A Routing Methodology for Achieving Fault Tolerance in Direct Networks

María Engracia Gómez, *Member*, *IEEE*, Nils Agne Nordbotten, José Flich, Pedro López, *Member*, *IEEE Computer Society*, Antonio Robles, *Member*, *IEEE Computer Society*, Jose Duato, *Member*, *IEEE*, Tor Skeie, and Olav Lysne, *Member*, *IEEE*

Abstract—Massively parallel computing systems are being built with thousands of nodes. The interconnection network plays a key role for the performance of such systems. However, the high number of components significantly increases the probability of failure. Additionally, failures in the interconnection network may isolate a large fraction of the machine. It is therefore critical to provide an efficient fault-tolerant mechanism to keep the system running, even in the presence of faults. This paper presents a new fault-tolerant routing methodology that does not degrade performance in the absence of faults and tolerates a reasonably large number of faults without disabling any healthy node. In order to avoid faults, for some source-destination pairs, packets are first sent to an intermediate node and then from this node to the destination node. Fully adaptive routing is used along both subpaths. The methodology assumes a static fault model and the use of a checkpoint/restart mechanism. However, there are scenarios where the faults cannot be avoided solely by using an intermediate node. Thus, we also provide some extensions to the methodology. Specifically, we propose disabling adaptive routing and/or using misrouting on a per-packet basis. We also propose the use of more than one intermediate node for some paths. The proposed fault-tolerant routing methodology is extensively evaluated in terms of fault tolerance, complexity, and performance.

Index Terms—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 (nuclear weapons simulations, protein folding, global climate modeling, galaxy interaction simulations, etc.). These applications require continued research and technology development to deliver computers with steadily increasing computing power. The required levels of computing power can only be achieved with massively parallel computers, such as the Earth Simulator [19], the ASCI Red [1], and the BlueGene/L [5].

The huge number of processors and associated devices (memories, switches, and links, etc.) significantly affects the probability of failure. Each individual component can fail and, thus, the probability of failure of the entire system increases dramatically. One of the JASON Defense Advisory Panel reports from 2003, about the requirements for ASCI, states that "Scaling to PetaFlop using present machine architectures implies very large number of processors—of order 100,000, perhaps—might be needed. Such large numbers raises serious questions of scalability of code performance and of machine reliability."

- *M.E. Gómez, J. Flich, P. López, A. Robles, and J. Duato are with the Department of Computer Engineering, Universidad Politécnica de Valencia, Camino de Vera, 14, 46071-Valencia, Spain. E-mail: {megomez, jflich, plopez, arobles, jduato}@disca.upv.es.*
- *N.A. Nordbotten, T. Skeie, and O. Lysne are with the Simula Research Laboratory, PO Box 134, N-1325, Lysaker, Norway. E-mail: {nilsno, tskeie, olavly}@simula.no.*
The first two authors are listed in alphabetical order.

Manuscript received 7 Feb. 2005; revised 5 Aug. 2005; accepted 5 Oct. 2005; published online 22 Feb. 2006.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0037-0205.

Thus, in these systems, it is critical to keep the system running, even in the presence of failures. In addition, failures in the interconnection network may isolate a large fraction of the machine, containing many healthy processors that otherwise could have been used. Although network components, like switches and links, are robust, they are working close to their technological limits and, therefore, they are 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 [25], [47], [26], [48], [37], [38].

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. In order to implement it, once a fault is detected, all the processes in the system are halted, the network is emptied, and a management application is run in order to deal with the faulty component. The management application detects where the fault is, computes the information required by the nodes in order to tolerate the fault, and distributes the information. Then, the system is rebooted and the processes are resumed. This fault model needs to be combined with checkpointing techniques in order to be effective. Applying checkpointing minimizes the fault's impact on applications because they are restarted from the latest checkpoint. In a dynamic fault model, once a new fault is found, actions are

0018-9340/06/$20.00 © 2006 IEEE Published by the IEEE Computer Society

taken in order to appropriately handle the faulty component while the system keeps running. For instance, a source node that detects a faulty component in a path can switch to a different path.

Although there are different ways to deal with faults in the interconnection network (see Section 2), most of the solutions proposed in the literature are based on designing fault-tolerant routing algorithms that are 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 in the case of failure. Alternatively, there exist some fault-tolerant routing strategies that use a very small number of extra resources to handle failures, at the expense of either disabling certain healthy nodes [25], thus reducing the processing power, or dramatically increasing the latencies for some packets [42]. Moreover, when faults occur, with most of those fault-tolerant strategies, link utilization may become significantly unbalanced, thus leading to premature network saturation and consequently degrading network performance even more.

In this paper, we overcome all these limitations, proposing a fault-tolerant methodology for interconnection networks that does not degrade performance at all in the absence of faults, inflicts minimal performance degradation in the presence of faults, and tolerates a reasonably large number of faults. Indeed, fault tolerance is achieved without disabling any healthy node, without requiring too many extra hardware resources, and without introducing any significant penalty (e.g., extra latency) when routing packets in a faulty network. The methodology assumes a static fault model and the use of a checkpoint/restart mechanism.

The methodology is based on the use of intermediate nodes for routing [22].[1] That is, for some source-destination pairs, packets are first forwarded to an intermediate node and then from that node to the destination, in this way splitting the routing path into two subpaths. Basically, we avoid faulty links by reducing the number of possible adaptive paths between source and destination nodes. In particular, we remove those paths along which packets could encounter any of the faults. Notice, though, that adaptive routing is still used along both subpaths. However, for some few paths, an intermediate node is not enough to avoid the faults. In order to increase the fault tolerance degree, two extensions of the methodology are proposed. These extensions restrict routing even more to ensure that packets will not encounter the fault. The first one disables adaptive routing and/or uses misrouting on a per packet basis ([23], [24]). The second extension extends the idea of intermediate nodes. Instead of using one intermediate node, it allows the use of multiple intermediate nodes for some paths [34], enabling adaptive routing to be used for all subpaths.

The methodology is valid for any network topology[2] and allows the use of fully adaptive routing even in the presence of failures.[3] To avoid deadlock, only three virtual channels are required even for tori.[4]

In this paper, we present the fault-tolerant routing methodology, fully exploring the impact of each of its mechanisms ([22], [23], [34]). We compare them from a uniform point of view and in the same scenarios, trying to meet the trade-offs between fault tolerance, network performance, and complexity.

The rest of the paper is organized as follows: Section 2 describes related work on fault tolerance. In Section 3, the methodology based on using intermediate nodes is presented. The extensions to the methodology are presented in Section 4. In Section 5, the different combinations of the proposed methodology are evaluated and compared using the same scenarios in terms of complexity, fault tolerance, and performance. Finally, in Section 6, some conclusions are drawn.

2 Related Work

Basically, there are three ways to tolerate faults in interconnection networks: component redundancy, fault-tolerant routing algorithms, and reconfiguration. Using component redundancy has been the easiest way to provide fault tolerance. Components in the system are replicated and, once a failed component is detected, it is simply replaced by its redundant copy. The main drawbacks of this approach are the high extra cost of the spare components and the nonnegligible probability that the circuits required to switch to the spare components may fail. An enhanced version of this technique does not require spare components. It simply bypasses faulty components, together with some healthy components, to maintain network regularity. For instance, in the BlueGene/L project [5], the nodes are connected by using a 3D torus. The full BlueGene/L supercomputer is constituted by 65,536 nodes, which are allocated over 64 racks of 1,024-nodes, with two 512-node midplanes per rack [20]. Once a failure is detected, all the nodes included in the midplane (512-nodes) that contains the faulty node/link are marked as faulty.

Another powerful technique is based on reconfiguring the routing tables in the case of failure, adapting them to the new topology after the failure [7], [44], [32], [33]. This approach is appropriate for switch-based networks (Myrinet [2], Quadrics [35], InfiniBand [27]) in which the topology is defined by the end user. When using reconfiguration, any number of faults is tolerated without requiring additional resources [38], as long as the network remains connected. This technique is extremely flexible, but this flexibility may also kill performance due to the need of using generic routing algorithms as a consequence of the irregularity in the resulting network topology. Often, generic routing algorithms achieve poor performance when applied to regular networks (e.g., 3D tori), as shown in [39]. This is because, in these cases, generic routing schemes are usually not able to provide minimal routing in all cases, regardless of whether they are deterministic or adaptive

1. Intermediate nodes were introduced by Valiant [45] for other purposes, such as traffic balancing.

2. For the sake of simplicity, we will focus on torus and mesh networks.

3. If intermediate nodes are used, fully adaptive routing refers to each subpath.

4. Note that two virtual channels are already required to provide deadlock-free fully adaptive routing [36].

[41].[5] Furthermore, they often provide worse traffic balance than that provided by the routing schemes specifically designed for these networks.

A large number of fault-tolerant routing algorithms for multiprocessor systems have been proposed, especially for mesh and torus topologies. Some approaches [30], [46] use global status information of the network, whereas others use only local status information. In particular, adaptive routing is used together with link status in [15] and [31]. However, these approaches require using a large number of virtual channels, depending on either the number of tolerated faults [15] or the number of dimensions of the topology [31]. Real systems implement very simple mechanisms that partly address the problem, such as the direction order routing used in the Cray T3E [43]. Other solutions consist of using routing algorithms together with additional resources (virtual channels). Some of these solutions are based on block faults [8], [4], [9], [10], [11], [47], whereas others allow individual faults [21], [17], [12]. In the former case, several healthy nodes must be marked as faulty, reducing the system's processing capacity, in order to build fault regions (either rectangular or nonconvex regions). Packets are routed around these fault regions. To this end, several virtual channels must be used. Finally, some routing solutions ([28], [16]) are based on performing misrouting and backtracking of packet headers. Despite tolerating any number of faults, these strategies often strongly penalize the network performance.

