Bermudez Garzon, DF.; Gómez Requena, C.; Gómez Requena, ME.; López Rodríguez, PJ.; Duato Marín, JF. (2016). A Family of Fault-Tolerant Efficient Indirect Topologies. IEEE Transactions on Parallel and Distributed Systems. 27(4):927-940. doi:10.1109/TPDS.2015.2430863.

A Family of Fault-Tolerant Efficient Indirect Topologies

D. Bermúdez Garzón, C. Gómez, M.E. Gómez, P. López and J. Duato
Universitat Politècnica de València, Valencia, Spain

Abstract—On the one hand, performance and fault-tolerance of interconnection networks are key design issues for high performance computing (HPC) systems. On the other hand, cost should be also considered. Indirect topologies are often chosen in the design of HPC systems. Among them, the most commonly used topology is the fat-tree. In this work, we focus on getting the maximum benefits from the network resources by designing a simple indirect topology with very good performance and fault-tolerance properties, while keeping the hardware cost as low as possible. To do that, we propose some extensions to the fat-tree topology to take full advantage of the hardware resources consumed by the topology. In particular, we propose three new topologies with different properties in terms of cost, performance and fault-tolerance. All of them are able to achieve a similar or better performance results than the fat-tree, providing also a good level of fault-tolerance and, contrary to most of the available topologies, these proposals are able to tolerate also faults in the links that connect to end nodes.

Index Terms—Regular Indirect Topologies, Fat-Trees, Adaptive and Deterministic Routing, RUFT, Fault–Tolerance.

1 Introduction

NOWADAYS, large parallel computers have been or are being built with thousands of nodes [1]. In such large systems, performance, fault-tolerance, and cost of the interconnection network play key roles in the whole system design. The required levels of computing power can only be reached by increasing the number of nodes that compose them. As the system grows also does the amount of network resources and therefore the probability of a network fault. As the availability of these systems is a concern, fault-tolerance mechanisms are often implemented based on increasing the network resources and its cost.

The fat-tree, which is a bidirectional multistage indirect topology, is one of the most widely used topologies in large machines (see the Top500 list [1]) since it provides good performance and fault-tolerance levels, but it may have a high hardware cost. The RUFT topology (Reduced Unidirectional Fat Tree) [2] is a unidirectional Multistage Interconnection Network (MIN) that was proposed as a simpler alternative topology with a lower cost than the fat-tree. RUFT obtains similar performance results with much less hardware cost (approximately half of the fat-tree one). The weakest point of RUFT is that it does not offer any fault-tolerance support.

In this work we focus on devising new MIN topologies that improve the performance, fault-tolerance, and/or cost with respect to common MINs, by organizing the hardware resources differently and keeping the hardware cost as low as possible. Additionally, these topologies, contrary to other ones, are able to cope with faults in the links that connect the computing nodes to the network. Taking the RUFT topology as a starting point, and keeping a lower or similar hardware cost to the fat-tree topology, we propose several MINs with different cost, performance, and fault-tolerance. The goal is to obtain MIN topologies with good performance, fault-tolerance, and with a relative low hardware cost. First, a straightforward topology is proposed, referred to as RUFT-PL. It duplicates in parallel the links in RUFT to have a similar hardware cost to that of the fat-tree topology. This topology is able to double the RUFT and fat-tree performance and also provides some fault-tolerance. Another proposal, referred to as FT-RUFT, increases the fault-tolerance degree of RUFT-PL that only tolerates one fault in the injection, network, and ejection links, by doubling injection and ejection and providing as much disjoint paths as possible.

This proposal has two variants referred to as FT-RUFT-212 and FT-RUFT-222. The first one only duplicates the links in the injection and in the ejection, keeping the network links as in RUFT, providing fault-tolerance and increasing the performance at the cost of some few extra links with respect to RUFT. The second one is a topology that doubles the network links as RUFT-PL does, but using the same strategy of FT-RUFT-212 to connect the injection and ejection links, increasing in this way the fault-tolerance. FT-RUFT-222 provides a high degree of fault-tolerance and up to $2\times$ the performance offered by the fat-tree topology at a similar hardware cost.

The rest of the paper is organized as follows. Section 2 summarizes some related work, and Section 3 provides background on the fat-tree and RUFT topologies. Section 4 describes the three new fault-tolerant topologies, analyzing them in Section 5. Finally, some conclusions and future work are drawn.

2 Related Work

A large amount of literature has been devoted to provide fault-tolerance in MINs [3]–[23]. The fault-tolerance in these designs are mainly based on using additional hardware. To provide fault-tolerance, researches propose three hardware alternatives: *1*) replicate the entire network *2*) add extra stages, and/or *3*) adding chaining

links. For example, the authors of CSMIN [3] introduce a new interconnection model to improve the Gamma Network [4], obtaining only two disjoint paths, thus tolerating a single fault. In CGIN [5], the network uses an excessive interconnection hardware, tolerating only a single fault. The PCGIN [6] adds one link to the switches of the first stage to generate two disjoint paths between any source–destination pair.

Other strategies such as 3DON [7] and 3DGIM [8] use extra hardware to get three disjoint paths, just tolerating two faults in the network. Another major problem of these latter topologies is that in order to increase the fault-tolerance in the network, the source has to send two identical packets to the destination along two different paths, causing more packet contention and degrading network performance. In [9], the authors propose adding an extra subnetwork (group of switches) which is used only to provide redundancy and not to connect the processing nodes, highly increasing the cost of the network. Other strategies such as those proposed in [10]–[13] also make use of additional hardware in terms of switches and links to increase the number of alternative paths, either by adding more links between the switches of the same stage, more links between stages, creating extra stages, or even full additional networks.

The work proposed in [14] uses two parallel fat-trees with crossover links between the switches at the same position of each network to provide dynamic fault-tolerance, but at a high hardware cost. In [15], the authors use several parallel MINs to create redundancy without any interconnecting link between them, also increasing the hardware cost of the network. In [16] the authors analyze the fault-tolerance properties of MIN topologies without additional redundancy. In [17] the authors introduce a fault-tolerant routing methodology that sacrifices a certain number of healthy nodes in order to use no more than two virtual channels, and to reduce the routing time. In [18], different methods are analyzed to provide multiple paths in MINs, and the authors describe methods for fault identification and network reconfiguration. They found that to achieve a good computational performance it is necessary to eliminate nodes with poor connectivity in order to maintain high network throughput.

Other works, as described in [21], propose a simply and novel re-design of the links of the fat-tree topology to reduce the number of affected paths when a failure is detected in the network. However, as the original fat-tree, its topology does not support failures in the switches of the first stage. In [22], the authors propose an improvement to the IASEN topology [24] to increase the fault-tolerance level, obtaining up to 12 paths for each pair of nodes. However, the problem with this proposal is that the intermediate stage could become a bottleneck if one or more switches fails.

In [23], fault-tolerance in MINs is provided by increasing the total number of switches in the network, besides of requiring a high number of multiplexers and demultiplexers to connect all processing nodes to the network. On the other hand, this proposal only relies on networks with 2×2 switches, being not suitable for networks with switches of higher arity.

Finally, in Bidirectional MINs (BMINs), other approaches different to replication have been proposed, since unlike Unidirectional MINs (UMINs), BMINs provide alternative paths between source–destination pairs. For example, in [19], the authors propose a fault-tolerant routing mechanism that makes use of the alternative paths available in a fat-tree when faults are detected.

Opposite to these aforementioned approaches, with a moderate increase on the hardware cost, our proposals allow UMINs *1*) to tolerate faults and *2*) to enhance network performance in the absence of faults.

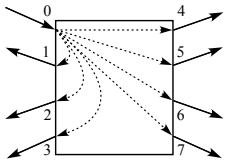
3 Background

3.1 Fat-Tree Topology

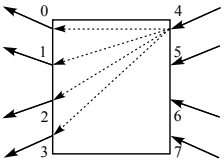
The fat-tree topology is based on a complete tree that thickens near the root. Switch arity increases as we go nearer to the root, which makes its implementation unfeasible. Hence, some alternative implementations have been proposed in order to use switches with fixed arity. In particular, the k-ary n-tree [25] is a parametric family of regular multistage topologies. The number of stages is n and k is the arity or the number of links of a switch that connects to the previous or to the next stage (i.e., the switch degree is $2k$). A k-ary n-tree is able to connect $N = k^n$ processing nodes using nk^{n-1} switches. Each processing node is represented as a n-tuple $\{0, 1, ..., k-1\}^n$, and each switch is defined as a pair $\langle s, o \rangle$, where s is the stage where the switch is located at, $s \in \{0..n-1\}$, and o is a $(n-1)$-tuple $\{0, 1, ..., k-1\}^{n-1}$ which identifies the switch inside the stage. Two switches $\langle s, o_{n-2}, ..., o_1, o_0 \rangle$ and $\langle s', o'_{n-2}, ..., o'_1, o'_0 \rangle$ are connected by an edge if $s' = s+1$ and $o_i = o'_i$ for all $i \neq s$. On the other hand, there is an edge between the switch $\langle 0, o_{n-2}, ..., o_1, o_0 \rangle$ and the processing node $p_{n-1}, ..., p_1, p_0$ if $o_i = p_{i+1}$ for all $i \in \{n-2, ..., 1, 0\}$. This edge is labeled with p_0. In what follows, we will assume that descending links are labeled from 0 to $k-1$, and ascending links from k to $2k-1$.

