A Low Cost Fault Tolerant Packet Routing for Parallel Computers

V. Puente, J.A. Gregorio, R. Beivide and F. Vallejo
Computer Architecture Group
University of Cantabria, Spain
{vpuente, jagm, mon, fernando}@atc.unican.es

Abstract

This work presents a new switching mechanism to tolerate arbitrary faults in interconnection networks with a negligible implementation cost. Although our routing technique can be applied to any regular or irregular topology, in this paper we focus on its application to k-ary n-cube networks when managing both synthetic and real traffic workloads. Our mechanism is effective regardless the number of faults and their configuration. When the network is working without any fault, no overhead is added to the original routing scheme. In the presence of a low number of faults, the network sustains a performance close to that observed under fault-free conditions. Finally, when the number of faults increases, the system exhibits a graceful performance degradation.

1. Introduction

Current [1] trends in computational demands are provoking the proliferation of parallel servers and supercomputers with a large number of processing elements. Reliability should be an important feature of these complex parallel systems. Traditionally, fault tolerance has referred to building systems from redundant components that, used in parallel, are normally applied to some critical mission or application. This design approach has not been broadly considered in general-purpose computers because the mean time between failures of an isolated component is usually sufficiently high. Nevertheless, in a parallel architecture with hundreds of processing nodes the sum of all the individual failure probabilities can be considerable and some mechanism should be incorporated to provide a graceful degradation system.

Reconfiguring a parallel system to get around its faults is a good approach to reliability enhancement, since the system may continue operating after reconfiguration. To provide this possibility in an efficient way, an appropriate design of the system interconnection network is needed. Moreover, as the interconnection subsystem itself is an important source of potential faults, robust network designs should be compulsory in massively parallel computers.

One of the main problems in designing a fault tolerant network is that deadlock avoidance mechanisms conceived for normal operation are no longer applicable in the presence of faults. In a partially operative system, the routing mechanisms should allow the rest of system to continue working in a deadlock-free condition. Actually, a truly fault tolerant interconnection network should allow for the communication between two nodes as long as there is an available physical path.

Due to the arbitrary nature of failures, finding trustworthy and inexpensive techniques to tolerate them can be a critical task when considering commercial solutions. In general, real systems implement very simple mechanisms that partly addressed the problem, such as the direction order routing used in the Cray T3E [13]. Nevertheless, the design of fault tolerant networks has been well documented in the technical literature. A fault tolerant algorithm for Meshes requiring 4 virtual channels to avoid deadlock in a network with rectangular regions in failure was proposed in [2]. The same authors improved their algorithm to tolerate non-convex failures in [4]. By adapting these ideas, the same methodology was applied to Torus networks but requiring up to 6 virtual channels to tolerate rectangular regions in failure [3]. More recent works have allowed the consideration of a broader range of failures while increasing the number of resources [14]. A different fault tolerant adaptive routing that uses deadlock detection and recovery mechanisms was presented in [17]. Other authors propose new topologies specifically conceived to improve the system fault tolerance [16]. Some of the drawbacks of such mechanisms are the limitation of dealing with a restricted number of network faults, the use of specific failure regions and the dependence of a particular topology. Furthermore, the high associated hardware costs, which could even re-

[1]This work has been supported by Spanish CICYT, project TIC2001-0591-C02-01.

duce the network performance in the absence of faults, limit the applicability of fault tolerant technology.

In this research the basis of a new fault tolerant packet routing for any kind of interconnection network is presented and evaluated. We presume the existence of a diagnosis mechanism and focus on how to use the diagnosis information to design a robust and reliable fault tolerant communication system. The analysis of the network performance under different failure conditions and workloads allow us to assure that our switching mechanism exhibits a graceful degradation. Specifically, our proposal relies on the use of Bubble Flow Control, a deadlock avoidance mechanism successfully applied to regular and irregular interconnection networks [12] [11]. Our fault tolerant routing is based on the permanent existence of a safe path able to communicate any pair of surviving nodes.

The proposed mechanism does not affect the network performance in absence of failure and it allows the system to handle any number and configuration of faults (obviously, assuming that the network remains connected). Furthermore, its hardware cost is almost negligible. Our technique is clearly suitable for networks having a high number of nodes, each of them with a low MTBF (Mean Time Between Failures). Besides, due to the slight performance degradation in the presence of a manageable number of faults, our fault tolerant routing is also a viable solution for systems with high MTTR (Mean Time To Repair).

