A Fault-Tolerant Routing Strategy for KNS Topologies Based on Intermediate Nodes

Roberto Peñaranda[1*], María Engracia Gómez[1], Pedro López[1], Ernst Gunnar Gran[2], Tor Skeie[2]

[1] *Universitat Politècnica de València; Valencia, Spain.*

[2] *Simula Research Laboratory; P.O. Box 134, N-1325 Lysaker, Norway*

SUMMARY

Exascale computing systems are being built with thousands of nodes. The high number of components of these systems significantly increases the probability of failure. A key component for them is the interconnection network. If failures occur in the interconnection network, they may isolate a large fraction of the machine. For this reason, an efficient fault-tolerant mechanism is needed to keep the system interconnected, even in the presence of faults. A recently proposed topology for these large systems is the hybrid k-ary n-direct s-indirect (KNS) family that provides optimal performance and connectivity at a reduced hardware cost. This paper presents a fault-tolerant routing methodology for the KNS topology that degrades performance gracefully in presence of faults and tolerates a large number of faults without disabling any healthy computing node. In order to tolerate network failures, the methodology uses a simple mechanism. For any source-destination pair, if necessary, packets are forwarded to the destination node through a set of intermediate nodes (without being ejected from the network) with the aim of circumventing faults. The evaluation results shows that the proposed methodology tolerates a large number of faults. For instance, it is able to tolerate more than 99.5% of fault combinations when there are ten faults in a 3-D network with 1,000 nodes using only one intermediate node and more than 99.98% if two intermediate nodes are used. Furthermore, the methodology offers a gracious performance degradation. As an example, performance degrades only by 1% for a 2-D network with 1,024 nodes and 1% faulty links.

To be published in CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE

KEY WORDS: fault-tolerant routing; exascale computing; hybrid topology; KNS topology

1. INTRODUCTION

The size of large supercomputers has been growing year after year. The topmost machines of the top 500 supercomputer list [1] are being built up by hundreds of thousands of processing nodes. For instance, the current (June 2016) number one has 10,649,600 computing cores comprising 40,960 nodes. All these processing nodes work jointly to solve a given problem as fast as possible. A high-speed interconnect among nodes allows running processes to communicate.

The large amount of hardware that can be found in the interconnection network of large high–performance machines significantly impacts the probability of having a fault in the system. Each component may independently fail, and therefore, the probability of having a single fault in the whole system drastically raises with the number of elements that compose it. Therefore, it is extremely important that systems can keep running despite the presence of several failures in the network.

A trivial alternative that has been followed in some proposals is to replicate network components, and use these additional elements as spare parts. The main problem of this alternative is that it significantly increases the cost of the network. Taking into account that interconnection network topologies usually provide different alternative paths between source–destination node pairs, it is possible to modify the routing algorithm to circumvent failures. Hence, destination nodes can be reached using alternative paths not affected by failures.

In [2] we presented a routing algorithm for the recently proposed KNS topology [3] that is able to tolerate multiple faults with minimal performance degradation in presence of failures. In this paper, we extend this previous work, providing a more detailed description of the mechanism, including implementation details and presenting new evaluation results.

*Correspondence to: Universitat Politècnica de València; Valencia, Spain. Email: ropeaceb@gap.upv.es

The rest of the paper is organized as follows. Section 2 briefly describes different fault tolerant algorithms previously proposed for other topologies. In Section 3, we describe KNS, the hybrid (direct-indirect) topology the proposed mechanism has been implemented for. In Section 4, we present the fault-tolerant methodology proposed in this paper that is based on using intermediate nodes. Section 5 evaluates different configurations of the new routing algorithm. Finally, in Section 6 some conclusions are drawn.

2. RELATED WORK

In this section, we will give some background on fault-tolerant routing algorithms. We will focus on direct topologies because of the similarity with the KNS topology.

There are two different categories of fault tolerant techniques that rely on the routing algorithm. The first category reconfigures the routing tables when a failure occurs. In this case, routing tables should be updated according to the new topology that results after the failure [4, 5, 6, 7]. This technique allows the network to tolerate any number of faults without requiring extra resources [8] as long as the network is still connected, thanks to its flexibility. However, this flexibility may kill performance due to the need of using topology agnostic routing algorithms [9], as the resulting network topology may become irregular. Irregular topologies do not consider and, hence, do not take advantage of the specific characteristics of the topology, thus they often provide inferior performance.

On the other hand, the second category covers fault-tolerant routing algorithms. A large number of fault-tolerant routing algorithms for interconnection networks have been proposed in the literature, and specially for direct network topologies like tori and meshes. Some of them require adding resources (virtual channels), usually depending on the number of tolerated faults [10] or the number of dimensions of the topology [11]. Other routing algorithms are based on disabling faulty regions [12, 13, 14, 15, 16] or individual nodes [17, 18, 19] to route the packets around these faulty regions. However, to do this, these algorithms usually disable healthy nodes. In [20], the authors use the technique of Valiant routing [21] to implement an algorithm that uses intermediate nodes to avoid

faults. This methodology requires a few virtual channels and does not disable healthy nodes. There are other routing algorithms that do not require extra virtual channels, like [22, 23] or [24]. The first two methods are able to tolerate only a few faults for low dimension meshes, while the third method can offer more tolerance against faults, but needs extra virtual channels in tori. These routing algorithms were specifically designed for meshes and tori and they provide bad traffic balance as a lot of traffic is directed towards a single link, which can be easily saturated, thus degrading interconnection network performance.

3. THE k-ARY n-DIRECT s-INDIRECT (KNS) TOPOLOGY

The topology of the interconnection network defines the connection pattern among its nodes. Topologies usually adopt a regular structure to simplify their implementation and the routing algorithm. Among the different taxonomies of regular topologies, the most commonly-used one divides them into direct and indirect topologies [25, 26].

Direct topologies usually adopt an orthogonal structure where nodes are organized in an n–dimensional space, and each processing node has an associated router. The nodes are connected in each dimension in a ring (torus) or array (meshes) fashion. 2-D or 3-D direct topologies are relatively easy to built as each topology dimension is mapped to a physical dimension. However, implementing direct topologies with more than three dimensions implies not only increasing its wiring complexity, but also the length of its links when they are mapped to the 3-D physical space. Indeed, the number of ports of the routers geometrically grows with the number of dimensions of the topology (as two ports per dimension are required). Therefore, for large topologies, the implementation limitation in the number of dimensions leads to an increase in the number of nodes per dimension, which increases the communication latency, negatively impacting performance.