3.1.1 Adaptive Routing in Fat-trees

In k-ary n-trees, minimal routing between a source-destination pair can be accomplished by sending packets upwards from the source to any of their nearest common ancestors and then from there downwards to the destination. When crossing stages in the upwards direction, several paths are possible, thus providing adaptive routing. In fact, each switch can select any of its k up output ports. Once a nearest common ancestor has been reached, the packet is turned around and sent downwards. The stage up to which the packet must be forwarded is obtained by comparing the source and destination components beginning from the most significant one. The first pair of components that differs indicates the last stage to forward up the packet. Once in that stage, the descending path is deterministic. At each stage, the descending link to choose is indicated by the component corresponding to that stage in the destination n-tuple.



(a) Requested ports in the upwards direction from port 0.



(b) Requested ports in the downwards direction from port 4.

Fig. 1: Ports that can be requested in a 4-ary n-tree using adaptive routing.

Switch complexity of fat-trees can be easily computed considering that each switch has k bidirectional input and output ports, leading to $2k \times 2k = 4k^2$ switching elements. However, this rationale does not account for the actual requirements of switching activity of the routing algorithm. As can be observed, in the upwards subpath, at each switch, the k input ports can forward packets through either any of the up k output ports if the packet continues in its upwards subpath, or any of its down output ports if the packet starts its downwards subpath. On the other hand, in the downwards subpath, there are k input ports that can only request k down output ports, since once a packet has started its downwards subpath, the packet must continue going downwards. Figures 1a and 1b show theoutput ports that can be requested in the upwards and downwards directions, respectively, in the switches of a 4-ary n-tree.

A common way of implementing switches is by using as many multiplexers as the number of required output ports. Each multiplexer has a number of inputs equal to the number of input ports that can request the corresponding output port. In the switch we are considering (Figure 1), ports in the upwards direction require k multiplexers with k inputs each one or a $k \times k = k^2$ complexity. On the other hand, ports in the downwards direction require k multiplexers with $2k$ inputs each one or a $k \times 2k = 2k^2$ complexity. Total switch complexity can be easily obtained as the sum of the upwards and downwards directions complexities, leading to a switch complexity of $3k^2$ switching elements.

3.1.2 Deterministic Routing in Fat-trees (DESTRO)

Contrary to the previously presented routing algorithm, DESTRO [26] is deterministic, that is, in both subpaths there is only one path for each source–destination pair. This algorithm not only obtains good results but also it is able to highly reduce the switch complexity. The high performance achived is due to the appropriate selection of packet upwards subpaths which distributes destinations in a very effective way to highly reduce the Head-of-Line (HoL) blocking effect. As stated above, the packet downwards subpaths are determined by the upwards subpath followed by the packet and, with DESTRO, the interferences among different destinations in the packet downwards subpaths are completely eliminated. All the packets destined to a particular node are kept inside the same sub-tree (See Figure 2), and have a unique and exclusive down path. This is performed by using the

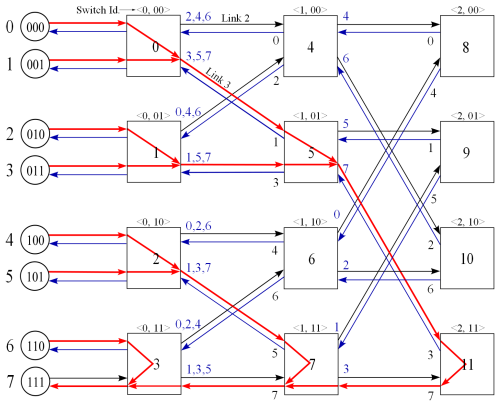


Fig. 2: Deterministic routing in a 2-ary 3-tree.

destination identifier to select one of the multiple available upwards subpaths. In DESTRO, the output port for routing a packet in a particular switch is given both by the destination identifier and the stage where the switch is located. In particular, it considers the component of the packet destination corresponding to that stage (i.e., a switch located at stage s considers the s^{th} component of the destination identifier, that is p_s). Therefore, at the switch $\langle s, o_{n-2}, ..., o_1, o_0 \rangle$, the selected output port for a packet with destination $p_{n-1}, ..., p_1, p_0$ will be $k + p_s$.

Figure 2 shows the destination node distribution in the ascending and descending links of a 2-ary 3-tree using DESTRO. In the first stage, the least significant component of the packet destination identifier (the least significant bit in this example) is used to select the ascending output port. At the second stage, the destinations of all packets that reach a given switch have the same least significant component. Hence, the component to consider in the selection of the up output port in this stage is the next one in the destination address. For instance, switch 4 is only reached by packets destined to nodes 0, 2, 4, and 6. These nodes have the same least significant component, which is 0. Among them, only packets destined to nodes 4 ($\langle 100 \rangle$) and 6 ($\langle 110 \rangle$) must be forwarded upwards. Packets destined to node 4 will select the first up link, and packets destined to node 6 the another one. Following this mechanism in all the upwards stages, finally, packets destined to a particular destination reach the same switch at the last stage and have a unique down subpath. Figure 2 highlights all the paths to node 7 and how all of them share the same downwards subpath.

By using DESTRO, the switch complexity can be highly reduced. The upwards switching activity is the same as in the fat-tree topology with adaptive routing. That is, each input port in the upwards subpath can request either any of the up output port, or any of the down output ports. However, in the down subpath, each link, input port and output port is used exclusively by packets sent to a unique destination. As a consequence, a given input port will always request the same output port, since all the packets that arrive to a particular input port, in their downwards subpath, are destined to the same node and are always forwarded to the same output port. This allows a noticeable reduction in switch

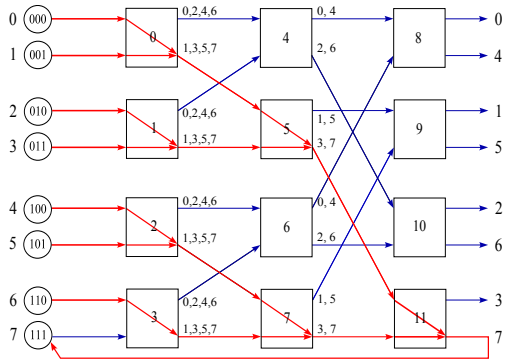


Fig. 3: A RUFT derived from the 2-ary 3-tree using DESTRO.

complexity. Using multiplexers to implement switches, ports in the upwards direction require k multiplexers, each with k inputs, or a $k \times k = k^2$ complexity, and ports in the downwards direction require k multiplexers with $k+1$ inputs (the k upwards ports plus the unique downwards one) or $k \times (k+1) = k^2 + k$ switching elements. Therefore the required switch complexity by DESTRO is $2k^2 + k$.

3.2 RUFT

The RUFT topology [2] is a simplification of the fat-tree topology obtained by taking advantage of the nice properties of DESTRO. In particular, since there is no switching activity in the downwards subpath, the switches are simplified by making them unidirectional. Therefore, the whole downwards subpaths are transformed in links that connect the last stage to the processing nodes (see Figure 3). Notice also that, as the topology is unidirectional, there is not chance to start the downwards subpath before reaching the last stage and, therefore, the paths are longer since all the packets must reach the last stage, contrary to the fat-tree topology where depending on the source–destination pair, different number of stages must be traversed.

Using RUFT, the switch complexity corresponds to the switch complexity of a unidirectional switch of k input ports and k output ports, where any of the k input ports can request any of the k output ports, so the switch complexity is k^2. As it can be seen, the switch complexity has been reduced more than twice when comparing it with DESTRO and three times when comparing with the fat-tree with adaptive routing.

4 The Proposal

As aforementioned, RUFT is a topology evolved from the fat-tree topology using DESTRO as routing algorithm. RUFT reduces the hardware cost compared to the fat-tree topology, maintaining a similar performance [2]. However, the weakest point of RUFT is that it does not provide fault-tolerance neither at injection/ejection links nor in the network links. Therefore, despite offering a good performance-cost tradeoff, the topology only provides a single path between each source destination pair.

To solve the lack of fault-tolerance of the RUFT topology, in this paper, we enhance this topology by adding some extra links. This results in a family of topologies that increases fault-tolerance and performance with respect to RUFT, but also with respect to the fat-tree topology, while having a hardware cost similar or lower than that of the fat-tree.

In particular, we propose three enhancements to RUFT. The first proposal, which is very straightforward, is referred to as RUFT-PL (RUFT with Parallel Links), and is based on duplicating the injection/ejection and the network links, also distributing the network traffic in a balanced manner to reduce the HoL blocking effect between these dual links. The second proposal, FT-RUFT-212, provides fault-tolerance by duplicating just the links that connect from/to end nodes and connecting them in a strategic way which entails a very small increase in hardware cost. FT-RUFT-212 uses the same amount of links as the RUFT topology, and the same connection pattern among switches. The last approach, FT-RUFT-222, strongly improves the fault-tolerance of RUFT-PL and the performance of FT-RUFT-212, by combining them.