In this paper, the authors demonstrate the advantages of this routing technique by means of its application to k-ary n-cube networks although any other topology could be also considered. Besides the typical synthetic workloads, several real applications running on a complete execution-driven cc-NUMA simulator have been carried out in order to offer a realistic scenario to evaluate our method. The rest of the paper is organized as follows: In Section 2 we will introduce the context where our routing mechanism is going to be used. In Section 3 we will consider the architecture and the implementation costs of the proposed interconnection subsystem. Section 4 will be devoted to analyzing the performance exhibited by our mechanism under both synthetic and real traffic workloads. Finally, in Section 5, the main conclusions of this work will be summarized.

2. Interconnection network characteristics

In this section, we present the context in which our proposal will be applied. The selected network topology, the router structure and the packet flow control function are introduced and analyzed.

Although our reconfigurable routing mechanism can be used without restrictions in any topology, as stated before, in this paper we will focus on analyzing its application to k-ary n-cube networks. As is known, these networks have frequently been implemented in several commercial systems due to both its good cost/performance ratio and scalability [8] [13]. Each router can inject packets from one or more computing elements to the network. Conversely, each router can eject packets from the network to one or more computing nodes. Obviously, the router's mission is to convey packets towards their destination. The design of this element has to maximize the use of the network resources avoiding communication anomalies such as packet deadlock, livelock and starvation.

Figure 1 shows our basic router organization. In addition to the usual hardware modules (crossbar, buffers, arbitration logic, synchronization, etc.), we employ a table to route packets toward their destination. Although arithmetic routing can be employed in k-ary n-cubes, table-based routing offers the necessary flexibility for implementing our fault tolerant switching mechanism. In fact, most modern parallel systems rely on this routing implementation. The routing table initialization will be carried out at boot time as in the SGI Spider [6] or the 21364 Alpha [8]. With current hardware technology the network scalability is not compromised by the table size.

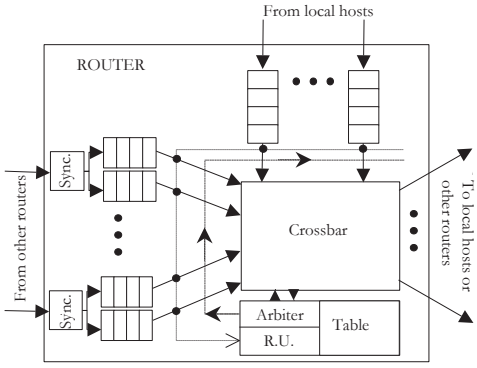

Figure 1. Basic router organization.

Our router must have two virtual channels per input link in order to support fully adaptive routing using a technique derived from [5]. A subset of the total virtual channels will be configured as an escape virtual network for potentially blocked packets and the rest will be configured as an adaptive virtual network. Bubble Flow Control (BFC) is going to be used to regulate packet injection on the escape virtual network to avoid exhausting its buffer resources. BFC will be applied to one or several virtual rings embedded in the network that includes all the network nodes. This set of virtual rings constitutes the escape virtual network. In our mechanism, any node in an escape virtual ring can transmit packets as regulated by Virtual Cut-Through flow control (VCT) [7]. To enable a packet transmission between two nodes, VCT flow control must verify the existence of a free buffer on the destination, which can eventually store

the whole packet in case it blocks at that node. Nevertheless, packet injection is a more restricted process that demands the existence of two free buffers in the virtual channel of every node trying to incorporate a new packet in a BFC ring (Bubble Condition). As a node in a BFC ring can simultaneously inject and receive a packet, we have to apply BFC at any router injecting a packet to assure that its buffer space will never be exhausted. As there always will be at least one free buffer in the ring (a *Bubble* under our terminology), transit packets can progress and deadlock never occurs. The Bubble Condition will be verified using only local information about the packet population in the router buffers.

A simplified example that illustrates how this mechanism operates is shown in Figure 2. When all buffers are exhausted no packet can advance to the following router and the network is in a deadlocked condition. If the injection of packets that can exhaust the last storage space is restricted, deadlock will never occur. It is possible that, if permission is granted to inject P_z from node 3 into node 1 because there is free space for it, simultaneously the packet P_y could begin to be transmitted to node 3. If this situation occurs simultaneously in all the routers composing the ring, packet deadlock is assured. Notwithstanding, by applying BFC, such a situation can never arise. In Figure 2, only P_x will be a candidate to be injected in the ring. Obviously, this packet must compete with P_y to obtain the output port once it is granted the permission by VCT flow control.