To overcome these drawbacks, a software-based fault-tolerant routing approach [42] can be used. When a packet encounters a fault, it is ejected from the network and is later forwarded through an alternative path. This mechanism is very flexible and supports many failure patterns, without either marking healthy nodes as faulty or requiring additional virtual channels. However, some packets may suffer high latencies due to the packet ejection and reinjection, and these packets also consume memory bandwidth in the node where the injection/reinjection is performed. An approach that minimizes the number of required virtual channels and tolerates a fairly large number of faults, at the expense of disabling some healthy nodes, was recently proposed in [25]. This algorithm is based on a static fault model and only requires two virtual channels per link. In the absence of faults, dimension-order routing (DOR) is used. When faults prevent the use of DOR, a set of nodes must be sacrificed (lamb nodes) in order to guarantee that every survivor node, a node that is neither faulty nor a lamb, can reach every survivor node by at most two rounds of DOR. Deadlock-freedom is guaranteed provided that a different virtual channel is used during each round. The main drawbacks of this routing algorithm are that a significant number of nodes must be disabled for packet transmission/reception (but not for routing) in order to support communication among the remaining nodes and that it does not support adaptive routing.

It is important to highlight the main differences between our proposal, described in Sections 3-4, and other approaches in the literature that also use a small number of extra resources. In particular, unlike [42], in no case does our proposed methodology require ejecting/reinjecting a packet at an intermediate node, thus reducing latency drastically. Moreover, unlike [25], our fault-tolerant methodology does not need to deactivate any *lamb* node to achieve good fault tolerance. Furthermore, the proposed methodology allows packets to be adaptively routed, thus increasing the overall network throughput.

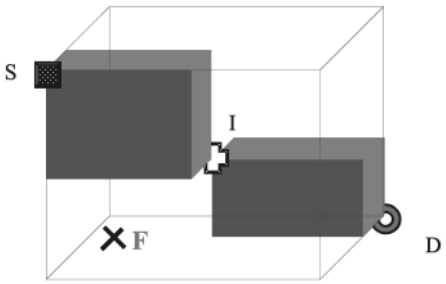

Fig. 1. The use of an intermediate node (I) limits the number of possible paths, from the source (S) to the destination (D), enabling the fault (F) to be avoided.

3 Methodology

In this section, we will describe the basic mechanism for achieving fault tolerance, that is, routing through intermediate nodes. For this purpose, we will assume a k-ary n-cube (torus) or an n-dimensional mesh network with minimal adaptive routing based on Duato's protocol [18]. In the absence of faults, packets are routed using fully adaptive routing, with at least two virtual channels (i.e., one adaptive channel and one escape channel) per physical link. The adaptive channels enable routing through any minimal path, whereas the escape channels guarantee deadlock freedom by using a deterministic routing function free from cyclic dependencies. At each hop, packets that cannot use any of the adaptive channels that provide a minimal path to their respective destinations use the escape channel provided by the deterministic routing function.[6] Also, a static fault model with checkpointing is assumed. Detection of faults, checkpointing, and distribution of routing info is performed as part of the static fault model and, thus, will not be further discussed in this paper.

If faulty components can be encountered when routing packets between a source-destination pair, the methodology avoids these faults by using intermediate nodes for routing. Packets are first forwarded to a suitable intermediate node and, then, from this node to their final destination. This way, intermediate nodes are used in order to obtain greater control over the paths followed by packets, thereby avoiding the faults. Notice that the packets are not ejected from the network at the intermediate node. Fig. 1 shows a source-destination pair that uses an intermediate node. The original routing algorithm (based on Duato's protocol) is used in both subpaths. By using intermediate nodes, areas

5. Some generic routing strategies are able to provide minimal paths at the expense of using a large number of virtual channels, often depending on the network size.

6. Notice that packets can again be routed through adaptive channels (when free) after using an escape channel.

containing faults are avoided at the expense of reducing the number of possible paths.

Deadlocks are avoided by using a separate escape channel for each phase. That is, one escape channel is used (if required) from the source to the intermediate node and another one from the intermediate node to the destination. This way, each phase defines a separate virtual network and packets change virtual network at the intermediate node. Thus, there are no dependencies between the two subpaths, allowing two independent paths to be followed.[7] Although each virtual network relies on a different escape channel, they share the same adaptive channel(s). Thus, a total of at least three virtual channels are required (one adaptive and two escape). Notice that deadlock-free minimal adaptive routing, based on Duato's protocol, requires at least two virtual channels, so only one additional virtual channel is required.

The escape channels use deterministic dimension order routing (DOR) and, for tori, also the bubble flow control mechanism [6]. The bubble flow control mechanism avoids deadlocks in the dimension rings of torus topologies by ensuring that there is always an empty buffer that allows packets to advance along the ring. With this mechanism, a packet that is injected into the network, crosses a network dimension, or originates from an adaptive channel requires two free buffers (i.e., one for the packet and one additional free buffer) in order to be allowed in the escape channel. Indeed, a packet changing virtual network at an intermediate node should also be considered as entering the ring and, therefore, requires two free buffers. Alternatively, if the bubble flow control mechanism is not used, deadlock freedom can be provided in torus topologies through the use of additional virtual channels [13]. If this latter approach were used, each subpath (i.e., virtual network) would require two escape channels, resulting in a total of five virtual channels (i.e., four escape channels and at least one adaptive channel). Anyway, this is an implementation issue, aimed at guaranteeing deadlock-freedom along the escape paths, that has no influence on the proposed fault-tolerant routing methodology.[8]

Next, a methodology for identifying the intermediate nodes is presented.

3.1 Intermediate Nodes for Adaptive Routing

In what follows, we will denote the source node as S and the destination node as D. The intermediate node is denoted as I. Faulty links are denoted as F_i. A node failure can easily be modeled as the failure of all the links of a node.

7. Note that, in certain cases, it may be unnecessary to carry out the virtual network transition at the intermediate node as long as cyclic channel dependencies are not introduced. However, these situations cannot be foreseen when using adaptive routing. Applying deterministic routing along the first subpath could help in some cases. However, the difficulties in guaranteeing in all cases that cyclic channel dependencies are not introduced prevent us from removing the need of using an additional virtual network.

8. We suggest the use of the bubble flow control mechanism in tori because it allows a more efficient implementation of the proposed methodology (i.e., it requires a smaller number of virtual channels to implement the escape paths). The bubble flow control mechanism is currently being used in the BlueGene/L supercomputer [5].

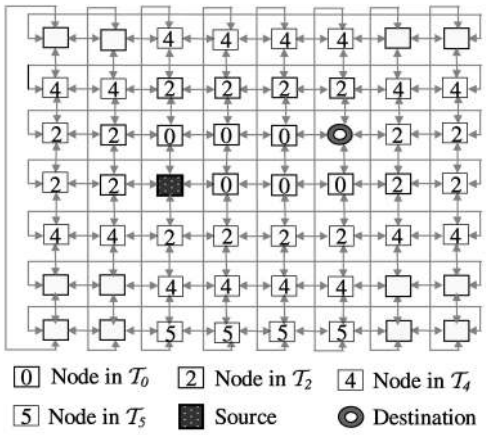

Fig. 2. The nodes in the sets $\mathcal{T}_j$, for $j \leq 5$, for a particular source-destination pair in a 2D torus.

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

1. I is reachable from S.
2. D is reachable from I.
3. There is no I' (fulfilling the previous requirements) giving a shorter path than I.

The first requirement guarantees that packets can be routed from S to I and the second one that packets can be routed from I to D. The third requirement guarantees that the final path is the shortest possible. We define 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, $\mathcal{T}_j$ for different values of j defines nonoverlapping sets of nodes, as shown in Fig. 2. These sets can easily be identified by starting with the nodes that are traversed on any minimal path from S to D (i.e., $j = 0$) and continuing outward.

Theorem 1. *Let j be the smallest integer for which $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}_D$ is nonempty. A node N fulfills all three requirements of an intermediate node I 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., for minimal routes):

- Let us assume that there is one node N in the set that does not fulfill the requirements of an intermediate node. 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 intersections, N must be in

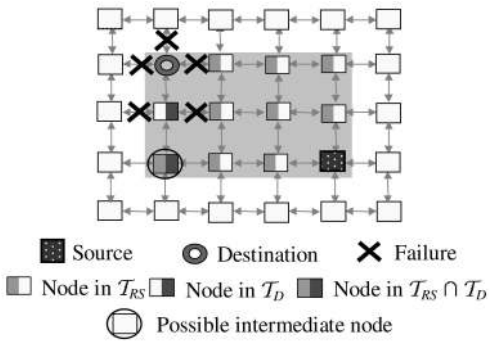

Fig. 3. The faults are avoided by the use of an intermediate node. The shaded area identifies the nodes in $\mathcal{T}_0$.

all 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 of an intermediate node. 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 $\mathcal{T}_0$, N is not on a minimal path from S to D. 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-D exists for $j < m + 1$. Indeed, as each increase of j adds one additional hop to the path S-I-D, all the intermediate nodes found when $j = m + 1$ yield paths S-I-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, to identify possible intermediate nodes, we start by considering the minimal paths ($j = 0$) and then, if necessary, nonminimal paths ($j > 0$) to avoid the fault(s). By minimizing j, preference is given to the shortest connected paths. We illustrate the intermediate node selection in the next section by applying Theorem 1 in two example scenarios.

3.2 Example Scenarios

Fig. 3 shows a scenario with five link 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. 3, there are several nodes within $\mathcal{T}_0$ that are either reachable from S or are able to reach D. However, we are only interested in nodes with all of these attributes, that is, the nodes given by the set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}_D$. In this scenario, there is only one such node, that is, the one identified as a possible intermediate node in Fig. 3. By using this node as the intermediate node, it is guaranteed that the faults are not encountered when packets are first routed from S to I and then from I to D.

Notice that, in a mesh, if all the minimal paths are faulty, it is not possible to find a suitable intermediate node even when considering nonminimal paths (i.e., $\mathcal{T}_j$ for $j > 0$) when fully adaptive routing is used. This is because it is then impossible to position the intermediate node in such a way that all the minimal paths from S to I and from I to D are fault free. In a torus, however, such faults can be avoided by using a nonminimal path given by taking the opposite direction to the minimal path. Thus, if all the minimal paths in the torus shown in Fig. 2 were blocked by faults, one could, for example, use the node two hops to the left of the source as an intermediate node and, thus, get a nonminimal path in the opposite direction of the ring. Because this node is in $\mathcal{T}_2$, the path length of this path equals the minimal path plus two. To handle such a situation in a mesh, it is necessary to use one of the complementary mechanisms described in the next section.