The idea of duplicating injection and ejection links is not new. For instance, the Black Widow [27] uses four injection links, where the ports of each node have been reduced to half of the bandwidth to keep a low-cost design. Our proposal could use three or more links, but we propose to use only two links because we are interested in keeping the hardware cost as low as possible.

4.1 RUFT with Parallel Links

RUFT with Parallel Links (RUFT-PL) uses the same number of switches as the fat-tree and RUFT topologies, but RUFT-PL has the same number of switch ports as the fat-tree. Our proposal pursues to implement the RUFT topology but with a similar hardware cost to the fat-tree topology. As fat-tree switches have bidirectional ports, RUFT-PL switches can double the number of unidirectional ports of RUFT switches, leading to $2k$ input ports and $2k$ output ports. We propose to use the available k additional ports to have two parallel links connecting each pair of switches of the original RUFT topology (see Figure 4 for an example). The parallel links provide fault-tolerance and additional routing flexibility that can be exploited in several ways.

4.1.1 Parallel link selection

As we can see in Figure 4, each processing node is connected to the network through two links. To provide fault-tolerance in this point, packet injection is dynamically distributed, i.e., through the link that connects with the switch port with most free buffer.

From there, at a given switch, the pair of output channels (i.e. output ports) to be used is given by the destination component corresponding to the stage of the switch (as in RUFT and DESTRO). Figure 4 shows how the destination nodes are distributed among the output parallel links of switches, for a topology with $k = 2$ and $n = 3$. However, to select which parallel channel will be

used, we could consider two different criteria: *1*) select the link corresponding to the least significant bit of the next destination component, or 2) select the link with most free buffer.

The former approach isolates different destinations between both parallel links. In this way, the remaining HoL blocking that still appears in the upwards subpaths of the RUFT topology could be reduced even more. Moreover, this scheme simplifies even more the switch. Each input port of a switch can only request the k output channels associated to $k/2$ output ports, and the implementation just requires k multiplexers with $2k$ inputs, which leads to a switch complexity of $k \times 2k = 2k^2$. The drawback of this scheme is that it implies a static selection which does not allow implementing fault-tolerance. In addition, when the network is suffering some non-uniform traffic patterns, some links are unused.

On the other hand, by applying a dynamic selection function to perform the routing, we can increase up to two the number of paths for each source–destination pair, thus tolerating a single fault at the network. Therefore, we can always use both parallel channels per port regardless of what kind of traffic is used. However, in this case, switch complexity is a little higher since each input channel can ask for any of the up output ports, so the switch complexity is $2k \times 2k = 4k^2$ switching elements. On the other hand, as stated, the fault-tolerance in the network is increased.

As we can see, at the last stage, the end nodes are also connected to two parallel links. At this point, again, the output port with most free buffer is selected.

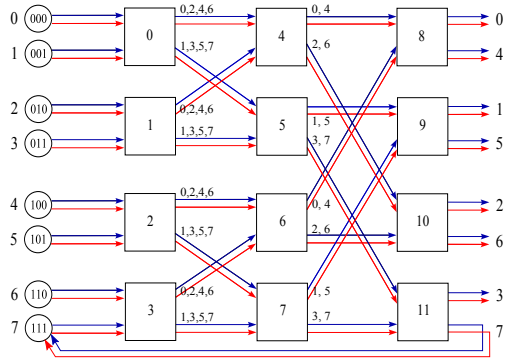


Fig. 4: A 2-ary 3-tree using RUFT-PL.

4.2 FT-RUFT-212

In this section, we present an alternative approach to provide fault-tolerance and increase performance of RUFT. The key point of this new proposal relies in the processing node connection to the network, that has been selected in a manner that allows achieving the most disjoint paths. Processing nodes can use two alternative paths through two different switches of the first stage. In particular, each node is connected to two different switches in such a way that each switch belongs to a different sub-tree in the network and provides completely disjoint paths for each source–destination pair and, therefore, fault-tolerance in the network in addition to tolerating also one fault in the injection.

Moreover, destination nodes have also dual connections in the last stage, which increases up to 4 the number of alternative paths for each source–destination pair. Destination nodes are connected also strategically in such a way that the alternative paths are also as disjoint as possible. This proposal is referred to as FT-RUFT-212 to indicate that it is a fault-tolerance variant of the RUFT topology in which there are 2 links connecting the source nodes to the network, only 1 link connecting the switches in the network, and destinations are connected to the network by 2 links. Notice that the switches of the first and last stages are asymmetrical with $2k \times k$ and $k \times 2k$ ports, respectively.

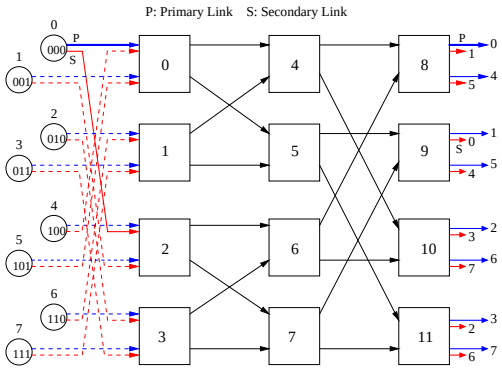


Fig. 5: A 2-ary 3-tree FT-RUFT-212.

Figure 5 shows the connections in a 2-ary 3-tree FT-RUFT-212. Every node is connected to the network through two links. The one that connects it to the same switch as RUFT will be referred to as the primary link (shown in blue in the figure), and it is obtained by dividing the node identifier by the arity of the network ($node\_id/k$), by 2 in this example. The alternative link that connects every node to the network will be referred to as the secondary link (shown in red in the figure). This alternative link is connected to a different switch. In particular, it is connected to the same switch as the node with the same identifier but inverting its most significant bit (i.e., $\langle 000 \rangle \rightarrow \langle 100 \rangle$). In the figure, the alternative link connects node 0 to switch 2, where node 4 is connected through its primary link. Following this criteria, we ensure to have double injection to two points of the network that provide two completely disjoint paths.

As stated, each node is also connected to two different switches of the last stage. This provides fault-tolerance at the ejection and additional fault-tolerance in the network by increasing the number of alternative paths. In particular, we connect each node to the switch where it is connected in RUFT (through the primary link, shown in blue in the figure) and, additionally, to the switch that connects to the node obtained by inverting its least significant bit (i.e., node 0 $\langle 000 \rangle$ is connected to switch 8 as in RUFT and additionally to switch 9, where node 1 $\langle 001 \rangle$ is connected through its primary link), which is referred to as the secondary link (shown in red in the figure). In this way, we obtain four alternative paths in the network, that is, two different paths per injection port.

4.2.1 Dual packet injection and ejection

As stated above, each node has two different injection links. In order to provide fault-tolerance and boost performance, the injection of packets is done through the link that connects with the switch port with more available space to allocate the packet. In this way, we can distribute the traffic load into two different network segments. If the two possible injection switch ports have the same buffer space, one of them is randomly selected. If there is a fault in the path to a destination through one of the injection links of a source node, the traffic to that destination will be forwarded through the other still healthy link.

Each switch in the last stage in this new topology has two ejection links. The double ejection allows to reach a given destination through the same path as in RUFT (i.e., through the primary ejection link) but also through the path that allows reaching the node with the LSB (least significant bit) inverted in RUFT (i.e., through the secondary ejection link). For example, to forward a packet to node 0 through the secondary ejection link, it must be routed as if its destination were node 1. The decision about which path is followed (the primary or the secondary one) is taken at the first stage, depending on the selected output port. In absence of faults, any selection function could be applied. However, to support faults, a dynamic selection function based on network status should be used. In this paper, we choose the ejection link in the first stage depending on the available space at the output ports buffers. The one with most free available slots is selected.

Although the way routing is performed in the first stage is changed with respect to RUFT, this is not the case of the rest of the switches of the network, which select the output port according to the remaining components of the packet destination (which are the same for both destinations, the original and the alternate one). When the packet reaches the last stage, it is forwarded either to the node connected by the primary or the secondary link, according to the least significant component of the destination.

4.2.2 Disjoint paths in the network

With the dual injection and ejection described above, we provide four different paths in the network for each source-destination pair, which are as disjoint as possible. This does not only allows fault-tolerance but also improves network performance, as analyzed in Section 5. As an example, Figure 6 shows the different paths that a packet could follow from source node 0 to destination node 7. In the absence of faults and following the mechanism previously described, the injection will be done through any of the injection links (red or blue links). At stage 1, depending on the utilization of the switch output buffers, the packet will either use the primary path (blue path) or the secondary path (red path). In case of faults in the network, injection or ejection links, the packet can be forwarded through any of the non-faulty available paths.

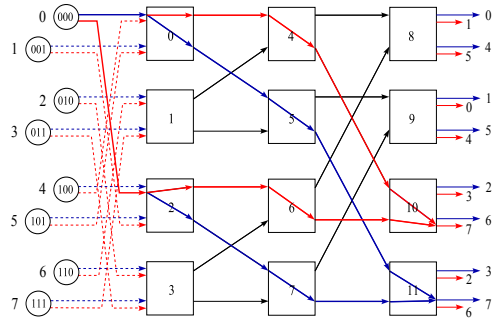


Fig. 6: Paths provided by FT-RUFT-212 from node 0 to node 7.

