A Survey and Evaluation of Topology-Agnostic Deterministic Routing Algorithms

José Flich, *Member*, *IEEE*, Tor Skeie, Andrés Mejía, Olav Lysne, *Member*, *IEEE*, Pedro López, *Member*, *IEEE Computer Society*, Antonio Robles, *Member*, *IEEE*, José Duato, Michihiro Koibuchi, *Member*, *IEEE*, Tomas Rokicki, and José Carlos Sancho

Abstract—Most standard cluster interconnect technologies are flexible with respect to network topology. This has spawned a substantial amount of research on topology-agnostic routing algorithms, which make no assumption about the network structure, thus providing the flexibility needed to route on irregular networks. Actually, such an irregularity should be often interpreted as minor modifications of some regular interconnection pattern, such as those induced by faults. In fact, topology-agnostic routing algorithms are also becoming increasingly useful for networks on chip (NoCs), where faults may make the preferred 2D mesh topology irregular. Existing topology-agnostic routing algorithms were developed for varying purposes, giving them different and not always comparable properties. Details are scattered among many papers, each with distinct conditions, making comparison difficult. This paper presents a comprehensive overview of the known topology-agnostic routing algorithms. We classify these algorithms by their most important properties, and evaluate them consistently. This provides significant insight into the algorithms and their appropriateness for different on- and off-chip environments.

Index Terms—Interconnection networks, routing algorithms, topology-agnostic routing.

1 Introduction

Clusters of PCs are often a cost-effective alternative to small- and medium-scale parallel computing systems. The performance of such clusters is closely related to the advances in their interconnection network. Currently, there are many proposals for high-performance clusters based on interconnects such as Myrinet [1], Servernet II [2], Gigabit Ethernet [3], InfiniBand [4], or Quadrics [5]. The size of these clusters has increased dramatically. The November 2010 list of top 500 supercomputer sites [6] includes several cluster-based machines with more than 100 K cores, some in the 10 top positions.

Cluster interconnects are usually arranged as switch-based networks whose topology is defined by the customer. The topology can be either regular or irregular. Regular topologies such as direct topologies and multistage networks are often used when performance is the primary concern. Many of the machines in the topmost positions of the top 500 supercomputers list use a 3D torus or a fat-tree topology. On the other hand, it makes no sense to use totally arbitrary irregular topologies in high-performance computing (HPC) systems. Although this was thought to be a possibility in the mid 1990s with the emergence of NOWs and COTS cluster systems, this never became a reality. However, a more common and realistic scenario is to consider regular patterns slightly modified by minor irregularities, such as those caused by the occurrence of failures. As the number of components increases, the probability of faults also increases. In such expensive HPC systems, it is often critical to keep the system running even in the presence of faults.

In order to tolerate faults, different mechanisms have been proposed. Unfortunately, most solutions either rely on specific logic (i.e., adaptive routing) not present in current cluster interconnects, or on disabling some regions of the network and otherwise working nodes. Other solutions rely on duplication of hardware, whereas others are based on dynamic reconfiguration of routing tables. For more details on these techniques, see [7]. Topology-agnostic routing algorithms provide a promising solution to tolerating faults by providing valid, deadlock-free routing no matter what combination of faults exists. The large body of research on routing algorithms for irregular networks may prove to be extremely useful not only for large clusters with faults, but also for on-chip networks.

The memory bandwidth requirements of massively multicore chips are a tremendous challenge to designers. A large shared multibank L2 cache on the chip can help reduce these requirements. To keep the L1 caches of the processors coherent, a high-speed, high-bandwidth on-chip

- *J. Flich, P. López, A. Robles, and J. Duato are with the DISCA, Universidat Politècnica de València, Camino de Vera, s/n, Valencia 46022, Spain. E-mail: {jflich, plopez, arobles, jduato}@disca.upv.es.*
- *T. Skeie and O. Lysne are with the Simula Research Laboratory and the University of Oslo, PO Box 134, Lysaker 1325, Norway. E-mail: {tskeie, olav.lysne}@simula.no.*
- *A. Mejía is with the Intel Corp, 2200 Mission College Blvd, SC12 3rd F2, Santa Clara, CA 95054-1549. E-mail: andres.mejia@intel.com.*
- *M. Koibuchi is with the National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan. E-mail: koibuchi@nii.ac.jp.*
- *T. Rokicki is with Instantis, 725 B Loma Verde, Palo Alto, CA 94303. E-mail: rokicki@instantis.com.*
- *J.C. Sancho is with the Barcelona Supercomputing Center, K2M Building, Office 107, C/Jordi Girona, 1-3, Barcelona 08034, Spain. E-mail: jose.sancho@bsc.es.*

Manuscript received 31 July 2010; revised 4 Jan. 2011; accepted 8 June 2011; published online 1 July 2011.
Recommended for acceptance by D. Wang.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2010-07-0457.
Digital Object Identifier no. 10.1109/TPDS.2011.190.

network is required; with many cores, a shared bus is no longer sufficient. Instead, direct networks such as the 2D mesh topology are being proposed for these networks on chip (NoCs). For some chips, such as those with heterogenous cores, a less regular topology may be used. In any case, manufacturing defects or runtime failures can convert the topology into an irregular one, requiring routing algorithms that can handle such topologies.

Therefore, NoC designers are faced with the problem of designing routing algorithms that deliver the lowest possible communication latency and use the internal bandwidth as efficiently as possible while being able to support irregular topologies, and even adapt to new topologies when failures occur. Topology-agnostic routing algorithms may provide a solution to this nontrivial problem.

This rising importance of topology-agnostic routing algorithms has led us to review the previous work in the field, collecting and comparing the different algorithms in the same context. This paper presents a taxonomy to characterize the algorithms. Although they appear very different, they have in common some fundamental characteristics (e.g., all of them guarantee deadlock freedom). We identify the main stages they have in common and present a method to describe all of them in the same framework, thus providing a global view.

These routing algorithms were developed for different technologies and each evaluated with different assumptions, network conditions, and traffic patterns, making it impossible to compare their performance. We have performed a comprehensive performance evaluation of all of these algorithms against the same set of topologies and workloads, making it possible to finally compare them all against each other.

Finally, we consider the relationship between current technology and the abstract routing algorithms. We describe the limitations and capabilities of each technology and discuss how this drives the selection of different routing algorithms.

The paper is organized as follows: Section 2 introduces some basic concepts regarding networks and routing algorithms. Section 3 describes the surveyed routing algorithms. Sections 4 and 5 present a taxonomy and a formal and unified description of all the routing algorithms. Section 6 gives an evaluation of all the routing algorithms. Section 7 discusses our performance results in light of present day technologies and scenarios. Finally, Section 8 summarizes our results.

2 Basic Concepts

Three issues dominate the design of an interconnection network: topology, switching technique, and routing algorithm [8]. The *switching technique* establishes how the network resources are allocated to packets and what to do while packets are waiting to acquire network resources. In lossless networks, packets waiting for acquiring a network resource are buffered. This buffering can be done either in units of packets, as in *store-and-forward* and *virtual cut-through*, or in smaller units of data commonly referred to as flits, as in *wormhole*. Unlike store-and-forward, virtual cut-through and wormhole allow transmission over each hop to be started as soon as the required resources are allocated without waiting for the entire packet to be received. Thus, in the absence of blocking, packets are effectively pipelined through the network. Several *virtual channels* multiplexed across the physical channel can be used to decouple buffer allocation from physical channel bandwidth. Each virtual channel is implemented with an independently managed buffer. This decoupling prevents a packet that is buffered from holding channel bandwidth idle.

An issue closely associated with switching technique is *flow control*. It is required in lossless networks to avoid losing packets/flits in transit when buffers become full. In general, flow control is a mechanism to report the availability of buffers at the far end of the channel and to allow the next packet/flit to be transmitted. When buffers become full, the flow-control mechanism provides the required backpressure to avoid packet/flit loss. If virtual channels are used, flow control is applied on a per-virtual channel basis. The flow-control mechanisms commonly used by current network technologies are *credit-based* (InfiniBand), *stop&go* (Myrinet), and *ack/nack*.

In order to efficiently route packets through a network, a routing algorithm must be used. Routing algorithms can be deterministic or adaptive. In *deterministic routing*, the path followed between a given source-destination pair is always the same. This is achieved by the switches providing only one routing option for a packet. With *adaptive routing*, several routing options may be provided by a switch to forward a packet. The selection of the routing option is usually made based on the current status of the links. Thus, with adaptive routing, packets can avoid congested areas in the network.

Routing algorithms can be implemented in several ways. Some routers use routing tables (also known as forwarding tables) with a number of entries equal to the number of destinations. These tables are associated with each network switch, and contain the next link a packet destined to the corresponding entry must follow. With a single entry per destination, only deterministic routing is allowed. The main advantage of table-based routing is that any topology and any routing algorithm can be used in the network. However, routing based on tables suffers from a lack of scalability, as the size of the table grows linearly with network size. An alternative is to place a specialized hardware at each node implementing a logic circuit to compute the output port to be used. This is the approach used in large parallel computers and there are some proposals in the context of irregular networks [9]. There are hybrid implementations, such as the FIR [10] or LBDR [11] approaches, which define the routing algorithm by programming a set of registers associated with each output port.

Another key issue in the design of routing algorithms is how to prevent deadlock and livelock. *Deadlock* occurs when no packet can advance toward its destination because the requested network resources (buffers/channels) are not released by the packets possessing them. In a deadlock situation, waiting packets are involved into a cycle of resource dependencies.[1] Deadlocks can be avoided by eliminating cycles in the resource dependency graph. This

1. There exists a dependence from i to j if a packet possessing the resource i requests the resource j.

can be achieved by imposing some restrictions on routing (e.g., imposing an ordering in the allocation of the resources [12] as in Dimension-Order Routing, DOR). Virtual channels are often used to allow this acyclic dependence graph [12]. However, some adaptive routing schemes allow cycles in their resource dependence graphs while still remaining deadlock free [13]. *Livelock* occurs when packets are allowed to advance, but they never reach their destination. Livelock can arise when nonminimal routing is allowed. It can be avoided by bounding the number of misroutings a packet is allowed, thus ensuring eventual packet progression.

The irregularity of the topology (induced by one or some faults or by the design of the topology) makes routing quite complicated. For instance, DOR has been proposed for meshes and tori networks. This routing algorithm forwards every packet through one dimension at a time, following an established order of dimensions. However, DOR is not able to route packets even in the presence of a single fault.

3 Surveyed Routing Algorithms

Many routing strategies have been proposed over the past decade for irregular networks, differing in goals and approaches. Some focus on obtaining minimal paths; others attempt to increase network bandwidth; yet others try to be quick to compute. Some exploit virtual channels, while others work on networks without this feature.

The first routing algorithm to appear was $\mathrm{up}^*/\mathrm{down}^*$ (UD) [14], and quickly gained popularity for its simplicity. The algorithm performs a breadth-first search (BFS) from a root node, assigning one direction of each link pair as *up* (toward the root) and the other direction as *down*. The uplinks form a directed tree toward the root, and the downlinks form a directed tree away from the root. Routes are chosen that traverse only a sequence of uplinks, followed by a sequence of downlinks; cyclic dependencies between links are avoided by forbidding any message to traverse a downlink followed by an uplink. Further refinements were proposed, such as the use of a depth-first spanning (DFS) tree [15], and the Flexible Routing scheme (FX) [39] which improved traffic balance by breaking cycles in each direction at different positions.

Another algorithm based on a BFS spanning tree, the left-up-first turn routing algorithm (LTURN) [16], uses a left to right directed graph, distributing the traffic around the root node of the spanning tree and achieving better network balance.

The Segment-based Routing (SR) algorithm [17] uses a divide-and-conquer approach, partitioning a topology into subnets, and subnets into segments, and placing bidirectional turn restrictions *locally* within each segment. This results in a larger degree of freedom when placing turn restrictions compared to the previous routing strategies that rely on heuristic rules.

Other routing algorithms, such as adaptive-trail [18], minimal adaptive [19], and smart-routing (SMART) [20], achieve performance improvements, but cannot be applied to every network technology, because they require extra functionality in the switches not usually present (for instance, distributed adaptive routing for adaptive-trail and minimal adaptive) or they have a very high computational cost (SMART).

Most of the above routing strategies (UD, DFS, LTURN, SMART) do not guarantee that all packets will be routed through minimal paths. This can lead to increased packet latency, especially for short packets. Use of nonminimal paths may also increase overall link utilization. The decision to use nonminimal paths is a key issue in selecting a routing algorithm. Some routing algorithms focus traffic on some network links, such as UD which forces a large percentage of the traffic to cross the root node. Such traffic imbalance can lead to rapid network congestion.

Several mechanisms have been proposed to achieve minimal routing while still avoiding deadlock. One mechanism is the use of In-Transit Buffers (ITB) [21], where packets are ejected from the network temporarily and stored in intermediate hosts when necessary to avoid deadlock. Other algorithms, such as Layered Shortest Path (LASH) [22] (a recent approach in [23]) use virtual channels to achieve deadlock-free minimal routing. The physical network is divided into a set of virtual networks (layers) using separate virtual channels; each layer is deadlock free and handles a subset of the required minimal paths.

The Multiple UD (MUD) [24], [25], [26] algorithm makes use of different virtual channels for multiple overlapping up/down trees, eliminating much of the traffic congestion around the root and using potentially fewer virtual channels, but not guaranteeing minimal paths.

The Transition Oriented Routing (TOR) [27] algorithm uses an underlying up/down tree, but permits otherwise "forbidden" transitions by transitioning from one virtual channel to the next. In order to guarantee minimal paths, a large number of virtual channels may be required. The Descending Layers (DL) algorithm [28] uses a similar approach, but the underlying base algorithm can be different (for instance, LTURN). In [29], the LASH and TOR algorithms are combined into LASH-TOR, using layers of differing base algorithms, with paths traversing the different algorithms and virtual channels. The enhanced flexibility of the different base algorithms among the virtual channel layers reduces the number of virtual channels required.

Each of these routing algorithms has been proposed for different network scenarios, relies on different mechanisms in the network, and focuses on achieving particular benefits. Some algorithms do not require virtual channels (UD, DFS, FX, SMART, LTURN, SR). Some of the rest require only a fixed number of virtual channels, while others require more and more virtual channels as the network size grows. Another distinction among them is how difficult they are to compute; some attempt to balance the predicted traffic, which can require excessive computation time for large networks, especially when the topology arises as the result of a fault and a new set of paths is required in short order. Technology limitations may also constrain the set of possible routing algorithms. Infiniband supports up to 15 virtual channels for routing purposes but real implementations may not implement virtual channels at all. The only common property among all of these routing algorithms is that they apply to any network topology.

