e., I_1) is identified. The shaded areas identify the nodes in $\mathcal{T}_2$

Figure 3b shows the same fault scenario as in the previous example, except that the source node is different. In this case, all the minimal paths between S and D are blocked by faults. The set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}^0_{D0}$, giving minimal paths with one intermediate node, is therefore empty. The set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}^0_{D1}$, giving minimal paths with two intermediate nodes, is also empty. Because preference is given to the paths with the least number of intermediate nodes when the path length is equal, we then try to find an intermediate node within $\mathcal{T}_2$ (because this is a mesh we are only interested in the even values of j) giving a non-minimal path with one intermediate node. However, this set, $\mathcal{T}_2 \cap \mathcal{T}_{RS} \cap \mathcal{T}^0_{D0}$, is also empty.

There are now two more sets giving the same path lengths as the previous one, but using two intermediate nodes instead of one. Which of these two sets are given preference is irrelevant for the correctness of the methodology as they both give the same value for $j + k$ (which should be minimized according to Theorem 2). Increasing j means adding one hop to the path. S-I_1, while increasing k adds one hop to the path I_1-I_2-D.

Anyway, of the two sets, the set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}^2_{D1}$ is empty, while $\mathcal{T}_2 \cap \mathcal{T}_{RS} \cap \mathcal{T}^0_{D1}$ gives us the possible intermediate nodes shown in Fig. 3b. Thus, the first intermediate node, I_1, can be selected among these three nodes. If I'_1 is the first intermediate node, then the second intermediate node, I_2, can be selected among the intermediate nodes that give I'_1 a path with one intermediate node to D. In this case, I_2 would be the same as the one identified as I_1 in the first example.

4 Evaluation of the Methodology

In this section, we evaluate the proposed methodology. In a first study, we analyze the fault-tolerance properties of the methodology, i.e., how many faults the mechanism is able to tolerate. The methodology is $n-$fault tolerant if it is able to tolerate any combination of n failures. A given combination of failures, again, is tolerated if the methodology is able to provide a valid path for every source-destination pair in the network. On the other hand, faults can physically disconnect some nodes in the network. In this situation, disconnected nodes are not taken into account and, provided that the paths for the remaining nodes can be computed, the fault combination is considered as tolerated.

Then, we evaluate how the methodology influences network performance. For this, network throughput has been measured for different numbers of faults. For each number of faults, 50 randomly generated fault combinations have been simulated, and the average network throughput for these combinations is provided.

We have applied the methodology to $3 \times 3 \times 3$ (27 nodes) torus and mesh networks, to a 3×3 (9 nodes) torus network, and to an $8 \times 8 \times 8$ (512 nodes) torus network. Although actual systems are built with larger topologies (e.g., a $32 \times 32 \times 64$ torus for the BlueGene/L), smaller networks can be evaluated exhaustively from a fault-tolerant point of view and the results can then easily be extended to larger networks.

4.1 Simulation Model

A detailed event-driven simulator has been used to analyze the performance exhibited by the proposed methodology. The simulator models a direct interconnect network with point-to-point bidirectional serial links. Each router has a non-multiplexed crossbar with queues only at the input ports. Each physical input port uses five virtual channels, each providing buffering resources in order to store up to two packets. A round-robin policy has been chosen to select among packets contending for the same output port.

In order to make a fair evaluation, the same number of virtual channels (i.e., five) is used regardless of the number of intermediate nodes used in the methodology. Virtual channels are used as adaptive or as escape channels, depending on the number of required intermediate nodes. If paths use at most one intermediate node, three virtual channels are used for adaptive routing, whereas the

remaining two virtual channels are used for the escape paths. For two intermediate nodes, there are two adaptive channels and three are escape channels, and so on. When faults are not present (i.e., when no intermediate nodes are required), four adaptive channels are used. In the escape channel(s), packets are deterministically routed following the DOR routing and the bubble flow control mechanism. Notice that for a given number of intermediate nodes, all the paths in the network will have the same number of adaptive virtual channels, regardless of whether they use intermediate nodes or not.

For each simulation run, we assume that the packet generation rate is constant and the same for all of the nodes. The destination of a message is randomly chosen with the same probability for all the nodes. This pattern has been widely used in other evaluation studies [3,8]. In all the simulations, the packet length is set to 128 bytes.

