Improving Non-Minimal and Adaptive Routing Algorithms in Slim Fly Networks

Pedro Yébenes*, Jesus Escudero-Sahuquillo*, Pedro J. García*, Francisco J. Quiles*, and Torsten Hoefler†
*Department of Computing Systems, University of Castilla-La Mancha, Spain.
†Department of Computer Science, ETH Zurich, Switzerland
Email: {pedro.yebenes,jesus.escudero,pedrojavier.garcia,francisco.quiles}@uclm.es, htor@inf.ethz.ch

***Abstract*—Interconnection networks must meet the communication demands of current High-Performance Computing systems. In order to interconnect efficiently the end nodes of these systems with a good performance-to-cost ratio, new network topologies have been proposed in the last years which leverage high-radix switches, such as Slim Fly. Adversarial-like traffic patterns, however, may reduce severely the performance of Slim Fly networks when using only minimal-path routing. In order to mitigate the performance degradation in these scenarios, Slim Fly networks should configure an oblivious or adaptive non-minimal routing. The non-minimal routing algorithms proposed for Slim Fly usually rely on Valiant's algorithm to select the paths, at the cost of doubling the average path-length, as well as the number of Virtual Channels (VCs) required to prevent deadlocks. Moreover, Valiant may introduce additional inefficiencies when applied to Slim Fly networks, such as the "turn-around problem" that we analyze in this work. With the aim of overcoming these drawbacks, we propose in this paper two variants of the Valiant's algorithm that improve the non-minimal path selection in Slim Fly networks. They are designed to be combined with adaptive routing algorithms that rely on Valiant to select non-minimal paths, such as UGAL or PAR, which we have adapted to the Slim Fly topology. Through the results from simulation experiments, we show that our proposals improve the network performance and/or reduce the number of required VCs to prevent deadlocks, even in scenarios with adversarial-like traffic.**

***Index Terms*—High-Performance Interconnection Networks, Slim Fly Topology, Routing**

I. Motivation

The interest in interconnection networks increases as the size of the system grows, especially in the High Performance Computing (HPC) systems such as the ranked in the Top 500 list [1]. These systems need to interconnect thousand of end nodes in order to meet with the growing computing and/or storage demands of current applications, thus larger HPC systems will appear requiring larger interconnection networks. If the interconnection network is unable to perform at the speed required by applications, it may become the system bottleneck, degrading the overall system performance.

Interconnect designers face several challenges when defining the architecture of large networks connecting tens of thousands of end nodes. The most typical ones are to reduce the number of elements in order to lower the network cost, guarantee high network bandwidth and low latency, save power consumption, and provide resiliency to link failures. A typical design decision is to lower the network diameter to diminish the number of network elements which also reduces the power consumption. Following this idea, the *flattened butterfly* [2], the Dragonfly [3], or the KNS [4] topologies were proposed. These topologies, in contrast with the traditional direct and indirect topologies, scale in cost while maintaining their performance, by reducing the network diameter and the number of switches. Another proposal is the Slim Fly topology [5] which maximizes the number of end nodes for a given switch radix and network diameter. It guarantees network diameter two by using MMS graphs. This topology, hence, needs fewer switches than the other topologies while offering high communication bandwidth and low latency.

Slim Fly networks require efficient routing algorithms that fully exploit their theoretical performance, regardless of the traffic distribution. In general, efficient routing algorithms compute one or multiple routes between two end nodes. The length of these routes can be minimal (i.e., the shortest distance between two nodes), or non-minimal. Minimal-path (MIN) routing algorithms obtain a high performance when traffic behaves uniformly, i.e., traffic is balanced among the available network paths. Adversarial traffic, however, leads MIN routing to overuse some network paths while others remain idle. In order to balance the traffic the among available network paths, routing algorithms, such as Valiant (VAL) [6] or UGAL [7], select non-minimal paths according to a criterion. Although these algorithms successfully distribute traffic in the adversarial scenarios, this criterion can be improved considering the Slim Fly topology features. Specifically, adaptive routing algorithms for Slim fly topologies usually rely on VAL to select the non-minimal path. However, the VAL routing is not specially designed for Slim Fly topologies, and it has several issues when computing a non-minimal routes, such as visiting switches previously visited along the same route, or selecting non-minimal paths too long. In this paper, we analyze these problems and propose several improvements to the VAL routing. Furthermore, other algorithms can also benefit from the proposals, such as UGAL or PAR [8], which can be also adapted to fit in Slim Fly topology.

The main contributions of this paper are the following:

- We analyze the VAL routing proposed for Slim Fly topology, identifying two problems: the "turn around problem" of traffic flows and the overhead introduced by non-minimal paths.
- We design strategies to solve the "turn-around problem" and to reduce the length of non-minimal paths. One of

them, moreover, requires fewer virtual channels to avoid deadlocks.