The alternative is to use an indirect topology. The main difference, compared to direct topologies, is that not all the routers have an associated processing node. The most common indirect topologies are multistage indirect networks (MINs), where switches are organized as a set of stages. For a large number of nodes, indirect topologies provide better performance than direct ones. However,

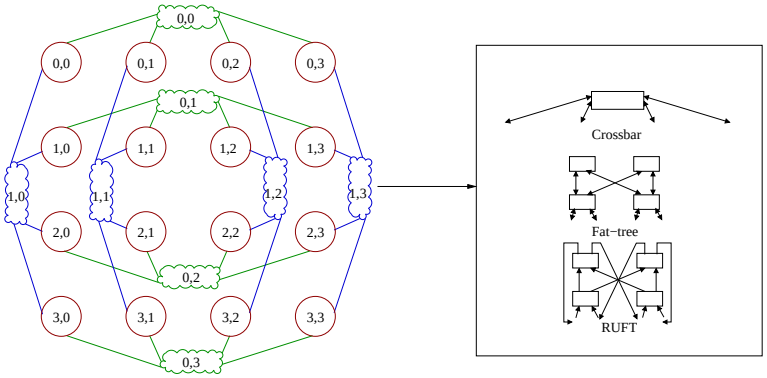


Figure 1. An example of the KNS topology with $n = 2$ and $k = 4$.

this is achieved by using a higher amount of switches and links. Furthermore, their physical implementation may become very complex due to the fact that the wiring complexity grows with the number of nodes in the system, unlike direct topologies where complexity grows with the number of topology dimensions.

To overcome the limitations of direct and indirect topologies, hierarchical and hybrid topologies have been proposed. In [27], the authors propose the Flattened Butterfly, a variation of the butterfly topology obtained by using high-radix switches, that results in a direct topology. This topology can be seen as a generalized hypercube with concentration (i.e., several processing nodes are attached to every router), as all the nodes in the same dimension are fully connected (i.e., there is a link from each node to all the others of the same dimension). As an extension of this topology, the Dragonfly topology was proposed in [28], which provides a hierarchical topology that is based on grouping routers in virtual ones to increase the effective radix of the network. This topology uses two different networks, one intra–group and one inter–group. There are global channels that link different groups. It is advisable to use a non-minimal global adaptive routing to balance the load among the global channels. However, global channels are long links so that a high latency can be expected.

To solve the limitations of previous topologies, the k-ary n-direct s-indirect (KNS) was recently proposed [3]. Like meshes and tori, it is organized as an n–dimensional topology, but the rings or arrays that connect the nodes in each dimension are replaced by small indirect networks. In this way, communication latency along each dimension no longer linearly grows with the number of nodes

per dimension. On the other hand, the small size of this indirect topology allows a reasonable wiring complexity opposite to large indirect topologies. This combination results in a family of topologies that provides high performance, with latency and throughput figures of merit close to the ones obtained with indirect topologies, but at a reduced hardware cost. The KNS topology is defined by three parameters: the number of network dimensions n, the number of nodes per dimension k, and the number of stages of the indirect subnetworks s that interconnect the nodes of each dimension. The total number of processing nodes is given by $N = k^n$.

Figure 1 shows an example of the KNS topology, with 2 dimensions and 4 nodes per dimension. Several solutions are feasible to connect the nodes of the same dimension. The simplest option is using a crossbar (i.e. a single switch, leading to a k-ary n-direct 1-indirect) to connect them, but a multistage topology like a fat-tree or a RUFT topology [29] can be used for a high number of nodes per dimension. Finally, any number of processing nodes can be attached to every network node (i.e., concentration). In KNS, we also refer to network nodes as routers, as opposed to the switches of the indirect subnets. In this paper, we focused only on k-ary n-direct 1-indirect topologies, although the proposed routing algorithm can be implemented in any KNS topology configuration.

4. DESCRIPTION OF THE FAULT-TOLERANT ROUTING METHODOLOGY

As stated above, in what follows, we will assume a KNS topology using crossbars as indirect subnetworks. Concerning routing, Hybrid-DOR [3] is used. It is a deterministic routing algorithm that crosses network dimensions in increasing order.

Hybrid-DOR works much like DOR (Dimension Order Routing) [26] for direct topologies. When a packet is injected, the router connected to the source node computes and compares the coordinates of the source (i.e., $s_{n-1} \dots s_1 s_0$) and destination (i.e., $d_{n-1} \dots d_1 d_0$) nodes in every dimension, and it sends the packet through the link that corresponds to the first dimension (i.e, f) the packet has to cross. Once the packet reaches the indirect subnetwork that corresponds to the crossed dimension, the packet is routed to reach the node of the same dimension that has as coordinate in this dimension the one corresponding to the destination node (i.e., d_f). How to route the packet in the subnetwork

depends on the subnetwork topology. If a single switch is used as an indirect subnetwork, the routing algorithm merely selects the output link which reaches the next router. If other topologies are used to implement the indirect subnetworks, any deadlock-free routing algorithm suitable for these topologies could be used. Notice that this is a local routing algorithm of the indirect subnetwork. This process is repeated for each dimension where the coordinates of the source and destination nodes are different. An increasing order is followed to ensure deadlock freedom. Notice that this routing algorithm is minimal and does not require virtual channels.

Let us analyze how to deal with network faults. We will only consider link faults, since a switch fault can be easily modeled as a switch with failures in all of its links. Moreover, as network links are bidirectional, we assume that if there is a link fault, then it fails in both directions. In this paper, we do not focus on how the failure information propagates to network nodes. In this way, a static fault model is assumed. This means that when a fault is discovered all the processes are stopped, the network is emptied, and a management application is run in order to deal with the fault. Checkpointing techniques must also be used so that applications can be brought back to a consistent state prior to the fault occurred. Detection of faults, checkpointing, and distribution of routing info is assumed to be performed as part of the static fault model, and are therefore not further discussed in this paper.

For each source-destination pair without failures in their path, packets are routed using Hybrid-DOR following minimal paths. But, if there is any fault in the path of a given source-destination pair, the methodology routes packets through intermediate nodes, like in [20]. The use of intermediate nodes was proposed in [21] for other purposes, such as traffic balancing. In our case, the idea is to avoid the faults by deviating the packet to an intermediate node. Therefore, a suitable intermediate node for each source-destination pair with faults in their path needs to be selected. The routing algorithm avoids faults by first sending the packet to an intermediate node and then, from this intermediate node, to the destination node. Several intermediate nodes could be also used for each pair of nodes, where packets are forwarded through these nodes until the destination node is reached. Notice that the Hybrid-DOR algorithm is used in all sub-paths. Using more than one intermediate

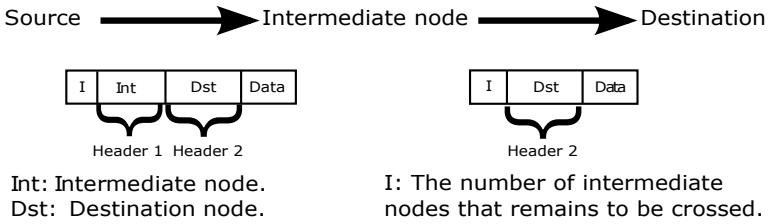


Figure 2. Header for packets using the intermediate node methodology.

node allows the mechanism to tolerate more faults by having more control over the global path followed by the packet. Notice also that the packets are not ejected from the network when they reach intermediate nodes. In Sections 4.1 and 4.2 we will describe in detail how intermediate nodes are selected for the case of requiring only one, or several of them, respectively.