Note that transit packets are more likely to advance in the network than new ones trying to be injected. Consequently, this strategy if used in isolation, may lead to packet starvation. However, when this deadlock-free network is combined with another adaptive virtual network, packet starvation is eliminated [12]. In the adaptive virtual network, all the packets, new or in transit, are regulated only by VCT flow control, so all packets will progress, including those at the injection queues.

In absence of faults, the virtual escape network for a k-ary n-cube topology is constituted by a collection of $2nk^{n-1}$ BFC rings of size k, as represented in Figure 3. When these rings are visited under Dimension Order Routing (DOR), deadlock-free communications are assured in the resulting escape network. Then, nk^{n-1} virtual channels will compose the escape virtual network. The other nk^{n-1} virtual channels will constitute the fully-adaptive virtual network. Changes from the escape to the adaptive network are possible and regulated by VCT flow control. Changes from the adaptive to the escape network are treated as a new packet injection and therefore, regulated under Bubble Flow Control. A more detailed description of BFC and a study showing its superior performance in respect to other traditional router alternatives can be seen in [12].

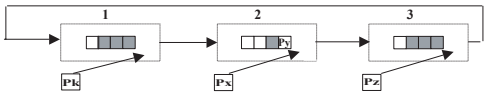

Figure 2. Simple example of BFC application over a ring.

3 Fault-Tolerant network architecture

In this Section, we will describe the kinds of faults considered in this research and the corresponding architectural support which palliates their effect on the performance and survivability of the networks under study. At the end of the Section, an evaluation of the added hardware costs will be considered.

3.1. Fault Model

Depending on their nature, two different kinds of network faults can be considered: link faults and router faults. The first class is related to physical faults in the media used to interconnect the routers. The fault can be uni-directional or bi-directional but we assume that both types cause a communication loss between two neighbor routers. When a router fault appears, the device interrupts communication with all the neighboring routers. Therefore, the computing node or nodes attached to it cannot communicate with any other processor in the system. A viable fault tolerant mechanism must be able to deal with any number and configuration of network faults and it must allow communication between two computing nodes while a physical path exists between them. Our method fulfills these conditions.

3.2. Fault-Tolerant routing mechanism

As stated before, one of the most complex problems inherent in handling any combination of link and/or node faults is that such faults will induce topological changes affecting the deadlock avoidance mechanism. For example, in a 2D Torus a fault in any link breaks down one BFC escape ring and, therefore, it is not always possible to use Dimensional Order Routing to route packets through the virtual escape network. Most of the proposed solutions add new resources to maintain deadlock-free communications but notably increasing the router complexity. Our approach, in contrast, is based on rearranging the shape and number of rings which compose the virtual escape network.

There are several algorithms for determining the topology of our escape network. The one proposed in this paper is based on a unique directed ring embedded on the network that can visit each node one or more times up to the node

degree. This ring is based on a specific tour through a spanning tree, always embedded in any arbitrary topology. To obtain the spanning tree, we employ an algorithm based on random link elimination, but any other of the existing methods could also be employed. The escape ring topology will be determined by a peripheral tour through this tree. We can illustrate the algorithm used to obtain the escape ring as follows: We trace, without lifting our pencil from the paper, a path through the tree connecting every vertex and visiting the leaves as soon as possible. We may return to each vertex as many times as needed to visit all its children, finally returning to the starting vertex. The resulting tour will visit all the nodes at least once and each edge twice. As the tree has $N-1$ links, the resulting directed escape ring will have $2(N-1)$ links. In fact, if we consider the spanning tree as a directed graph having unidirectional links, our escape ring constitutes an Eulerian tour inside this tree. Such a virtual ring, like the one shown in Figure 4 for a 4-ary 2-cube with 12 faulty links, always exists in a connected graph and is easy to find regardless of the type and number of faults present. In fact, there are a number of simple algorithms that can be explored to find a safe ring traversing all the nodes. The quasi-linear complexity of our algorithm based on a tree tour, makes it suitable to be employed even in very large networks. In a different context, a similar trip-based model has been employed in [15] to support multicast communications in wormhole-routed networks. As a possible optimization, it is clear that a Hamiltonian path through the network would provide us with two opposite minimal length directed escape rings. Although almost any regular network is Hamiltonian, the search of Hamiltonian paths in irregular graphs is an NP complete problem. Moreover, an arbitrary graph does not necessary have a Hamiltonian path. In our experiments, in parallel with the search of the tour-based escape ring, we will employ a backtracking algorithm to find Hamiltonian paths on undirected graphs, as the one proposed in [1]. For example, in the 4-ary 2-cube with 4 faulty links shown in Figure 5 it is possible to find a Hamiltonian path leading to two opposite virtual escape rings. We will abort the backtracking algorithm if it does not provide a solution within the time required to the establishment of the tour-based ring.