4 Complementary Mechanisms

In some situations, like the one in the last example, it is impossible to avoid all the faults by using the methodology presented in the previous section. Therefore, we now present some alternative extensions to the methodology. First, we present how the intermediate node concept can be extended to use more than one intermediate node for some source-destination pairs. Then, we present alternative solutions based on using additional mechanisms, that is, disabling adaptive routing and/or using misrouting for some paths.

4.1 Multiple Intermediate Nodes

By using more than one intermediate node, additional control over the paths followed by packets is gained, enabling more faults to be avoided while still using adaptive routing for all the paths. In order to still guarantee deadlock freedom, an additional virtual channel is needed for each additional intermediate node.[9] This way, each subpath continues to use a different escape channel. So, when at most two intermediate nodes are used in each path, a total of four virtual channels is required (i.e., three escape channels and one adaptive channel).

When using multiple intermediate nodes, we refer to the intermediate nodes as I_x, where I_1 denotes the first intermediate node in a route. We will first present a methodology for using two intermediate nodes. Then, we generalize this methodology so that it can be used, in a recursive way, for any number of intermediate nodes.

4.1.1 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.

9. If the bubble flow control mechanism was not used in a torus topology, two virtual channels would be required for each additional intermediate node.

- There are no I'_1 and I'_2 (fulfilling the previous requirements), giving a path shorter than I_1 and I_2.

Indeed, 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, where I_1 must be selected in such a way that all four requirements for I_1 and I_2 stated above are fulfilled when I_2 follows from Theorem 1.

In order to solve this problem, let us introduce a variation of $\mathcal{T}_D$, namely, $\mathcal{T}^k_{D1}$. 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 the value of k here equals the value of j, in the set $\mathcal{T}_j$, used in Theorem 1 for identifying it, e.g., the set $\mathcal{T}^0_{D1}$ consists of the nodes that have a minimal path, via one intermediate node, to D. The set $\mathcal{T}^1_{D1}$, on the other hand, consists of the nodes that 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 integers (i.e., so that their sum is minimized) for which $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}^k_{D1}$ is nonempty. A node N fulfills all four requirements of an intermediate node I_1 if and only if $N \in \mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}^k_{D1}$.*

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., for minimal paths):

- Let us assume that there is one node N in the set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}^0_{D1}$ that gives a path S-N-I_2-D that does not fulfill the requirements of an intermediate node I_1. It follows from Theorem 1 and the definition of $\mathcal{T}^k_{D1}$ 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 cannot be on a minimal path from S to D for the path S-N-I_2-D to be a nonminimal 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 of an intermediate node I_1. 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 arguments used 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.

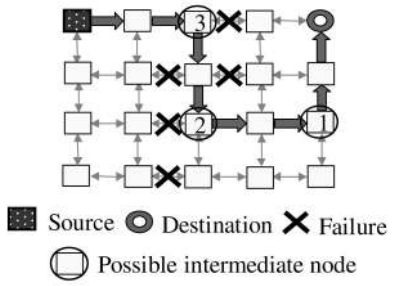

Fig. 4. Examples of nodes in $\mathcal{T}^k_{D_z}$ for different values of k and z. Node 1 is in $\mathcal{T}^0_{D0}$, node 2 is in $\mathcal{T}^0_{D1}$, node 3 is in $\mathcal{T}^4_{D2}$, and the source is in $\mathcal{T}^4_{D3}$.

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, to find the first intermediate node in a path with two intermediate nodes, we start by considering the minimal paths (i.e., $j + k = 0$) and then consider nonminimal paths (i.e., $j + k > 0$), if necessary, to avoid all the faults. The second intermediate node is given by Theorem 1. We will now extend this concept to any number of intermediate nodes and then provide an example scenario.

4.1.2 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 $\mathcal{T}^k_{D_z}$ 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 $\mathcal{T}_D$).
- $\mathcal{T}^k_{D_z}$ (for $z > 0$ and $k \geq 0$): The set of nodes from which at least one node in $\mathcal{T}_{j'} \cap \mathcal{T}^{k'}_{Dz-1}$ is reachable without the use of any intermediate node, where $j' + k' = k$.

Thus, a node $N \in \mathcal{T}^k_{D_z}$ reaches D through z intermediate nodes and k is, here, the number of additional hops, in the path from N to D, compared to the minimal path. Fig. 4 shows some examples of nodes in $\mathcal{T}^k_{D_z}$ for different values of k and z. Node 1 here has a minimal path without any intermediate nodes to the destination (D) and, thus, belongs to the set $\mathcal{T}^0_{D0}$. Node 2 has a minimal path, via one intermediate node (i.e., node 1) to D, and therefore belongs to the set $\mathcal{T}^0_{D1}$. Node 3 has a path to D via two intermediate nodes (i.e., nodes 2 and 1), with a length of four hops more than the minimal path, and thus belongs to the set $\mathcal{T}^4_{D2}$. More specifically, $j' = 4$ because the first intermediate node in this path (i.e., node 2) is in $\mathcal{T}_4$ (relative to node 3 and D) and $k' = 0$ because the path from node 2 to D is a minimal one. Thus, $k = j' + k' = 4$. The source node (S) has a path to D through three intermediate nodes and this path is four hops longer than the minimal path, thus S is in the set $\mathcal{T}^4_{D3}$. In this case, $j' = 0$ because node 3 is on a minimal path between S and D, while $k' = 4$ because node 3 is in $\mathcal{T}^4_{D2}$ (i.e., the path from node 3 to D is four hops longer than the minimal path).

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, thus resulting in paths with one

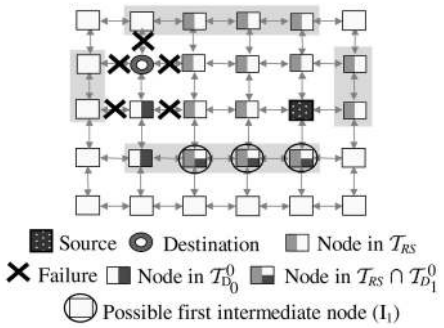

Fig. 5. Two intermediate nodes must be used in order to avoid the faults. The figure shows how the first of these intermediate nodes (i.e., I_1) is identified. The shaded areas identify the nodes in $\mathcal{T}_2$.

intermediate node. The set $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D1}^k$ is that given by Theorem 2, resulting in paths with two intermediate nodes. Similarly, $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D2}^k$ gives paths with three intermediate nodes. Continuing this way, an arbitrary number of intermediate nodes can be obtained. When paths of equal length exist, preference should be given to paths with fewer intermediate nodes.

4.1.3 Example Scenario

Fig. 5 shows the same scenario as previously shown in Fig. 3, except that the source node is different. In this case, all the minimal paths between S and D are blocked by faults. The set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D0}^0$, giving minimal paths with one intermediate node, is therefore empty. The set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D1}^0$, 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 $\mathcal{T}_2$ ($\mathcal{T}_j$ is empty for odd values of j in meshes) giving a nonminimal path with one intermediate node. However, this set, $\mathcal{T}_2 \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D0}^0$, 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). Anyway, of the two sets, the set $\mathcal{T}_0 \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D1}^2$ is empty, while the set $\mathcal{T}_2 \cap \mathcal{T}_{RS} \cap \mathcal{T}_{D1}^0$ gives us the possible intermediate nodes shown in Fig. 5. 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 node as the one earlier identified as I_1 in Fig. 3.

4.2 Disabling Adaptive Routing

A deterministic minimal routing function uses a subset of the paths returned by an adaptive minimal routing function. Therefore, a node given by Theorem 1 for adaptive routing can also be used as an intermediate node when deterministic routing is used. However, there are scenarios where the set $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}_D$ is empty but where it is still possible to find a suitable intermediate node if routing is restricted to a deterministic route. 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:

- For all i, F_i is not on the $S-I$ deterministic path.
- For all i, F_i is not on the $I-D$ deterministic path.
- There is no I' (fulfilling the previous requirements) giving a shorter path than I.

As can be observed, these are the same requirements as those that formed the basis for Theorem 1, except that deterministic routing now is assumed. Therefore, we can obtain the desired nodes by applying Theorem 1 for deterministic routing. In order to do this, 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.

Corollary 1. *Let j be the smallest integer for which $\mathcal{T}_j \cap \mathcal{T}_{RS}^d \cap \mathcal{T}_D^d$ is nonempty. A node N fulfills all three requirements of an intermediate node, when deterministic routing is used along the subpaths if and only if $N \in \mathcal{T}_j \cap \mathcal{T}_{RS}^d \cap \mathcal{T}_D^d$.*

This gives intermediate nodes (I) where deterministic routing is used from S to I and from I to D. However, even when it is not possible to use adaptive routing all the way from S via I to D (i.e., when the set $\mathcal{T}_j \cap \mathcal{T}_{RS} \cap \mathcal{T}_D$ is empty), it may still be possible to use adaptive routing from S to I or from I to D.

Thus, 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. As before, j is minimized in order to get the shortest path length possible.

4.2.1 Example Scenario

Fig. 6a shows a mesh network where all the shortest paths between the source and the destination are blocked by faults. Thus, the set $\mathcal{T}_j \cap \mathcal{T}_{RS}^d \cap \mathcal{T}_D^d$ is empty for $j=0$. It is also empty for $j=1$ as the set $\mathcal{T}_j$ is empty for all odd values in meshes. We must therefore try with $j=2$. The set $\mathcal{T}_2$ consists of the nodes within the shaded areas in the figure. Among the nodes within $\mathcal{T}_2$ that can be reached from S, using deterministic (dimension order) routing, one that also has a deterministic route to D (i.e., a node in $\mathcal{T}_2 \cap \mathcal{T}_{RS}^d \cap \mathcal{T}_D^d$) can be chosen as the intermediate node. As shown in Fig. 6a, there is one such node in this scenario. This node should therefore be used as the intermediate node. Notice that packets could encounter a fault if adaptive routing was used in any of the subpaths and it is therefore necessary to use deterministic routing (i.e., disable adaptive routing) both from S to I and from I to D.