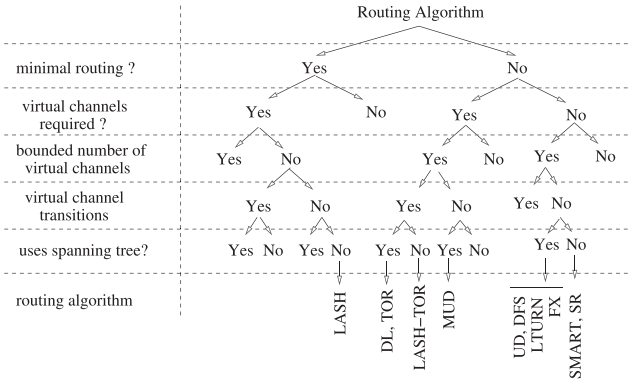


Fig. 1. Routing taxonomy.

Most of the routing algorithms are deterministic (e.g., FX, SR, LASH, and LASH-TOR). Some of them optionally permit distributed adaptive routing (e.g., UD, DFS, and MUD), whereas others are adaptive by design (e.g., adaptive-trail and minimal adaptive). However, current commercial network technologies do not support adaptive routing. This is mainly due to its higher implementation complexity (additional hardware support is often required) and its difficulties to guarantee in-order packet delivery (costly reorder buffers should be allocated at the NICs), which is required by many applications. Therefore, we focus only on those routing methods that are able to provide deterministic routing.

Recent work on routing of irregular topologies includes minor modifications of the algorithms we have presented above. For example, in [30], a variant of UD routing is presented, and in [31], a general methodology encompassing UD and LTURN is described. We have chosen to let UD and LTURN represent these developments here. Regarding efforts that specially target networks on chip, the developments most relevant to our work are the ACES approach [32] and the ASPRA approach [33]. These do, however, generate routing algorithms from the previous knowledge of the communication graph of the application that runs on the chip. Indeed, the topology is customized at design time, and thus, the best set of paths are computed based on the knowledge of the topology being designed. For this reason, they do not fit into the general framework of this paper.

4 A Routing Taxonomy

This section presents a taxonomy of all the deterministic topology-agnostic routing algorithms, shown in Fig. 1. The algorithms are separated by key objectives and required resources; first, by whether they guarantee minimal routing, then by their use of virtual channels, and finally by whether the paths are derived from a spanning tree or not.

The taxonomy first classifies the routing algorithms by whether or not they guarantee minimal routing. Only one of the routing algorithms we consider, LASH, guarantees minimal routing. While the others may well achieve minimal routing for many topologies, it is not guaranteed by design. LASH guarantees minimal routing only by requiring an unbounded number of virtual channels.

The next classification of the routing algorithms is by whether they need virtual channels or not. Notice that virtual channel multiplexing can be applied to every routing algorithm, but our distinction is made on whether virtual channels are a requirement or not. In this sense, UD, DFS, LTURN, FX, SR, and SMART do not require virtual channels, while DL, TOR, LASH-TOR, MUD, and LASH require at least two virtual channels. This distinction is significant as some network technologies like Myrinet do not support virtual channels. Notice as well that the use of certain number vc of virtual channels for routing purposes may reduce the QoS capabilities. However, it does not prevent QoS from being provided because it is orthogonal to routing. In this case, vc virtual channels should be dedicated to each service class provided.

The next distinction is whether the number of required virtual channels is bounded or not. Even those network technologies that support virtual channels may not provide very many virtual channels. Depending on network size and topology, an unbounded approach may require more virtual channels, and thus might not be suitable. Indeed, even technologies that provide many virtual channels, such as Infiniband, may not make them all available for just routing; virtual channels are often reserved for providing QoS, yielding only two or three effective virtual channels for routing. Thus, MUD, DL, TOR, and LASH-TOR can always be applied to a network with a limited set of virtual channels (with the trade-off that they do not always guarantee minimal paths, and that more virtual channels may provide increased performance). LASH's guarantee of minimal routing comes at a cost of no bound on the number of virtual channels that may be required.

The next characteristic used by the taxonomy is transitions between virtual channels on a path. If all paths generated by a routing algorithm lie within the same virtual channel, this permits the use of more network technologies. For instance, in InfiniBand, the selection logic of virtual channels does not take into account the virtual channel used at the previous switch; it only takes into account a packet field fixed throughout the path (the service level field). Thus, routing algorithms that require transitions between virtual channels may generate routings incompatible with Infiniband. The DL, TOR, and LASH-TOR algorithms require the use of virtual channel transitions.

The final property used by the taxonomy is the use of a spanning tree. UD, DFS, FX, LTURN, and MUD use a spanning tree in order to compute their paths, permitting very fast computation of routings. DL and TOR also use a spanning tree, but only to determine where virtual channel transitions must occur. On the other hand, SR, LASH-TOR, and LASH do not use any spanning tree at all.

5 A Framework for Generic Routing

This section unifies the routing algorithms within a framework suitable for describing generic routing algorithms. The purpose of this framework is to provide a context with respect to how to derive the final set of paths from the initial set of all network paths. This framework is outlined as a sequence of functional and algorithm stages, as depicted in Fig. 2. All the topology-agnostic routing methodologies can be described based on how they approach the functional stages shown. Moreover, the framework puts the routing

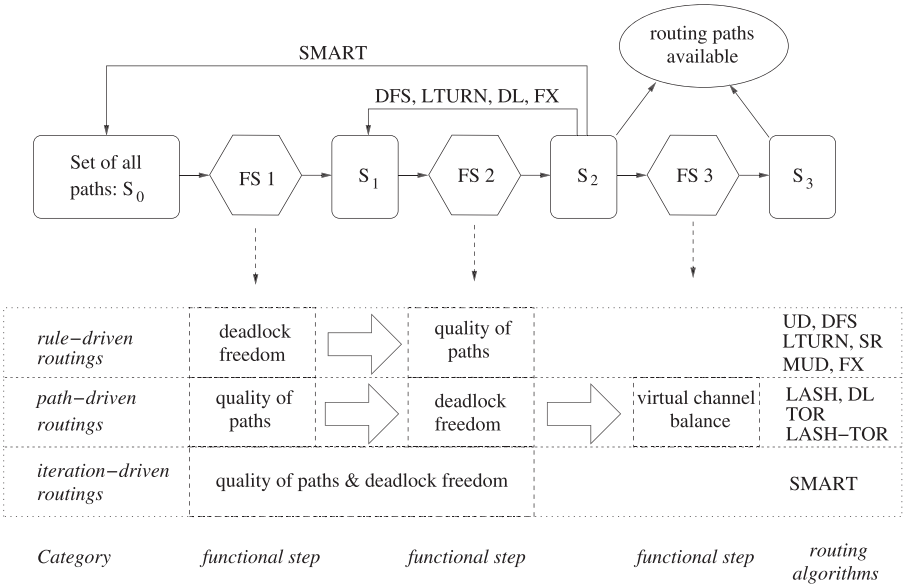


Fig. 2. A framework for deriving (describing) generic routing functionality.

taxonomy into perspective by reflecting the various design options available when developing routing algorithms.

Fig. 2 consists of two parts. The upper part defines three functional steps (FS_i). Each step takes a set of paths and produces a new (smaller) set, for instance, selecting a single deterministic path among several candidate paths according to some quality criterion, or providing deadlock freedom by selecting a set of paths without cyclic dependencies. Note that the initial set is the set of all possible paths, both minimal and nonminimal, between each source-destination pair.

The lower part of the figure sketches three alternative ways of implementing the functional steps (*rule-driven*, *path-driven*, and *iteration-driven*), defining three different categories of routing algorithms. Within each category, each of the functional steps may be approached in various ways. All of the analyzed algorithms fit into this framework, as listed in Fig. 2.

Guaranteeing deadlock freedom and obtaining a good traffic balance are the most important issues of generic routing algorithms. These tasks can be managed at different stages depending on whether virtual channels are available or not and so forth. Fig. 2 illustrates how the deadlock problem can be handled either in the first stage (with rule-driven routings) or in the second stage (with path-driven routings).

Though the framework is laid out as three main sequential stages, some routing algorithms may integrate two or more functional steps, or even repeat some of the stages until a certain condition is met. This is, for example, the case for SMART routing [20]. This type of algorithm is referred to as iteration-driven routing. Below, we will explore each functional step and introduce possible approaches to their objective relative to the routing algorithms surveyed in this paper.

5.1 Rule-Driven Algorithms

The first class of algorithms in the framework encourages deadlock freedom in the first functional stage by imposing routing rules in such a way that cyclic dependencies cannot be formed. These *rule-driven algorithms* are UD, DFS, LTURN, SR, MUD, and FX.

5.1.1 Providing Deadlock Freedom in Rule-Driven Algorithms

Enforcing deadlock freedom through a simple rule on the allowed paths is a compelling approach. These rules are frequently based on a spanning tree of the network. Take, for instance, $\mathrm{up}^*/\mathrm{down}^*$, first presented in connection with Autonet [14] and later used in Myrinet [1]. After generating a spanning tree, up directions are assigned to all links such that any path following only links in the up direction ends up at the root of the spanning tree. Routing is restricted so that no packet can traverse a link in the up direction after having traversed a link in the down direction; it is easy to see that this prevents the occurrence of cyclic dependencies. This technique has drawbacks; however, the root tends to become a congested area and frequently nonminimal paths must be used.

Several improvements to UD have been proposed. One approach is to permit more turns (link-to-link transitions), selecting carefully to avoid introducing cyclic dependencies, as proposed by Koibuchi et al. [34]. Their method, called L-turn routing (LTURN), relies on building a left-right tree (directed graph), which extends the conventional up-down tree by assigning directions to horizontal links (links between nodes at the same level in the breadth-first search). A second approach, proposed by Sancho and Robles, computes the spanning tree based on a depth-first search (called DFS), instead of breath-first search, yielding fewer down-up restrictions [35]. By carefully selecting the order of children to explore in the search, through the use of some heuristics, more minimal paths are made available. In addition, the algorithm considers all possible root nodes and selects the best one based on behavioral metrics in order to improve traffic balance. Fig. 3 shows the graphs obtained by LTURN and DFS for a 2D mesh topology whose nodes are consecutively numbered by rows (left to right and top to bottom).

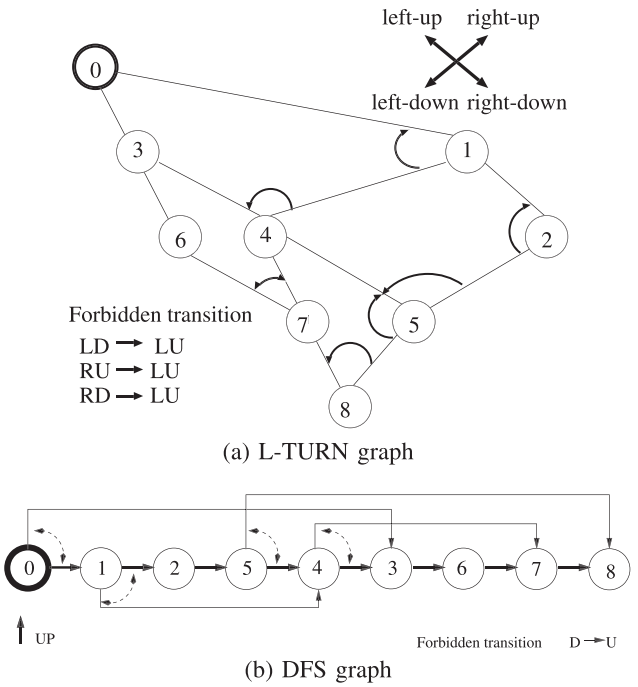


Fig. 3. Graphs obtained by L-TURN and DFS for a 3×3 mesh.

Segment-Based Routing algorithm [17] is another routing algorithm that eliminates deadlocks as its first step. The idea behind SR is to form a spanning tree by dividing the topology into subnets and subnets into segments. It finds a first segment that forms a complete cycle, then it continues looking for new segments or extensions to that cycle until it covers the entire topology. Within each segment, a bidirectional (or even two unidirectional) routing restriction is introduced. The result is to break each cycle independently while maintaining connectivity among the nodes. SR allows great flexibility when placing routing restrictions and increments fault tolerance capabilities for the topology, as each segment is independent.

A significant difficulty with UD-based routings is that breaking cycles at the same node in both directions can push traffic away from that node and therefore onto other nodes. This problem tends to occur on regular topologies, where UD-based routings introduce an asymmetry leading to traffic imbalance. The FX routing algorithm [39] attempts to address this problem by formulating two rules for breaking cyclic dependencies, one for each direction, on every four-switch cycle in the network. A third rule is also devised removing even more dependencies, in order to decrease the maximum number of paths crossing each channel.

Virtual channels permit the traffic imbalance and nonminimal path problems of spanning-tree-based routings to be attacked further. Simply put, each virtual channel can be assigned a distinct underlying UD-based routing, with separate root nodes, and then each source-destination pair can be assigned to virtual layers to obtain the shortest path. This approach, which we refer to as Multiple UD, was proposed in a number of papers [24], [25], [26].

5.1.2 Selecting Paths in Rule-Driven Algorithms

Once the rules are in place to ensure deadlock freedom, the second functional step must select a single best path between each source-destination pair to obtain a deterministic routing with good traffic balance. Different polices can be adopted for selecting a single best path. In this section, we will discuss the selection strategies used by the rule-driven algorithms. These same selection policies can be applied to many routing algorithms, and have shown to be effective [34].

The simplest path selection approach simply uses random or round-robin selections. These approaches have been used with classical UD [14] and LTURN routings. However, random and round-robin selection may not achieve sufficient traffic balance. The DFS algorithm advocates a more sophisticated traffic balancing algorithm [35] by repetitively removing routing paths from the link having the largest number of crossing paths, so long as the affected source-destination pairs have alternative paths. This procedure terminates when the number of paths between each pair of nodes is reduced to 1.

A similar approach was proposed in [36]. All the possible minimal paths are computed for every source-destination pair, after which only a single path is selected for every pair of nodes, attempting to minimize the deviation in the number of paths crossing each link (we will refer to it as link weight).

Traffic balance comes in two flavors for the MUD algorithms [25], [24], [26]. First, the selection of multiple roots should be chosen carefully to alleviate the hot-spot problem of a single root. In [25], [24], the authors choose the root of a new layer in such a way that it maximizes the minimal distance between the new root and any of the existing roots. Second, since each of the separate UD structures in MUD may offer several candidate paths between each node pair, a single best deterministic path must be selected; any of the traffic balancing approaches discussed so far can be used.