Anyway, whichever escape topology is used, one or two directed virtual rings traversing all the healthy network nodes are going to be used as escape paths. As these rings use non-minimal routing, packet livelock could arise. A packet traveling through a non-minimal routing escape ring can be incorporated into the adaptive network at any router, provided that there is room in the selected adaptive buffer. The packet may need again to enter the escape ring, getting further from its destination. Thus, this packet may indefinitely travel among virtual networks and never arrive to destination. Nevertheless, the livelock anomaly disappears just by limiting the number of times that a packet can abandon the escape channels.

In conclusion, the routing operation mode in both a healthy and a faulty network only differs in the escape network used. Without faults, we will use as many independent virtual rings as the topological cycles dictated by the wrap-around connections, visiting them under DOR routing. When faults arise, we try to obtain a Hamiltonian path to be used as an escape ring. If a quick answer is not obtained due to the number and configuration of the faults, then it is always possible to use the longer escape ring derived from the tour through the spanning tree.

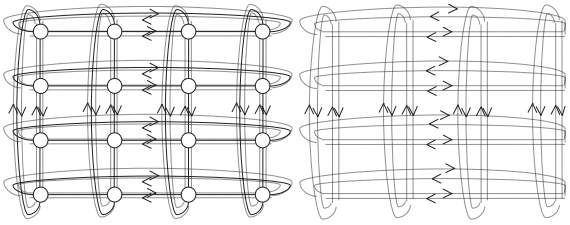

Figure 3. Escape paths on a fault-free 4x4 Torus.

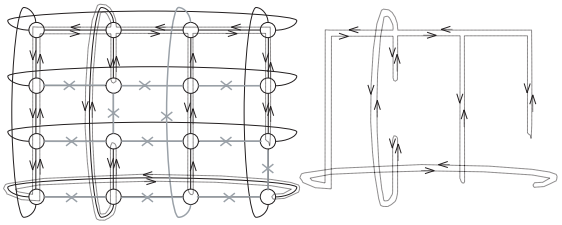

Figure 4. Tree-based escape path for a 4x 4 torus with 12 faulty links.

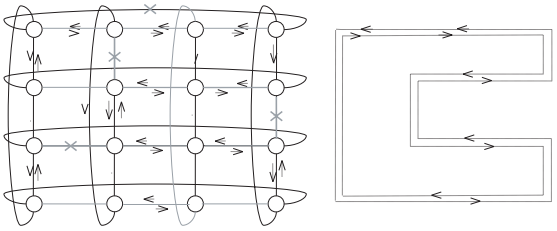

Figure 5. Hamiltonian-based escape paths for a 4x 4 torus with 4 faulty links.

3.3. Hardware cost

When a fault arises, the routing tables must be updated to reflect the topological changes and the resources lost caused by the fault. The table reprogramming can occur at boot time [6] [8], or dynamically without resetting the system [13]. In the case of dynamic reconfiguration, only local information neighbor's status would be necessary. To implement our fault tolerant routing mechanism, it must be possible to reconfigure in each router the local structure of the escape network. For example, in a fault-free network any router has a configuration of escape paths similar to the one shown in Figure 6(a). Using our methodology, in the case of a *West* link failure, we could reconfigure the internal escape connections in the way reflected in Figure 6(b). To apply BFC, we have to know the pairs "input channel/output port" belonging to the escape network. Hence, we must take into account that the relationship between input and output router terminals can change over time.