4.3 FT-RUFT-222

This section proposes a topology that combines the double injection and ejection of FT-RUFT-212 with the parallel network links as in RUFT-PL. As a result, symmetric and identical switches are used in all stages. This topology combines the performance benefits from RUFT-PL and the fault-tolerance from FT-RUFT-212. This topology is referred to as FT-RUFT-222 to indicate that it is a fault-tolerant variant of RUFT, that uses 2 links to connect the processing nodes to the network, 2 links for the switch interconnection, and 2 links to connect the last stage switches to the processing nodes. FT-RUFT-222 has double injection and ejection links as FT-RUFT-212, but all the switches in the network are $2k \times 2k$ (as in the fat-tree and RUFT-PL topologies). This new topology, as shown in Section 5, achieves good performance and increases the fault-tolerance with respect to the previous ones. Figure 7 shows an example of this new topology, a 2-ary 3-tree FT-RUFT-222.

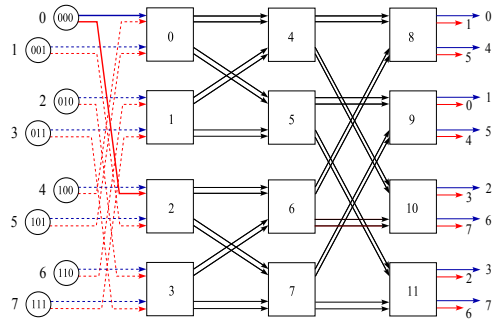


Fig. 7: A 2-ary 3-tree using FT-RUFT-222.

The greatest advantage offered by this topology is that, by having dual connection between all switches and the FT-RUFT-212 injection/ejection, we can increase the number of tolerated faults. As we have seen, FT-RUFT-212 offers four paths in the network, provided by the double injection/ejection. Now, FT-RUFT-222, thanks to the parallel network links, offers up to four paths per injection port, i.e., eight alternative paths in total, tolerating so up to seven faults in the network links.

On the other hand, in absence of faults, having parallel network links allows us to boost the number of paths in the routing algorithm to further increase performance. The strategy to inject traffic in the network is the same

as the ones used by FT-RUFT-212. Also, to distribute the traffic at the first stage we use the same criterion used in FT-RUFT-212 (i.e., selecting the output port with most buffer space) with the advantage that, in this new topology, we have four possible paths to send packets for each source–destination pair, thanks to the parallel links. From the second stage up to the last stage there is only an available pair of ports in each switch, which are also selected considering the output port buffer space availability. At the last stage, there is only one available port to deliver the packet.

Figure 8 shows all the available paths that a packet may follow from source node 0 and destination node 7.

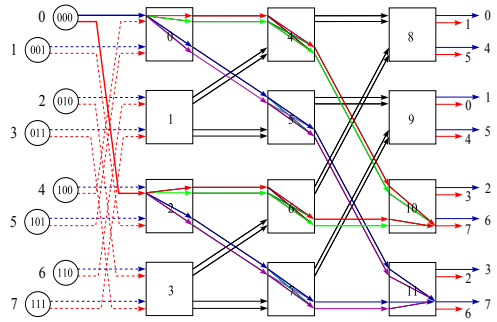


Fig. 8: Paths provided by FT-RUFT-222 from node 0 to node 7.

5 Evaluation

In this section, we evaluate the fault-tolerance, performance and cost of the three topologies proposed in this paper, comparing them against RUFT and the fat-tree topology with adaptive routing. The fat-tree is only evaluated with adaptive routing because is the one that can provide fault-tolerance. Anyway, a fat-tree with DESTRO obtains roughly the same performance as a fat-tree with adaptive routing.

Network performance is evaluated both without any fault in the network links and the performance degradation with faults in the network links.

Performance degradation of the proposed topologies is evaluated including a set of faults in the network links. For this analysis, we have considered a static fault model. In this model, once a fault is detected, the system activity is stopped, appropriate actions to handle the fault are taken, and then, the system activity is resumed. Notice that RUFT's performance degradation is not evaluated given that this topology does not support any fault.

5.1 Evaluation Methodology

To obtain the results of this section, two different tools have been developed, one for measuring the fault-tolerance of the network and the another one for performance analysis. For the fault-tolerance analysis, we have developed a tool that injects faults in the network and, for each fault combination, it calculates the number of paths still available between each source–destination pair in order to obtain the fault-tolerance degree of the topology. If the topology is able to provide at least one non-faulty path between every source and every destination for that combination of faults, then the combination of faults is tolerated. If there is at least a source–destination pair that does not have an available path for a fault combination, then the fault combination will not be tolerated. To analyze a high number of fault combinations and, for each combination, to check all source–destination pairs and all paths in the network, a high computation power is required. The tool has been implemented and run on a GPU coprocessor. For a network and a given number of faults, we tested all the possible fault combinations in the network, with a limit of 1 million of fault combinations in the largest networks due to the required execution time of the tool.

Timing Parameters	Cycles
Crossbar	1
Fly	1
Link	1
Long Fly	$Fly * stages + 1$
Routing	4

Other Parameters	Units
Virtual Channels	1
Buffer Size	2 Packets

TABLE 1: General parameters used in the simulations.

For performance evaluation, a detailed event-driven simulator that models several indirect virtual cut-through networks have been used. The basic timing configuration for the simulated networks is shown in Table 1. The "Long Fly" parameter specified in the table is the time to traverse the link from the last stage to the end nodes (this only applies for RUFT-like networks). In the fat-tree, packets are adaptively routed (with the FTA routing algorithm, see Section 3.1.1) and in RUFT, RUFT-PL, FT-RUFT-212, and FT-RUFT-222 packets are routed in a deterministic way as in RUFT (see Section 3.2) [2], with the variations presented in Section 4.

Several synthetic traffic patterns have been considered: uniform, hot-spot, complement, and perfect shuffle. Uniform is the most widely used traffic pattern, where each node randomly sends packets to all other nodes with the same probability for each one. Hot-spot traffic is also evaluated to model those cases where a considerable amount of the traffic is targeted to a specific endpoint. In this paper, we send 15% of overall traffic to a randomly selected hot-spot node. Complement traffic is a specific traffic pattern that stresses MIN topologies where each node sends traffic only to the destination that is obtained by inverting the bits of the source node (for instance, node 3 $\langle 011 \rangle$ will only send messages to node 4 $\langle 100 \rangle$). In this way, packets are forced to always reach the switches of the last stage of the network. Shuffle traffic is characterized by sending messages to a destination node whose bits correspond to the bits of the source but rotating to the left 1 bit (for example, node 3 $\langle 011 \rangle$ will only send messages to node 6 $\langle 110 \rangle$).

To evaluate the degradation of network performance in presence of faults, each topology has been simulated by injecting several faults in the network. These faults are static and permanent, and remain constant throug-

hout the entire evaluation.

The evaluation of the fat-tree topology has been performed using Flexible Interval Routing (FIR) with multiple exclusion intervals, to avoid victim nodes, as described in [19].

For each simulated number of faults and network, 50 fault combinations have been randomly selected to finally calculate the average throughput and get the average degradation of performance.Each fault combination considered in the performance evaluation is tolerated by the evaluated topologies, that is, there exist at least a path between each source-destination pair that has not been affected by any fault. Although only faults in the links are considered, a switch fault can be easily modelled as if all its links had failed.

5.2 Fault-tolerance Evaluation

For the fault-tolerance evaluation we show three different analysis. First, the maximum number of faults tolerated by each topology is shown in Table 2. For that number of faults, all the fault combinations are tolerated, which means that all the source–destination pairs are able to communicate for all the combinations of that number of faults. The second analysis (Figure 9) shows the percentage of non-tolerated fault combinations for each number of faults and each topology. The third analysis presents an alternative point of view by showing the percentage of source–destination pairs (Figure 10) that are able to communicate for each topology as the number of faults is increased.

Table 2 presents the maximum number of tolerated faults for each topology and several different arities. The table considers faults in both network and injection/ejection links. As it can be seen, RUFT cannot tolerate any fault, FTA can tolerate as many faults as the arity minus 1 in the network and none in the injection/ejection, RUFT-PL can tolerate 1 fault in the network and 1 fault in the injection/ejection links, FT-RUFT-212 can tolerate 3 network link faults and 1 fault in the injection/ejection links, and FT-RUFT-222 tolerates up to 7 network link faults and 1 fault in the injection/ejection links. In addition, FT-RUFT-212 and FT-RUFT-222, thanks to the disjoint injection/ejection, can also cope with faulty switches, i.e., where all its links have failed. RUFT-PL is not able to tolerate switch faults because all the alternative paths use the same switches. FTA does not tolerate faults in the first stage switches.