4.3 Misrouting

The last complementary mechanism is the use of misrouting. For this, we use direction-order routing

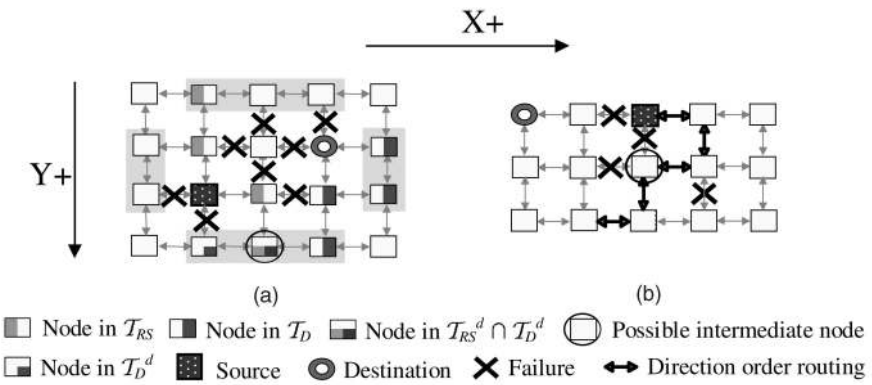

Fig. 6. (a) Adaptive routing must be disabled in both subpaths to avoid all the faulty links. The shaded areas identify the nodes in $\mathcal{T}_2$. (b) Misrouting is used in the first subpath and in the beginning of the second subpath in order to circumvent all the faults

TABLE 1
Variations of the Methodology

Variation	Description
I	All packets are adaptively routed. An intermediate node is used when needed.
D	Adaptive routing can be disabled (i.e., deterministic routing is applied) on a per packet basis.
M	Packets can be forced along up to three directions (at most eight hops in each direction) at the beginning of the path to the destination (following minimal or non-minimal paths). Then, packets are adaptively routed.
I+D	Packets can use an intermediate node (I) and/or disable adaptive routing (D). When an intermediate node is used, adaptive routing can be disabled along a subpath or along the entire path.
I+M	If an intermediate node (I) is used, misrouting (M) can be applied for one or both of the subpaths. Otherwise, misrouting can be applied at the beginning of the path to the destination. In all cases, packets are adaptively routed after the misrouting phase.
D+M	Misrouting (M) and/or disabling adaptive routing (D) can be applied. When both mechanisms are applied, packets are deterministically routed after the initial misrouting phase. Otherwise, packets are adaptively routed.
I+D+M	If an intermediate node is used, misrouting and/or deterministic routing (D+M) can be applied for one or both of the subpaths. Otherwise, misrouting and/or deterministic routing can be applied for the path to the destination.
I×N	All packets are adaptively routed. At most N intermediate nodes can be used for each path.
I×2+D	At most two intermediate nodes can be used for each path, and adaptive routing can be disabled along any of the subpaths.

$(X+Y+Z+X-Y-Z-)$ instead of dimension order routing. This is because direction order routing allows packets to be routed in both directions of a dimension and, therefore, offers greater flexibility to avoid faults. Furthermore, direction order routing allows routing through nonminimal paths. We restrict packet misrouting to at most three directions, up to eight hops in each direction, at the beginning of each path/subpath.

4.3.1 Example Scenario

Fig. 6b shows a scenario where the shortest paths between the source and the destination are blocked by faults. By using direction order routing, packets are first misrouted one hop in the $X+$ direction, then one hop in the $Y+$ direction, and one hop in the $X-$ direction. By having an intermediate node at this point, the packets are again allowed to travel in the $Y+$ direction. So, from the intermediate node, the packets are misrouted one hop in the $Y+$ direction and then one hop in the $X-$ direction before they continue toward the destination using adaptive routing.

5 Evaluation

In this section, we will evaluate the methodology in terms of fault tolerance, complexity, and performance. The three basic mechanisms are analyzed individually: one intermediate node (I), disabling adaptive routing (D), and misrouting (M). Additionally, the following combinations of these mechanisms are evaluated: I+D, I+M, M+D, and I+D+M. The extension to more than one intermediate node is also analyzed and is referred to as I×N, where N is the maximum number of intermediate nodes used in a path. We also analyze the combination of two intermediate nodes (I×2) and disabling adaptive routing (D), that is, I×2+D. Table 1 summarizes each variation of the methodology.

5.1 Fault Tolerance

A methodology is n-fault-tolerant if it is able to tolerate any combination of n link failures. Indeed, we will say that a given combination of failures is tolerated if the methodology is able to provide a nonfaulty route between every source-destination pair in the network. If, for a combination of n faults, there is at least one source-destination pair whose path is not fault-free, then that combination is not tolerated and the methodology is not n-fault-tolerant.

Faults may disconnect some nodes in the network. As these nodes can no longer send or receive messages, such a situation is not considered as not tolerated. We only consider link failures since a node failure can be modeled as the failure of all the links of a node.[10]

We present fault tolerance results for a $3 \times 3 \times 3$ (27 nodes) torus network. Although current systems are being built with larger topologies (e.g., a $32 \times 32 \times 64$ torus for BlueGene/L), smaller networks can be exhaustively evaluated from a fault-tolerant point of view and the results can then be easily transferred to larger networks. In larger networks, the faults will be at the same or a greater distance, so it is reasonable to expect that the fault tolerance will be the same and that the percentage of not tolerated combinations will be even lower.

5.1.1 Fault Analysis Models

For a limited number of faults, 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 scenarios where the faults are closely located are difficult to solve 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 obtained results cannot be directly extended to the generic case, where the faults may be located over the entire network, it gives us an approximation of the effectiveness of the methodology in a critical situation.

The analyzed region is formed by all the links of the nodes that are one hop away from a node (the center node), which is randomly selected.[11] We will refer to this region as a *distance 1* region. Fig. 7 shows a *distance 1* region, formed by 36 links. In a $3 \times 3 \times 3$ torus, it only consists of 33 links though, as three of the links then are shared by nodes within the region. Notice that, with a high number of faults, the center node is hardly accessible.

In the second approach, a statistical analysis is performed. A subset of the total number of fault combinations for a given number of faults is analyzed in order to check if they are tolerated. The faults in each of these combinations are randomly located over the entire network. From the obtained results, we perform a statistical analysis to obtain the fault tolerance of the proposed methodology.

5.1.2 Fault Tolerance Results

Table 2 shows the percentage of fault combinations not tolerated by the different mechanisms and their possible combinations in a $3 \times 3 \times 3$ torus network. Results for the three fault analysis models (exhaustive, *distance 1*, and probabilistic) are shown. As can be observed, the I+D+M combination achieves a very good fault tolerance degree. In particular, all the evaluated fault combinations with up to seven faults are tolerated. Notice that, for six-faults and up, only the *distance 1* and statistical analysis models are used.

10. When a link fails, we assume that it fails in both directions.

11. The selection of the center node does not affect the results due to the symmetry property of torus networks.

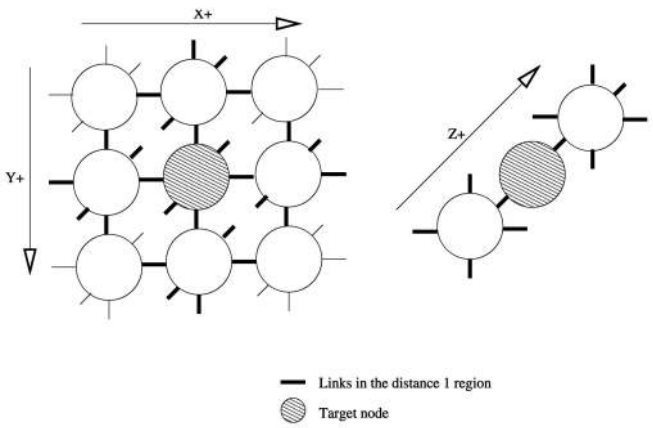

Fig. 7. Distance 1 region in a 3D torus.

As the *distance 1* represents a worst-case analysis, this strongly indicates that the mechanism is 7-fault-tolerant. Moreover, for the statistical fault analysis model, all the analyzed fault combinations (up to 14 faults) were tolerated. However, in the *distance 1* analysis model, some fault combinations are not tolerated from eight faults upward.

Table 2 also shows the fault tolerance of the mechanisms when used individually. The D mechanism is neither able to tolerate any fault nor any combination of faults. This result was expected since the deterministic routing used (i.e., DOR routing) is not even 1-fault-tolerant. Regarding the I mechanism alone, we can observe that it is only 1-fault-tolerant. Moreover, the number of not tolerated fault combinations increases significantly as the number of faults in the network increases. For five faults, roughly a quarter of the fault combinations are not tolerated. Finally, the M mechanism obtains a better fault tolerance degree than the I mechanism by being 3-fault-tolerant. However, for a higher number of faults, the percentage of not tolerated combinations increases significantly. Therefore, in light of these results, it must be noticed that an acceptable fault tolerance degree cannot be achieved by using any of the mechanisms separately, requiring additional support to be effective.

Looking at the combinations of mechanisms, we can observe that disabling adaptive routing is effective when it is combined with intermediate nodes. In fact, I+D is 5-fault-tolerant and the percentage of not tolerated fault combinations remains low as the number of faults in the network increases. The percentage of not tolerated fault combinations for eight faults is lower than 1.5 percent in the *distance 1* region. These results are obtained thanks to the use of dimension order $(X \pm Y \pm Z \pm)$ routing. However, when using direction order $(X + Y + Z + X - Y - Z -)$ routing instead of dimension order routing, only three faults were tolerated by I+D. This is because, when an intermediate node is used, $X \pm Y \pm Z \pm$ provides more flexibility in the entire path (i.e., taking into account both the $S - I$ stretch and the $I - D$ stretch) since the dimensions are always traveled in the same order, regardless of the direction.