Regarding the structure of packets, several fields should be added to the packet header to support routing through intermediate nodes. Figure 2 shows the packet header. First, the number of intermediate nodes that the packet has to cross is stored in a new field (I in the figure). In addition to the destination node, the addresses of intermediate nodes are also stored in the packet header. Every time an intermediate node is reached, its address is removed from the packet header and the I field is decreased. Other implementations are possible, such as storing in the packet header a pointer to the field that should be considered for routing. As soon as the packet reaches an intermediate node, this field will point to the next intermediate one or, finally, to the destination node.

For each source–destination pair, the mechanism checks whether the deterministic Hybrid-DOR path is fault-free or not. If not, routing through intermediate nodes is required. A list of intermediate nodes should be computed for paths with faults. This list will be stored in a table at every source node. There is a table entry for each destination node that requires routing through intermediate nodes. This table can be implemented as linear (i.e., with as many entries as the network size) or as random (i.e., as content-addressable memories) tables. The size of the latter table depends on the number of faults the network has to tolerate. The higher this number, the higher the number of affected source–destination pairs. In practice, though, it is expected that it should be enough to

tolerate a relatively low number of faults, as if a network suffers a high number of faults, the problem should be solved in another way.

However, the size of these tables also depends on how we codify the info stored in them. For example, considering a 2-D network, if the fault is located at the first dimension link of a node (i.e. the one that connects to the nodes of the same row), each source node in the same row (except the one connected to the faulty link) requires routing through intermediate nodes to reach every other node located in the same column of the fault. However, in the same example, the source node that is connected with the faulty link needs an intermediate node for each destination node in every other columns. This means that $k * (k-1)$ destination nodes need an intermediate node to reach them from this source node. Generally speaking, for more network dimensions, if a fault occurs at a link of the i dimension, the source node requires using intermediate nodes to reach $(k^{(n-i-1)}) * (k-1)$ destinations. Moreover, if there are more faulty links in this source node, each fault will involve a number of destinations according to this equation. Therefore, a table design where there is an entry for each destination that needs an intermediate node could require a big size in large networks. In such cases, an alternative design based on the use of masks, similar to IP routing tables, could be used. The table would have two fields (id to compare and mask to select the bits) and an option flag. The option flag indicates whether the corresponding entry is considered when the comparison is satisfied or not. Of course, there is also a field to store the possible intermediate nodes. In general, each entry can be used for a set of destinations, thus reducing the size of the table. However, some destination nodes may need specific intermediate nodes. Additional entries for them will be included in the table, which will lead to several hits for the same destination node. The solution is to insert the entries in the table following a given priority order and then selecting the entry with higher priority.

As in [20], we will refer to the source node as S and the destination node as D. For each intermediate node, we will use the notation I_x, where x represents the index of the intermediate node (I_1 for the first one, I_2 for the second one and so on). To ensure deadlock freedom, the routing algorithm needs at least as many additional virtual channels as intermediate nodes for fault-tolerant routing. For example, if we use up to two intermediate nodes, we need at least three virtual channels

(i.e., the original plus two additional ones). When a packet reaches an intermediate node, the packet is re-injected into the network using a new virtual channel to avoid deadlocks. For instance, assume that two intermediate nodes are used. One virtual channel (v_1) is used from S to I_1, another one (v_2) from I_1 to I_2, and the last one (v_3) from I_2 to D. In this way, deadlocks are avoided because the network is split into three virtual networks, deadlock-free routing algorithm is used within each virtual network, and each virtual network transition is performed following a strict order ($v_1 \rightarrow v_2 \rightarrow v_3$ in the example). Adaptive routing could also be used. In this case, several virtual channels can be used for adaptive routing provided that there is an escape channel to break cyclic dependencies for each subpath [30]. Routing in the escape channels uses Hybrid-DOR. Adaptive channels can be used in any of the sub-paths. However, in such a case, a different escape channel is required in each sub-path to ensure deadlock freedom. In this paper, though, we only focus on deterministic routing.

There are some combinations of failures that physically disconnect one or more nodes of the network. As the proposed methodology does not add new resources to the network, these sets of faults can not be supported. The aim of the proposed methodology is to provide a path for every source–destination pair, provided that they are physically connected by the interconnection network.

Next, we will present how the intermediate nodes are selected. First, we will focus on using only one intermediate node. After that, we show how to extend the methodology to use multiple intermediate nodes.

4.1. One Intermediate Node

In this case, we will use only one intermediate node when one or more faults affect the minimal path provided by Hybrid-DOR between a pair of nodes. This intermediate node, I_1, has to satisfy two rules:

R1. I_1 is reachable from S.

R2. D is reachable from I_1.

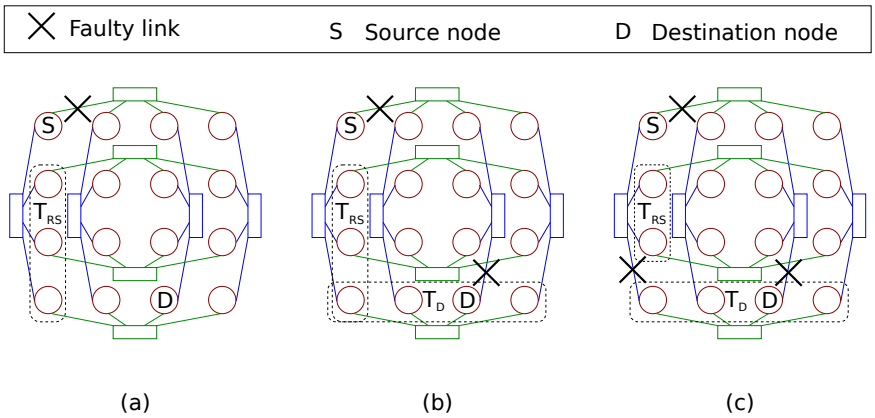


Figure 3. Examples of fault combinations in a KNS topology with $n = 2$ and $k = 4$.

We say that a node Y is reachable from X if there is a minimal path provided by Hybrid-DOR between this pair of nodes, and there is no fault in it.

For direct topologies like tori or meshes, the choice of this intermediate node is very important, because the number of hops can greatly increase, depending on the selected intermediate node. However, in the KNS topology, the number of hops only depends on the number of dimensions that the packet must cross. The proposed methodology prioritizes the use of those intermediate nodes which allow the packet to reach the destination node without additional hops compared to the original minimal path between the source and the destination nodes.

For instance, Figure 3.(a) shows a 4-ary 2-direct 1-indirect network with a fault at the source node S in the x dimension link. So, it cannot reach the destination node D using Hybrid-DOR. In this case, the possible intermediate nodes are all the nodes of the same column (surrounded by the dashed line in the figure). The best choice, however, is the node that is located in the same row as the destination node, because it allows to reach the destination node by using a minimal path.