Arity	FTA	RUFT	RUFT-PL	FT-RUFT-212	FT-RUFT-222
2	1/0	0/0	1/1	3/1	7/1
4	3/0	0/0	1/1	3/1	7/1
8	7/0	0/0	1/1	3/1	7/1
16	15/0	0/0	1/1	3/1	7/1

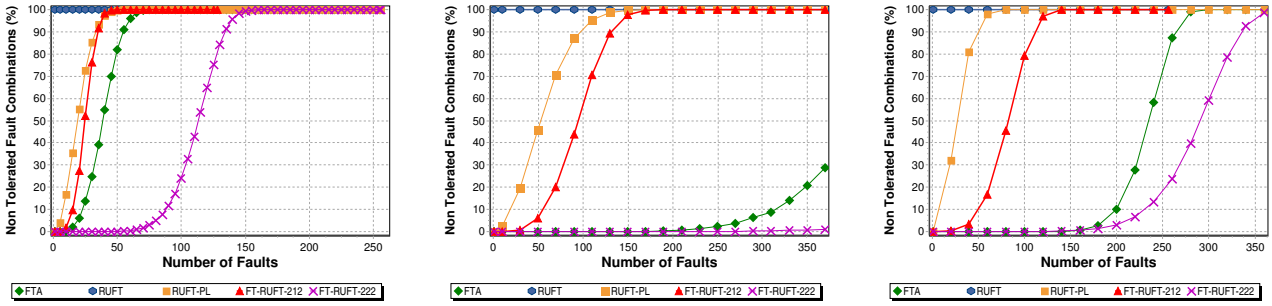
TABLE 2: Number of tolerated link faults at network links (left) and injection/ejection links (right).

Figure 9 presents the percentage of non-tolerated fault combinations at network links for three network sizes when increasing the number of faults in the network. Network sizes analyzed have been chosen to see the effects of changing the arity of the switches and the number of stages in the network. First, notice that RUFT does not tolerate any fault because this topology only provides one path for each source–destination pair, so, if this path has a fault, it cannot be avoided. Although RUFT-PL has the same number of links as FTA[1], the former only supports 1 fault since the parallel link can be used when a link fails, but there are no more alternative paths. When the arity is $k = 4$ (see Figure 9a), FTA tolerates a higher number of fault combinations than FT-RUFT-212. Although both tolerate the same absolute number of faults, FTA can route packets through more paths in the network than RUFT-212 and, as a consequence, it tolerates a higher percentage of fault combinations.

Concerning FT-RUFT-222, it is able to tolerate a considerably higher number of fault combinations than the other topologies. Thanks to the parallel network links and the disjoint injection/ejection scheme, it provides eight paths in the network for each source–destination pair, tolerating up to seven faults and also many more fault combinations for more than seven faults. For instance, in Figure 9a, FT-RUFT-222 can tolerate almost all combinations for up to 50 faults, and no other topology can tolerate any fault combination for such a number of faults. As we can see in Figures 9b and 9c, as the switch arity increases, the percentage of non tolerated fault combinations decreases considerably, being FT-RUFT-222 the topology that provides the best behavior against a high number of faults, but, in both scenarios, FTA also considerably increases its percentage of tolerated fault combinations. The key point here is that, when injection and ejection links are considered, neither FTA nor RUFT tolerate any fault, while both RUFT-PL and FT-RUFT variants are able to tolerate one fault in these links. For this analysis, only network links have been considered. The reason why the injection/ejection links are not evaluated is because FTA does not support any fault in these edges.

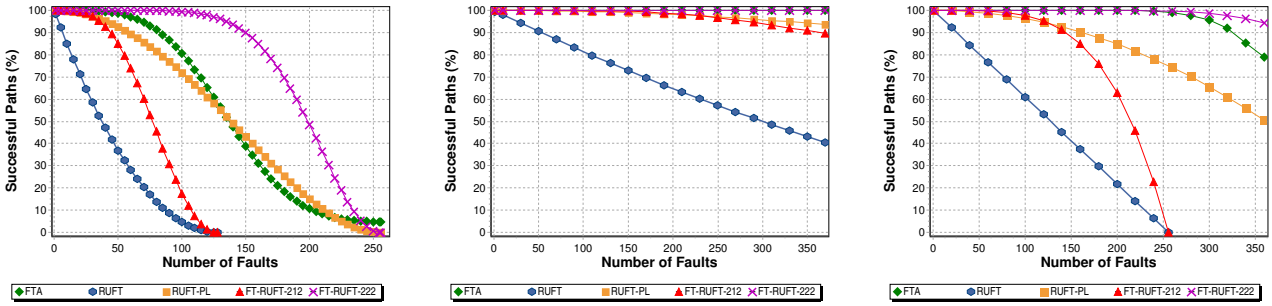
As an alternative point of view, Figure 10 shows the percentage of source–destination pairs that are able to communicate in each topology for different number of faults at network links. Instead of showing which combinations are non-tolerated, we show how many paths are still connected for fault combinations of that number of faults. As it can be seen, FT-RUFT-212 obtains better results than RUFT, communicating a high percentage of source–destination pairs by adding just a few links in the injection/ejection edge. On the other hand, the number of source–destination pairs that can keep communicating with RUFT-PL is higher than FT-RUFT-212, thanks to the parallel links in the interconnection between switches. Even so, FT-RUFT-222 offers better results than FTA despite using the same amount of network resources. In particular, it can cope with a relatively high number of faults while still keeping a high percentage of source–

1. Remember that FTA uses bidirectional links whereas RUFT-like topologies implement unidirectional links. Thus, FTA doubles the number of links with respect to RUFT-like topologies, and it has the same number of links that the RUFT tolopogies with double links in the network.



(a) 4-ary 3-tree. (b) 8-ary 3-tree. (c) 16-ary 2-tree.

Fig. 9: Percentage of non-tolerated fault combinations at network links for different network sizes.



(a) 4-ary 3-tree. (b) 8-ary 3-tree. (c) 16-ary 2-tree.

Fig. 10: Percentage of source–destination paths that can communicate for different network sizes and number of faults at network links.

destination paths even when there are more than 350 faults in the network.

5.3 Performance Evaluation

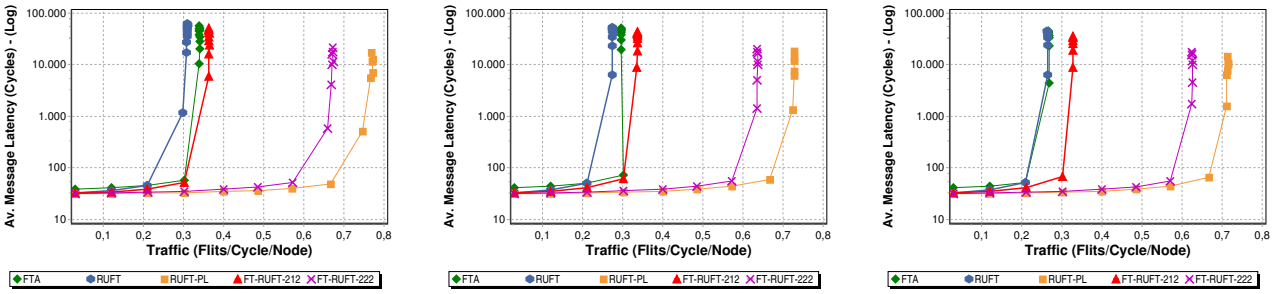
Several network sizes were analyzed, from 64 to 4096 nodes, with different topology arities. For the sake of shortness, results for the largest networks will only be shown for the uniform traffic pattern. Concerning packet size, although we evaluated 8B, 128B, 256B, and 1024B-packets, for the sake of shortness, for uniform traffic we will show results for 8B and 128B; and for the other traffic patterns only for 128B, since similar conclusions can be drawn with all packet sizes.

In Figures 11, 12, 13, and 14, we compare the performance achieved by FTA, RUFT, RUFT-PL, FT-RUFT-212, and FT-RUFT-222 for uniform, hot-spot, complement, and shuffle traffic patterns, respectively. As can be seen in Figure 11, for uniform traffic, RUFT-PL is able to achieve more than 2× the performance of FTA, RUFT, and FT-RUFT-212. However, FT-RUFT-212 increases the RUFT performance by only adding some injection and ejection links, while also providing fault-tolerance. On the other hand, although this topology uses less hardware resources than the fat-tree, it is able to outperform it. Concerning FT-RUFT-222, with a similar hardware cost to fat-tree, this topology is able to obtain more throughput than FTA, RUFT, and FT-RUFT-212, in addition to provide a high fault-tolerance degree as shown in Section 5.2. As it can be seen, the differences among the studied topologies are consistent through all network sizes, and the packet size does not influence the respective behavior of each topology, thus, from now on, we only show results for a packet size of 128B.