- We apply the proposed strategies to improve adaptive routing algorithms, such as UGAL or PAR, which can be also adapted to fit in Slim Fly topology.
- We evaluate our proposals by means of simulation through uniform and adversarial traffic scenarios and compare them to other routing algorithms.

The rest of the paper is organized as follows. Section II explains the basics of the Slim Fly topology: connection pattern, routing algorithms, and deadlock freedom. In Section III we identify the problems that may appear in the non-minimal routing algorithms in Slim Fly networks. In Section IV we propose new techniques which face with the identified problems. In Section V we evaluate our proposal through simulation results. Section VI describes the related work. Finally, Section VII draws some conclusions and future work.

II. The Slim Fly Topology

Slim Fly topology bases its connection pattern on *MMS graphs* [9], [10] to be a close-to-optimum topology that maximizes the number of endpoints for network diameter two and radix k' [5]. It offers full global bandwidth, provides resiliency, and reduces network costs, energy consumption and latency. Compared to other topologies, Slim Fly networks require fewer elements for a given diameter and switch radix. Table I lists the symbols that describe this topology.

TABLE I: Symbols used to describe the Slim Fly topology

N	Number of end nodes in the network
p	Number of end nodes attached to a switch (concentration)
k'	Number of channels to other switches (network radix)
k	Switch radix ($k = k' + p$)
N_s	Number of switches in the network

A. Connection pattern

The construction of a Slim Fly requires a prime power q such that $q = 4l+\delta$, where $\delta \in \{-1, 0, 1\}$ and $l \in \mathbb{N}$. For such q, a MMS graph is generated with network radix $k' = \frac{3q-\delta}{2}$ and a number of switches $N_s = 2q^2$. Switches are connected among them by performing the following steps:

1) *Constructing the Galois field* $\mathbb{F}_q$: Let $\mathbb{F}_q$ be the Galois field of order q. A *primitive element* ξ of $\mathbb{F}_q$ has to be found. ξ is an element of $\mathbb{F}_q$ that generates $\mathbb{F}_q$, i.e., all non-zero elements of $\mathbb{F}_q$ can be written as ξ^i mod q ($i \in \mathbb{N}$). There exists no universal method for finding ξ, but it can be calculated for smaller fields.
2) *Constructing the generator sets X and X'*: ξ is utilized to construct the sets known as *generators*. Depending on δ, these sets are generated in a different way:
   - If $\delta = 1$
     - $X = \{1, \xi^2, \xi^4, ..., \xi^{q-3}\}$
     - $X' = \{\xi, \xi^3, \xi^5, ..., \xi^{q-2}\}$
   - If $\delta = -1$
     - $X = \{1, \xi^2, \xi^4, ..., \xi^{2l-2}, \xi^{2l-1}, \xi^{2l+1}, ..., \xi^{4l-3}\}$
     - $X' = \{\xi, \xi^3, \xi^5, ..., \xi^{2l-1}, \xi^{2l}, ..., \xi^{4l-4}, \xi^{4l-2}\}$

   All operations applies modulo q, e.g., ξ^{4l-2} means ξ^{4l-2} mod q.
3) *Connecting switches*: The set of all switches is a Cartesian product: $\{0, 1\} \times \mathbb{F}_q \times \mathbb{F}_q$. Switches are divided into two subgraphs: the composed of switches $(0, x, y)$ and the composed of switches $(1, m, c)$. Switches are connected using the following equations [10]:

$$\text{switch } (0, x, y) \text{ is connected to } (0, x, y') \text{ iff } y - y' \in X \quad (1)$$

$$\text{switch } (1, m, c) \text{ is connected to } (1, m, c') \text{ iff } c - c' \in X' \quad (2)$$

$$\text{switch } (0, x, y) \text{ is connected to } (1, m, c) \text{ iff } y = mx + c \quad (3)$$

4) *Attaching end nodes*: Each switch should be connected to $p \approx \frac{k'}{2}$ end nodes to ensure full global bandwidth [5].

The following example details the construction of a small Slim Fly for $q = 5$. As $q = 5 = 4 \times 1 + 1$, l is 1, δ is 1, the network radix is $k' = \frac{3 \times 5 - 1}{2} = 7$, and $N_s = 2 \times 5^2 = 50$. The Galois field is $\mathbb{F}_5 = \{0, 1, 2, 3, 4\}$ and the primitive element is $\xi = 2$. This primitive element generates all elements of $\mathbb{F}_5$ ($1 = 2^4 \bmod 5, 2 = 2^1 \bmod 5, 3 = 2^3 \bmod 5, 4 = 2^2 \bmod 5$). The generator sets are $X = \{1, 2^2 \bmod 5, ..., 2^{5-3} \bmod 5\} = \{1, 4\}$ and $X' = \{2, 2^3 \bmod 5, ..., 2^{5-2} \bmod 5\} = \{2, 3\}$. Then, Equations 1, 2, and 3 are applied to connect switches. Fig. 1 exemplifies the result of this process. Equation 3 connections are omitted for the sake of clarity. Finally, three or four end nodes can be attached to switches ($p \approx \frac{7}{2}$) which results in a network with 150 or 200 end nodes, respectively.