On the other hand, we can observe that disabling adaptive routing together with misrouting does not help much. In particular, D+M exhibits fault tolerance capabilities similar to those obtained with M. I+M appears to be 7-fault-tolerant as I+D+M, although it has a slightly higher percentage of not tolerated combinations in the *distance 1* region for eight faults. The similar results are due to the fact

TABLE 2
Percentages of Link Fault Combinations Not Tolerated by Each Mechanism and Their Combinations in a $3 \times 3 \times 3$ Torus

Link faults	# of combinations	D	I	M	D+M	I + D	I+M	I+D+M	I×2	I×3	I×2+D
Exhaustive analysis											
1	81	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%
2	3,240	100%	2.5%	0%	0%	0%	0%	0%	0%	0%	0%
3	85,320	100%	7.44%	0%	0%	0%	0%	0%	0%	0%	0%
4	1,663,740	100%	14.67%	0.95%	0.84%	0%	0%	0%	0%	0%	0%
5	25,621,596	100%	24.06%	-	-	0%	0%	0%	0%	0%	0%
distance-1 analysis											
5	237,336	100%	38.16%	8.47%	7.09%	0%	0%	0%	0%	0%	0%
6	1,107,568	100%	54.52%	20.39%	17.29%	0.057%	0%	0%	0.01%	0%	0%
7	4,272,048	100%	70.31%	36.95%	31.96%	0.35%	0%	0%	0.06%	0%	0%
8	13,884,156	100%	83.30%	55.33%	49%	1.25%	0.0006%	0.0004%	0.31%	0%	0%
9	38,567,100	100%	92.15%	-	-	-	-	-	1.06%	0%	0%
10	92,561,040	100%	96.97%	-	-	-	-	-	2.99%	0.0007%	0.0003%
Statistical analysis											
5	> 14M	100%	24.07%	4.23%	2.11%	0%	0%	0%	0%	0%	0%
6	> 6M	100%	35.46%	11.22%	8.11%	0%	0%	0%	0%	0%	0%
7	> 4.5M	100%	48.72%	22.55%	19.33%	0%	0%	0%	0%	0%	0%
8	> 3.4M	100%	62.98%	41.04%	37.60%	0%	0%	0%	0%	0%	0%
9	> 2.7M	100%	76.51%	54.14%	47.82%	0.00008%	0%	0%	0%	0%	0%
10	> 2.2M	100%	87.40%	70.29%	69,71%	0.48%	0%	0%	0.09%	0%	0%
11	> 1.7M	100%	94.47%	83.08%	74.47%	1.063%	0%	0%	0.23%	0%	0%
12	> 1.5M	100%	98.05%	91.69%	90.12%	2.79%	0%	0%	0.52%	0%	0%
13	> 1.3M	100%	99.46%	96.60%	93.67%	3.16%	0%	0%	1.10%	0%	0%
14	> 1M	100%	99.88%	98.86%	98.17%	8.47%	0%	0%	2.13%	0%	0%

In the statistical results, the error is always less than 1 percent.

that misrouting can also be used in the same way as deterministic routing.

We also consider the combination of several intermediate nodes. When using two intermediate nodes, the methodology greatly increases its fault tolerance degree. In particular, it is 5-fault-tolerant. With six faults in the network, two intermediate nodes are not enough for all the fault combinations. This is similar to the result obtained for the I+D combination. However, for more than five faults, I×2 obtains lower percentages of not tolerated combinations. With three intermediate nodes, the methodology achieves a very high fault tolerance degree. Taking into account the *distance 1* analysis, we can observe that I×3 tolerates as many as nine faults, while there are some not tolerated combinations with 10 faults. In the statistical analysis, the I×3 combination could provide a valid path for every nondisconnected pair of nodes for up to 14 faults in all the analyzed fault combinations. The combination of two intermediate nodes and disabling adaptive routing (I×2+D) achieves very similar results, that is, it tolerates nine faults with a slightly lower number of not tolerated combinations for 10 faults.

Table 3 shows, in the third column, for each analyzed combination of mechanisms and each number of faults, the percentage of paths (source-destination pairs) that require using any mechanism(s) of the combination in order to avoid the faults.[12] As can be seen, for one fault, 6.85 percent of the paths are affected by faults. This percentage increases as the number of faults increases. For five faults, more than a quarter of the paths are affected by faults.

Additionally, Table 3 shows, for each analyzed mechanism combination and each number of faults, the percentage of the affected paths that use each of the available mechanisms, either alone or in combination, in order to avoid faults. For instance, when using I+D+M, for one fault, all the faults are avoided by using only one intermediate node. For five faults, several mechanisms, either alone (I,D) or in combination (I+M, I+D), are required to provide fault tolerance. It can be noticed that intermediate nodes (I) is the most frequently used mechanism (when it is available). In particular, for all the combinations of mechanisms that allow the use of intermediate nodes, more than 98 percent of the affected paths use only an intermediate node (fifth column) to deal with faults when there are less than six faults. Notice, though, that the additional mechanisms are still required in order to ensure fault tolerance. When I is not available (D+M combination), D is the most commonly used mechanism since it provides paths shorter than M. Remember that, when a path can use several options to avoid the faults, it selects the one that avoids the faults and provides a shorter path. Also, the option that allows adaptive routing is preferred.

5.2 Required Resources and Complexity

In order to support the methodology, routing info must be stored at each source node. For every destination, the intermediate node(s) to use (if required) and, if applicable, information about misrouting and/or switching off adaptive routing must be stored. Notice that the amount of memory required to store the routing info is low, e.g., if the I×2 combination is used in a large system with 65,536 nodes, 256KB of memory would be required.[13] This table can easily be compacted by only storing information about the faulty paths.

12. Because only the tolerated combinations are included when calculating the percentage of affected paths, the percentage of affected paths may differ for the different mechanism combinations.

13. Two bytes are used to store the address of each intermediate node ((2B+2B)*65,536=256KB).

TABLE 3
Percentages of the Affected Paths in a $3 \times 3 \times 3$ Torus Network that Make Use of a Specific Mechanism when a Given Mechanism Combination Is Used

			Percentage of the affected paths that utilize mechanism								
Mech. Comb.	# faults	Affected Paths	M	I	I+M	D	I+D	D+M	I+D+M	I×2	I×3
Exhaustive analysis											
I+D+M	1	6.85%	0%	100%	0.00%	0.00%	0.00%	0%	0.00%	-	-
	2	13.03%	0%	99.70%	0.04%	0.07%	0.14%	0%	0.07%	-	-
	3	18.60%	0%	99.26%	0.07%	0.16%	0.34%	0%	0.16%	-	-
	4	23.62%	0%	98.68%	0.11%	0.26%	0.60%	0%	0.51%	-	-
	5	28.16%	0%	98.05%	0.18%	0.41%	0.92%	0%	0.43%	-	-
I+M	1	6.85%	0.00%	100%	0.00%	-	-	-	-	-	-
	2	13.03%	0.00%	99.90%	0.10%	-	-	-	-	-	-
	3	18.60%	0.00%	99.78%	0.23%	-	-	-	-	-	-
	4	23.62%	0.00%	99.58%	0.42%	-	-	-	-	-	-
	5	28.16%	0.00%	99.36%	0.65%	-	-	-	-	-	-
D+M	1	6.85%	23.36%	-	-	64.08%	-	12.68%	-	-	-
	2	13.03%	24.38%	-	-	62.40%	-	13.20%	-	-	-
	3	18.60%	25.38%	-	-	60.82%	-	13.77%	-	-	-
	4	23.63%	25.64%	-	-	59.24%	-	15.11%	-	-	-
	5	28.203%	26.62%	-	-	57.69%	-	15.81%	-	-	-
I+D	1	6.85%	-	100%	-	0.00%	0.00%	-	-	-	-
	2	13.03%	-	99.70%	-	0.07%	0.25%	-	-	-	-
	3	18.60%	-	99.26%	-	0.17%	0.57%	-	-	-	-
	4	23.62%	-	98.72%	-	0.27%	1.00%	-	-	-	-
	5	28.16%	-	98.06%	-	0.39%	1.53%	-	-	-	-
I×N	1	6.85%	-	100%	-	-	-	-	-	0.00%	0.00%
	2	13.03%	-	99.70%	-	-	-	-	-	0.30%	0.00%
	3	18.60%	-	99.30%	-	-	-	-	-	0.67%	0.00%
	4	23.62%	-	98.69%	-	-	-	-	-	1.31%	0.00%
	5	28.16%	-	98.01%	-	-	-	-	-	1.99%	0.00%
distance-1 analysis											
I+D+M	6	29.45%	0.00%	91.48%	0.30%	1.19%	6.41%	0.00%	0.61%	-	-
	7	32.49%	0.00%	89.41%	0.42%	1.66%	7.60%	0.00%	0.91%	-	-
	8	35,21%	0.00%	87.05%	0.53%	2.21%	8.83%	0.00%	1.33%	-	-
I+D	6	29.27%	-	96.12%	-	0.58%	3.30%	-	-	-	-
	7	32.21%	-	94.72%	-	0.75%	4.55%	-	-	-	-
	8	34.8%	-	93.14%	-	0.91%	5.96%	-	-	-	-
I+M	6	29.45%	0.00%	98.22%	1.78%	-	-	-	-	-	-
	7	32.49%	0.00%	97.45%	2.57%	-	-	-	-	-	-
	8	34.9%	0.00%	96.34%	3.65%	-	-	-	-	-	-
I×N	6	29.45%	-	95.93%	-	-	-	-	-	4.06%	0.0001%
	7	32.49%	-	94.47%	-	-	-	-	-	5.53%	0.0012%
	8	34.76%	-	92.77%	-	-	-	-	-	7.22%	0.0057%

For up to five faults, all the possible combinations have been analyzed. From six faults upward, the study is reduced to the distance I region. The percentages are based on the fault scenarios that are tolerated by the respective combinations of mechanisms.

In addition, a proper packet header must be used, according to the combination of mechanisms in use, in order to allow the routing to be hardwired in the switches. In particular, a packet routed through an intermediate node will have two subheaders. The first one will be used for routing the packet toward the intermediate node and the second one for routing the packet toward the final destination. If more than one intermediate node is used for some paths, an additional subheader is needed for each one.