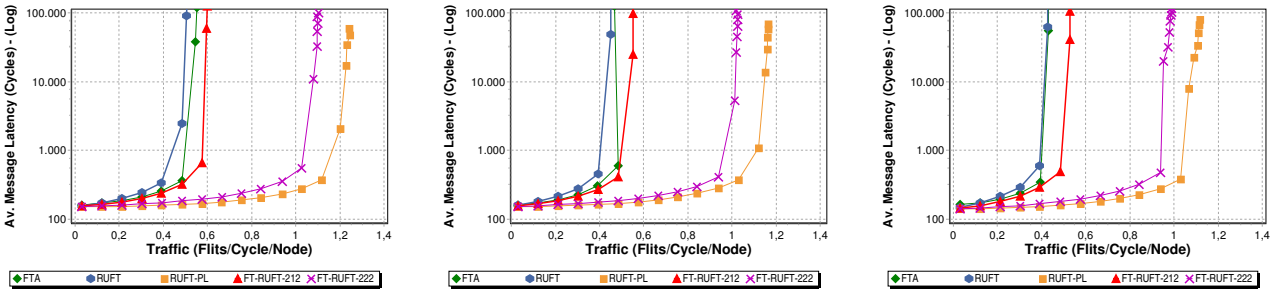
For hot-spot traffic (see Figure 12), all proposed topologies are able to outperform FTA and RUFT thanks to the double injection/ejection that allows to isolate the concentration of traffic, reducing the HoL blocking effect and allowing the topologies to get a higher performance.

For the complement traffic pattern (see Figure 13), each source sends packets to a single destination and RUFT-PL achieves the highest performance, taking into account that every source–destination pair has a unique path and, thanks to the parallel links and the dynamic port selection, packets are distributed between them, outperforming the other topologies. Likewise, RUFT also has an exclusive path for every source–destination pair, so that, HoL blocking does not exist and it is possible to get a good performance. On the other hand, as FT-RUFT-212 uses dual injection/ejection and FT-RUFT-222 uses dual links in the whole network, the distribution of packets in the network is different to RUFT, modifying the effective traffic pattern. However, FT-RUFT-222 still improves performance over RUFT thanks to the dual links.

Concerning the shuffle traffic pattern, any of the three proposals are able to outperform FTA and RUFT topologies, being FT-RUFT-222 the one that uses all the hardware resources and adaptivity of the provided paths in the most effective way.

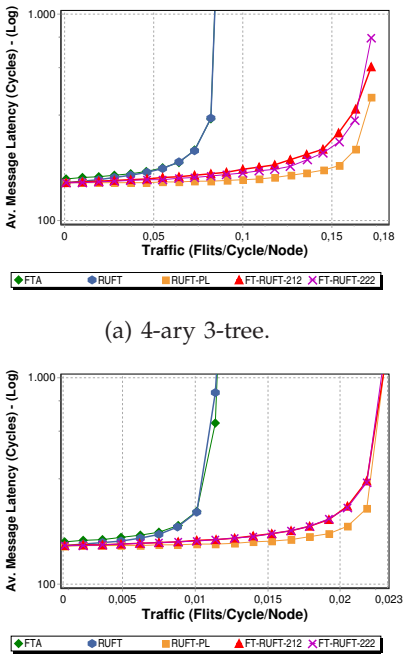


(a) 4-ary 3-tree, PS 8B. (b) 8-ary 3-tree, PS 8B. (c) 16-ary 3-tree, PS 8B.



(d) 4-ary 3-tree, PS 128B. (e) 8-ary 3-tree, PS 128B. (f) 16-ary 3-tree, PS 128B.

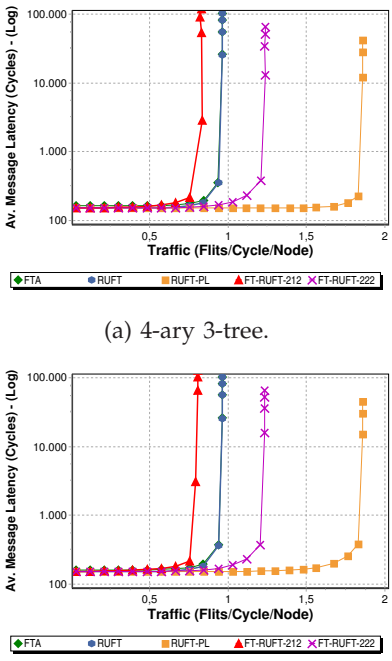
Fig. 11: Packet latency from generation versus accepted traffic for different network sizes with uniform traffic and a packet size (PS) of 8B and 128B.



(a) 4-ary 3-tree.

(b) 8-ary 3-tree.

Fig. 12: Packet latency from generation versus accepted traffic for different network sizes with a packet size of 128B and hotspot traffic (15%).



(a) 4-ary 3-tree.

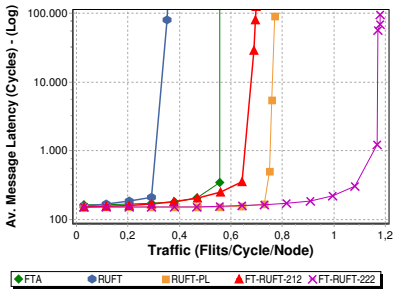
(b) 8-ary 3-tree.

Fig. 13: Packet latency from generation versus accepted traffic for different network sizes with a packet size of 128B and complement traffic.

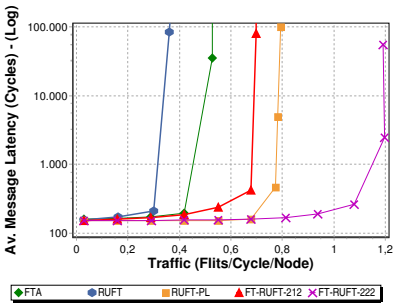
5.4 Performance Degradation

The performance degradation analysis is performed to obtain the behavior of the topology in the presence of faults at network links. For this analysis, we have evaluated the proposed topologies under the same traffic patterns, as in the previous evaluation, analyzing several network sizes. Regarding packet size, only results for 128B will be shown.

Figure 15 shows the results for the performance degra-



(a) 4-ary 3-tree.



(b) 8-ary 3-tree.

Fig. 14: Packet latency from generation versus accepted traffic for different network sizes with a packet size of 128B and shuffle traffic.

dation of RUFT-PL, FT-RUFT-212, FT-RUFT-222 and FTA under uniform traffic. As it can be seen, RUFT-PL is the topology which obtains the best throughput; however, as faults are introduced in the network, its performance degradation becomes more noticeable because the topology is not able to cope with the traffic with a reduced number of paths per source–destination pair, increasing the HoL blocking effect in those points where there is only one available interconnection link to forward the traffic due to faults. Moreover, the relative performance degradation with FT-RUFT-212 is lower than in RUFT-PL thanks to the dual injection/ejection and an appropriate routing, taking advantage of the still useful links. Concerning FTA, we can see that the performance degradation is very similar to FT-RUFT-212 despite using more links and a greater switch complexity, especially when the number of stages in the network increases.

Finally, FT-RUFT-222 achieves a very low performance degradation despite having a high number of faults in the network links.

For the hot-spot traffic (Figure 16), RUFT-PL and FT-RUFT-222 are able to keep the traffic as uniform as possible thanks to the dual links and to the double injection/ejection that allows balancing the traffic between them. Although FT-RUFT-212 also has dual injection/ejection, the performance degradation is higher when the number of network links is small (Figure 16a), but as the number of links increases, the topology is less affected by the failures (Figure 16b) and the performance degradation is minimal. Concerning FTA, the topology has a low performance degradation, thanks to the number of paths that provides the adaptive routing, however, its initial throughput is very low compared to the other proposals.

Topology	Switch complexity
FTA	$3k^2$
RUFT	k^2
RUFT-PL	$4k^2$
FT-RUFT-212	k^2 (interm. stages) | $2k^2$ (first and last stages)
FT-RUFT-222	$4k^2$

TABLE 3: Switch complexity comparison.

On the other hand, for the complement traffic pattern, in RUFT-PL, each source-destination pair has a unique path with two available links to route the packets. For this reason the topology can reach a very high performance. In small networks (Figure 17a), the performance degradation is high, because as links fail, the number of paths in the network decreases. However, as in other traffic patterns, as the network becomes larger (Figure 17b), the performance degradation is lower. FT-RUFT-212 and FT-RUFT-222, thanks to the dual injection/ejection, are able to exploit the different paths to keep the performance as uniform as possible, while in FTA the degradation is worse because of the HoL blocking effect generated by the routing algorithm.

For the shuffle traffic (see Figure 18), FT-RUFT-222 suffers the worst degradation. However, is able to keep the network performance, even, over the other topologies, despite the high number of failures in the network. Concerning the FTA, it also suffers from performance degradation, and its throughput without faults is very low compared to the other proposals. FT-RUFT-212 achieves, in this case, not only a better performance than FTA, but also its behavior is similar or better than RUFT-PL.