We say that a fault-tolerant routing algorithm is able to tolerate f failures if it can provide a valid path between every source-destination pair for any combination of up to f failures.

Lemma 1

Given a KNS network with n dimensions, using one intermediate node, the routing algorithm can tolerate up to $n - 1$ failures.

Proof

For a KNS network with n dimensions, each node has n links. Each link allows connecting to the other nodes of the same dimension by an indirect subnetwork. Let T_{RS} be the set of nodes reachable from S using Hybrid-DOR, and T_D the set of nodes from which D is reachable using Hybrid-DOR. As long as $T_{RS} \cap T_D$ is not empty for any source-destination pair, the network is able to handle the fault combination by using one intermediate node located in this set. Therefore, the worst fault combinations are the ones that reduce the number of nodes in T_{RS} and/or T_D set.

For instance, in Figure 3.(a), while T_D set comprises all network nodes (except S), the fault reduces T_{RS} to only 3 nodes. $T_{RS} \cap T_D$ cardinal is 3 and therefore the fault is tolerated. However, assume that, then, there is another link failure that reduces T_D. The worst scenario occurs when the fault is located at the Y dimension link of D (Figure 3.(b)). In this case, $T_{RS} \cap T_D$ is reduced to one node, and the fault is still tolerated. However, if a new failure appears in the links of this unique intermediate node (thus, there will be 3 faults in the network), the source–destination pair S–D will become disconnected. Remember that this example corresponds to one intermediate node. This situation can be seen in Figure 3.(c).

However, the worst scenario occurs when the source and the destination nodes are located in the same row, i.e., when all coordinates but the one of the first dimension are the same for both the source and destination nodes. In this case, with only two faults (the first two faults of the previous example: a link failure at the x link of the source node and another one at the y link of the destination node), the destination node is no longer reachable from the source node, and $T_{RS} \cap T_D$ is empty (see Figure 4). This means that the network is able to tolerate all the fault combinations of 1 fault, but neither of 2 faults, and therefore, the network supports one fault. Additionally, there are also 2-fault combinations that physically disconnect nodes (i.e. a failure in all links of a node) and therefore they are not reachable, and not all the source-destination pairs are able to communicate. These cases are not tolerated.

In general, for any n-dimensional KNS network, there is a non-tolerated scenario that happens when the source and the destination nodes have the same coordinates but the coordinate of the

first dimension. If T_D is reduced due to faults on all links of the destination node but the one of the first dimension (which physically disconnects the node) and there is a fault on the link of the first dimension of the source node, both nodes become disconnected as there is not any suitable intermediate node.

Therefore, the minimal number of faults needed to disconnect the network with only one intermediate node is n faults ($n-1$ failures at the destination node links plus one failure at the source). Thus, the routing algorithm is able to tolerate $n-1$ failures. □

4.2. Multiple Intermediate Nodes

There are cases where only one intermediate node is not enough to handle the network fault combination. In these cases, the routing algorithm can use more than one intermediate node to have more chances of finding a set of fault-free deterministic Hybrid-DOR paths between the source and destination nodes. Assume that a number of x intermediate nodes $I_1, I_2, \ldots I_x$ are required. Intermediate nodes are selected according to the following rules:

R1. I_1 is reachable from S.

R2. I_{i+1} is reachable from I_i, for $0 < i < x$, $x > 1$.

R3. D is reachable from I_x.

Therefore, we can guarantee that the packet is able to reach its destination node following the path S-I_1-...-I_i-I_{i+1}-...-I_x-D.

In order to avoid non-minimal routing, the methodology tries to use a set of intermediate nodes that does not increase the number of hops beyond a minimal path. In particular, if there are non-minimal available paths using i intermediate nodes and also a minimal path using j intermediate nodes, where $i < j$, the latter will be finally selected.

Lemma 2

Given a KNS network with $n > 2$ dimensions, using two intermediate nodes, the routing algorithm can tolerate $2 * (n-1) + k - 3$ failures. If $n = 2$, the routing algorithm can tolerate $2 * k - 1$ faults.

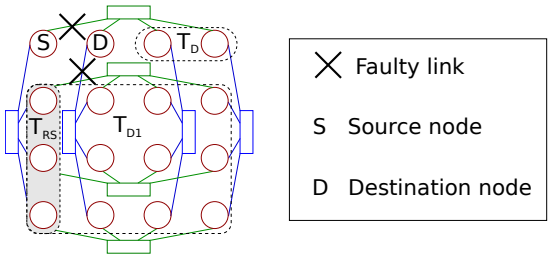


Figure 4. An example of the KNS topology with $n = 2$ and $k = 4$ and 2 faults.

Proof

Let T_{D1} be the set of nodes from which the destination node is reachable using Hybrid-DOR using one intermediate node. That is, the set of nodes that can reach the destination node using an intermediate node from T_D. In Figure 4 we can see an example for a 4-ary 2-direct 1-indirect network with 2 faults. The first failure is located at the source node x-link and the second one at the destination node y-link. The Figure shows the set of reachable nodes from the source node (T_{RS} with gray background), the set of nodes which can reach the destination node (T_D), and the set of nodes that can reach the destination node using any node within T_D as an intermediate node (T_{D1}). Thus, in this case, we have to use one node of $T_{RS} \cap T_{D1}$ as the first intermediate node, and another one of T_D as the second intermediate node. As long as $T_{RS} \cap T_{D1}$ is not empty for any source-destination pair, the network is able to handle the fault combination. Therefore, the worst fault combinations are those that reduce the number of nodes in T_{RS}, T_D and/or T_{D1}.

As in Lemma 1, if all the destination node links fail except the link of the first dimension (to avoid disconnecting the network), T_D will be reduced to $k - 1$ nodes. T_D includes the nodes which share all coordinates with the destination node, except the coordinate of the first dimension. On the other hand, if all links of the source node fail, except the one of the last dimension, T_{RS} will be reduced to $k - 1$ nodes. T_{RS} includes the nodes which share all coordinates with the source node except the coordinate of the last dimension. In this way, the possible second intermediate nodes, T_D, are only reached by the link of the last dimension. But if we assume that these links are also faulty, the destination node will be no longer reachable. So, with $n - 1$ faults at the source node links, $n - 1$ faults at the destination node links and $k - 1$ faults in the set of possible second intermediate nodes, the fault combination is not tolerated.

However, the worst scenario occurs when the source node shares all coordinates with the destination node, except the coordinate of the first dimension (see Figure 4). In this case, the destination node will not be reachable with only $k-2$ faulty links in the T_D set. Hence, the source–destination pair becomes disconnected with $2*(n-1)+k-2$ faults, ergo, the network can tolerate $2*(n-1)+k-3$ faults.

Notice, though, that for 2-D networks, adding faults for every possible second intermediate node (which comprises T_D) physically disconnects the row where the destination node is located at, as we can see in Figure 4. These corner cases are not tolerated by the methodology.