5.2 Path-Driven Algorithms

New interconnect technologies, such as InfiniBand, provide a certain number of virtual channels. This has led to new routing algorithms that are able to guarantee minimal-path routing while requiring a modest number of virtual channels. These methods fall into the category called *path-driven* algorithms, which we describe below.

5.2.1 Providing Paths in Path-Driven Algorithms

Because deadlock freedom can be guaranteed more easily with virtual channels than without, the first step of path-driven algorithms is to select a set of paths with minimal length and good traffic balance. This set may be composed of a single path for each source-destination pair, or multiple paths, depending on the particular algorithm.

The TOR routing algorithm, as proposed by Sancho et al., guarantees minimal routing requiring only a modest number of virtual channels [27]. This algorithm requires all minimal paths available when executing the second functional step, which we will explain below, so the first functional step for TOR simply computes all possible minimal paths for the next step. A similar approach is taken by Koibuchi et al. in [28] in a method referred to as DL. Unlike TOR, the underlying routing algorithm used by DL can be different from UD (e.g., LTURN).

The LASH [22] and LASH-TOR [29] algorithms take a different approach, forwarding only a single path for each source-destination pair to the succeeding deadlock freedom step. Since just one single path will be passed to the next step, the quality of this path has to be assessed; traffic balance should be considered at this point. LASH and LASH-TOR manage this issue by first computing all the minimal paths, similar to the TOR algorithm, and then selecting among the candidates. The versions of LASH and LASH-TOR evaluated in this paper deploy random selection. However, other selection policies can of course be applied.

5.2.2 Avoiding Deadlocks in Path-Driven Algorithms

The second functional step in path-driven algorithms focuses on obtaining a deadlock-free set of paths through the use of virtual channels. The algorithms use a variety of techniques to accomplish this.

The TOR algorithm relies on an UD spanning tree of the network and a set of virtual channels to break cyclic path dependencies. Wherever one of the input paths would require a down to up transition in the UD spanning tree, TOR introduces a virtual channel transition to a higher numbered virtual layer (where the virtual layers are numbered from 1 to n). Every packet inserted into the network will start in layer 1, and it will move to the next virtual layer every time a forbidden transition is encountered. Notice that the path crossing the largest number of forbidden transitions will determine the number of virtual layers required to guarantee minimal routing.

The TOR algorithm considers all minimal paths between each source-destination pair, selecting the one with the fewest forbidden transitions, in order to minimize the number of virtual layers required. In practice, the number of virtual layers is modest [27]. Notice that after this selection process, some pairs of nodes may still have more than one single path (all of them being deadlock free). To generate a deterministic, balanced routing, TOR adopts the generic DFS traffic balancing algorithm to select among these paths.

The DL algorithm is similar to TOR, changing forbidden transitions into virtual layer transitions. DL uses a bounded number of virtual channels, however, by following the original underlying routing restrictions in the last virtual layer.

LASH takes a different approach for guaranteeing deadlock freedom [22], [23]. The idea behind LASH is that each virtual layer in the network has a set of source-destination pairs assigned to it, in such a way that all source-destination pairs are assigned to exactly one virtual layer. Thus, all packets assigned to a single virtual layer stay in that virtual layer until they reach their destination. LASH assigns paths to virtual layers one by one, always assigning a path to the first virtual layer it can be added to such that no cyclic dependencies would be introduced. This may require the introduction of another virtual layer. Since no path can transit virtual layers, and since each virtual layer is deadlock free, the entire routing is deadlock free. The need for virtual layers in order to guarantee minimal routing follows a logarithmic curve as the size of the network grows [22]. However, LASH demands more virtual layers than the TOR algorithm to ensure minimal routing.

The LASH-TOR methodology is an extension of LASH that reduces the number of required virtual layers by allowing transitions between the layers. It is a hybrid between the LASH and TOR algorithms [29]. Like LASH, LASH-TOR assigns source-destination pairs (paths) onto virtual layers, by ensuring that the path under assignment does not introduce cycles. However, as in TOR, a path can be split into several subpaths, each assigned to different virtual layers. A path will be split at the point (switch) where the next dependency associated with this path would introduce a cycle in the dependency graph of the current virtual layer. A transition will then be made to the next virtual layer, where the path continues until it is completely assigned or else has to make a transition to the following virtual layer, and so forth.

LASH-TOR requires significantly fewer virtual layers than LASH as the network size grows, and is comparable to the TOR methodology [29]. On the other hand, unlike LASH, LASH-TOR introduces transitions between layers that can serve to spread congestion from one virtual layer into another. Both LASH and LASH-TOR conduct physical traffic balance in the first functional step, while TOR performs traffic balance last.

5.2.3 Balancing Virtual Channels in Path-Driven Algorithms

Virtual channels are implemented with a set of buffers that are assigned to each virtual channel. It is important to balance the use of these resources much like physical traffic balancing balances the link utilization. Each of the path-driven algorithms described assigns virtual layers greedily, filling the first virtual layer with as many paths or subpaths as possible, and only using subsequent layers as necessary. This virtual layer imbalance must be corrected with a third functional step.

For the TOR and LASH-TOR algorithms, most of the paths assigned to the first virtual layer are complete paths. One example 32-switch network assigns about 75 percent of the total paths the first layer [27]. The TOR algorithm addresses this imbalance by moving completely mapped first-layer paths to the other virtual layers until a balancing criterion is met. Since those candidate paths do not introduce any forbidden down-up turns (since each virtual layer uses the same UD tree), deadlock freedom continues to be guaranteed. LASH-TOR approaches the virtual balancing task similarly.

A slightly different selection policy can also be used. Instead of selecting the virtual channel with the largest number of paths in the procedure above, and removing paths from this channel, one alternative is to pick the virtual channel having the smallest number of crossing paths and let those routing paths become fixed [34], removing alternative paths for their corresponding source-destination pairs. The motivation behind this strategy is to avoid virtual channels with low utilization. However, this policy has shown to be less effective than DFS traffic balancing, which focuses on avoiding having heavily congested links because, as shown in [35], they noticeably penalize network throughput.

5.3 Iteration-Driven Algorithms

The *iteration-driven* algorithms integrate the traffic balancing and deadlock avoidance functional steps in an attempt to obtain good performance characteristics (Fig. 2), unlike the

TABLE 1
Computing Complexity of the Routing Algorithms

Routing alg.	Traffic balancing	Complexity	Routing alg.	Traffic balancing	Complexity
LASH	Random	$O(n^3)$	DL	Optimized	$O(n^3)$
TOR	Optimized	$O(n^3)$	SR	Optimized	$O(n^3)$
LASH-TOR	Random	$O(n^3)$	MUD	Random	$O(n^2)$
UD	Random	$O(n^2)$	FX	Optimized	$O(n^3)$
DFS	Optimized	$O(n^3)$	SMART	Optimized	$< O(n^9), typ. O(n^4)$
LTURN	Optimized	$O(n^3)$			

modular approach used in the previously discussed algorithms. The algorithm falling into the *iteration-driven* category is SMART [20] routing. This routing algorithm does not require the use of virtual channels.

The idea behind SMART routing is to break cyclic dependencies at strategic points (that is, performing *cuts* in the channel dependency graph). For each cycle in the dependency graph, the SMART algorithm will break the dependency that minimizes the average path length of the topology. The procedure terminates when the channel dependency graph has no cycles. Each time a cycle is found, all possible cuts are considered and the average path length after that cut is computed. Because of the removal of channel dependencies, one cannot calculate minimal paths directly, since the existence of a path from i to j and another from j to k does not imply a path from i to k via j. Therefore, SMART routing must use breadth-first search from every source to all destinations in order to compute this path length.

This procedure is not guaranteed to yield a deadlock-free routing function because one may have carried out poor selection of previous cuts, such that one later possibly encounters a cycle where breaking any dependency will cause the routing function to be disconnected. In such cases, SMART must be restarted in a tree mode by which a spanning tree structure of the topology is computed (seen as a "backbone" network), where the connecting links in the tree are marked as essential and considered nonbreakable. The potential restart of SMART routing is also captured by Fig. 2, shown as the backward arrow from the functional stage FS_2 to the initial stage. SMART routing is rather complex, where the runtime is bounded by n^9 (n is the number of switches). Typically, the runtime is bounded by n^4 [20].

5.4 Computing Complexity of the Routing Algorithms

All routing algorithms have at least a $O(n^2)$ complexity, n being the number of switches in the network, as they have to compute all the paths for every source-destination pair. Path-driven routing algorithms have to search for cycles, increasing the computational complexity by an additional n. Depending on the strategy used to balance traffic, computing complexity may increase even more. If random path selection is used, their complexity remains the same ($O(n^2)$ or $O(n^3)$). However, with more sophisticated load-balancing schemes, the computing complexity may strongly increase. For instance, DFS balances traffic by computing the spanning tree considering all the network nodes as possible roots of the tree, selecting as the definitive root the one that gives the lowest average number of paths per link. Therefore, DFS increases complexity by another factor of n. More complex schemes, as the heuristic used in SMART, may strongly increase the computing complexity. Table 1 shows the computing complexity of each of the routing algorithms analyzed in this paper, as stated in the original papers describing them.

6 Performance Evaluation

This section presents a performance evaluation of all the routing algorithms using the same scenarios and evaluation environment. We present both an analysis of path lengths and traffic balance, and a simulation-driven empirical evaluation.

6.1 Simulation Environment

Our simulator models switch-based networks with point-to-point links. Each switch uses virtual cut-through switching [37] and credit-based flow control. A full crossbar configuration and a round-robin arbitration scheme are used. The crossbar arbiter has one queue per output port, and buffers of 1 KB at both the input and output sides. It is based on *FIFO* request queues, and it is able to transmit one byte per connection, per cycle. Routing decisions are made at every switch by accessing the local routing table. This table maps destinations to output ports, with one mapping for each possible destination. We use a constant packet size of 32 bytes, and a credit size of 32 bytes. The routing time at each switch is assumed to be 100 ns (taken from Myrinet). This includes the time to access the routing table, the crossbar arbitration time, and the time to set up the crossbar connections.

We also model the fly time, which depends on the link length and the propagation delay of the cable. We model 12 m in length copper cables with a propagation delay of 5 ns/m, leading to 60 ns of fly time.[2] Link bandwidth is assumed to be 1 Gbps. The clock cycle is adjusted to the time required to transfer one byte (10 ns at 1 Gbps with 8/10 coding).

We have evaluated 2D meshes and tori networks of sizes 4×4, 8×4, 8×8, and 16×8. Each switch has attached one processing node. Additionally, we have modeled 240 different topologies derived from torus and meshes with 1, 3, and 5 percent of randomly injected link failures. Notice that the simulated 2D topologies require the use of low-radix switches. The availability of high-radix switches would allow to attach more processing nodes to each switch, thus

2. Fly time was obtained from the maximum cable length in an example system composed of 128 switches located in four racks, obtaining that using a 12 m Myrinet cable length was enough to comfortably build the network.

stressing the network more. It also allows meshes or tori with a higher dimensionality, but these topologies are not used in practice. Alternatively, high-radix switches allow increases in network connectivity (for instance, by using express channels [38]), therefore decreasing the average routing distance (ARD) and reducing the differences among the routing algorithms analyzed (see Section 6.4). We have not evaluated fat trees as they have many alternative paths by design. Therefore, it is somewhat easy to deal with faults in fat trees without the need of using a topology-agnostic routing. Moreover, the application of some of the analyzed routing algorithms, in particular those based on a spanning tree, would be equivalent to the up/down routing commonly used in fat trees. As a consequence, it is expected that their behaviors are very similar.

For each simulation run, we assume the same constant packet rate for each end node. We evaluate the full range of loads, from low load to saturation, for uniform, bit-reversal, and hot-spot traffic patterns. Each simulation is run until a given number of messages (500,000) have been delivered to the end nodes, ignoring the first 50,000 received ones (warm-up period). On the other hand, we account for the variability in simulation results due to the simulation tool by executing the same simulation several times for each network configuration using different random seeds and calculating the 95 percent confidence interval for the results, which are not shown in the plots because the size of error bars was negligible. Regarding the variability due to the network irregularity (induced through fault injection), we have simulated, for each number of faults injected, 10 random fault configurations.

The evaluation of the routing algorithms will be performed using the path balancing mechanism originally suggested by each routing scheme. We will not separately evaluate the influence of the additional path selection mechanisms, but the routing algorithms as a whole. However, there are some cases (DFS, SR, UD, LTURN, DL, and FX, for networks with 128 nodes) where the paths have been calculated using random path selection. In these cases, we will label those routings with an asterisk and in bold face.

6.2 Average Routing Distance and Link Weight Analysis

This section presents an analysis of all the topology-agnostic routing methods considered in this paper. In particular, we are interested in two main characteristics of each routing. The first one is their ability to achieve shortest path for all (or most) of the source-destination pairs. For this, we will collect the average routing distance achieved by each routing algorithm. ARD is computed as the sum of all path lengths (measured as the number of visited switches) divided by the number of paths.[3] The second issue we are interested on is their ability to achieve a good set of paths in terms of traffic balance. For this, we will analyze the *link weight* obtained by each routing algorithm. The weight of a link is defined as the number of paths that cross that link. In this study, we have differentiated each direction of the link. In order to get representative numbers, we obtain the average (*Avg*) link weight and its standard deviation (*St.D*) for each routing algorithm.

TABLE 2
Average Routing Distance (Routing Algorithms with No Virtual Channel Requirements)

	Mesh topologies						
Size	DOR	UD	DFS	LTURN	SMART	FX	SR
4×4	3.50	3.50	3.50	3.50	3.50	3.50	3.50
8×4	4.88	4.88	4.88	4.88	4.88	4.88	4.88
8×8	6.25	6.25	6.25	6.27	6.25	6.25	6.25
16×8	8.94	8.94	**8.94***	**8.95***	8.94	8.94	**8.94***

	Torus topologies						
Size	DOR	UD	DFS	LTURN	SMART	FX	SR
4×4	3.00	3.00	3.00	3.00	3.00	3.00	3.13
8×4	4.00	4.25	4.43	4.14	4.12	4.25	4.36
8×8	5.00	5.50	5.75	5.32	5.26	5.50	5.36
16×8	7.00	8.13	**7.89***	**7.47***	7.44	7.01	**8.06***

DOR routing uses two VCs in tori to avoid cycles.