Fig. 8 shows the packet header used for the I+D+M combination. The "I" bit indicates whether the packet is being routed via an intermediate node or not and can be used to select the proper escape channel. At the intermediate node, the first subheader is removed and the "I" bit is set to zero. As shown, the packet subheaders include information about the directions and the number of hops to misroute. Because we assume that misrouting can be performed in three directions, three bits are used to indicate the direction and three more to indicate the number of hops to misroute in that direction. Furthermore, a bit in each of the subheaders is used to disable adaptive routing.

The computational cost of the proposed methodology is not too high, especially when taking into account that the routing info is computed offline. Notice that most of the affected paths will only use an intermediate node to avoid faults and the cost of computing each intermediate node is $O(1)$. Therefore, for all the paths, the computational cost is $O(n^2)$, where n represents the number of nodes. However, when misrouting is used, the algorithm will have to explore all the possible hops along each dimension (up to the network radix, k) until a fault-free path is found. For a 3D torus (with $n = k^3$ nodes), in the worst case, the

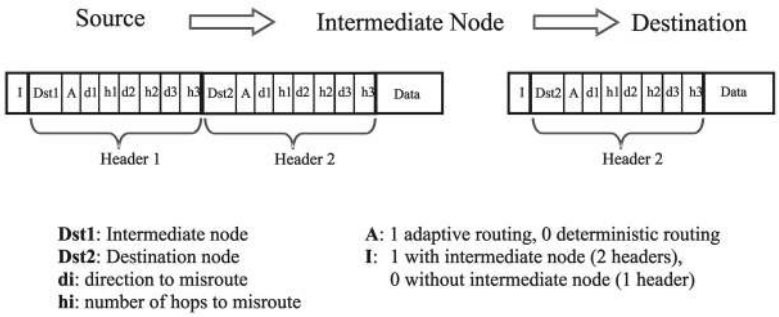

Fig. 8. Packet header for the methodology when the I+D+M combination is used.

computational cost will be $O(k^3) = O(n)$. Therefore, the computational cost for all the paths by using the proposed methodology with misrouting will be $O(n^3)$. Similarly, the computational cost, in the worst case, when multiple intermediate nodes are used is also $O(n^3)$.

5.3 Performance

In this section, we analyze how the different variations of the methodology influence network performance for different numbers of failures. Moreover, we are interested in comparing the proposed methodology with real fault-tolerant mechanisms used in current systems. In particular, we compare the performance degradation of the proposed methodology with a mechanism similar to the one used in the IBM BlueGene/L supercomputer. Thus, we will first provide a brief overview of the BlueGene/L supercomputer. Then, the simulation model is described in detail and, finally, the performance results are presented and analyzed.

5.3.1 BlueGene/L Supercomputer

The BlueGene/L supercomputer [5] is based on a new architecture that exploits system-on-a-chip technology to get a target peak processing power of 360 teraFLOPS. The machine currently holds the number one position on the top 500 list. BlueGene/L is configured as a 3D torus of $64 \times 32 \times 32$ compute nodes, with point-to-point serial links between the routers. Each node has two processors and is directly connected with six neighbors, one in each of the six directions. The standard fault tolerance mechanism implemented in the BlueGene/L supercomputer uses a static fault model with checkpointing. Fault tolerance is achieved by bypassing planes. In each rack, there are additional link boards that connect links from the backplane with connectors in the front panel so that each rack can be connected with neighboring ones. Internal switches on the link boards allow a plane to be connected to the next one or to skip one plane (each plane containing 512 nodes). By shutting down the bypassed section of the machine, the fault can be repaired. The BlueGene/L also has an additional software-based fault-tolerant mechanism. When using this mechanism, it is ensured that packets are injected in a manner that forces them to avoid the faults. This mechanism uses nonminimal paths and can handle up to three faults, provided they are not collinear. However, this mechanism has great performance impacts and is not intended for general use [20].

The torus network uses virtual cut-through [29] and provides both adaptive and deterministic minimal-path routing. Physical channels are multiplexed into four virtual channels. Two virtual channels are used for minimal adaptive routing [18].[14] The remaining two virtual channels are used for minimal deterministic routing. The first of these is used to implement the escape paths for the adaptive routing, whereas the second one is reserved for high priority packets. In the deterministic virtual channels, the bubble flow control mechanism [36] is used to avoid deadlocks inside a ring, whereas dimension order routing (DOR) is used to avoid deadlocks when routing through different dimensions.

5.3.2 Simulation Model

A detailed event-driven simulator has been developed to model the performance behavior exhibited by the proposed methodology and a mechanism similar to the one used in the BlueGene/L. The simulator models a direct interconnection network with point-to-point bidirectional serial links. Each router has a nonmultiplexed crossbar with queues only at the input ports. The crossbar has an input for each input queue and an output for each output port. A round-robin policy is used to select among packets contending for the same output port.

A 3D torus has been used for all the simulations. Thus, each node in the network has six ports connecting it to other nodes. In addition, four internal ports connect the router to the processing node. We will present results for an $8 \times 8 \times 8$ (512 nodes) torus network.

Each physical input port is split into several virtual channels (four or five), each providing buffering resources in order to store two packets. The number of escape channels will depend on the number of intermediate nodes that can be used in a single path. Virtual channels are used as adaptive or as escape channels, depending on the number of required escape channels. In order to make a fair evaluation, the same total number of virtual channels has been used, regardless of the number of intermediate nodes in the methodology.

Packets are adaptively routed through minimal paths by using the adaptive virtual channels. In the escape channels, packets are deterministically routed following the $X+Y+Z+X-Y-Z-$ ordering when misrouting is available.

14. It is expected that most of the traffic will use adaptive routing, so, in order to reduce the head-off-line blocking effect, two virtual channels are used.

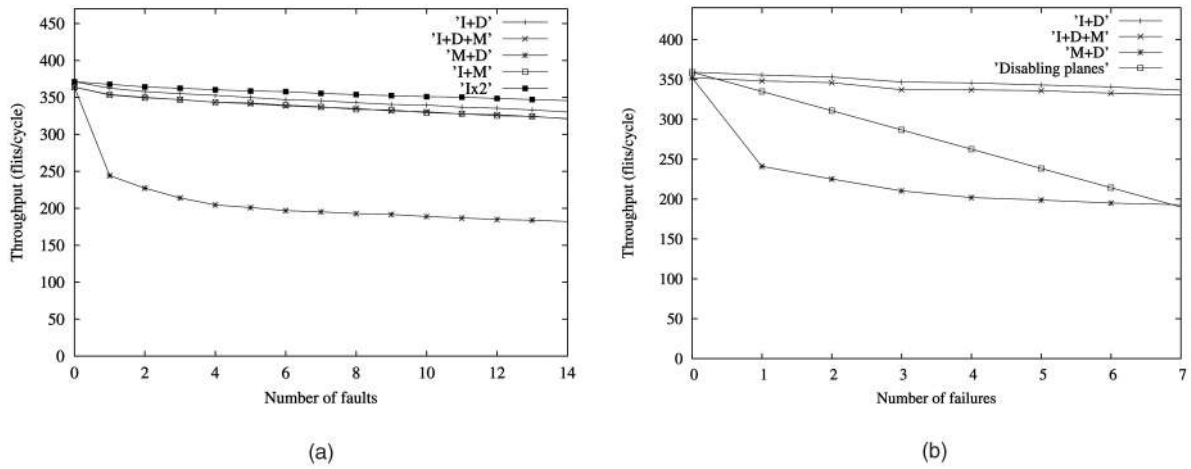

Fig. 9. Throughput (flits/cycle) for several combinations of mechanisms in an $8 \times 8 \times 8$ torus network. Error bars are not shown as they are too small to be seen. (a) Five virtual channels are used. (b) Four virtual channels are used. The throughput for the mechanism used in the BlueGene/L supercomputer is also shown.

Otherwise, dimension-order routing $(X \pm Y \pm Z \pm)$ is applied. The bubble flow control mechanism is used in the escape channels.[15] If a packet arrives at an input port and the adaptive queues are full, the proper escape channel is selected depending on the packet's current destination $(I_1, I_2, \ldots, D)$. The output port is selected based on the information located in the packet header (i.e., destination node, adaptive routing disabled/enabled, and misrouting info). If there are several possible output ports, then the status of the available ports and the status of the queues at the neighbor nodes are also taken into account.

In order to make the performance results independent of the relative positions of the faults, a large number of simulations has been performed. When required, confidence intervals are provided. For each simulation run, the packet generation rate is constant and equal for all 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], [14]. In all the simulations, the packet length is set to 128 bytes.

5.3.3 Performance Results

We have analyzed the network performance of the following variations of the methodology: I+D, I+D+M, I+M, D+M, and I×2. Up to two intermediate nodes are used for I×N because all the combinations randomly generated for the performance evaluation could be solved by the use of just two intermediate nodes.[16] Furthermore, because most paths (all in this case) can be resolved using two intermediate nodes alone, the performance of I×2+D is very similar to that of I×2 and has not been included in the results.

The evaluation takes into account the overhead in the packet header that is required for each combination of mechanisms. We have run 50 simulations for each number of faults and each combination of mechanisms. That is, for each number of faults, 50 fault combinations were selected randomly. In each of them, the faults were located randomly over the links of the entire network. For each of these fault combinations, we have run a simulation for each combination of mechanisms.

Each point in Fig. 9a represents the mean overall network throughput for the 50 simulations (that correspond to a given fault combination). Results for different numbers of failures in an $8 \times 8 \times 8$ torus network with five virtual channels are shown. Notice that, in all cases, there are at least two adaptive channels.[17] The confidence intervals for the 50 simulations are always lower than ± 5. As can be seen, D+M provides the lowest throughput. Indeed, once a fault is present in the network, the achieved throughput is decreased by 31.6 percent. This is due to several factors, such as the use of deterministic routing for a large number of paths, the use of nonminimal paths, and a larger packet header. For larger numbers of faults, the throughput decreases very slowly.