For 2-D topologies, the scenario is different. For instance, keeping the same number of faults at the destination node ($n-1=1$ fault) and at the source node ($n-1=1$ fault), and setting faults on the x-links of nodes in T_{RS} except one of them ($k-2$, see Figure 4), to avoid physically disconnecting the column, we have only one node in $T_{RS} \cap T_{D1}$, which will be the first intermediate node. From this first intermediate node, there are $k-2$ possible paths to the destination, i.e., one path for each possible second intermediate node which comprises T_D. If there are faults in all these paths, the destination node will not be reachable by the source node, i.e., $1+1+(k-2)+(k-2)=2*k-2$ faults are necessary to not tolerating the fault combination. Thus, the routing algorithm is able to tolerate $2*k-1$ faults in a 2-D KNS network. □

4.3. Extension To Any Indirect Subnetwork

In this paper, we have focused on KNS topologies that use crossbars as indirect subnetworks. However, we can extend the methodology to KNS topologies that use other indirect subnetworks like fat-trees or RUFT. To do this, a specifically designed methodology should also be used to tolerate faults on each indirect subnetwork. Therefore, intermediate nodes are used globally, while a specific methodology to tolerate faults will be used locally on each subnetwork.

While the subnetworks can avoid faults, the direct routers will work normally. However, if a node becomes unreachable due to a fault located at a given subnetwork, it will be modeled like a link fault at this node, in the link of the corresponding dimension of this faulty subnetwork.

5. EVALUATION RESULTS

To evaluate the proposed methodology, we have performed two kind of analysis. First, we analyze the number of network failures that can be tolerated. Remember that a fault-tolerant routing algorithm is able to tolerate n failures if it can provide a valid path between every source-destination pair for any combination of n failures. Notice that there are situations where the failures physically disconnect the network. We consider these situations as combinations where there is no path for all source-destination pairs and therefore the combination is not tolerated.

Second, we evaluate the network performance degradation in presence of faults when using the proposed methodology. To do this, we have simulated different tolerated network configurations with a varying number of faulty links under uniform traffic. For each number of faults, we have tested 50 random fault combinations to obtain the average network throughput and latency. In those experiments, the combinations where some nodes are physically disconnected are discarded and not simulated since only tolerated combinations are simulated.

We have analyzed the proposed methodology for two network configurations: a 32-ary 2-direct 1-indirect topology and a 10-ary 3-direct 1-indirect topology. Both configurations have a similar number of nodes (1,024 and 1,000 nodes, respectively), so we can analyze the impact of the number of dimensions on the behavior of the fault-tolerant routing algorithm proposed in this paper.

5.1. Simulation Model

To perform the simulations, we have used an event-driven simulator which models KNS topologies with bidirectional links. This simulator uses virtual cut–through switching. Each switch has a full crossbar with queues of 4 packets both at their input and output ports. Credits are used to implement the flow control mechanism. Packet length is 16-flit. We assume a pipelined router with a latency of 4 clock cycles, while the switch and link bandwidth is assumed to be one flit per clock cycle.

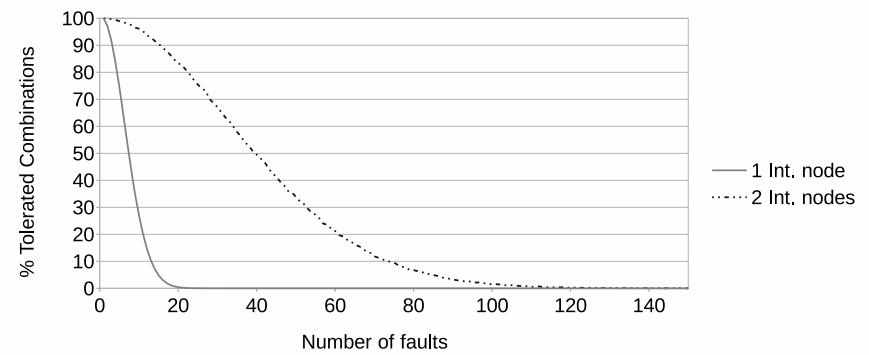


Figure 5. Fault combinations tolerated by the methodology when using one or two intermediate nodes in a 2-D network with 1,024 nodes.

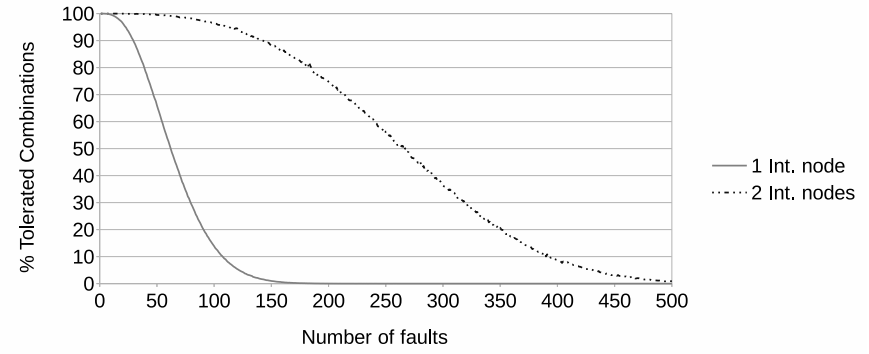


Figure 6. Fault combinations tolerated by the methodology when using one or two intermediate nodes in a 3-D network with 1,000 nodes.

5.2. Fault Analysis

The number of possible fault combinations exponentially increases with the number of faults. For this reason, it is not possible to explore all possible fault combinations for a given number of faults in a reasonable amount of time. Therefore, we have used statistical analysis as a tool. Specifically, we have analyzed a subset of the fault combinations, where the faults are randomly chosen. This subset is large enough to obtain results with a confidence level of 99% and an error lower than 1%.

Figures 5 and 6 show the percentage of tolerated fault combinations for different number of faults using one or two intermediate nodes for a 2-D network with 1,024 nodes and a 3-D network with 1,000 nodes, respectively. The results are shown for a number of link faults up to 150 for 2-D networks and 500 for 3-D networks. First, as expected, the percentage of tolerated combinations of faults strongly increases when using two intermediate nodes instead of only one because there is

Table I. Percentage of paths that use one or two intermediate nodes when at most two intermediate nodes are used: (a) 1,024-node 2-D network and (b) 1,000-node 3-D network.