We will group the routing algorithms in terms of whether virtual channels are required or not. In a first set, we group the routing algorithms that do not use any virtual channel (UD, DFS, LTURN, SMART, FX, and SR), and in the second set, we group the routing algorithms that require virtual channels (MUD, DL-UD, DL-LTURN,[4] LASH, LASH-TOR, and TOR). In this case, we will bound the number of virtual channels to two and three. For comparison purposes, when analyzing regular networks without faults, we will also show results for the DOR routing. When evaluating DOR in mesh networks, we will use one and two virtual channels. For torus networks, two virtual channels will be used to avoid cycles along rings.

Table 2 shows the average routing distance for the algorithms requiring no virtual channels in the different topologies analyzed. For meshes, ARD for all algorithms is the same and the minimum achievable, except for the LTURN, where ARD is slightly higher for the largest topology. However, for torus networks, despite the fact that the achieved ARD values are lower than that achieved in meshes thanks to the use of wrap-around links, the ARD values achieved by most of the algorithms are higher than the minimum achievable (indicated by DOR routing). As can be observed, all the algorithms increase the average path length as the network becomes larger. In fact, none of the routing algorithms is able to achieve shortest paths for the 8×4 torus and upward. However, one of the best routing methods in ARD terms is the FX routing, with a very small increase in the ARD value. Notice that DOR routing achieves the minimum ARD as it uses two virtual channels to avoid cycles along each ring. However, the rest of algorithms do not use virtual channels at all; thus, in order to avoid deadlocks, some paths must be nonminimal.

Table 3 shows the average and standard deviation of link weights for each routing algorithm with no virtual channel requirements For meshes, as shown, link weights are not so similar. Although UD achieves the highest standard deviation, in some cases, DFS and LTURN algorithms attain the highest number of paths crossing a link (max values not shown but reflected in high St.D.). For UD and LTURN concerning the 16×8 mesh, the maximum link weight is high. In the case of DFS, LTURN, and SR, the original traffic balancing algorithm could not be used due to

3. In a system with N nodes, there are N^2 paths. For paths with the same source and destination, the path length is 1.

4. DL-LTURN is the DL routing when using LTURN as the underlying algorithm to set the virtual channel transitions. The same applies to DL-UD.

TABLE 3
Analytical Results (Routing Algorithms with No Virtual Channel Requirements)

	4×4 mesh		8×4 mesh		8×8 mesh		16×8 mesh		4×4 torus		8×4 torus		8×8 torus		16×8 torus	
algorithm	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.
DOR	13.33	1,91	38.15	15.01	96.00	27.77	280.28	133.79	8.0	2.85	24.00	9.63	64.00	9.82	192.00	66.88
UD	13.33	4,80	38.15	20.09	96.00	43.40	361.21	321.10	8.0	3.96	26.00	19.30	72.00	50.35	228.00	201.89
DFS	13.33	3,57	38.15	18.55	96.00	42.58	**280.28**	**155.18**	8.0	2.51	27.44	14.80	76.00	33.53	220.51	145.36
LTURN	13.33	2,12	38.15	15.40	96.00	34.50	**281.23**	**227.21**	8.0	1.70	25.12	11.60	69.06	25.72	206.91	107.05
SMART	13.33	2,81	38.15	18.37	96.00	34.41	280.28	151.89	8.0	0.00	24.97	9.38	68.20	14.45	206.15	100.63
FX	13.33	1,91	38.15	15.01	96.00	27.77	280.28	133.79	8.0	0.00	26.00	14.13	72.00	28.06	192.46	70.41
SR	13.33	3,14	38.15	15.57	96.00	29.75	**280.28**	**143.25**	8.5	3.01	26.88	13.85	69.78	24.45	225.94	133.94

Mesh and torus networks.

its extremely large computation time; thus, a random path selection was performed. Although the random path selection is also performed in other routing algorithms (i.e., LASH, LASH-TOR, and TOR), in the case of UD and LTURN, its impact is quite severe. The reason is that the bad set of paths UD and LTURN are using regardless of the path selection algorithm, because they are based on a spanning tree. On the other hand, the minimum standard deviation is achieved by DOR and FX. In fact, they present the same values as the use of unidirectional restrictions allows the FX algorithm to achieve the same set of paths that DOR uses in regular meshes [39]. That is, the FX algorithm resembles the DOR algorithm for mesh topologies. However, for most cases, SR, DFS, SMART, and FX present roughly the same results, thus achieving a good traffic balance in mesh networks.

For torus networks, we also observe here the DOR mentioned capability to obtain the minimum ARD (not shown but reflected in St.D.) and minimum standard deviation in nearly all cases (remember DOR is using two virtual channels to avoid cycles). However, SR, SMART, LTURN, DFS, and FX continue to exhibit on average an acceptable behavior, while UD achieves the worst figures in all cases due to its well known tendency to concentrate traffic around the root of the tree. It is also interesting to note that SMART and FX obtain perfect and desired traffic balancing for the torus 4×4 topology.

Now, let us turn our attention to the routing algorithms requiring several virtual channels. Table 5 shows the results for routing algorithms using two virtual channels (also for three VCs for LASH in torus networks). For fault-free meshes, all the algorithms achieve the minimum possible ARD. This was expected due to the use of virtual channels and their ability to obtain shortest paths for every source-destination pair. Also, all the routing algorithms achieve the minimum ARD for a 4×4 torus. However, for larger torus networks, MUD and LASH routings experience higher ARD values. This is due to the fact that both algorithms require more than two virtual channels to achieve minimal paths for every source-destination pair. In the case of MUD, the computation of two different $\mathrm{up}^*/\mathrm{down}^*$ trees is not enough to obtain a minimal path for every pair of nodes. For LASH, instead, by using a bounded number of separate virtual layers, some minimal paths could lead to deadlock, thus requiring a separate layer implementing $\mathrm{up}^*/\mathrm{down}^*$ paths (thus, some paths being nonminimal). For LASH with three VCs (shown in table), the ARD is decreased. For 8×4 torus, the ARD achieved with three VCs is the minimum. However, as network size increases, even with three VCs, LASH does not guarantee minimal routing. Also, it is interesting to note that TOR, LASH-TOR, and DL (either using UD or LTURN as underlying routing algorithms) algorithms are able to get minimal paths for every source-destination pair, even when using only two virtual channels. Remember that these algorithms allow the transition of virtual channels at the packet level. On the contrary, neither LASH nor MUD allows virtual channel transitions.

Table 4 shows that TOR, LASH-TOR, and DL achieve the best traffic balance with just two VCs. Moreover, the use of a third VC does not contribute to improve the traffic balance. Note that, except in DL, this is achieved regardless of the traffic balancing algorithm applied. In the case of DL, if a random path selection is used in DL-UD*; and DL-LTURN*, instead of the original traffic balancing scheme, a great impact on traffic balance is observed. As it can be seen in the table, DL-UD* and DL-LTURN* have some links crossed by a very large number of paths, increasing the standard deviation of the link weight. As it can be also observed,

TABLE 4
Analytical Results (Routing Algorithms with Virtual Channel Requirements)

	4×4 mesh		8×4 mesh		8×8 mesh		16×8 mesh		4×4 torus		8×4 torus		8×8 torus		16×8 torus	
algorithm	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.	Avg	St.D.
MUD	13.3	3.10	38.1	16.51	96	32.82	280.3	183.40	8	2.46	24.0	10.37	64.6	23.07	194.4	104.63
MUD 3VC	13.3	4.15	38.1	18.05	96	45.69	N/A	N/A	8	2.27	24.0	9.41	64.3	15.86	N/A	N/A
LASH 2VC	13.3	4.00	38.1	17.99	96	40.40	280.3	160.00	8	2.68	25.0	15.22	68.3	45.48	213.0	203.06
LASH 3VC	13.3	4.00	38.1	17.49	96	40.78	280.3	161.05	8	2.68	24.2	9.42	66.5	32.00	207.9	180.33
TOR	13.3	3.45	38.1	16.34	96	33.10	280.3	153.43	8	2.42	24.0	9.61	64.0	12.94	192.0	78.29
TOR 3VC	13.3	4.05	38.1	17.37	96	44.13	280.3	143.43	8	2.46	24.0	10.12	64.0	17.52	192.0	78.63
LASH-TOR 2VC	13.3	4.45	38.1	19.39	96	39.87	280.3	151.18	8	2.34	24.0	10.29	64.0	17.00	192.0	78.22
LASH-TOR 3VC	13.3	4.45	38.1	19.39	96	39.87	280.3	151.18	8	2.34	24.0	10.29	64.0	17.00	192.0	78.94
DL_LT	13.3	2.31	38.1	15.43	96	30.23	N/A	N/A	8	1.13	24.0	8.81	64.0	5.72	N/A	N/A
DL_UD	13.3	2.50	38.1	15.78	96	30.88	N/A	N/A	8	1.02	24.0	9.14	64.0	6.18	N/A	N/A
*DL_LT	13.3	4.49	38.1	19.51	96	60.32	**280.3**	**183.25**	8	2.51	24.0	10.82	64.0	24.46	**192.1**	**96.91**
*DL_UD	13.3	6.25	38.1	19.63	96	61.10	**280.3**	**183.40**	8	3.24	24.0	11.50	64.0	26.34	**192.0**	**103.96**

Two/three VCs per physical link. Mesh and torus networks.

TABLE 5
Average Routing Distance (Routing Algorithms with Virtual Channel Requirements)

	Mesh topologies					
Top.	MUD	LASH	TOR	LASH-TOR	DL-LT	DL-UD
4×4	3.50	3.50	3.50	3.50	3.50	3.50
8×4	4.88	4.88	4.88	4.88	4.88	4.88
8×8	6.25	6.25	6.25	6.25	6.25	6.25
16×8	8.94	8.94	8.94	8.94	**8.94***	**8.94***

	Torus topologies						
Top.	MUD	LASH	LASH 3 VC	TOR	LASH-TOR	DL-LT	DL-UD
4×4	3.00	3.00	3.00	3.00	3.00	3.00	3.00
8×4	4.00	4.12	4.00	4.00	4.00	4.00	4.00
8×8	5.04	5.27	5.16	5.00	5.00	5.00	5.00
16×8	7.08	7.68	7.50	7.00	7.00	**7.00***	**7.00***

Two VCs per physical channel.

LASH routing achieves the highest standard deviation in almost all the torus networks. This is due to the random path selection policies used and, in some degree, to the need of an additional virtual channel. This fact can be observed for torus networks. With three VCs, maximum link weight is decreased significantly (not shown but reflected in St.D.) for 8×4 and 8×8 torus networks. However, for a 16×8 torus network, a fourth virtual channel would be required. Finally, for MUD, TOR, and LASH-TOR, the ARD value is low regardless of the network topology, size, and number of virtual channels. To be added here is that LASH-TOR (in this evaluation) deploys a random path selection, contrary to the other analyzed methods, which use more sophisticated algorithms for path selection.

As a summary of the analytical evaluation, we can conclude the following. First, using virtual channels is necessary to achieve minimal routing for every pair of nodes in torus networks. However, in meshes, almost all the routing algorithms do achieve minimal routing. Also, when using two virtual channels, allowing transitions between them is necessary to achieve minimal paths. Second, tree-based routings with no virtual channels (UD, DFS) experience higher link weights, although the most noticeable cases occur in torus networks. Finally, the path balancing algorithm is key to reduce the link weight and obtain good performance.

6.3 Performance Evaluation

We expect those routing algorithms that generate good analytical results to also perform well in simulations (and real life). Indeed, those routing algorithms with a low standard deviation of link weights must achieve a good performance under uniform traffic. However, they may behave differently in other traffic situations. We will also explore how they behave in regular networks with some failures. In most cases, we plot the average packet latency[5] versus the average accepted traffic[6] measured in bytes/ns/switch. We mainly use uniform and bit-reversal[7] traffic patterns. In addition, hot-spot traffic is eventually explored for regular networks. In this case, 5 percent of messages are directed to switch 0 and the remaining ones uniformly distributed. In all the cases, packet size is 32 bytes.

6.4 Regular Topologies without Failures

Fig. 4 shows the performance results for all the routing algorithms requiring no virtual channels at all (DOR, LTURN, UD, FX, DFS, SMART, and SR) for a uniform traffic pattern. The results show that the FX algorithm always performs the best in meshes but not for tori. FX is equivalent to the DOR algorithm in meshes as both of them achieve roughly the same results. However, in torus networks, the FX algorithm experiences worse behavior with respect to DOR. This is because DOR uses two virtual channels in order to avoid cycles whereas FX uses no virtual channels, requiring longer paths.

Comparing DOR and FX in meshes, there is no clear winner. It seems that SR achieves better behavior but in larger mesh networks, DFS beats SR. Anyway, in general terms, we can observe that SR and SMART tend to perform better than UD, DFS, and LTURN in meshes. One important observation is the degradation experienced by LTURN for 16×8 mesh networks. The reason is that its path selection algorithm could not be applied in such a network, due to its high computation complexity. Thus, this phase is crucial for LTURN to achieve acceptable performance levels.

In torus networks, we can see that the tree-based routing algorithms (UD and DFS) experience low performance numbers whereas the intensive path computation algorithms (SMART and FX) achieve very good figures, close to the DOR performance. Fig. 5 shows results for routing algorithms requiring no virtual channels for the bit-reversal traffic pattern. From the figures, we see a totally different performance behavior. In particular, for meshes, SR routing beats all the routing algorithms by far for 8×4 and 8×8 meshes. However, SR is beaten by SMART for 16×8 meshes. In this case, SR performed a random path balancing algorithm as its original path balancing algorithm exhibited a high computation complexity. For the remaining algorithms in meshes, all of them exhibit similar performance levels, including DOR.

For tori, we can see that SMART is the winner for 8×8 and 16×8, but SR wins when it comes to the 8×4 topology. The reason for the lower performance of SR in larger torus networks is that its segmentation process has been optimized for meshes. Anyway, its performance levels are still acceptable. Also, we can see the varying performance of FX and DFS. For DFS, its path selection algorithm largely impacts the final performance.

Finally, Fig. 6 shows results for hot-spot traffic. In such scenario, most of the routing algorithms collapse their performance plot around the same region. Indeed, there is no clear winner. The exception is for UD routing which achieves half the performance in torus networks. The behavior for this kind of traffic is related to the bottleneck introduced by the traffic. The network saturates at lower levels due to the traffic and not due to the different performance levels of the algorithms.

For the routing methodologies with virtual channel requirements, Fig. 7 shows the results for uniform traffic. As it can be seen, differences among routing algorithms

5. Latency is the elapsed time between the injection of a packet at the source host until it is delivered at the destination host.

6. Accepted traffic is the amount of information delivered per time unit.

7. In bit-reversal traffic, the node with binary coordinates $a_{n-1}, a_{n-2}, \ldots, a_1, a_0$ communicates with the node $a_0, a_1, \ldots, a_{n-2}, a_{n-1}$.