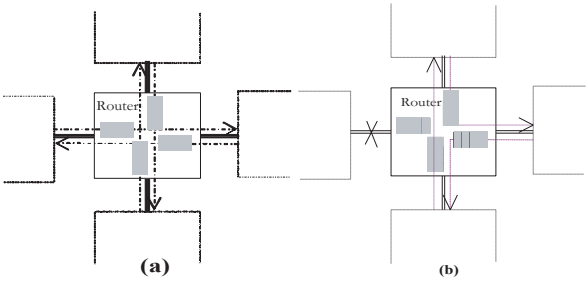

Figure 6. Escape paths reconfiguration (a) before and (b) after, a link fault.

To perform internal reconfiguration, the router has an additional small table of $(p^2 + p)$ bits, p being the number of the router ports, as shown in Figure 7. This table records the needed information about the escape paths configuration. BFC implementation is quite simple. The main routing table must contain all the profitable output channels for a given destination. In each router, at least, four virtual channels are adaptive ones. Similarly, at most, four virtual channels could belong to the escape paths. If a profitable output channel for advancing a packet belongs to the escape virtual network and the packet movement implies an injection in such a network, it is necessary to check the fulfilling of the Bubble Condition before sending the request to the arbiter

To illustrate the above mechanism we focus in the example showed in Figure 7. The escape paths configuration is represented by doted lines at the right part of the Figure. All the adaptive profitable channels labeled as "vc1" in the main routing table can be requested to the arbiter without any limitation. In some cases, before requesting the remaining profitable channels labeled as "vc2", the Bubble Condition must be checked. For example, for advancing a packet stored in the "vc2" channel associated to input port 0 to the "vc2" channel associated to the output port 3, Bubble condition is irrelevant as the packet continues traveling through the escape ring. Nevertheless, if the same packet tries to advance towards the "vc2" channel associated to output port 1, Bubble condition must be verified as this packet movement represents a new injection in the escape ring. If a packet stored at any "vc1" channel tries to advance to any of the "vc2" channels associated to output ports 0, 1 or 3, again Bubble condition must be fulfilled. To distinguish among all these cases, we use the additional small table. The value of any bit at position (i, j) indicates when Bubble condition must be checked. A zero means that Bubble condition fulfillment is required to advance a packet from input channel i to output port j.

As it can be seen, the required additional table and its control logic is fairly simple. We will employ in our experiments a pipelined router having five stages as the one presented in [12]. Although this table is located in the critical path of the routing stage, it is known that this pipeline stage does not determine the router clock cycle. Usually, the crossbar arbitration stage is more costly, so there will be no increment in the router clock cycle. In conclusion, the added cost in respect to a router without fault tolerance capabilities is imputable only to the reconfiguration of the escape paths. Moreover, no overhead is added to the router pass time.

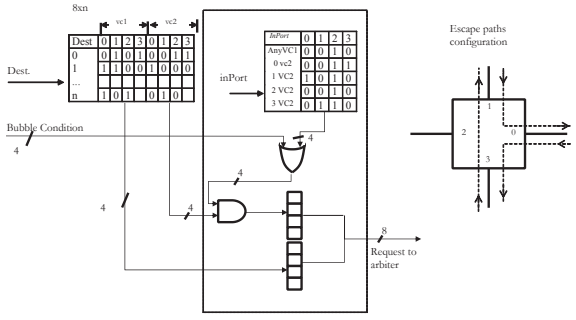

Figure 7. Added complexity in the routing unit to support escape path reconfiguration in bi-dimensional networks (example).

4 Network performance analysis

In order to assess the viability of our proposal, several performance measurements on healthy and faulty k-ary n-cube networks are going to be presented, analyzed and compared. The results obtained show a graceful system

degradation under any combination of network faults. In this work, we have decided to compare performance results among different healthy and faulty networks only using our fault tolerant routing. A number of reasons support this decision. First of all, our original routing mechanism without fault tolerant capabilities outperforms any other typical routing algorithm having similar hardware costs [12]. Second, regardless the operating conditions of the network, the use of our fault tolerant routing does not imply any increment on the router pass time. In addition, up to our knowledge, the proposed mechanism is the cheapest one in terms of the hardware costs. Finally, our intention here is to highlight the graceful degradation exhibited by our switching mechanism.