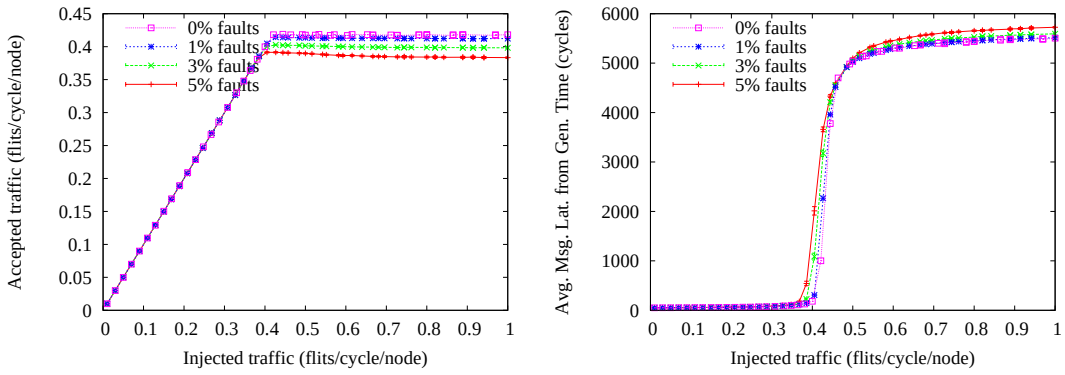
Link faults	% of paths	
	1 inter.	2 inter.
1	0.19%	0%
2	0.38%	0.000003%
3	0.57%	0.000009%
4	0.75%	0.00018%
5	0.94%	0.000031%
6	1.13%	0.000046%
7	1.32%	0.000067%
8	1.50%	0.000087%
9	1.69%	0.000115%
10	1.88%	0.000145%
11	2.06%	0.000179%
12	2.25%	0.000216%
13	2.43%	0.000257%
14	2.61%	0.000307%
15	2.80%	0.00036%

(a)

Link faults	% of paths	
	1 inter.	2 inter.
1	0.18%	0%
2	0.36%	0%
3	0.54%	<0.000001%
4	0.72%	<0.000001%
5	0.90%	<0.000001%
6	1.08%	<0.000001%
7	1.25%	<0.000001%
8	1.43%	<0.000001%
9	1.61%	<0.000001%
10	1.79%	<0.000001%
11	1.96%	<0.000001%
12	2.14%	0.000001%
13	2.32%	0.000002%
14	2.50%	0.000002%
15	2.67%	0.000002%

(b)

more control over the path followed by the packet. That is, using two intermediate nodes instead of only one provides more alternative paths or, what is the same, we have more options to configure the final path to the destination node, avoiding the faults. On the other hand, in the 2-D network, the percentage of tolerated combinations of faults is considerably lower than in the 3-D one because more alternative paths are available in the latter for each source-destination pair. In the case of the 3-D network, the methodology is able to tolerate more than 99.5% of the 10-fault combinations with only one intermediate node and more than 99.98% for the configurations of 15 faults with 2 intermediate nodes. The fact of having more dimensions gives more probablity to route the packet through different paths that do not share resources, being able to avoid more faults. However, the 3-D network has more resources. There are 3,000 links in the 3-D network versus 2,048 links in the



(a) (b)

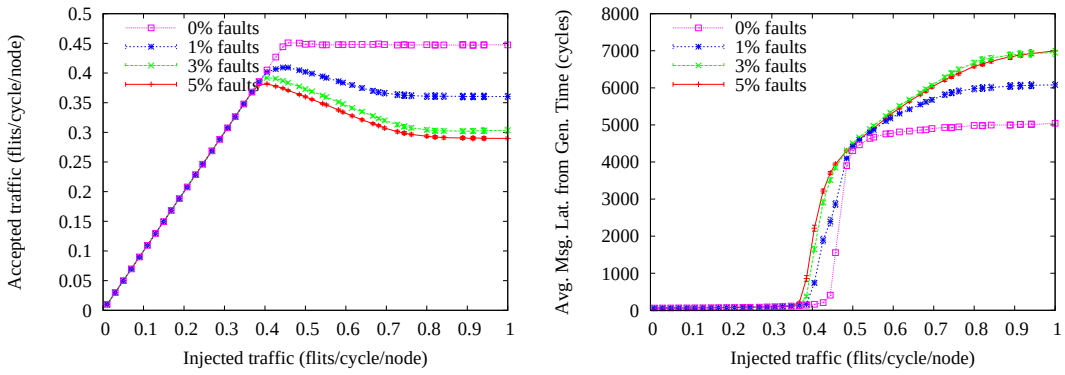
Figure 7. 32-ary 2-direct 1-indirect under uniform traffic: (a) Accepted traffic and (b) average latency versus injected traffic.

2-D network. This is why, for 23 faults in the 2-D network even with two intermediate nodes, the percentage of tolerated combinations is lower than 80%. However, in the 3-D network, more than 100 faults are needed to reach a percentage of tolerated combinations lower than 80%.

Tables I.(a) and I.(b) show the percentage of paths that have to use intermediate nodes to reach the destination node, and how many intermediate nodes are needed when up to two intermediate nodes can be used for a 2-D network with 1,024 nodes and a 3-D network with 1,000 nodes, respectively. These tables only show the information for up to 15 faulty links. Regarding the percentage of paths using intermediate nodes, we can see that it is quite low for one intermediate node, and much lower using a second intermediate node. Even for 15 faults, less than 3% of the paths use intermediate nodes. Therefore, it is expected that the extra latency due to the intermediate nodes does not significantly impact the final average latency, as the number of affected paths is extremely low. In the case of 3-D networks, the percentage of paths that use intermediate nodes is even lower.

5.3. Performance Analysis

In this Section we analyze the performance degradation suffered by the network when applying the fault-tolerant routing methodology in presence of faults. To do this, we have simulated several network scenarios with 1% faulty links, 3% faulty links and 5% faulty links. For each fault scenario,



(a) (b)

Figure 8. 10-ary 3-direct 1-indirect under uniform traffic: (a) Accepted traffic and (b) average latency versus injected traffic.

we have generated 50 random fault combinations, all of them tolerated by the methodology. That is, all source-destination pairs are able to communicate. Up to 2 intermediate nodes can be used, since this allows to test fault combinations with a higher number of faults. However, the fact of using two rather than only one intermediate node does not strongly impact network performance. This is because the methodology is able to avoid the fault using only one intermediate node in most of the cases, although it is able to use two (as shown in Table I.(a) and I.(b)).

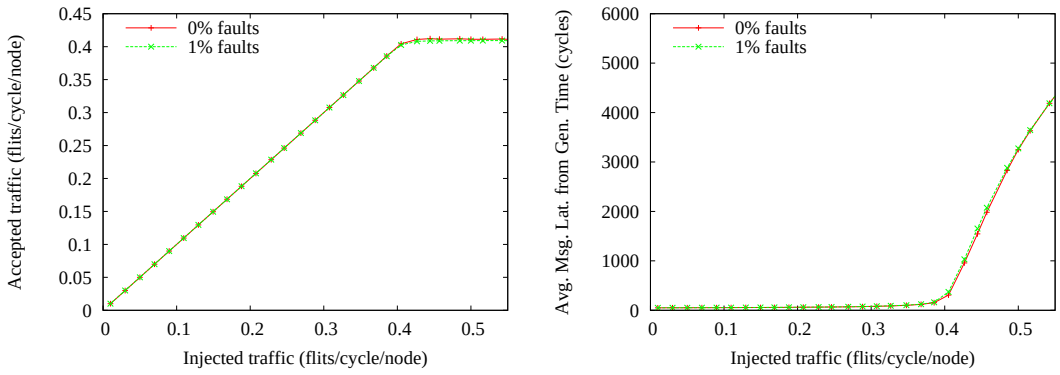
In order to measure the performance degradation, we obtained the accepted traffic and average message latency versus injected traffic. Accepted traffic is measured as the amount of data per node and per time that the network can accept (flits/cycle/node). Network throughput is the peak value of accepted traffic. Average message latency is measured as the mean of the elapsed time from message injection into the network at the source node until its ejection at the destination node. For network load, source nodes inject traffic following a uniform traffic pattern (i.e., randomly selecting the destination node).