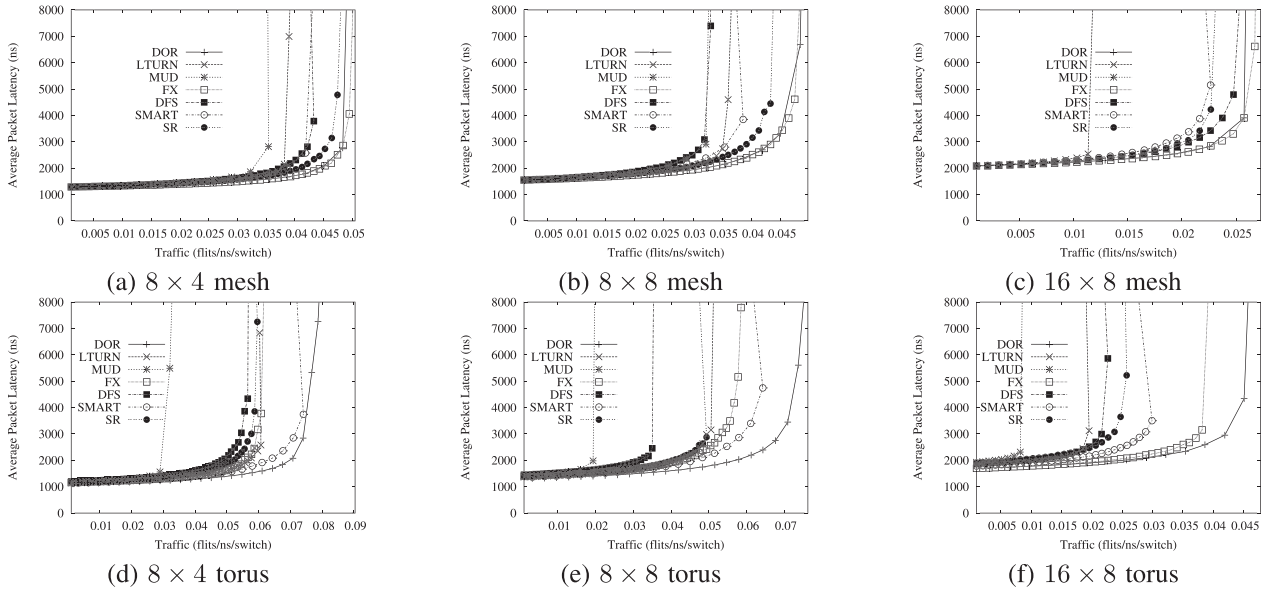


Fig. 4. Average packet latency versus accepted traffic. One VC per physical channel. Packet length is 32 bytes. Uniform traffic pattern.

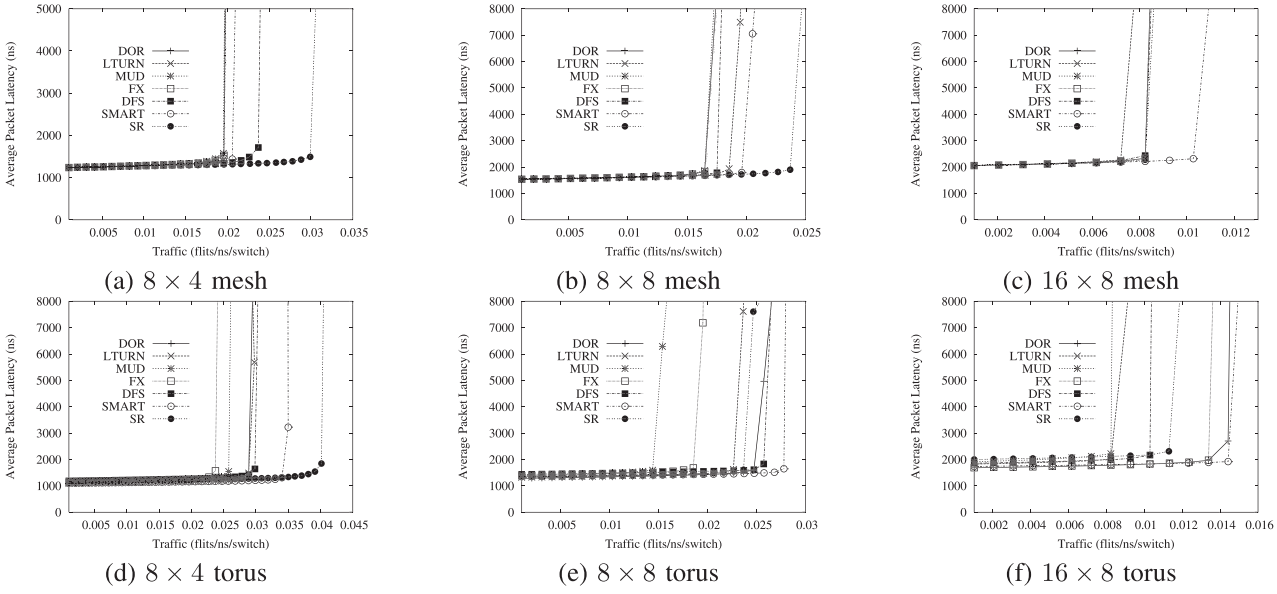


Fig. 5. Average packet latency versus accepted traffic. One VC per physical channel. Packet length is 32 bytes. Bit-reversal distribution of packet destinations.

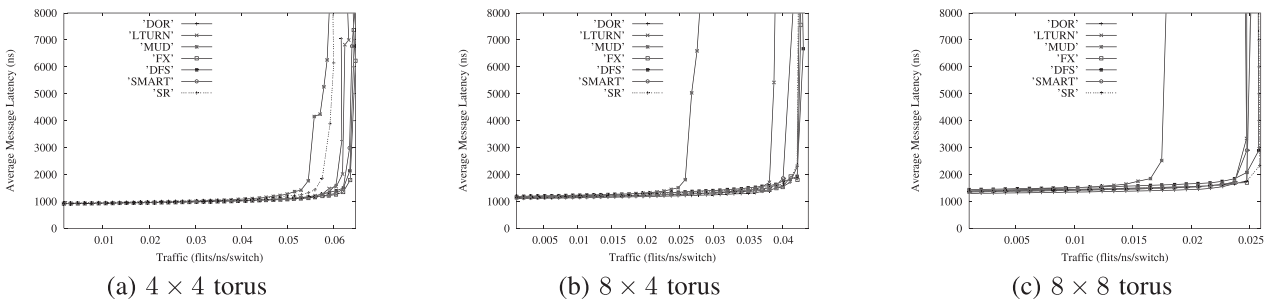
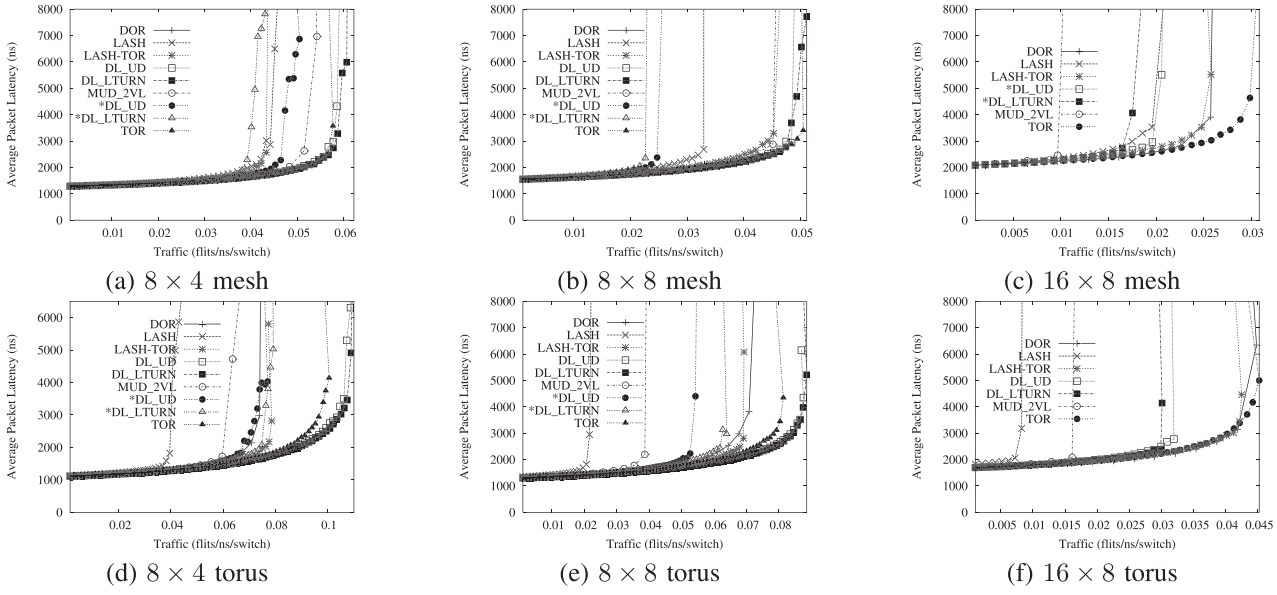


Fig. 6. Average packet latency versus accepted traffic. One VC per physical channel. Packet length is 32 bytes. Hot-spot distribution of packet destinations.

increase as network size become larger. In particular, the routings with lower performance are LASH and MUD. In both cases, as the network becomes larger, they have problems in guaranteeing shortest paths. On the contrary, the packet-based transition routings (TOR, LASH-TOR, and DL) achieve much better performance results. Indeed, TOR is the one with very remarkable results for all the topologies. On the other hand, DL-UD and DL-LTURN have performance problems in 16×8 meshes and tori due to its path balancing algorithm. This underlines the



(a) 8×4 mesh (b) 8×8 mesh (c) 16×8 mesh

(d) 8×4 torus (e) 8×8 torus (f) 16×8 torus

Fig. 7. Average packet latency versus accepted traffic. Two VCs per physical channel. Packet length is 32 bytes. Uniform traffic pattern.

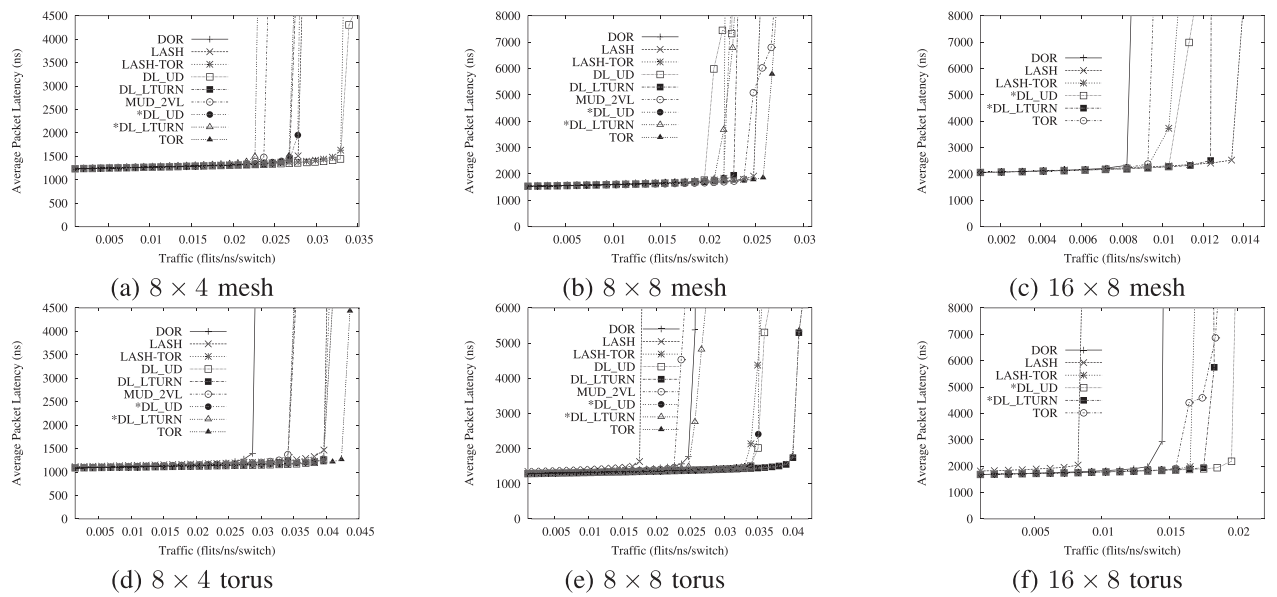
importance of applying a good path selection algorithm (recall that also LASH-TOR suffers from not having implemented a proper path selection algorithm). For bit-reversal traffic pattern (Fig. 8), slight differences appear. However, in most of the topologies, the packet-based transition algorithms (TOR, DL, and LASH-TOR) usually perform better. This is the case of TOR, which achieves the highest performance for the 8×8 mesh, 8×4 torus, and 8×8 torus topologies.

Comparing the performance evaluation results with that obtained by analyzing the link weight and ARD metrics (Section 6.2), we observe that the lower the standard deviation of link weight, the higher the performance obtained under uniform traffic. However, this relation does not hold when traffic is nonuniform. So, for routing algorithms that use one VC, those that achieve higher throughput (DOR, FX, and SMART) are those that exhibit the lowest values for the standard deviation of the link weight, both for meshes and tori. In general, these algorithms are also the same as that obtaining the lowest value for ARD. Indeed, this metric determines the latency for low and medium traffic rates (not appreciated clearly in plots). Similar conclusions are obtained for routing algorithms that require virtual channels. In particular, TOR and DL achieve higher throughput and at the same time exhibit the lowest values for the standard deviation of the link weight.

Table 6 summarizes the throughput (maximum accepted traffic rate) achieved by each routing algorithm for different topologies and network sizes under uniform traffic.

6.5 Regular Topologies with Failures

In the previous section, we have evaluated the performance of the topology-agnostic routings in regular topologies. In these networks, the use of topology-agnostic



(a) 8×4 mesh (b) 8×8 mesh (c) 16×8 mesh

(d) 8×4 torus (e) 8×8 torus (f) 16×8 torus

Fig. 8. Average packet latency versus accepted traffic. Two VCs per physical channel. Packet length is 32 bytes. Bit-reversal traffic pattern.