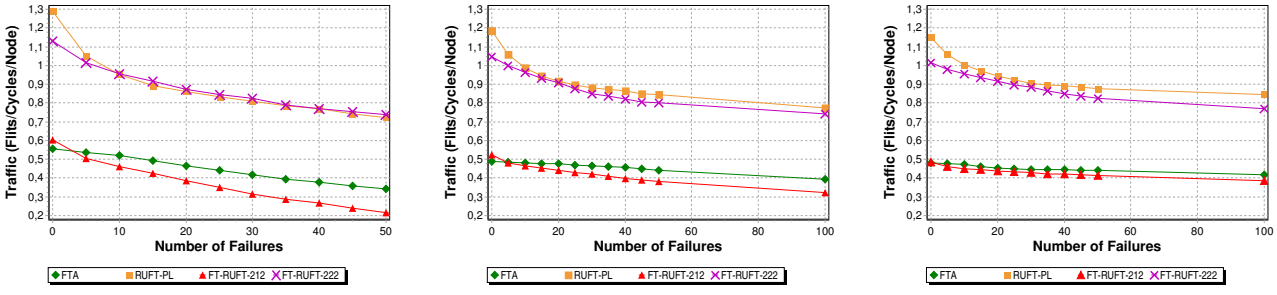
Overall, the three proposals provides fault-tolerance to the interconnection network with a moderate performance degradation in presence of faults, being FT-RUFT-222, thanks to the high availability of paths, the one that offers the best results.

5.5 Cost Evaluation

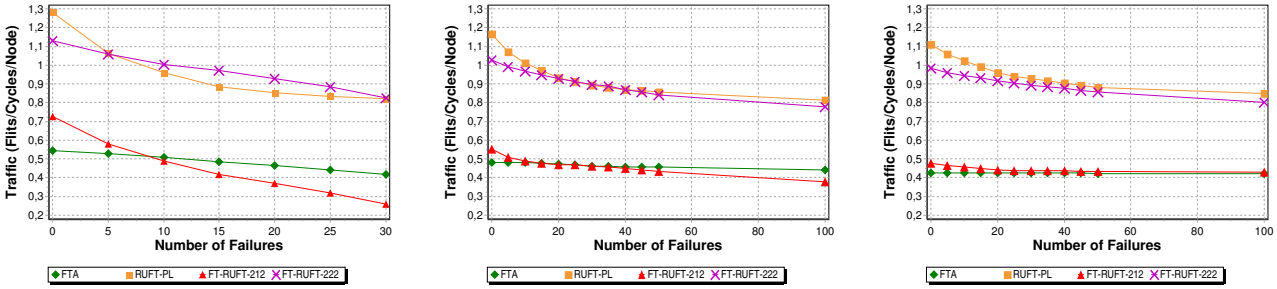
To analyze and compare the hardware cost of each topology proposed in this paper, we have considered the most outstanding elements in the network design: the number of unidirectional links and the number of switching elements. The later involves the number of switches and its degree.

Table 3 summarizes switch complexity for all the analyzed topologies measured as the number of switching elements required at the switch to connect the input and output ports. As we can see, the switch complexity of FT-RUFT-212 depends on the stage of the switch, since first and last stages switches have a complexity of $2k^2$ (remember that they are asymmetric) while the intermediate stages switches have a complexity of k^2.

In RUFT-PL and FT-RUFT-222, if we considered a static selection of the parallel links based on the destination identifier, as described in Section 4.3, then the switch has $2k^2$ switching elements. However, with a more general policy and, most important, to tolerate network link faults, transitions between the parallel links are required,

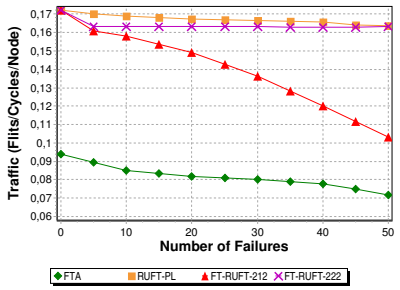


(a) 4-ary 3-tree. (b) 4-ary 4-tree. (c) 4-ary 5-tree.

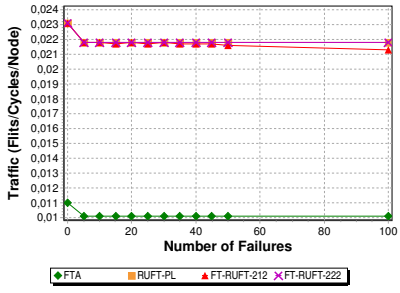


(d) 8-ary 2-tree. (e) 8-ary 3-tree. (f) 8-ary 4-tree.

Fig. 15: Performance degradation with faults for different network sizes with uniform traffic and a packet size of 128B.

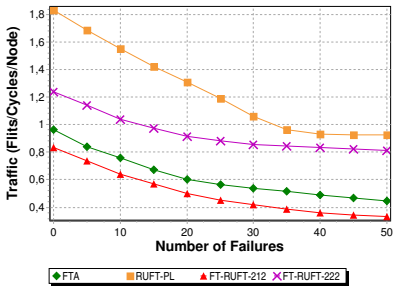


(a) 4-ary 3-tree.

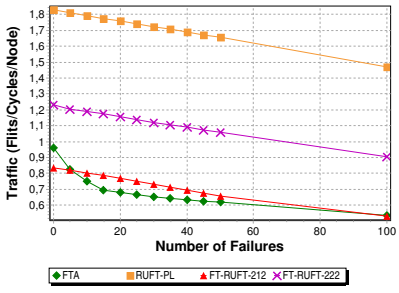


(b) 8-ary 3-tree.

Fig. 16: Performance degradation with faults for different network sizes with hot-spot traffic (15%) and a packet size of 128B.



(a) 4-ary 3-tree.

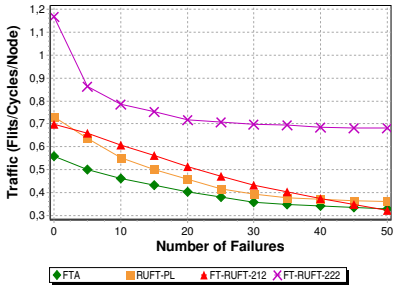


(b) 8-ary 3-tree.

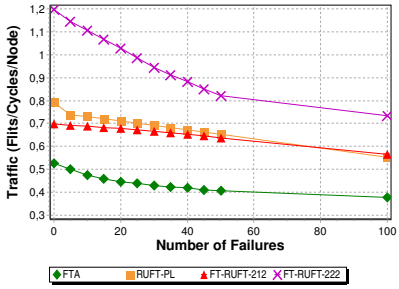
Fig. 17: Performance degradation with faults for different network sizes with complement traffic and a packet size of 128B.

leading to $4k^2$ switching elements. As expected, the topologies that have a higher switch degree (FTA, RUFT-PL, and FT-RUFT-222) are the ones that require more switching elements. RUFT is the cheapest one but it is also the one that provides the worst performance and fault-tolerance results (no fault-tolerance is provided).

Table 4 shows the number of links and switching elements required by each topology for two different network sizes. Notice that, the Links column of Table 4, besides of the network links, also considers the injec-



(a) 4-ary 3-tree.



(b) 8-ary 3-tree.

Fig. 18: Performance degradation with faults for different network sizes with shuffle traffic and a packet size of 128B.

	Topology	Links[1]	Switching Elements	Network Throughput[2]	Base Latency[3]
4-ary 3-tree	FTA	384	2304	0.55	161
	RUFT	256	768	0.51	156
	RUFT-PL	512	3072	1.24	152
	FT-RUFT-212	384	1280	0.60	154
	FT-RUFT-222	512	3072	1.10	153
8-ary 3-tree	FTA	3072	36864	0.48	162
	RUFT	2048	12288	0.45	159
	RUFT-PL	4096	49152	1.16	153
	FT-RUFT-212	3072	20480	0.55	156
	FT-RUFT-222	4096	49152	1.03	154

[1] Number of unidirectional links used by each topology.
[2] Network throughput is measured in flits/cycle/node.
[3] Network latency is measured in cycles.

TABLE 4: Performance-Cost for different network sizes, uniform traffic and a packet size of 128B.

tion/ejection links. To perform a cost-performance comparison, we have also shown some performance results for the uniform traffic pattern. As already stated, RUFT uses the least number of links and the least number of switching elements (less than half the resources of the fat-tree and less resources than the other topologies). However, RUFT provides the worst throughput and does not provide any fault-tolerance (see Table 2). Moreover, FT-RUFT-212 increases the network throughput and provides fault-tolerance (it supports 3 network faults, see Table 2), while the number of required switch elements remains lower than the fat-tree one. Notice that, the number of links in the system is the same as the fat-tree, for networks with three stages, but as the number of stages increases, the number of required links will be lower in FT-RUFT-212. Finally, RUFT-PL and FT-RUFT-222 require some extra links and 33% more switching elements than FTA, but the throughput achieved is nearly twice the one offered by the other topologies and also obtains the lowest latency. However, when considering fault-tolerance, FT-RUFT-222 is definitively the best choice, as it is able to support up to 7 network faults versus 1 fault of RUFT-PL (see Table 2).

6 Conclusions

This work presents three MIN topologies which offer good throughput and fault-tolerance levels. These topologies are derived from the RUFT topology. They are referred to as RUFT-PL, FT-RUFT-212, and FT-RUFT-222.

Each topology has been evaluated under different traffic patterns and different packets sizes. Although RUFT-PL offers the best performance, it only tolerates one fault, and it does not support switch faults. FT-RUFT-212 does not only provide fault-tolerance to RUFT but also increases the network performance with a minimal hardware increase. This topology is a good alternative to the FTA, given that provides a good fault-tolerance level and a similar/better performance that the previous one, but with a lower design cost.