4.2 Fault Analysis Models

For a reduced 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 tackle this problem with two approaches. In the first approach, we focus on faults bounded into a limited region of the network. Notice that the worst combinations of faults to be solved by the methodology are those where the faults are closely located. This is because the number of fault-free paths in that region is reduced. Because the number of fault combinations within such a region is much lower 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 may be located over the entire network, it gives 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 attached to the nodes that are one hop away from a node (the center node). Therefore we refer to this as a distance 1 region, and it consists of 36 links. However, in a $3 \times 3 \times 3$ torus it only consists of 33 links, as three of the links then are shared by nodes within the region. The center node is randomly selected.[4] Notice that with a high number of faults and for a large number of fault combinations, the center node is hardly accessible, as very few links are non-faulty. So, the distance 1 region actually represents a real worst case to access the center node.

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

[4] The selection of the center node does not affect the results in a torus network due to the symmetry of the topology.

Table 1. Fault tolerance achieved by the methodology when using at most one ($I \times 1$), two ($I \times 2$), or three ($I \times 3$) intermediate nodes in a $3 \times 3 \times 3$ torus network. The three rightmost columns show the percentages of the paths that use each number of intermediate nodes when at most three intermediate nodes are used

Link faults	Analysis type	Combinations analyzed	Not tolerated combinations			I×3 Paths using # I		
			I×1	I×2	I×3	1	2	3
1	Exhau-	81	0%	0%	0%	6.86%	0%	0%
2	stive	3,240	2.50%	0%	0%	12.99%	0.04%	0%
3		85,320	7.44%	0%	0%	18.46%	0.13%	0%
4		1,663,740	14.67%	0%	0%	23.32%	0.31%	0%
5		25,621,596	24.06%	0%	0%	27.62%	0.56%	0%
6		324,540,216	35.49%	0.0002%	0%	31.41%	0.90%	0.000001%
6	Dist.1	1,107,568	54.52%	0.01%	0%	28.09%	1.19%	0.00003%
7		4,272,048	70.31%	0.06%	0%	30.41%	1.78%	0.0004%
8		13,884,156	83.30%	0.31%	0%	32.25%	2.51%	0.002%
9		38,567,100	92.15%	1.06%	0%	33.67%	3.38%	0.008%
10		92,561,040	96.97%	2.99%	0.001%	34.71%	4.36%	0.02%
11		193,536,720	99.01%	6.51%	0.01%	35.42%	5.44%	0.05%
12		354,817,320	99.67%	12.88%	0.62%	35.84%	6.58%	0.11%
6	Stati-	10,000,000	35.46%	0.00%	0%	31.41%	0.90%	0.000001%
7	stical	10,000,000	48.72%	0.00%	0%	34.72%	1.34%	0.00001%
8		10,000,000	62.98%	0.01%	0%	37.61%	1.88%	0.00007%
9		10,000,000	76.51%	0.03%	0%	40.10%	2.53%	0.0002%
10		10,000,000	87.40%	0.09%	0%	42.21%	3.29%	0.0008%
11		10,000,000	94.47%	0.23%	0%	43.98%	4.16%	0.002%
12		10,000,000	98.05%	0.52%	0.00001%	45.44%	5.14%	0.005%
13		10,000,000	99.46%	1.10%	0.0003%	46.60%	6.22%	0.01%
14		10,000,000	99.88%	2.13%	0.0009%	47.50%	7.41%	0.02%

4.3 Evaluation Results

Table 1 shows the fault tolerance achieved by the methodology for a $3 \times 3 \times 3$ torus network. The table shows the results for the three different types of analysis performed (exhaustive, distance 1, and statistical). From the exhaustive analysis results, we can observe that the methodology is only 1-fault tolerant when using only one intermediate node. For two faults present in the network, 2.5% of the fault combinations are not tolerated when using one intermediate node. As the number of faults increases, the percentage of not tolerated combinations grows fast. For six faults, 35.49% of the fault combinations are not supported when using one intermediate node.

By using two intermediate nodes, the methodology greatly increases its fault tolerance degree. In particular, it is 5-fault tolerant as all the fault combinations of up to and including five faults are tolerated. With six faults in the network, two intermediate nodes where sufficient for almost all the fault combinations, except for 0.0002% of the combinations.

With three intermediate nodes, the methodology achieves a very good fault tolerance degree. From the exhaustive analysis results, we observe that using three intermediate nodes allows tolerating all the possible fault combinations up to and including six faults. In the statistical analysis, where 10 million randomly generated fault combinations were analyzed for up to 14 faults[5], the methodology could provide a valid path for every non-disconnected pair of nodes for up to 11 faults. For 12 faults one not tolerated combination was found (in 10,000,000) and this number increased to 89 when 14 faults were present.