TABLE 6
Effectiveness of Each Routing Algorithm in Terms of Maximum Accepted Traffic Rate for Different Topologies and Network Sizes

		Low-traffic		Medium-traffic		High-traffic	
Size	Topology	1VC	2VC	1VC	2VC	1VC	2VC
8×4	mesh			MUD,LTURN	DL*,MUD,LASH,LASH-TOR	FX,SR,SMART,DFS	DL,TOR
8×4	torus	MUD	LASH	DFS,SR,FX	DL*,MUD,LASH-TOR	SMART	DL,TOR
8×8	mesh			MUD,DFS,SMART,LTURN	LASH,DL*	FX,SR	TOR,DL,LASH-TOR
8×8	torus	MUD,DFS	LASH,MUD	LTURN,SR,FX	DL*,LASH-TOR	SMART	DL,TOR
16×8	mesh	MUD	MUD	SMART,SR	DL*,LASH	FX,DFS	TOR,LASH-TOR
16×8	torus	MUD	LASH,MUD	LTURN,DFS,SR,SMART	DL*	FX	TOR,LASH-TOR

The considered traffic rates are relative to that achieved by the routing algorithm with the highest throughput. Empty cells mean that all routing algorithms are effective for low traffic rates in 8×4 and 8×8 meshes.

routings is not a requirement since there are other specialized routings (e.g., DOR) that achieve good performance. The main motivation for using topology-agnostic routings are those cases where regular networks experience some failures. In this section, we evaluate the routing algorithms in such a scenario.

We evaluate some topologies derived from the previous experiments. For each topology, we inject a certain number of randomly generated link failures. In particular, we will analyze topologies with 1, 3, and 5 percent of link failures, rounding up the number of failures (e.g., in a 4×4 network, we set one, two, and three link failures for each percentage). In order to get results independent of the failure location, we evaluate 10 different topologies for each case. To simplify the plotting of results, we will show results using Kiviat diagrams, showing the relative throughput achieved by each routing (the relative throughput will be computed as the percentage of achieved throughput relative to the maximum one) for each topology.

Fig. 9 shows the normalized throughput for the routing algorithms with no virtual channel requirements under uniform traffic. The first thing we can observe is the remarkable good results achieved by SMART. In most topologies, and regardless of the number of failures, SMART achieves roughly the maximum throughput. Only for 16×8 networks, it loses some ground. The reason again is its very good traffic balancing algorithm. When fully deployed (up to 8×8 networks), it allows achieving maximum throughput with uniform traffic. For larger networks (16×8), SMART is beaten by LTURN. However, LTURN does not behave as good for smaller networks. Also, it is noticeable the progressive degradation of tree-based routing algorithms (MUD, DFS) as network size increases. The same happens with FX (that was a good candidate for regular networks but no longer for irregular networks). Finally, SR also experiences both good and bad performance levels. For small networks, it achieves in most cases maximum performance, whereas in larger networks

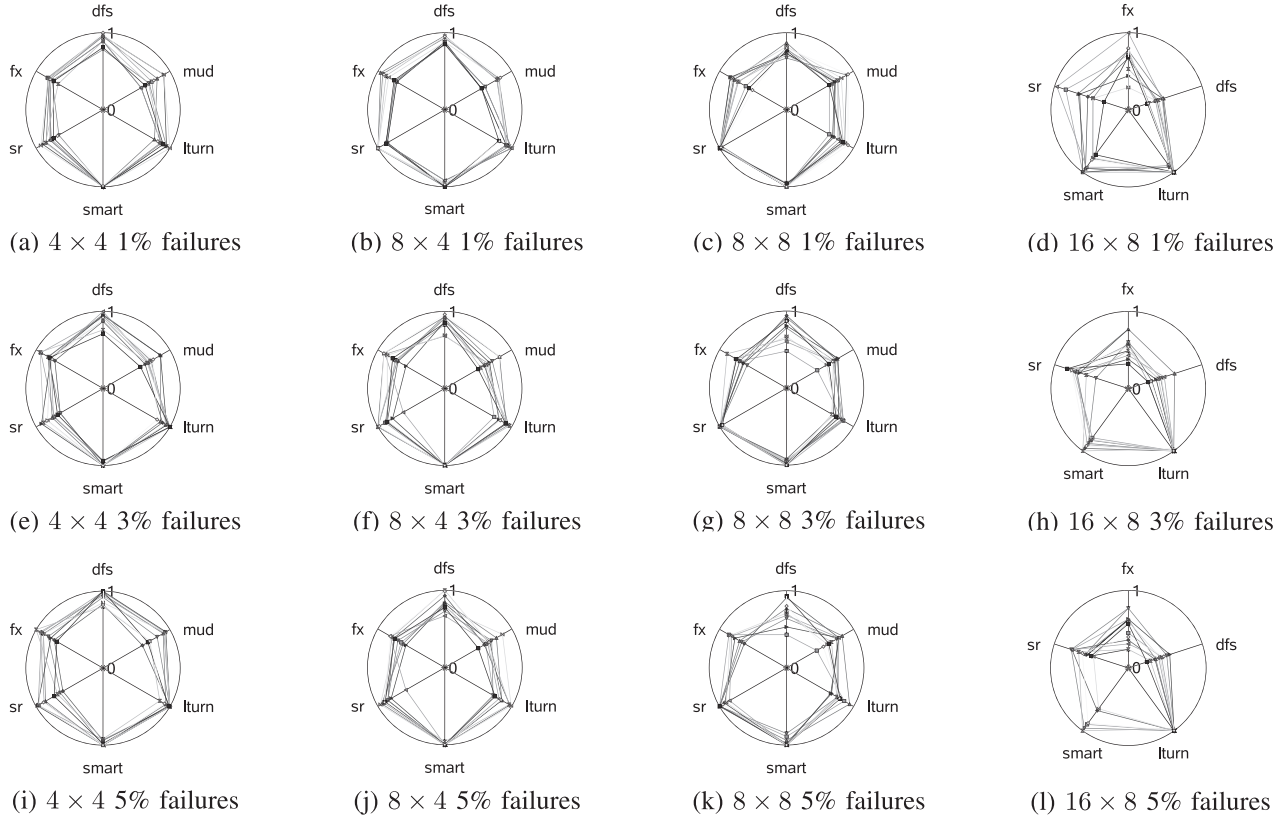


Fig. 9. Normalized throughput for irregular topologies. For each topology, 10 random topologies analyzed. One VC. Uniform traffic pattern.

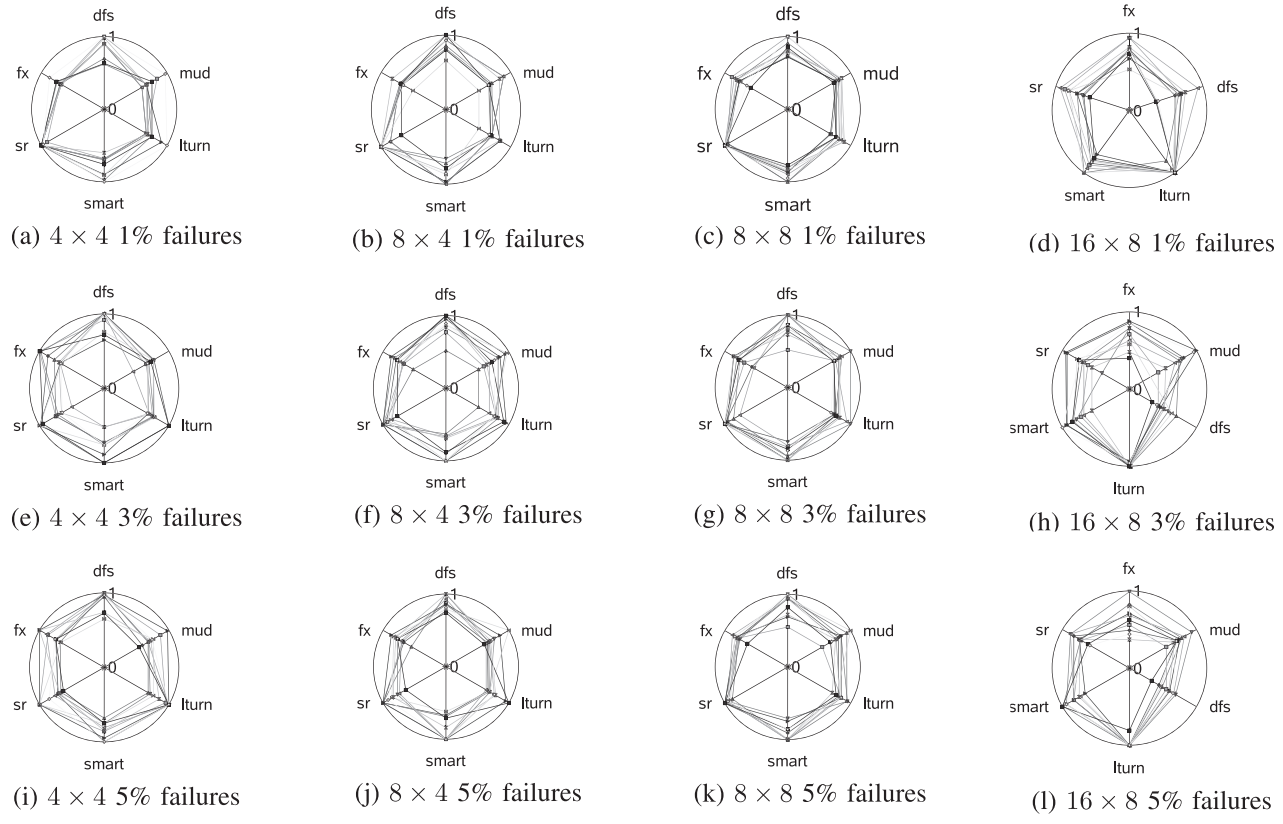


Fig. 10. Normalized throughput for irregular topologies. For each topology, 10 random topologies analyzed. 1 VL. Bit-reversal traffic pattern.

(16×8), it achieves lower results (due to its impossibility to apply its traffic balance algorithm).

For bit-reversal traffic (shown in Fig. 10), we can observe similar results, but there is not a clear winner. Depending on network size and the percentage of link failures (related to the degree of irregularity), some routing algorithms behave better. This is the case for SR in 4×4 with 1 percent failures and in 8×8 with 1, 3, and 5 percent link failures and for LTURN in 16×8 networks with any percentage of link failures. For tori networks with failures, similar results were obtained for uniform and bit-reversal traffic. However, they are not shown due to lack of space.

Finally, let us focus our attention on the routing algorithms with virtual channel requirements (Fig. 11) under uniform traffic. In this case, more differences are observed. First, transition packet-based routings (DL, TOR, and LASH-TOR) are the clear winners for any topology. In small- and medium-sized networks, DL achieves the highest performance. This is due to its traffic balancing algorithm. Indeed, for the 16×8 topologies, it looses ground in benefit of TOR. Anyway, TOR routing achieves very acceptable performance levels for any topology. Regarding LASH-TOR, it exhibits varying results (mainly because it here uses random paths selection), but otherwise tending to achieve higher performance for larger networks. On the other hand, LASH experiences very low performance levels. This is because of its random traffic balancing algorithm and its need of more virtual channels in order to guarantee minimal routing. For MUD, it achieves acceptable levels of performance but it is not able to achieve the highest performance for any topology. For bit-reversal traffic patterns and for torus networks with link failures, similar results were obtained (not shown).

As a summary of the performance evaluation, we can obtain the following conclusions. First, when not using virtual channels, the SMART algorithm is the one that achieves the highest performance in most of the topologies. Remember, though, the high computational complexity of this routing algorithm. When using two virtual channels, the packet-transition-based routing algorithms (TOR, LASH-TOR, and DL) are the ones that achieve the highest performance. Thus, allowing transition among virtual channels per-packet basis is also significant, as it allows minimal paths. Also, the traffic balance algorithm is key to allow high performance.

6.6 Impact of Path Selection

One thing that is hard to measure and control is the effect of path selection. Because path selection for the different methods works under different constraints, it is impossible to devise a common path selection algorithm for all methods. Ideally, we should have compared all methods using the best possible path selection algorithm for that particular method, but clearly the combinatorial explosion of all possible strategies disallows us from following such an approach. On the other hand, it is clear from our results that the strategy for path selection is a very important factor for the end performance of the methods. For example, in the case of a fault-free mesh, we can make all path-driven methods be equal to DOR (by choosing the DOR paths).

Let a *configuration* be a set of paths (with transitions when applicable) from every source to every destination (minimal or nonminimal). Then, the end result of any

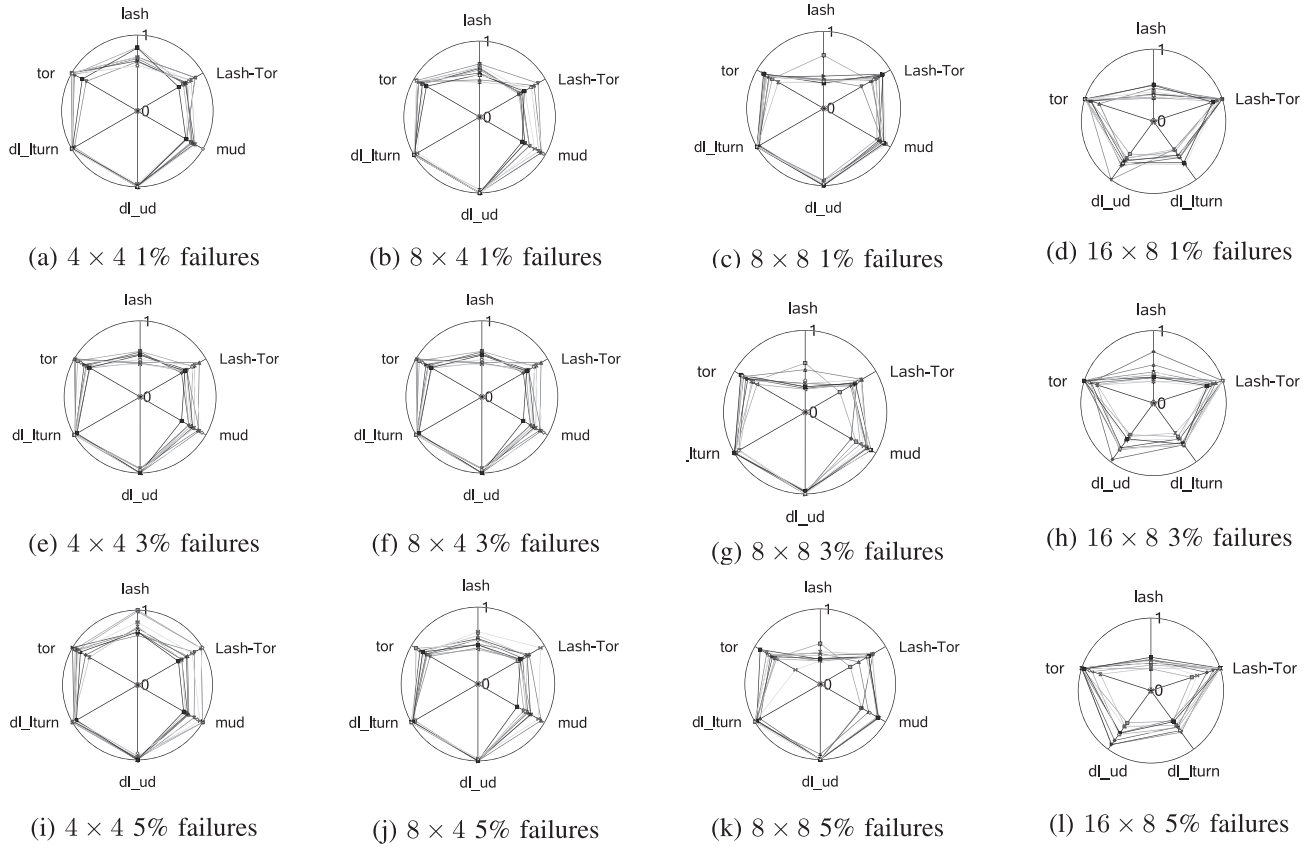


Fig. 11. Normalized throughput for irregular topologies. For each topology, 10 random topologies analyzed. 2 VL. Uniform traffic pattern.

routing method will be a configuration. It is then clear that the routing method that provides the best configuration is the one that will achieve the best overall performance. For methods with only one virtual channel, this will be the method that imposes the fewest and most flexible routing restrictions. It is well known that LTURN introduces few routing restrictions, and that SR, FX, and SMART introduce the most flexible routing restrictions of the methods that only require one virtual channel, so in this case, this is compatible with our simulation results.