A simulator denoted as SICOSYS has been employed to carry out this study [10]. Its main advantages with respect to hardware-level simulators are its similar high accuracy and its lower computational cost. To study the performance degradation suffered by a faulty state-of-the-art CC-NUMA multiprocessor running realistic workloads, ED-SICOSYS has been employed [10]. This execution-driven simulator has been derived from RSIM [9] by replacing its original network module, NETSIM, with our more detailed and flexible SICOSYS simulator.

4.1. Study under synthetic traffic

Firstly, we will focus on analyzing the network response to a progressive fault injection process under random traffic conditions. Failures were randomly generated and each experiment contemplating more than one faulty element was simulated 20 times. For a clear analysis, we will consider node and link faults separately and only bi-directional fault links. The network under study was a 64-node (8x8) Torus managing packets of 40 phits. Figure 8 and Figure 9 show the average results of packet throughput for a different number of faulty links and faulty nodes respectively.

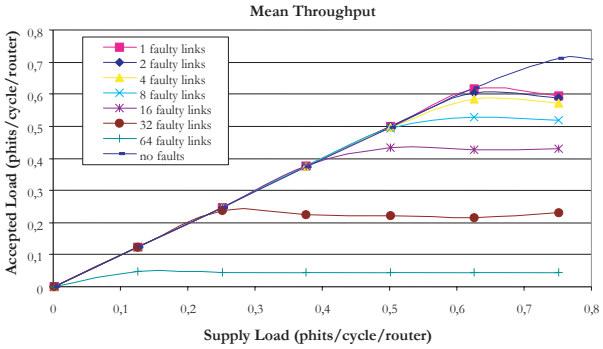

Figure 8. Impact of link faults of an 8x8 torus under uniform traffic pattern.

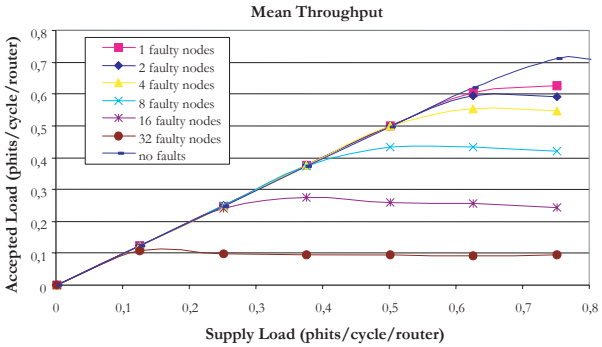

Figure 9. Impact of node faults of an 8x8 torus under uniform traffic.

With a low number of faults, just a small degradation in network throughput can be observed. Under one fault of any kind (link or node), the maximum achievable throughput falls less than 15%. The throughput remains practically unchanged until four faults occur. It must be noted that the system can always sustain a throughput level close to its maximum value beyond the network saturation point. Additionally, in the presence of a low number of faults, base latency degradation is almost negligible. The main reason explaining this behavior is that most of the packets use minimal paths to reach their destination without traveling through the escape paths. Besides, the average distance from a topological point of view remains nearly unchanged when the number of faults is low. When the proportion of faults increases, the topological average distance is longer, which translates in a higher base latency. Finally, we must highlight that even with a very high number of faults, the network remains operative. Note that 64 faulty links in a 64-node Torus represent half of the total network links. It is important to remark that the performance of a fault-free network using Bubble Flow Control is higher than those offered by other current proposals [12].

For the previously shown results, in 40% of the networks it was not possible to find, a Hamiltonian path for more than 2 faults. It is clear that the suitability of our controlled injection routing mechanism will depend on the impact of the selected topology for implementing the escape network on the overall network performance. Fortunately, we can assure that this impact is almost negligible. To prove this fact, we analyze the performance in a healthy 8x8 Torus using the two different previously considered virtual escape networks separately, as shown in Figure 10. It can be seen that the differences between both approaches are negligible. This behavior can be explained by examining the way in which the escape network is used and by considering the average length traversed by potentially blocked packets. It must be remembered that the escape network in our switching mech-

anism is used only as the last routing alternative. Moreover, as changes from the escape network to the adaptive one are permitted at any time, packets always try to travel through the shorter adaptive routes. Besides, the restricted injection mechanism controlling the escape network reduces the volume of traffic this virtual network can manage. In conclusion, the average use of the escape virtual channels is clearly lower than the use of the adaptive virtual channels.