Figures 7.(a) and 7.(b) show results for a 32-ary 2-direct 1-indirect network under uniform traffic. In this case, the network suffers a performance degradation of about 1% in throughput with 1% faulty links (21 links) compared to the same network without faults. For 3% and 5% faulty links, performance degradation increases to 3,8% and 6,5%, respectively. Latency is affected as well,

increasing the average value with respect to the fault-free case. In particular, at saturation, latency is increased by 67% with 1% faulty links.

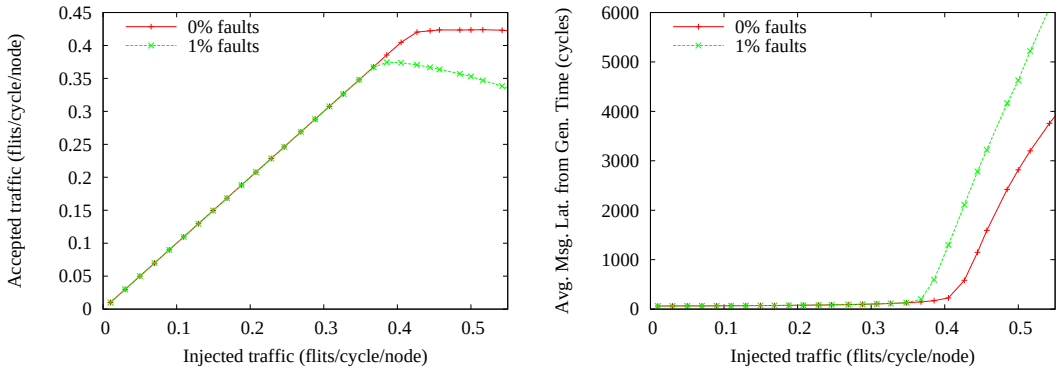
For the 10-ary 3-direct 1-indirect network (Figures 8.(a) and 8.(b)) the performance degradation, for the same percentage of faulty links, is higher when compared to the 32-ary 2-direct 1-indirect network. In particular, network throughput degrades by about 9% with 1% faulty links (30 faults) compared to the fault-free network, and by 13% and 15% with 3% and 5% faulty links, respectively. The increase in latency for the 1% link faults case, is 1.8 times at the saturation point. Notice that the increase in latencies is because the network with faults saturates before than the one without faults. If we focus on the base latency or the latency before saturation, we can see that there is almost no impact. This is because the use of intermediate nodes is only required for a low percentage of source–destination pairs (Tables I.(a) and I.(b)). Although both networks have roughly the same number of nodes, they have a different number of links and, for the same number of relative link faults, the 3-D network involves more faulty links. Therefore, more paths are affected. On the other hand, as shown in Figures 5 and 6, the fact of having a higher number of dimensions with the same number of nodes improves the probability of avoiding a given fault combination. Therefore, although throughput is degraded by 3.8% with 3% faulty links in the 2-D network versus a performance degradation of 13% with 3% faulty links in the 3-D network, the probability of supporting a combination with this number of faults in the 3-D network is about 97%, against only 16% in the 2-D network.

We can see the same behavior in Figures 9.(a), 9.(b), 10.(a), 10.(b), 11.(a) and 11.(b), where results for larger networks are shown. In this case, we consider 4,096-node networks and we only checked a number of faults equal to 1% of the links. Again, for 2-D networks, the performance is not affected to any large extent. On the other hand, for 3-D networks, the performance degradation is more intense, but remember that this network has a higher absolute number of faulty links for the same percentage of faults. Remember also that the higher the number of network dimensions, the higher the number of tolerated faults. Therefore, there is a trade-off. If we have a network with a very low probability of faults, or the faults can be repaired in a short period of time, we should



(a) (b)

Figure 9. 64-ary 2-direct 1-indirect under uniform traffic: (a) Accepted traffic and (b) average latency versus injected traffic.

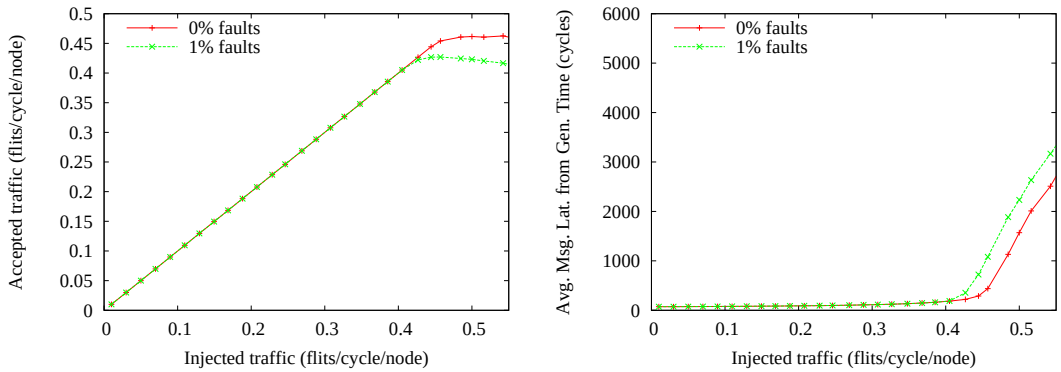


(a) (b)

Figure 10. 16-ary 3-direct 1-indirect under uniform traffic: (a) Accepted traffic and (b) average latency versus injected traffic.

choose a network configuration with a low number of dimensions. Otherwise, if the probability of faults is higher, network configurations with more dimensions would be a better choice.

To summarize, the proposed methodology offers a better tolerance of faults for networks with more dimensions, but this does not necessarily mean a better performance, because it depends on the fault combination and the selected intermediate nodes. With more dimensions, the selected intermediate nodes can increase the number of hops to a greater extent, as the distance between two nodes in a KNS topology increases with the number of dimensions.



(a) (b)

Figure 11. 8-ary 4-direct 1-indirect under uniform traffic: (a) Accepted traffic and (b) average latency versus injected traffic.