For the methods that require more than one virtual channel, we make the following observations. First, the set of configurations that would be allowed in DL-UD routing is equal to the set that would be allowed in TOR. Whenever our results are different for these two methods, this is due to different path selection, not the routing method itself. Regarding the relationship between TOR, DL, and LASH-TOR, recall that they differ only in where transitions must be carried out in order to avoid deadlock. The more transitions they require, the less paths will be minimal and the more dependencies will be introduced between the layers. TOR and DL require a transition when the path violates a rule-driven routing (UD or LTURN), and they do this regardless of whether the current path could actually generate a deadlock. LASH-TOR requires transitions if *and only if* the path closes a deadlock cycle. Therefore, all configurations that may come out of TOR and DL will also be valid for LASH-TOR, and for that reason, we may conclude that LASH-TOR with the best possible path selection should perform at least as well as TOR and DL with the best possible path selection algorithm. This is only partially reflected in our results, something which underlines the importance of path selection for any of these methods.

7 Applying the Results to Technologies

In the previous text, we have discussed the routing methods in an idealized way. This section analyzes the constraints imposed by different network technologies on which routing methods are implementable, which are summarized in Table 7.

7.1 Myrinet

Myrinet [1] is a technology that is able to handle any topology. Therefore, it has a flexible routing scheme that

TABLE 7
Applicability of the Algorithms to Different Technologies

	Myrinet	Servernet II	InfiniBand	Ethernet	Quadrics
DOR	✓	✓	✓	✓	✓
UD	✓	✓	✓	✓	✓
DFS	✓	✓	✓	✓	✓
LTURN	✓	✓	✓	✓	✓
SR	✓	✓	✓	✓	✓
SMART	✓	✓	✓	✓	✓
FX	✓	✓	✓	✓	✓
MUD			✓	*	
LASH			✓	*	
TOR			Partly		
LASH-TOR			Partly		
DL			Partly		

*For InfiniBand, three algorithms are usable partially due to the limited size of the SLtoVL table. * indicates the routing algorithms are applicable if Ethernet ultimately can work on a per-priority basis.*

does not restrict the way by which packets are forwarded. On the other hand, it does not support virtual channels. This means that for Myrinet, the routing algorithms that are possible are DOR, UD, DFS, LTURN, SR, SMART, and FX. Our results for these cases vary depending on the size of the network and whether there are faults in the network. In general, it can be said that as long as the network is fault free, and thus regular, FX and SMART are the ones that perform the best. In this case, however, there is no need for topology-agnostic routing algorithms, so the more interesting case is perhaps where there are faults in the network. Here, it becomes clear that SMART, and in some cases SR and LTURN, outcompetes the others.

7.2 Servernet II

Although the Servernet II [2] architecture allows virtual channels, its current implementation does only have one virtual channel. Thus, the same limitations apply as for Myrinet.

7.3 InfiniBand

InfiniBand [4] provides flexible routing tables and supports up to 16 virtual lanes, but current implementations generally provide four or eight virtual lanes. There are mechanisms that allow packets to transition between layers in Infiniband. However, the flexibility of these mechanisms is limited. For some of the routing methods that require such transitions, it has, though, been shown that the existing mechanism (SL to VL mapping tables) is sufficient for networks of reasonable size.

Basically, this means that in this technology, there is a choice of which resources to use for effective routing. One may constrict oneself to methods that require only one virtual channel (DOR, UD, DFS, LTURN, SR, SMART, and FX) if virtual lanes are to be used for something else than effective routing (e.g., QoS). On the other hand, if multiple virtual lanes can be used, we can in addition use MUD and LASH. In this case, our results show that MUD outperforms LASH when the number of virtual channels is as low as 2. It must, however, be noted that the full benefit of LASH will not appear until one has more than two virtual channels. In [24], it is reported that LASH outperforms MUD by far when the number of virtual channels is sufficient to guarantee shortest paths.

It will also be possible to use the existing mechanisms for transitions between virtual channels, opening for TOR, LASH-TOR, and DL. Our simulation results indicate that TOR is the overall best one. However, we believe that this is mainly due to the balancing effort we implemented for the paths in TOR. If such balancing had been performed for LASH-TOR and DL as well, there is every reason to believe that the results would have been comparable for all three, with a possible advantage for LASH-TOR. What is consistent, however, is the fact that with few virtual channels available, transitions between the virtual channels are important to gain throughput.

7.4 Ethernet

Ethernet [3] is a technology that comes in many variants. In its more pure form, the routing algorithm is already given. It is based on a spanning tree protocol, and is independent of topology. It is, however, well known that in many cases, the spanning tree protocol leads to a massive underutilization of network resources. In particular, it will disable a subset of the links in the network in order to make the network free of cycles. Therefore, the spanning tree protocol can be bypassed in many commercial switches, and other routing strategies can be implemented through upload of pregenerated routing tables.

In its basic version, Ethernet discards packets, and is therefore not prone to deadlocks. These versions are out of scope for the routing algorithms we discuss here, that imposed deadlock freedom as one of their design criteria. The flow-controlled version of Ethernet (802.1Q) may be run in a modus where packets are not discarded, and delivery is guaranteed. In this mode, Ethernet is prone to deadlocks, and the routing algorithms discussed here are of interest.

The routing algorithms that are applicable in Ethernet with flow control are the ones that do not require virtual channels (DOR, UD, DFS, SR, LTURN, SMART, and FX). In fact, in [40], SR tackles the described spanning tree weakness. There is a mechanism in Ethernet called *priority tagging* that supports different treatment of packets, and that comes very close to virtual channels (version 802.1Q). There are, however, two obstacles to this. The first is the ability to treat the different packet priorities according to different routing schemes. This problem can be circumvented by using VLAN tags [41]. The second obstacle is that the flow control in Ethernet works on a per-link basis, not on a per-priority basis. This means that deadlock freedom cannot be treated separately for each priority; thus, the different priorities cannot immediately be viewed as virtual layers. A rather small modification in the Ethernet flow control would, though, remedy this. Changing priorities on the fly would, however, require a more profound change. Therefore, the routing strategies that are available for Ethernet as it is today are those that do not require virtual channels. With a small change in the flow-control scheme, MUD and LASH could be used as well.

7.5 Quadrics

Quadrics [5] is a source routed technology and supports two virtual channels. It does, however, not support transitions between virtual channels. This basically means that TOR, LASH-TOR, and DL cannot be applied. The small number of virtual channels makes LASH of limited value, meaning that MUD is the best choice when both virtual channels may be used for efficient routing. However, all the routings requiring a single virtual channel could be applied to Quadrics. Therefore, our comments for Myrinet are also relevant for Quadrics.

7.6 Networks on Chip

Networks on chip are different from the above technologies in that the set of features they will be able to support is not fixed. First, we need to differentiate between application-specific chip designs, usually found in the embedded market, and chip multiprocessor (CMP) systems, usually targeting high-performance computing. In the first case, topology is usually totally irregular and fitted to the running application. In that case, the routing algorithm is set by the application at design time and there is no need to use topology-agnostic routing algorithms. In the other case, CMPs, the main topology is becoming the 2D mesh (e.g.,

Polaris chip [42] and SCC chip from Intel [43], Tilera products [44]). However, during recent years, manufacturing defects have been identified as a primary source of failures for current and future NoCs. Indeed, recently, there have been publications addressing this issue [11], [45] as well as other sources that may induce irregularities in the original design (e.g., heterogeneity of the multicore components, variability [46], DVFS domains [47], and power saving strategies [45]). In addition, as technology scales, virtualization is appealing in order to get an efficient use of all the computing resources within a chip. To maximize chip utilization, the chip might require a partition into irregular regions. In this scenario, CMPs with induced irregularities, the use of topology-agnostic routing algorithms makes sense and could fit the necessities.

Compared against macro multicomputer systems, the design of scalable and reliable NoCs for CMPs exhibits some different requirements; the primary consideration is to minimize area and power dissipation, such limitations impose the choice of topology, packetization, routing algorithms, and architectural implementations. At first sight, regular topologies combined with adaptive routing algorithms and wormhole switching are the preferred choices. Unfortunately, due to the aforementioned issues, the resulting interconnection topology is not longer regular preventing the use of traditional routing algorithms for regular topologies. It is worth differentiating between the routing algorithm and its implementation. In NoCs, this distinction is fundamental as an efficient implementation will enable the use of a routing algorithm (even if it is complex in its rules). Incoming challenges and the way to implement topology-agnostic routing algorithms are detailed in [48].

Basically, there are two factors that determine what routing strategy is applicable: the availability of virtual channels, and the ability to let packets transition between them. For NoCs where the footprint of the chip disallows the extra buffer space required for multiple virtual channels, we are in the same situation as the one described for Myrinet and Servernet above. In the cases where virtual channels are there, but where the extra resources needed for transitions between layers are prohibitive, the considerations will be identical to those made for Quadrics. Finally, if we allow both virtual channels and some resources for making transitions between them, the above discussion for Infiniband applies.

8 Summary and Conclusions

The increasing probability of failures and the flexibility offered by interconnection networks for clusters may cause their interconnection topologies to become irregular. In such a situation, topology-agnostic routing algorithms may become a simple and effective solution to keep the system running even in the presence of failures. Also, irregular networks may appear in on-chip systems due to fabrication processes and heterogeneous IP nodes, thus obtaining benefit from the use of topology-agnostic routing algorithms.

In this paper, we have put in the same context all the topology-agnostic routing algorithms we are aware of. In a first effort, we have classified them based on their requirements and their foundations. Routings have been classified primarily based on their requirements for virtual channels and their capability to guarantee minimal routing. As a second contribution, we have provided a unified description of all the routing algorithms. This description helped us to understand better the anatomy of each topology-agnostic routing scheme.

As a third effort, we have evaluated all the routing algorithms under the same traffic and network conditions. Results have shown that using virtual channels helps in achieving minimal paths for each pair of nodes. Also, allowing virtual channel transitions reduces the required number of virtual channels. However, for achieving maximum network throughput, it is compulsory to use a path balancing algorithm.

Finally, as a fourth and last contribution, we have provided insights of application of the different routing algorithms to current state-of-the-art interconnection networks for off-chip communication and for on-chip networks. We have shown that among the analyzed topology-agnostic routings, we always can find enough candidates able to fulfill the constraints imposed for each network technology or application field.

Acknowledgments

This work was supported by the Spanish MEC and MICINN, as well as by the European Commission FEDER funds, under Grants CSD2006-00046, TIN2009-14475-C04, and RYC2009-03989.

References

[1] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, and W.-K. Su, "Myrinet: A Gigabit-per-Second Local Area Network," *IEEE Micro,* vol. 15, no. 1, pp. 29-36, Feb. 1995.

[2] R.W. Horst, D.P. Sonnier, and W.J. Watson, "A Flexible Servernet-based Fault-Tolerant Architecture," *Proc. 25th Int'l Symp. Fault-Tolerant Computing (FTCS '95),* p. 2, 1995.

[3] R. Seifert, *Gigabit Ethernet: Technology and Applications for High-Speed LANs.* Addison-Wesley Longman Publishing Co., Inc., 1998.

[4] I.T. Assoc. "Infiniband Architecture Specification Release 1.2.1," http://www.infinibandta.org/specs/register/publicspec/, Jan. 2008.

[5] F. Petrini, W.C. Feng, A. Hoisie, S. Coll, and E. Frachtenberg, "The Quadrics Network (qsnet): High-Performance Clustering Technology," *Proc. Ninth Symp. High Performance Interconnects (HOTI '01),* p. 125, 2001.

[6] Top500 "Top500 Supercomputer List," www.top500.org, 2011.

[7] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach.* Morgan Kaufmann, 2003.

[8] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks.* Morgan Kaufmann Publishers, Inc., 2003.

[9] H.-C. Chi and C.-T. Tang, "A Deadlock-Free Routing Scheme for Interconnection Networks with Irregular Topologies," *Proc. Int'l Conf. Parallel and Distributed Systems,* pp. 88-95, Dec. 1997.

[10] M.E. Gómez, P. López, and J. Duato, "A Memory-Effective Routing Strategy for Regular Interconnection Networks," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp.,* p. 41b, 2005.

[11] S. Rodrigo, S. Medardoni, J. Flich, D. Bertozzi, and J. Duato, "Efficient Implementation of Distributed Routing Algorithms for NoCs," *IET Computers and Digital Techniques,* vol. 3, pp. 460-475, 2009.

[12] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers,* vol. C-36, no. 5, pp. 547-553, May 1987.

[13] 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.

[14] M.D. Schroeder, A.D. Birrell, M. Burrows, H. Murray, R.M. Needham, and T.L. Rodeheffer, "Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links," *IEEE J. Selected Areas in Comm.,* vol. 9, no. 8, pp. 1318-1335, Oct. 1991.

[15] J.C. Sancho, A. Robles, and J. Duato, "A New Methodology to Compute Deadlock-Free Routing Tables for Irregular Networks," *Proc. Workshop Comm., Architecture and Applications for Network-Based Parallel Computing (CANPC '00),* Jan. 2000.