The I+D, I+D+M, I+M, and I×2 combinations obtain a highly stable throughput, regardless of the number of faults in the network. With one fault in the network, the throughput is only slightly decreased. This is because only intermediate nodes (see Table 3) are used for the affected paths most of the time. Therefore, adaptive routing is not seriously limited. As the number of injected faults increases, the throughput decreases slightly (less when two intermediate nodes are used). This behavior is due to the normal traffic unbalance that the faults progressively introduce in the network. Anyway, even with 14 faults present in the network, the throughput only decreases by about 11 percent for I+D, I+D+M, and I+M. The differences between these combinations of mechanisms are due to the use of nonminimal paths and the extra bits required in the packet headers, by the I+D+M and I+M combinations, for misrouting information. However, although I+D+M and I+M provide a slightly worse performance, note that they tolerate a larger number of faults (up to seven faults). I+M

15. Remember that, alternatively, deadlock freedom along the escape paths could be achieved by using two virtual channels such as stated by Dally and Seitz [13].

16. This is because, in an $8 \times 8 \times 8$ torus, the percentage of not tolerated fault combinations, when using two intermediate nodes, is much lower than in a $3 \times 3 \times 3$ torus.

17. Two adaptive channels for I×2, three for I+D, I+D+M, and I+M, and four for D+M.

provides performance results very similar to I+D+M. For I×2, the performance degradation is even lower. In particular, when there are 14 faults in the network, the throughput is only decreased by 6.9 percent.

Finally, we compare the performance degradation of our methodology against the performance degradation that would be obtained when using a fault-tolerant mechanism similar to the standard mechanism used in the BlueGene/L supercomputer. The BlueGene/L system disables a 512-node plane in order to deal with a fault. As we are using a smaller torus network, we will model the mechanism of the BlueGene/L system by only disabling 32-nodes. Fig. 9b shows the network throughput obtained with three combinations of mechanisms (I+D, I+M+D, and D+M) and the mechanism similar to the one used in the BlueGene/L supercomputer when there are up to seven faults in an $8 \times 8 \times 8$ torus. As the BlueGene/L uses four virtual channels per physical link, in order to do a fair comparison, all the simulations have been run with four virtual channels.

Notice that this is a worst case for the BlueGene/L-like mechanism because it assumes that the seven faults are located in different planes. If the seven faults were located in the same plane, only that plane (32 nodes) would be disconnected. However, our mechanism obtains a higher network throughput for seven faults than the BlueGene/L-like mechanism obtains for one fault (i.e., when only one plane is disconnected). Network throughput degrades only by 6.4 percent and 6.25 percent for I+M+D and I+D, respectively, with seven random faults. When using the BlueGene/L-like mechanism, network performance drops by 6.8 percent and 47.1 percent, with one and seven faults, respectively. However, if the seven faults were located in the same plane (best scenario), the BlueGene/L-like mechanism would reduce performance by 6.8 percent.

Also notice that, in a full-size BlueGene/L system, in the presence of seven faults, the BlueGene/L mechanism disables 512 nodes in the best case and 3,584 nodes in the worst case, thereby reducing the total processing power. Our methodology, on the other hand, inflicts no such decrease in processing power.

5.4 Discussion

In light of the evaluation results, we observe that routing through an intermediate node is the fault-tolerant mechanism most widely used by the methodology. This is good because it allows packets to continue being routed adaptively in the presence of faults, thus avoiding an excessive performance degradation. However, an intermediate node alone is not sufficient in order to guarantee an acceptable fault tolerance degree. In particular, we have observed that also disabling adaptivity for some paths is enough to increase the fault tolerance degree of the methodology from one to five faults. Furthermore, this combination of mechanisms (I+D) is able to tolerate up to 14 faults with a high probability. This is an interesting result, especially when taking into account that the improvement in the fault tolerance degree hardly increases the implementation cost of the methodology. Moreover, it has been shown that only a small percentage of the routing paths need to disable adaptivity. This shows that the combination of using intermediate nodes and disabling adaptivity represents a cost-effective strategy.

An alternative way to tolerate up to five faults is to use an additional intermediate node. The I×2 approach mainly has two advantages. First, the mechanism does not require modifying the way in which packets are routed, which allows us to maintain the same router design used in the absence of intermediate nodes. Second, the I×2 combination allows all packets to be adaptively routed, which contributes to keeping a good network performance. On the other hand, this approach requires using an additional escape channel.

Moreover, if we want to improve the fault tolerance degree of the methodology, misrouting or additional intermediate nodes can be used. In particular, the methodology becomes 7-fault-tolerant when combining the I and M mechanisms (I+M). Furthermore, this combination of mechanisms tolerates up to 14 faults with a very high probability. However, the application of misrouting leads to an increase in the computation time of the fault-free routing paths. However, this fact is insignificant because the new routing paths are computed offline during the recovery process when a static fault model is used. On the other hand, we have observed that it is not worth combining the three mechanisms (I+D+M) because it roughly achieves the same fault tolerance degree and performance level as I+M but at the expense of a slightly higher cost.

Alternatively, when using three intermediate nodes (I×3), the methodology is 9-fault-tolerant. Its main drawback is that it requires a total of four escape channels, although virtual channels are nowadays inexpensive. Considering that very few paths are not resolved using two intermediate nodes alone, the combination of two intermediate nodes with disabling adaptive routing (I×2+D) may be considered a more cost-effective way of providing the same fault tolerance (i.e., tolerating nine faults), saving one escape channel.

Finally, it is observed that, by applying only misrouting and disabling adaptive routing (i.e., without using intermediate nodes), either separately or jointly, is not enough to provide a cost-effective fault-tolerant routing methodology.

6 Conclusions

In this paper, we have proposed a fault-tolerant routing methodology for direct networks that assumes a static fault model and tolerates a reasonably large number of faults without significantly degrading performance. Moreover, unlike other fault-tolerant approaches, the resulting routing strategy does not need to disable any healthy node, does not require too many extra hardware resources, and does not degrade performance in the absence of failures. The methodology requires at least one additional virtual channel.

The proposed methodology can potentially combine three different fault-tolerant mechanisms. Among them, intermediate nodes (I) is the most effective and elegant mechanism to provide fault tolerance, providing the best network performance in the case of failure. To achieve a sufficient fault tolerance degree (tolerating five link faults), it is necessary to use at least two intermediate nodes (I×2) for some paths, requiring another additional virtual channel. The fault tolerance of this mechanism can be further increased by combining it with a simple mechanism such as disabling adaptive routing (D) on a per packet basis. The resulting mechanism (I×2+D) is then 9-fault-tolerant.

However, we have also shown that, by combining one intermediate node with disabling adaptive routing on a per

packet basis (D), it is possible to guarantee an acceptable fault tolerance degree (up to five link faults are tolerated in a $3 \times 3 \times 3$ torus) without significantly affecting the implementation cost or performance. Therefore, the resulting mechanism (I+D) could be considered a cost-effective mechanism.

If a higher fault tolerance degree is required, without adding additional virtual channels, an intermediate node combined with misrouting (M) can be used, at the expense of increasing complexity. This mechanism (I+M) is 7-fault-tolerant (in a $3 \times 3 \times 3$ torus) with very high probability and exhibits a network performance slightly lower than that of I+D.

To sum up, we can conclude by stating that routing through intermediate nodes is the key mechanism to providing a cost-effective fault-tolerant routing methodology for direct networks.