However, taking into account the distance 1 analysis (representing the worst case situation) we can observe that with 10 faults in the network there were some not tolerated combinations. Therefore, the methodology is not 10-fault tolerant. Even, from seven up to and including nine faults we can not deduce for sure that the methodology is n-fault tolerant since not all the fault combinations have been tested. However, this strongly indicates that the methodology tolerates nine faults. Anyway, even with a high number of faults, the percentage of not supported fault combinations is very low when using three intermediate nodes. So, the methodology achieves a high fault-tolerance degree.

Table 1 also shows the percentage of paths that use a certain number of intermediate nodes. As can be seen, most of the paths avoid faults by using just one intermediate node, and very few paths need a third intermediate node. Notice that although the third intermediate node is little used, it makes a large difference for the fault-tolerance degree.

Table 2 shows the results achieved for a 3×3 torus network and for a $3 \times 3 \times 3$ mesh network. In the 3×3 torus, all the combinations of up to and including six faults (i.e., 1/3 of the total number of links) have been exhaustively analyzed. The methodology tolerates one fault when using one intermediate node, three faults when using two intermediate nodes, and five faults when using three intermediate nodes. So, the fault-tolerance degree in a 2D torus is lower than in a 3D torus. This is not unexpected considering that a 2D torus provides lower routing flexibility.

For the $3 \times 3 \times 3$ mesh network, the results are not as good as for the torus networks. The methodology requires at least two intermediate nodes in order to be 1-fault tolerant.[6] When using three intermediate nodes, the methodology is 4-fault tolerant, and it is 6-fault tolerant when using four intermediate nodes.

Finally, Fig. 4 shows the performance degradation exhibited by the methodology in an $8 \times 8 \times 8$ torus network when up to two intermediate nodes are used. Notice that in a larger network, like the one used in the performance analysis, the percentage of not tolerated fault combinations, when using two intermediate nodes, is much lower than in a $3 \times 3 \times 3$ torus. Thus, all the randomly generated combinations for the performance evaluation could be solved by the use of at most two intermediate nodes. When only one fault is present in the network,

[5] The error due to not analyzing all the combinations is lower than 0.05.

[6] Notice that in a mesh the wraparound links do not exist and it is therefore impossible to communicate to a node on the direct opposite side of the fault without using at least two intermediate nodes (i.e., when S, F, and D are in the same row/column).

Table 2. Fault tolerance degree achieved by the methodology for a 3×3 torus network and a $3 \times 3 \times 3$ mesh network. The table shows the percentage of the total number of combinations that are not tolerated. The results have been obtained by exhaustively analyzing all the possible fault combinations

Link	3×3 torus			$3 \times 3 \times 3$ mesh			
faults	I×1	I×2	I×3	I×1	I×2	I×3	I×4
1	0%	0%	0%	100%	0%	0%	0%
2	11.76%	0%	0%	100%	0%	0%	0%
3	33.82%	0%	0%	100%	0.97%	0%	0%
4	67.06%	1.18%	0%	100%	4.23%	0%	0%
5	91.81%	10.71%	0%	100%	11.65%	0.05%	0%
6	96.49%	40.24%	2.33%	100%	24.89%	0.28%	0%
7	N/A	N/A	N/A	100%	43.67%	1.02%	0.002%
8	N/A	N/A	N/A	100%	64.53%	2.83%	0.02%

only one intermediate node is used. The figure shows for every number of faults, the average throughput achieved. The presented throughput is the average of the individual results obtained when evaluating the 50 randomly generated fault combinations.[7] As can be observed, the throughput decreases as the number of faults in the network increases. However, the decrease in throughput, is very low. In particular, when there are 14 faults, the throughput is on average only decreased by 6.49% compared to the fault-free case (from 474 flits/cycle to 443 flits/cycle). In particular, this degradation is lower than the one obtained with the methodology proposed in [13], where with 5 virtual channels the degradation from the fault-free case to 14 faults was 11.02%.

5 Conclusions

In this paper we have proposed a fault-tolerant routing methodology based on the use of intermediate nodes. The proposed methodology can be applied with any minimal routing function in n-dimensional mesh and torus networks, and it has been applied with minimal adaptive routing in this paper. The main advantage of the proposed mechanism is its simplicity, since the same original routing (e.g., minimal adaptive) continues to be valid. The only requirement on switches is that they should provide the required number of extra virtual channels. However, only a low number of virtual channels is required.

The paper provides the necessary and sufficient conditions, for selecting the intermediate nodes, in order to tolerate as many faults as possible and to provide the shortest paths possible.