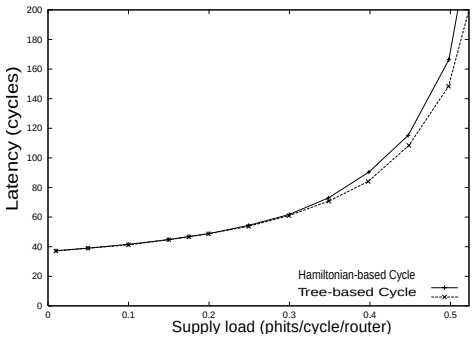

Figure 10. Latency values for the two different escape paths.

We can illustrate the low utilization of the escape network by analyzing the packet average distances in respect to the traffic volume for the two different alternatives, as shown in Figure 11. In both cases, only a light increment with respect to the network average distance was measured. This increment is, obviously, proportional to the number of packets which uses the escape path. Nevertheless, only a tiny difference in throughput of around 1% can be observed when the two escape network alternatives are compared. These experimental measures show that the performance of our routing mechanism is quite independent of the selected ring for implementing the escape virtual network, which confirms the versatility of our fault tolerant routing mechanism.

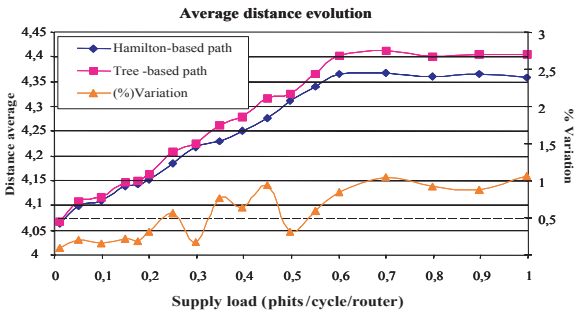

Figure 11. Average distance variation between hamiltonian and tree escape paths.

4.2. Study under realistic workload conditions

To assess the network behavior under realistic workload conditions, the impact of an increasing number of faults on the execution time of different parallel applications has been analyzed. To assure the finalization of the programs we have considered only link faults that cannot isolate any computing node. We will emulate a multiprocessor system with 64 nodes assuming that each network router has attached a single-processor computing node. Also, given the high computational cost of this analysis, just one of the 20 random samples for each faulty network was considered. That is, we simulate a single network in failure for each number of faulty links. This network was the one whose performance under synthetic traffic was closest to the average value observed with the 20 samples previously considered.

The parameters of the CC-NUMA multiprocessor emulated in this paper (cache coherence protocol, processor architecture, memory hierarchy, etc.) have the default values set by RSIM except for the cache line size (32 bytes), the command packet size (8 bytes) and the processor speed which has been established at 650 MHz. As the physical channel width or phit size is 2 bytes, a data packet will contain 40 bytes or 20 phits. The command packets, request or invalidation, are consequently 4 phits long. The router clock was set to 177MHz, as derived from a specific implementation presented in [12].

To carry out this realistic evaluation, we fed our simulation platform with three applications selected from the SPLASH-2 suite: Radix, FFT and LU, which had already been ported into RSIM by researchers at Rice University [9]. These three applications were selected because they have significant communication demands, and each one represents a different case of network load. Radix applies a high pressure in terms of volume of information to be handled by the network while exhibiting a practically uniform communication pattern. FFT, however, applies a medium load on the network but the communication pattern has no spatial locality. Finally, LU applies lower load on the network but it gives rise to hot spots. The default problem size for FFT is 64K double complexes. Due to the high demand for computational resources, the problem size for LU has been reduced from its default value of 512x512 to 256x256. The problem size for Radix has also been reduced from one million integer keys to a half-million using a radix of a half-million. For the emulated system size, these changes do not compromise the accuracy of the results. The capacity of the different levels of the memory hierarchy was chosen in such a way that the results obtained are significant for the selected problem sizes and for the dimensions of the global system.

The normalized execution times of the applications under study are represented in Figure 12. At first glance, it

can be seen that our routing proposal provides the system with a graceful performance degradation. The maximum degradation in presence of a high number of faults is in the order of 25%. Amazingly, the performance degradation executing the LU application with 16 faulty links is close to 10%. These results are consistent with the behavior observed under synthetic traffic conditions. Remember that although some degradation was observed on network throughput, base latencies remained constant. As it is known, parallel applications demand a high packet throughput only on certain phases of their execution. Hence, the increase on the parallel execution time can be proportionally inferior to the throughput degradation observed under synthetic traffic.