FT-RUFT-222 is a topology derived from RUFT-PL and FT-RUFT-212 that combines the best properties of both proposals, allowing the topology to achieve up to twice the fat-tree network performance and a very high level of fault-tolerance (it tolerates 7 faults in the network links and 1 fault in the injection/ejection links with 100% probability and a large number of fault combinations with a very high probability). In addition, both, FT-RUFT-212 and FT-RUFT-222 are able to deal with failures of switches in the network. In contrast, FTA does not support failures in the switches of the first stage. Finally, the three new topologies proposed in this paper are able to tolerate faults in the injection and ejection links.

As future work, we are working on developing a routing mechanism that can take advantage of the fault-tolerance provided by the topologies.

Acknowledgments

This work was supported by the Spanish Ministerio de Economía y Competitividad (MINECO) and by FEDER funds under Grant TIN2012-38341-C04-01.

References

[1] Top 500 supercomputers sites. http://www.top500.org.
[2] C. Gomez, F. Gilabert, M.E. Gomez, P. Lopez, and J. Duato. RUFT: Simplifying the Fat-Tree Topology. In *Parallel and Distributed Systems, 2008. 14th IEEE Intl. Conf. on*, 2008.
[3] C.W. Chen and C.P. Chung. Designing A Disjoint Paths Interconnection Network with Fault Tolerance and Collision Solving. *The Journal of Supercomputing*, 34(1):63–80, 2005.
[4] D.S. Parker and C.S. Raghavendra. The Gamma Network. *Computers, IEEE Trans. on*, 1984.
[5] P.J. Chuang. CGIN: A Fault Tolerant Modified Gamma Interconnection Network. *Parallel and Distributed Systems, IEEE Trans. on*, 1996.
[6] C.W. Chen, N.P. Lu, T.F. Chen, and C.P. Chung. Fault-tolerant Gamma Interconnection Networks by Chaining. *Computers and Digital Techniques, IEE Proc.*, 2000.
[7] R. Rastogi, Nitin, and D. Chauhan. 3-Disjoint Paths Fault-tolerant Omega Multi-stage Interconnection Network with Reachable Sets and Coloring Scheme. In *Computer Modelling and Simulation (UKSim), 2011 UkSim 13th Intl. Conf. on*, 2011.

[8] R. Rastogi, R. Verma, Nitin, and D. Chauhan. 3-Disjoint Paths Fault-tolerant Multi-stage Interconnection Networks. In *Advances in Computing and Communications*, volume 190 of *Communications in Computer and Information Science*. 2011.

[9] S. Wei and G. Lee. Extra Group Network: a cost-effective fault-tolerant multistage interconnection network. In *Computer Architecture, 1988. Conf. Proceedings. 15th Annual Intl. Symp. on*, 1988.

[10] S. Konstantinidou. The Selective Extra-Stage Butterfly. *Computer Design: VLSI in Computers and Processors, 1992. ICCD '92. Proc., IEEE 1992 Intl. Conf. on*, pages 502–506, 1992.

[11] N. Kamiura, T. Kodera, and N. Matsui. Design of a fault tolerant multistage interconnection network with parallel duplicated switches. In *Defect and Fault Tolerance in VLSI Systems, 2000. Proc. IEEE Intl. Symp. on*, 2000.

[12] S. Bataineh and B. Allosl. Fault-tolerant multistage interconnection network. *Telecommunication Systems*, 17(4):455–472, 2001.

[13] C.W. Chen, P.S. Gan, and C.H. Chang. Designing a High Performance and Fault Tolerant Multistage Interconnection Network with Easy Dynamic Rerouting. In *Parallel and Distributed Processing and Applications*, volume 3358 of *LNCS*, pages 1007–1016. 2005.

[14] F.O. Sem-Jacobsen, T. Skeie, O. Lysne, O. Toerudbakken, E. Rongved, and B. Johnsen. Siamese-Twin: A Dynamically Fault-Tolerant Fat-Tree. In *Parallel and Distributed Processing Symp., 2005. Proc. 19th IEEE Intl.*, 2005.

[15] M. Valerio, L.E. Moser, and P.M. Melliar-Smith. Fault-tolerant orthogonal fat-trees as interconnection networks. In *Algorithms and Architectures for Parallel Processing, 1995. ICAPP 95. IEEE First ICA/sup 3/PP., IEEE First Intl. Conf. on*, 1995.

[16] I. Gazit and M. Malek. Fault tolerance capabilities in multistage network-based multicomputer systems. *Computers, IEEE Transactions*, 37(7):788–798, 1988.

[17] C.T. Ho and L. Stockmeyer. A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers. In *Vehicle Navigation and Information Systems Conf., 1993., Proceedings of the IEEE-IEE*.

[18] F.T. Chong and T.F. Knight, Jr. Design and Performance of Multipath MIN Architectures. In *Proc. of the 4th Annual ACM Symp. on Parallel Algorithms and Architectures. 1992*.

[19] C. Gómez, M.E. Gómez, P. López, and J. Duato. FT^{2}EI: A Dynamic Fault-Tolerant Routing Methodology for Fat-Trees with Exclusion Intervals. *IEEE TPDS*, 20(6), 2009.

[20] G. Zarza, D. Lugones, D. Franco, and E. Luque. A Multipath Fault-Tolerant Routing Method for High-Speed Interconnection Networks. *Europar*, 5704:1078–1088, 2009.

[21] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson. F10: A Fault-tolerant Engineered Network. In *Proc. of the 10th USENIX Conf. on Networked Systems Design and Implementation, 2013.*

[22] V.P. Bhardwaj and N. Nitin. A new fault tolerant routing algorithm for advance irregular augmented shuffle exchange network. In *Computer Modelling and Simulation (UKSim), 2012 UKSim 14th Intl. Conf. on*, 2012.

[23] F. Bistouni and M. Jahanshahi. Pars network: A multistage interconnection network with fault-tolerance capability. *Journal of Parallel and Distributed Computing*, 75(0):168 – 183, 2015.

[24] R.R. Aggarwal and Dr. L. Kaur. Fault-Tolerance and Permutation Analysis of ASEN and its Variant. *Intl. Journal of Computer Science and Information Technologies*, 1(1):24–32, 2010.

[25] F. Petrini and M. Vanneschi. k–ary n–trees: High performance networks for massively parallel architecture. *IEEE Micro*, 15, 1995.

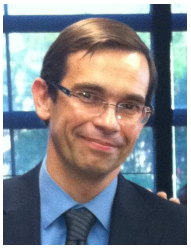
[26] C. Gómez, F. Gilabert, M.E. Gómez, P. López, and J. Duato. Deterministic versus Adaptive Routing in Fat-Trees. In *Parallel and Distributed Processing Symp., 2007. IPDPS 2007. IEEE Intl.*, 2007.

[27] S. Scott, D. Abts, J. Kim, and W.J. Dally. The BlackWidow High-Radix Clos Network. In *Computer Architecture, 2006. ISCA '06. 33rd Intl. Symp. on*, 2006.

Diego F. Bermúdez G. obtained his MS in Computer Science from the Universidad Politécnica de Valencia, Spain, in 2008. He joined the Parallel Architecture Group at Universidad Politécnica de Valencia in 2011, where he is conducting his PhD. His research interests are in the field of topology for off-chip interconnection networks and GPU architectures.

Crispín Gómez Requena obtained his MS in Computer Science from the Universidad Politécnica de Valencia, Spain, in 2005. He joined the Parallel Architecture Group at Universidad Politécnica de Valencia in 2006 where he presented his thesis. On 2010, he joined the Intel Barcelona Research Center group inside Intel Labs under the leadership of Prof. Antonio Gonzalez. His research interests are in the field of topology and fault–tolerance for off–chip and on–chip interconnection networks, and in microprocessor architectures.

María Engracia Gómez obtained her MS and PhD degrees in Computer Science from the Universidad Politécnica de Valencia, Spain, in 1996 and 2000, respectively. She joined the Department of Computer Engineering (DISCA) at Universidad Politécnica de Valencia in 1996 where she is currently an Associate Professor of Computer Architecture and Technology. Her research interests are in the field of interconnection networks, networks-on-chips, cache coherence protocols and memory hierarchy. In those fields she has published more than 50 refereed conference and journal papers.



Pedro López is a full professor in computer architecture and technology at the Department of Computer Engineering (DISCA), Universidad Politécnica de Valencia, Spain. He received the BEng degree in electrical engineering and the MS and Ph.D. degrees in computer engineering from the same university in 1984, 1990 and 1995, respectively. He has taught several courses on computer organization and architecture. His research interests include high performance interconnection networks for multiprocessor systems and clusters and networks on chip. Prof. López has published more than 100 refereed conference and journal papers. He served as a member of the editorial board of Parallel Computing journal.

José Duato received the MS and PhD degrees in electrical engineering from the Technical University of Valencia, Spain, in 1981 and 1985, respectively. Currently, Dr. Duato is Professor in the Department of Computer Engineering (DISCA) at the Polytechnic University of Valencia. 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. Prof. Duato has published over 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. Prof. Duato is the first author of the book "Interconnection Networks: An Engineering Approach". Dr. Duato 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.