[16] M. Koibuchi, A. Funahashi, A. Jouraku, and H. Amano, "L-Turn Routing: An Adaptive Routing in Irregular Networks," *Proc. Int'l Conf. Parallel Processing (ICPP '01),* pp. 383-392, 2001.

[17] A. Mejia, J. Flich, J. Duato, S.-A. Reinemo, and T. Skeie, "Segment-Based Routing: An Efficient Fault-Tolerant Routing Algorithm for Meshes and Tori," *Proc. 20th Int'l Parallel and Distributed Processing Symp. (IPDPS '06),* Apr. 2006.

[18] W. Qiao and L.M. Ni, "Adaptive Routing in Irregular Networks Using Cut-Through Switches," *Proc. Int'l Conf. Parallel Processing (ICPP '96),* pp. 52-60, 1996.

[19] F. Silla and J. Duato, "Improving the Efficiency of Adaptive Routing in Networks with Irregular Topology," *Proc. Fourth Int'l Conf. High-Performance Computing (HIPC '97),* p. 330, 1997.

[20] L. Cherkasova, V. Kotov, and T. Rokicki, "Designing Fibre Channel Fabrics," *Proc. IEEE Int'l Conf. Computer Design (ICCD '95),* pp. 346-351, 1995.

[21] J. Flich, M.P. Malumbres, P. Lopez, and J. Duato, "Performance Evaluation of a New Routing Strategy for Irregular Networks with Source Routing," *Proc. Int'l Conf. Supercomputing (ICS '00),* 2000.

[22] T. Skeie, O. Lysne, and I. Theiss, "Layered Shortest Path (LASH) Routing in Irregular System Area Networks," *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS '02),* 2002.

[23] J. Domke, T. Hoefler, and W. Nagel, "Deadlock-Free Oblivious Routing for Arbitrary Topologies," *Proc. 25th IEEE Int'l Parallel and Distributed Processing Symp.,* May 2011.

[24] O. Lysne, T. Skeie, S.-A. Reinemo, and I. Theiss, "Layered Routing in Irregular Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 17, no. 1, pp. 51-65, Jan. 2006.

[25] I. Theiss and O. Lysne, "FROOTS - Fault Handling in Up*/Down* Routed Networks with Multiple Roots," *Proc. Int'l Conf. High Performance Computing (HiPC '03),* 2003.

[26] J. Flich, P. Lopez, J.C. Sancho, A. Robles, and J. Duato, "Improving Infiniband Routing through Multiple Virtual Networks," *Proc. Fourth Int'l Symp. High Performance Computing (ISHPC '02),* pp. 49-63, 2002.

[27] J.C. Sancho, A. Robles, J. Flich, P. Lopez, and J. Duato, "Effective Methodology for Deadlock-Free Minimal Routing in Infiniband Networks," *Proc. Int'l Conf. Parallel Processing (ICPP '02),* pp. 409-418, 2002.

[28] M. Koibuchi, A. Jouraku, K. Watanabe, and H. Amano, "Descending Layers Routing: A Deadlock-Free Deterministic Routing Using Virtual Channels in System Area Networks with Irregular Topologies," *Proc. Int'l Conf. Parallel Processing (ICPP '03),* Oct. 2003.

[29] T. Skeie, O. Lysne, J. Flich, P. Lopez, A. Robles, and J. Duato, "Lash-Tor: A Generic Transition-Oriented Routing Algorithm," *Proc. IEEE Int'l Conf. Parallel and Distributed Systems (ICPADS '04),* pp. 595-604, 2004.

[30] Y. Liu, C. Dwyer, and A. Lebeck, "Routing in Self-Organizing Nano-Scale Irregular Networks," *ACM J. Emerging Technologies in Computing Systems,* vol. 6, no. 1, pp. 1-21, 2010.

[31] R. Moraveji, H. Sarbazi-Azad, and A. Zomaya, "A General Methodology for Direction-Based Irregular Routing Algorithms," *J. Parallel and Distributed Computing,* vol. 70, no. 4, pp. 363-370, 2010.

[32] J. Cong, C. Liu, and G. Reinman, "ACES: Application-Specific Cycle Elimination and Splitting for Deadlock-Free Routing on Irregular Network-on-Chip," *Proc. 47th ACM/IEEE Design Automation Conf. (DAC),* pp. 443-448, 2010.

[33] R. Holsmark, M. Palesi, and S. Kumar, "Deadlock Free Routing Algorithms for Irregular Mesh Topology NoC Systems with Rectangular Regions," *J. Systems Architecture,* vol. 54, nos. 3/4, pp. 427-440, 2008.

[34] M. Koibuchi, A. Jouraku, and H. Amano, "Routing Algorithms Based on 2D Turn Model for Irregular Networks," *Proc. Int'l Symp. Parallel Architectures, Algorithms and Networks (ISPAN '02),* 2002.

[35] J.C. Sancho, A. Robles, and J. Duato, "An Effective Methodology to Improve the Performance of the Up*/Down* Routing Algorithm," *IEEE Trans. Parallel Distributed Systems,* vol. 15, no. 8, pp. 740-754, 2004.

[36] J. Flich, P. Lopez, M.P. Malumbres, J. Duato, and T. Rokicki, "Combining In-Transit Buffers with Optimized Routing Schemes to Boost the Performance of Networks with Source Routing," *Proc. Third Int'l Symp. High Performance Computing (ISHPC '00),* pp. 300-309, 2000.

[37] P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks,* vol. 3, no. 4, pp. 267-286, Sept. 1979.

[38] W.J. Dally, "Express Cubes: Improving the Performance of K-Ary n-Cube Interconnection Networks," *IEEE Trans. Computers,* vol. 40, no. 9, pp. 1016-1023, Sept. 1991.

[39] J.C. Sancho, A. Robles, and J. Duato, "A Flexible Routing Scheme for Networks of Workstations," *Proc. Third Int'l Symp. High Performance Computing (ISHPC '00),* pp. 260-267, 2000.

[40] A. Mejia, J. Flich, J. Duato, S.-A. Reinemo, and T. Skeie, "Boosting Ethernet Performance by Segment-Based Routing," *Proc. 15th Euromicro Conf. Parallel, Distributed and Network-Based Processing (PDP '07),* Feb. 2007.

[41] S.-A. Reinemo and T. Skeie, "Ethernet as a Lossless Deadlock Free System Area Network," *Proc. Int'l Symp. Parallel and Distributed Processing and Applications,* Y. Pan, D. Chen, M. Guo, J. Cao, and J. Dongarra, eds., pp. 901-914, 2005.

[42] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz Mesh Interconnect for a Teraflops Processor," *IEEE Micro,* vol. 27, no. 5, pp. 51-61, Sept. 2007.

[43] J. Rattner "Single-Chip Cloud Computer: An Experimental Many-Core Processor from Intel Labs," http://download.intel.com/pressroom/pdf/rockcreek/SCC_Announcement_JustinRattner.pdf, 2011.

[44] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. Miao, J.F. Brown III, and A. Agarwal, "On-Chip Interconnection Architecture of the Tile Processor," *IEEE Micro,* vol. 27, no. 5, pp. 15-31, Sept./Oct. 2007.

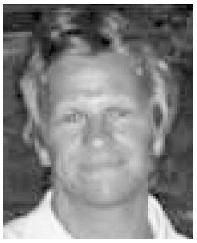
[45] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato, "Addressing Manufacturing Challenges with Cost-Efficient Fault Tolerant Routing," *Proc. Fourth ACM/IEEE Int'l Symp. Networks-on-Chip,* pp. 25-32, 2010.

[46] C. Hernández, A. Roca, F. Silla, J. Flich, and J. Duato, "Improving the Performance of GALS-Based NoCs in the Presence of Process Variation," *Proc. Fourth Int'l Symp. Networks-on-Chip,* May 2010.

[47] U.Y. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu, "Voltage-Frequency Island Partitioning for GALS-Based Networks-on-Chip," *Proc. Design Automation Conf.,* pp. 110-115, June 2007.

[48] J. Flich and D. Bertozzi, *Designing Network On-Chip Architectures in the Nanoscale Era.* CRC Press, Taylor and Francis, 2010.

José Flich received the MS and PhD degrees in computer science from the Technical University of Valencia, Spain, in 1994 and 2001, respectively. He joined the Department of Computer Engineering, Universidad Politécnica de Valencia, in 1998, where he is currently an associate professor of computer architecture and technology with the Parallel Architectures Group. He has published more than 100 papers in peer-reviewed conferences and journals. His current research interests include high-performance interconnection networks for multiprocessor systems, cluster of workstations, and networks on chip. He has served as a program committee member in different conferences, including NOCS, DATE, ICPP, IPDPS, HiPC, SC, CAC, ICPADS, and ISCC. He is an associate editor of the *IEEE Transactions on Parallel and Distributed Systems* and currently the cochair of the CAC and INA-OCMC workshops. He is the coordinator of the NaNoC FP7 EU-Funded Project (http://www.nanoc-project.eu) and editor of the book *Designing Network On-Chip Architectures in the Nanoscale Era*. He is a member of the IEEE and the IEEE Computer Society.



Tor Skeie received the MS and PhD degrees in computer science from the University of Oslo in 1993 and 1998, respectively. He is a professor at the Simula Research Laboratory and the University of Oslo. His work is mainly focused on scalability, effective routing, fault tolerance, and quality of service in switched network topologies. He is also a researcher in the Industrial Ethernet area. The key topics here have been the road to deterministic Ethernet end to end and how precise time synchronization can be achieved across switched Ethernet. He has also contributed to wireless networking, hereunder quality of service in WLAN, and cognitive radio.

Andrés Mejía received the PhD degree from Technical University of Valencia, Spain, in 2008. He is currently a senior research scientist at Intel Research Labs in Santa Clara, California. His research interests include different areas of high-speed interconnects including power optimization, performance modeling, analysis, and validation.

Olav Lysne received the master's degree in 1988 and the Dr. Scient. degree in 1992 from the University of Oslo. He is the head of a research department as well as professor in computer science at Simula Research Laboratory and the University of Oslo. His research interests are in network architectures. In particular, he works on how routing functions, network topologies, and strategies for buffer management influence on network performance, fault tolerance, and quality of service. He is the coinventor of LASH (Layered Shortest Path Routing) that is implemented in the OpenFabrics Enterprise Distribution (OFED), and is used in many high-performance compute clusters around the world. In 2007, his research team was the first in the world to demonstrate a working solution for IP-fast reroute. This is a technology that provides recovery in IP-networks in less than 50 milliseconds after a link or router has failed. He has more than 100 peer-reviewed publications, and several patents. He is a member of the IEEE.

Pedro López received the BEng degree in electrical engineering and the MS and PhD degrees in computer engineering from the same university in 1984, 1990, and 1995, respectively. He is a full professor in computer architecture and technology at the Department of Computer Engineering (DISCA), Universitat Politècnica de València, Spain. He has taught several courses on computer organization and architecture. His research interests include high-performance interconnection networks for multiprocessor systems and clusters and processor microarchitecture. He has published more than 100 refereed conference and journal papers. He is a member of the editorial board of *Parallel Computing* journal. He is a member of the IEEE Computer Society.

Antonio Robles received the MS degree in physics (electricity and electronics) from the Universidad de Valencia, Spain, in 1984, and the PhD degree in computer engineering from the Universidat Politècnica de València in 1995. He is currently a full professor in the Department of Computer Engineering at the Universidat Politècnica de València, Spain. He has taught several courses on computer organization and architecture. His research interests include high-performance interconnection networks for multiprocessor systems and clusters, and scalable cache coherence protocols for SMP and CMP. He has published more than 70 refereed conference and journal papers. He has served on program committees for several major conferences. He is a member of the IEEE and the IEEE Computer Society.

José 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, and a researcher at Simula Research Laboratory, Oslo, Norway. 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 400 refereed papers. He proposed a powerful theory of deadlock-free adaptive routing for wormhole networks. Versions of this theory have been used in the design of the routing algorithms for the MIT Reliable Router, the Cray T3E supercomputer, the on-chip router of the Alpha 21364 microprocessor, and the IBM BlueGene/L supercomputer. He also developed RECN, a scalable congestion management technique, and a very efficient routing algorithm for fat trees that has been incorporated into Sun Microsystem's 3456-port InfiniBand Magnum switch. Currently, he leads the Advanced Technology Group in the HyperTransport Consortium, whose main result to date has been the development and standardization of an extension to HyperTransport (High Node Count HyperTransport Specification 1.0). He is the first author of the book *Interconnection Networks: An Engineering Approach*. He served as a member of the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, and *IEEE Computer Architecture Letters*. He has been the general cochair for the 2001 International Conference on Parallel Processing, the program committee chair for the 10th International Symposium on High Performance Computer Architecture (HPCA-10), and the program cochair for the 2005 International Conference on Parallel Processing. Also, he served as the cochair, member of the Steering Committee, vice chair, or member of the Program Committee in more than 60 conferences, including the most prestigious conferences in his area (HPCA, ISCA, IPPS/SPDP, IPDPS, ICPP, ICDCS, Europar, and HiPC).

Michihiro Koibuchi received the BE, ME, and PhD degrees from Keio University, Yokohama, Japan, in 2000, 2002, and 2003, respectively. He was a visiting researcher at the Technical University of Valencia, Spain, in 2004, and a visiting scholar at the University of Southern California in 2006. He is currently an associate professor in the Information Systems Architecture Research Division, National Institute of Informatics, Tokyo, and the Graduate University for Advanced Studies, Japan. His research interests include the areas of high-performance computing and interconnection networks. He is a member of the IEEE and the IEEE Computer Society.

Tomas Rokicki received the BS degree in electrical engineering from Texas A&M University in 1985 and the PhD degree in computer science from Stanford University in 1993. He joined Hewlett-Packard Laboratories in 1993, where he pursued his interests in computer networking and architecture. In 1999, he founded Instantis with a few colleagues, where he is the director of technology.

José Carlos Sancho received the MS and PhD degrees in computer science from the Technical University of Valencia, Spain, in 1998 and 2002, respectively. In 2003, he joined Los Alamos National Laboratory, New Mexico, where he participated on the performance characterization of Roadrunner, the first supercomputer in the world that achieved a petaflop. Since 2010, he has been a senior researcher at the Barcelona Supercomputing Center, Spain. His current research interests include cost-effective interconnection networks and dataflow parallel programming languages. Also, he served as a reviewer of prestigious conferences in parallel computing (IPDPS, ICS, SC, ICPP, Europar, and HiPC). Currently, he serves as a member of the Program Committee of the International Conference on Parallel Processing. And he has just received recognition for the best innovative idea of 2011 in the Barcelona university campus.