ACKNOWLEDGMENTS

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

REFERENCES

[1] ASCI Red Web Site, http://www.sandia.gov/ASCI/Red/, 2003.
[2] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J. Seizovic, and W. Su, "Myrinet—A Gigabit-per-Second Local Area Network," *IEEE Micro,* pp. 29-36, Feb. 1995.
[3] R. Bopana, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W. Su, "A Comparison of Adaptive Wormhole Routing Algorithms," *Proc. 20th Ann. Int'l Symp. Computer Architecture,* pp. 351-360, May 1993.
[4] R.V. Boppana and S. Chalasani, "Fault-Tolerant Wormhole Routing Algorithms for Mesh Networks," *IEEE Trans. Computers,* vol. 44, no 7, pp. 848-864, July 1995.
[5] The BlueGene/L Team, "An Overview of the BlueGene/L Supercomputer," *Proc. ACM/IEEE Conf. Supercomputing,* pp. 1-22, Nov. 2002.
[6] C. Carrion, R. Beivide, J.A. Gregorio, and F. Vallejo, "A Flow Control Mechanism to Avoid Message Deadlock in K-Ary N-Cube Networks," *Proc. Fourth Int'l Conf. High Performance Computing,* pp. 332-329, Dec. 1997.
[7] 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 Trans. Parallel and Distributed Systems,* vol. 12, no. 2, pp. 115-132, Feb. 2001.
[8] A.A. Chien and J.H. Kim, "Planar-Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors," *Proc. 19th Int'l Symp. Computer Architecture,* pp. 268-277, May 1992.
[9] S. Chalasani and R.V. Boppana, "Fault-Tolerant Wormhole Routing in Tori," *Proc. Eighth Int'l Conf. Supercomputing,* pp. 146-155, July 1994.
[10] S. Chalasani and R.V. Boppana, "Communication in Multicomputers with Nonconvex Faults," *IEEE Trans. Computers,* vol. 46, no. 5, pp. 616-622, May 1997.
[11] C.L. Chen and G.M. Chiu, "A Fault-Tolerant Routing Scheme for Meshes with Nonconvex Faults," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 5, pp. 467-475, May 2001.
[12] C.M. Cunningham and D.R. Avresky, "Fault-Tolerant Adaptive Routing for Two-Dimensional Meshes," *Proc. First Ann. Int'l Symp. High Performance Computing Architecture,* pp. 122-131, Jan. 1995.
[13] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers,* vol. 36, no. 5, pp. 547-553, May 1987.
[14] W.J. Dally, "Virtual-Channel Flow Control," *IEEE Trans. Parallel and Distributed Systems,* vol. 3, no. 2, pp. 194-205, 1992.
[15] W.J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 4, pp. 466-475, 1993.
[16] B.V. Dao, J. Duato, and S. Yalamanchili, "Configurable Flow Control Mechanisms for Fault-Tolerant Routing," *Proc. 22nd Int'l Symp. Computer Architecture,* pp. 220-229, June 1995.
[17] J. Duato, "A Theory of Fault-Tolerant Routing in Wormhole Networks," *Proc. Int'l Conf. Parallel and Distributed Systems,* pp. 600-607, Dec. 1994.
[18] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-and-Forward Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 8, pp. 841-854, Aug. 1996.
[19] Earth Simulator Center, http://www.es.jamstec.go.jp/esc/eng/index.html, 2006.
[20] A. Gara et al., "Overview of the Blue Gene/L System Architecture," *IBM J. Research & Development,* vol. 49, no. 2, pp. 195-212, Mar./May 2005.
[21] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," *Proc. Int'l Symp. Computer Architecture,* pp. 278-287, May 1992.
[22] M.E. Gómez, J. Duato, J. Flich, P. Lopez, 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.
[23] M.E. Gómez, J. Duato, J. Flich, P. Lopez, A. Robles, N.A. Nordbotten, T. Skeie, and O. Lysne, "A New Adaptive Fault-Tolerant Routing Methodology for Direct Networks," *Proc. Int'l Conf. High Performance Computing,* pp. 462-473, Dec. 2004.
[24] M.E. Gómez, J. Flich, P. Lopez, A. Robles, and J. Duato, N.A. Nordbotten, O. Lysne, and T. Skeie, "An Effective Fault-Tolerant Routing Methodology for Direct Networks," *Proc. Int'l Conf. Parallel Processing,* pp. 222-231, Aug. 2004.
[25] C.T. Ho and L. Stockmeyer, "A New Approach to Fault-Tolerant Wormhole Routing for Mesh-Connected Parallel Computers," *IEEE Trans. Computers,* vol. 53, no. 4, pp. 427-439, Apr. 2004.
[26] Z. Jiang, J. Wu, and D. Wang, "A New Fault Information Model for Fault-Tolerant Adaptive and Minimal Routing in 3-D Meshes," *Proc. Int'l Conf. Parallel Processing,* pp. 500-507, June 2005.
[27] InfiniBand℠ Trade Assoc., http://www.infinibandta.com, 2006.
[28] P.T. Gaughana and S. Yalamanchili, "A Family of Fault-Tolerant Routing Protocols for Direct Multiprocessor Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 6, no. 5, pp. 482-497, May 1995.
[29] P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks,* vol. 3, pp. 267-286, 1979.
[30] T. Lee and J.P. Hayes, "A Fault-Tolerant Communication Scheme for Hypercube Computers," *IEEE Trans. Computers,* vol. 41, no. 1, pp. 1242-1256, Oct. 1992.
[31] D.H. Linder and J.C. Harden, "An Adaptive and Fault-Tolerant Wormhole Routing Strategy for k-Ary n-Cubes," *IEEE Trans. Computers,* vol. 40, no. 1, pp. 2-12, Jan. 1991.
[32] O. Lysne, T. Pinkston, and J. Duato, "A Methodology for Developing Dynamic Network Reconfiguration Processes," *Proc. Int'l Conf. Parallel Processing,* pp. 77-86, Oct. 2003.
[33] O. Lysne, J.M. Montañana, T.M. Pinkston, J. Duato, T. Skeie, and J. Flich, "Simple Deadlock-Free Dynamic Network Reconfiguration," *Proc. Int'l Conf. High Performance Computing,* pp. 504-515, Dec. 2004.
[34] N.A. Nordbotten, M.E. Gómez, J. Flich, P. Lopez, A. Robles, T. Skeie, O. Lysne, and J. Duato, "A Fully Adaptive Fault-Tolerant Routing Methodology Based on Intermediate Nodes," *Proc. IFIP Int'l Conf. Network and Parallel Computing,* pp. 341-356, Oct. 2004.
[35] F. Petrini, W. Feng, A. Hoisie, S. Coll, and E. Frachtenberg, "The Quadrics Network (QsNet): High-Performance Clustering Technology," *Proc. Ninth IEEE Hot Interconnects,* Aug. 2001 (original version), *IEEE Micro,* pp. 46-57, Jan./Feb. 2002 (extended version).
[36] 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. Int'l Conf. Parallel Processing,* pp. 58-67, Sept. 1999.
[37] V. Puente, J.A. Gregorio, R. Beivide, and F. Vallejo, "A Low Cost Fault-Tolerant Packet Routing for Parallel Computers," *Proc. Int'l Parallel and Distributed Processing Symp.,* Apr. 2003.
[38] V. Puente, J.A. Gregorio, F. Vallejo, and R. Beivide, "Immunet: A Cheap and Robust Fault-Tolerant Packet Routing Mechanism," *Proc. Int'l Symp. Computer Architecture,* pp. 198-211, June 2004.
[39] J.C. Sancho, A. Robles, and J. Duato, "A Flexible Routing Scheme for Networks of Workstations," *Proc. Int'l Conf. High Performance Computing,* pp. 260-267, Oct. 2000.

[40] R. Schwitters, "Requirements of ASCI," The MITRE Corp., JASON Program Office, 2003.
[41] F. Silla, "Routing and Flow Control in Networks of Workstations," PhD thesis, Mar. 1999.
[42] Y.J. Suh, B.V. Dao, J. Duato, and S. Yalamanchili, "Software-Based Rerouting for Fault-Tolerant Pipelined Communication," *IEEE Trans. Parallel and Distributed Systems,* vol. 11, no. 3, pp. 193-211, 2000.
[43] S.L. Scott and G.M. Thorson, "The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus," *Proc. Hot Interconnects IV,* pp. 147-156, Aug. 1996.
[44] T.M. Pinkston, R. Pang, and J. Duato, "Deadlock-Free Dynamic Reconfiguration Schemes for Increased Network Dependability," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, no. 8, pp. 780-794, Aug. 2003.
[45] L.G. Valiant, "A Scheme for Fast Parallel Communication," *SIAM J. Computing,* vol. 11, no. 2, pp. 350-361, 1982.
[46] J. Wu, "Unicasting in Faulty Hypercubes Using Safety Levels," *Proc. Int'l Conf. Parallel Processing,* vol. 3, pp. 132-136, Aug. 1995, also available as Technical Report TR-CSE-95-2, Dept. of Computer Science and Eng., Florida Atlantic Univ.
[47] J. Wu, "A Fault-Tolerant and Deadlock-Free Routing Protocol in 2D Meshes Based on Odd-Even Turn Model," *IEEE Trans. Computers,* vol. 52, no. 9, pp. 1154-1169, Sept. 2003.
[48] J.P. Zhou and F.C. M. Lau, "Multi-Phase Minimal Fault-Tolerant Wormhole Routing in Meshes," *Parallel Processing,* vol. 30, no. 3, pp. 423-442, 2004.

María Engracia Gómez received the MS and PhD degrees in computer science from the Universidad Politécnica de Valencia, Spain, in 1996 and 2000, respectively. She joined the Department of Computer Engineering (DISCA) at the Universidad Politécnica de Valencia in 1996, where she is currently an associate professor of computer architecture and technology. Her research interests are in the field of interconnection networks. She is a member of the IEEE.

Nils Agne Nordbotten received the Master's degree in communication systems in 2003 from the University of Oslo, Norway. He is currently a PhD student at the Simula Research Laboratory, where the focus of his work is on fault tolerance in interconnection networks. Previously, he has also been involved in the areas of service discovery, Bluetooth, and ad hoc networks.

José Flich received the MS and PhD degrees in computer science from the Technical University of Valencia (Universidad Politécnica de Valencia), Spain, in 1994 and 2001, respectively. He joined the Department of Computer Engineering (DISCA), Universidad Politécnica de Valencia in 1998, where he is currently an associate professor of computer architecture and technology. His research interests are related to high performance interconnection networks for multiprocessor systems and cluster of workstations.

Pedro López received the BEng degree in electrical engineering and the MS and PhD degrees in computer engineering from the Universidad Politécnica de Valencia, Spain, in 1984, 1990, and 1995, respectively, where he is a full professor of computer architecture and technology in the Department of Computer Engineering (DISCA). He has taught several courses on computer organization and architecture. His research interests include high-performance interconnection networks for multiprocessor systems and cluster of workstations. Prof. López has published more than 70 refereed conference and journal papers. He is a member of the editorial board of *Parallel Computing*. He is a member of the IEEE Computer Society.

Antonio Robles received the MS degree in physics (electricity and electronics) from the University of Valencia, Spain, in 1984 and the PhD degree in computer engineering from the Technical University of Valencia in 1995. He is currently a full professor in the Department of Computer Engineering at the Technical University of Valencia. His current research interests include multiprocessor systems, clusters of workstations, interconnection networks, fault tolerance, and network-based computing. He has served on program committees for several major conferences. He is a member of the IEEE Computer Society.

Jose Duato received the MS and PhD degrees in electrical engineering from the Technical University of Valencia, Spain, in 1981 and 1985, respectively. Currently, he is a professor in the Department of Computer Engineering (DISCA) at the same university. He was also an adjunct professor in the Department of Computer and Information Science, The Ohio State University. His current research interests include interconnection networks and multiprocessor architectures. He has published more than 300 refereed papers. He proposed a powerful theory of deadlock-free adaptive routing for wormhole networks. He is a coauthor of the book *Interconnection Networks: An Engineering Approach*. He is a member of the IEEE and the IEEE Computer Society.

Tor Skeie received the PhD degree in computer science from the Department of Informatics, University of Oslo in 1998. He is affiliated with the Simula Research Laboratory and the Department of Informatics, University of Oslo, Norway. He has been working for years in the interconnection networking problem domain, focusing on the science and technology of how to connect point-to-point links and switches into scalable network topologies and how to route packets effectively in these networks so that they yield the highest possible performance. This includes fulfillment of various fault tolerance, quality of service, and other nonfunctional requirements.

Olav Lysne received the Master's degree in 1988 and Dr.Scient. degree in 1992, both from the University of Oslo. He is the research director at the Simula Research Laboratory and a professor of computer science at the Simula Research Laboratory and the University of Oslo. His early research contributions were in the field of algebraic specification and term rewriting, with a particular emphasis on automated deduction. While working in this field, he was a visiting researcher at Université de Paris-Sud. In later years, he has mainly been working on switching techniques, such as wormhole switching and virtual cut through, focusing on problems like effective routing, fault tolerance, and Quality of Service. He has published approximately 70 academic papers. He is a member of the IEEE and the IEEE Computer Society.