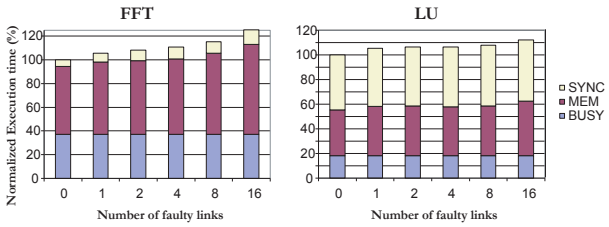

Figure 12. Performance degradation for FFT and LU using 64 processors.

5 Conclusions

A new fault tolerant routing for any kind of interconnection networks has been proposed in this paper. Our routing is based on the application of a controlled packet injection policy to a virtual deadlock-free network. Faults are tolerated by rearranging the shape and number of rings which compose the virtual escape network. The viability of our proposal has been demonstrated in realistic scenarios. Our technique is effective regardless the number of faults and their configuration. When operating in fault-free conditions, the network does not support any overhead. In the presence of a low number of faults, the use of our mechanism allows the network to achieve a sustained performance close to that observed under fault-free conditions. When the number of faults increases, our system exhibits a graceful performance degradation. Furthermore, even with a high number of links and/or node faults, the network remains operative.

In addition, the hardware cost of our technique can be considered negligible when compared with other current fault tolerant switching mechanisms. Although the results shown in this paper have been obtained for k-ary n-cube networks, the same technique can be applied to any other regular or irregular topology without additional cost.

References

[1] B. Bollobas, T. I. Fenner, and A. M. Frieze. An algorithm for finding hamiltonian paths and cycles in random graphs. *Combinatorica*, 7(4):327–341, 1987.

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

[3] S. Chalasani and R. Boppana. Fault-tolerant wormhole routing in tori. *Proceedings of 8th ACM Conference on Supercomputing*, July 1994.

[4] S. Chalasani and R. V. Boppana. Communication in multicomputers with nonconvex faults. *IEEE Trans. on Computers*, 46(5):616–622, 1997.

[5] J. Duato. A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks. *IEEE Trans. on Parallel and Distributed Systems*, 7(8):841–854, 1996.

[6] M. Galles. Spider: a high-speed network interconnect. *IEEE Micro*, 17(1):34–39, Jan-Feb 1997.

[7] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[8] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb. The Alpha 21364 network architecture. *IEEE Micro*, 22(1):26–35, Jan-Feb 2002.

[9] V. S. Pai, P. Ranganathan, and S. V. Adve. RSIM: An execution-driven simulator for ilp-based shared-memory multiprocessors and uniprocessors. *IEEE TCCA Newsletter*, October 1997.

[10] V. Puente, J. A. Gregorio, and R. Beivide. SICOSYS: An integrated framework for studying interconnection network in multiprocessor systems. *Proc. of the IEEE 10th Euromicro Workshop on Parallel and Distributed Processing*, 2002.

[11] V. Puente, J. A. Gregorio, R. Beivide, F. Vallejo, and A. Ibaez. A new routing mechanism for networks with irregular topology. *Supercomputing 2001*, November 2001.

[12] V. Puente, C. Izu, R. Beivide, J. A. Gregorio, F. Vallejo, and J. M. Prellezo. The adaptative bubble router. *Journal of Parallel and Distributed Computing*, 61(9):1180–1208, September 2001.

[13] S. L. Scott and G. M. Thorson. The Cray T3E network: Adaptive routing in high performance 3D torus. *Hot Interconnects IV*, 1996.

[14] J. Shih. Wormhole routing for torus networks with faults. *Parallel Computing*, 27:1817–1829, 2001.

[15] X. X. Lin, P. K. McKinley, and L. M. Ni. Deadlock-free multicast wormhole routing in 2D mesh multicomputers. *IEEE Trans. on Parallel and Distributed Systems*, 5(8):793–800, 1994.

[16] L. Ziang. Fault tolerant networks with small degree. *SPAA*, 2000.

[17] J. Zin, Z. Liu, and A. A. Chen. Compressionless routing: a framework for adaptive and fault-tolerant routing. *IEEE Trans. on Parallel and Distributed Systems*, 8(3):229–244, 1997.