6. CONCLUSIONS

This paper proposes and evaluates a fault-tolerant routing algorithm for k-ary n-direct 1-indirect topologies. This new routing algorithm is based on the use of intermediate nodes assuming a static fault model. It is able to tolerate a large number of faults without suffering a great fall in performance. This routing algorithm does not disable any healthy node, unlike other algorithms, and does not require any additional resources except one extra virtual channel per each intermediate node used. The mechanism is able to tolerate 99.5% of 10-fault combinations with only one intermediate node and more than 99.98% for the configurations of 15 faults with 2 intermediate nodes in a 3-D network with 1,000 nodes. The proposed fault-tolerant routing algorithm has been evaluated by simulation under uniform traffic and the results show that the network performance only suffers a small degradation. For instance, using only two intermediate nodes (2 extra virtual channels), the evaluation results show a performance degradation of 1% for a 2-D network with 1024 nodes and 1% faulty links (21 faults). The proposed methodology can be easily extended to other configurations of the KNS network topology.

ACKNOWLEDGEMENTS

This work was supported by the Spanish Ministerio de Economía y Competitividad (MINECO), by FEDER

funds under Grant TIN2015-66972-C5-1-R, by Programa de Ayudas de Investigación y Desarrollo (PAID) from Universitat Politècnica de València and by the financial support of the FP7 HiPEAC Network of Excellence under grant agreement 287759.

REFERENCES

1. TOP500 Supercomputer Site. http://www.top500.org.
2. Peñaranda R, Gran EG, Skeie T, Gómez ME, López P. A New Fault-Tolerant Routing Methodology for KNS Topologies. *2016 2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, IEEE, 2016; 1–8, doi:10.1109/HIPINEB.2016.9. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7457761.
3. Peñaranda R, Requena CG, Gómez ME, López P, Duato J. A New Family of Hybrid Topologies for Large-Scale Interconnection Networks. *NCA*, IEEE Computer Society, 2012; 220–227.
4. Casado R, Bermúdez A, Duato J, Quiles FJ, Sánchez JL. A protocol for deadlock-free dynamic reconfiguration in high-speed local area networks. *Parallel and Distributed Systems, IEEE Transactions on* 2001; **12**(2):115–132.
5. Pinkston TM, Pang R, Duato J. Deadlock-free dynamic reconfiguration schemes for increased network dependability. *Parallel and Distributed Systems, IEEE Transactions on* 2003; **14**(8):780–794.
6. Lysne O, Pinkston TM, Duato J. A methodology for developing dynamic network reconfiguration processes. *Parallel Processing, 2003. Proceedings. 2003 International Conference on*, IEEE, 2003; 77–86.
7. Lysne O, Montañana JM, Pinkston TM, Duato J, Skeie T, Flich J. Simple deadlock-free dynamic network reconfiguration. *High Performance Computing-HiPC 2004*. Springer, 2005; 504–515.
8. Puente V, Gregorio JA, Vallejo F, Beivide R. Immunet: A Cheap and Robust Fault-Tolerant Packet Routing Mechanism. *ISCA*, IEEE Computer Society, 2004; 198–211.
9. Flich J, Skeie T, Mejia A, Lysne O, Lopez P, Robles A, Duato J, Koibuchi M, Rokicki T, Sancho JC. A Survey and Evaluation of Topology-Agnostic Deterministic Routing Algorithms. *IEEE Transactions on Parallel and Distributed Systems* 2012; **23**(3):405–425, doi:10.1109/TPDS.2011.190. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5953590.
10. Dally WJ, Aoki H. Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels. *IEEE Trans. Parallel Distrib. Syst.* 1993; **4**(4):466–475.
11. Linder DH, Harden JC. An Adaptive and Fault Tolerant Wormhole Routing Strategy for k-Ary n-Cubes. *IEEE Trans. Computers* 1991; **40**(1):2–12.
12. Boppana RV, Chalasani S. Fault-tolerant wormhole routing algorithms for mesh networks. *Computers, IEEE Transactions on* 1995; **44**(7):848–864.
13. Chien AA, Kim JH. Planar-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors. *ISCA*, Gottlieb A (ed.), ACM, 1992; 268–277.

14. Chalasani S, Boppana RV. Communication in multicomputers with nonconvex faults. *Computers, IEEE Transactions on* 1997; **46**(5):616–622.
15. Chen CL, Chiu GM. A Fault-Tolerant Routing Scheme for Meshes with Nonconvex Faults. *IEEE Trans. Parallel Distrib. Syst.* 2001; **12**(5):467–475.
16. Chalasani S, Boppana RV. Fault-tolerant wormhole routing in tori. *Proceedings of the 8th International Conference on Supercomputing*, ACM, 1994; 146–155.
17. Cunningham CM, Avresky DR. Fault-Tolerant Adaptive Routing for Two-Dimensional Meshes. *HPCA*, IEEE Computer Society, 1995; 122–131.
18. Glass CJ, Ni LM. The turn model for adaptive routing. *ACM SIGARCH Computer Architecture News*, vol. 20, ACM, 1992; 278–287.
19. Duato J. A Thory of Fault-Tolerant routing in Wormhole Networks. *ICPADS*, Ni LM (ed.), IEEE Computer Society, 1994; 600–607.
20. Gómez ME, Duato J, Flich J, López P, Robles A, Nordbotten NA, Lysne O, Skeie T. An Efficient Fault-Tolerant Routing Methodology for Meshes and Tori. *Computer Architecture Letters* 2004; **3**.
21. Valiant LG. A Scheme for Fast Parallel Communication. *SIAM J. Comput.* 1982; **11**(2):350–361.
22. Glass CJ, Ni LM. Fault-Tolerant Wormhole Routing in Meshes without Virtual Channels. *IEEE Trans. Parallel Distrib. Syst.* 1996; **7**(6):620–636.
23. Lysne O, Skeie T, Waadeland T. One-fault tolerance arid beyond in wormhole routed meshes. *Microprocessors and Microsystems* 1998; **21**(7):471–480.
24. Nordbotten NA, Skeie T. A routing methodology for dynamic fault tolerance in meshes and tori. *High Performance Computing–HiPC 2007*. Springer, 2007; 514–527.
25. Dally W, Towles B. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.
26. Duato J, Yalamanchili S, Lionel N. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc.: USA, 2002.
27. Kim J, Dally W, Abts D. Flattened butterfly: a cost-efficient topology for high-radix networks. *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, ACM: New York, NY, USA, 2007; 126–137, doi:10.1145/1250662.1250679.
28. Kim J, Dally W, Scott S, Abts D. Technology-Driven, Highly-Scalable Dragonfly Topology. *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, IEEE Computer Society: Washington, DC, USA, 2008; 77–88, doi:10.1109/ISCA.2008.19.
29. Gómez C, Gilabert F, Gómez M, López P, Duato J. RUFT: Simplifying the Fat-Tree Topology. *Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on*, 2008; 153–160, doi:10.1109/ICPADS.2008.44.
30. Duato J. A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-and-Forward Networks. *IEEE Transactions on Parallel and Distributed Systems* 1996; **7**:841–854, doi:10.1109/71.532115.