The methodology has been shown to be five fault-tolerant when using two intermediate nodes in a 3D torus network. When using three intermediate nodes,

[7] The 95% confidence intervals are always smaller than 0.796.

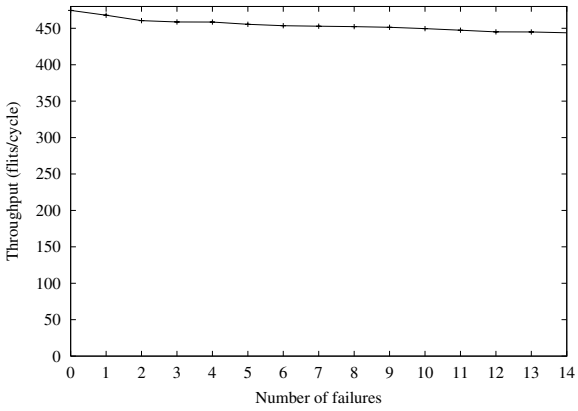

Fig. 4. Overall throughput (flits/cycle) for the proposed methodology in an $8 \times 8 \times 8$ torus network. Five virtual channels are used

the method is nine fault-tolerant for 3D torus networks, five fault-tolerant in 2D torus, and four fault-tolerant in 3D mesh topologies.

Regarding performance, the methodology does not degrade performance in the absence of faults, whereas in the presence of faults it provides a gracious performance degradation. Specifically, it has been shown that the average performance degradation, in an $8 \times 8 \times 8$ torus network with 14 faults, is only 6.49%.

References

1. ASCI Red Web Site. http://www.sandia.gov/ASCI/Red/.
2. IBM BG/L Team. *An Overview of the BlueGene/L Supercomputer.* ACM Supercomputing Conference, 2002.
3. R. Bopana and S. Chalasani. *A Comparison of Adaptive Wormhole Routing Algorithms.* Proc. 20th Annual Int. Symp. Comp. Architecture, 1993.
4. C. Carrion, R. Beivide, J.A. Gregorio, and F. Vallejo. *A Flow Control Mechanism to Avoid Message Deadlock in K-ary N-Cube Networks.* 4th International Conference on High Performance Computing, pp. 332-329, 1997.
5. R.Casado, A. Bermúdez, J. Duato, F.J. Quiles, and J.L. Sánchez. *A protocol for deadlock-free dynamic reconfiguration in high speed local area networks.* IEEE Transactions on Parallel and Distributed Systems, vol. 12, No. 2, pp. 115-132, 2001.
6. A.A. Chien and J.H. Kim. *Planar-adaptive routing: Low-cost adaptive networks for multiprocessors.* Proceedings of the 19th International Symposium on Computer Architecture, pp. 268-277, 1992.
7. S.Chalasani and R.V. Boppana. *Communication in multicomputers with nonconvex faults.* IEEE Transactions on Computers, vol. 46, no. 5, pp. 616-622, 1997.
8. W.J. Dally. *Virtual-channel flow control.* IEEE Transactions on Parallel and Distributed Systems, vol. 3, no. 2, pp. 194-205, 1992.

9. W.J. Dally and H. Aoki. *Deadlock-free adaptive routing in multicomputer networks using virtual channels.* IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 4, pp 466-475, 1993.
10. 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.
11. J. Duato. *A theory of fault-tolerant routing in wormhole networks.* Proc. International Conference on Parallel and Distributed Systems, pp. 600-607, 1994.
12. Earth Simulator Center. http://www.es.jamstec.go.jp/esc/eng/index.html.
13. M.E. Gómez, J. Duato, J. Flich, P. López, and A. Robles / N.A. Nordbotten, O. Lysne, and T. Skeie. *An Efficient Fault-Tolerant Routing Methodology for Meshes and Tori.* Computer Architecture Letters, vol. 3, May 2004.
14. 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.
15. 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.
16. InfiniBand^{TM} Trade Association, *InfiniBand^{TM} architecture. Specification vol. 1. Release 1.0.a.* Available at http://www.infinibandta.com.
17. D.H. Linder and J.C. Harden. *An Adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes.* IEEE Transactions on Computers, vol. C-40, no. 1, pp. 2-12, 1991.
18. 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.* 22nd International Conference on Parallel Processing, 1999.
19. 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, pp. 193-211, 2000.
20. S.L. Scott and G.M. Thorson. *The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus.* Symposium on High Performance Interconnects, 1996.
21. L.G. Valiant, *A Scheme for Fast Parallel Communication.* SIAM J. Comput. 11, pp. 350-361, 1982.