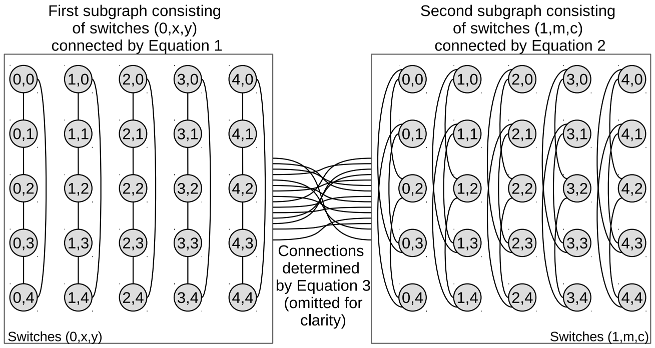

First subgraph consisting of switches (0,x,y) connected by Equation 1
Second subgraph consisting of switches (1,m,c) connected by Equation 2
0,0 1,0 2,0 3,0 4,0
0,1 1,1 2,1 3,1 4,1
0,2 1,2 2,2 3,2 4,2
0,3 1,3 2,3 3,3 4,3
0,4 1,4 2,4 3,4 4,4
Connections determined by Equation 3 (omitted for clarity)
Switches (0,x,y)
Switches (1,m,c)

Fig. 1: Basic Slim Fly topology diagram.

B. Routing in Slim Fly topology

Slim Fly connection pattern has a single minimal path between a source end node S attached to a switch Sw_S and a destination end node D attached to a switch Sw_D [5]. The minimal-path routing (MIN) algorithm, hence, consists of three routing cases:

1) 0-hop path: $Sw_S = Sw_D$. Packets are sent directly to the destination, end node D.
2) 1-hop path: $Sw_S \neq Sw_D$ and Sw_S is connected to Sw_D. Packets are routed to Sw_D.

3) 2-hop path: $Sw_S \neq Sw_D$ and Sw_S is not connected to Sw_D. Packets are routed to a middle switch Sw_M connected with Sw_D and Sw_S. Then, they travel from Sw_M to Sw_D. Note that between each pair of switches Sw_S-Sw_D, there is only a single switch Sw_M because network diameter is two. Fig. 2 displays the minimal path between Sw_S and Sw_D.

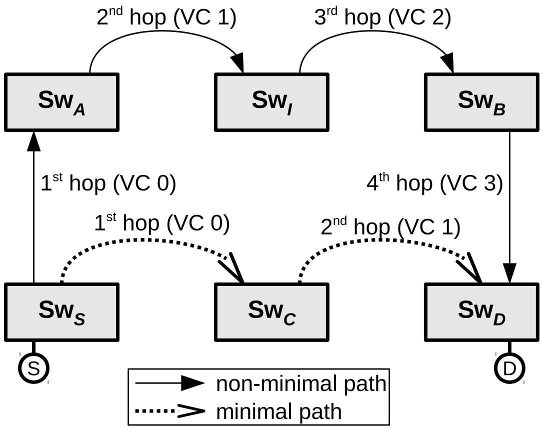


Fig. 2: Minimal and non-minimal paths in a Slim Fly topology. Virtual channels avoid deadlocks in each hop.

MIN routing achieves high performance under uniform traffic patterns, since traffic flows are balanced fairly among the available routes. Unfortunately, adversarial traffic causes an overuse of some network ports so that non-minimal routing algorithms are required using alternative routes.

Non-minimal path routing is based on the Valiant Random Routing (VAL) algorithm [6]. For each packet, VAL first selects a random intermediate switch Sw_I different from Sw_S or Sw_D to route packets before moving to Sw_D. Fig. 2 illustrates the non-minimal path between Sw_S and Sw_D through Sw_I. Packets follow a minimal path when traveling to both Sw_I and to Sw_D, thus packets may perform 2, 3, or 4 hops, depending on whether switches Sw_S, Sw_D, and Sw_I are directly connected. Non-minimal paths are always one hop, at least, longer than the minimal ones.

C. Deadlock Freedom

Slim Fly topology offers path diversity allowing multiple routes between two end nodes at the cost of containing physical cycles that may lead to deadlock situations in the network [11]. Nevertheless, deadlocks can be avoided using virtual channels (VCs) [5]. In a n-hop path between two end nodes, packets performing hop k are assigned to VC $k-1$, where $1 \leq k \leq n$. Fig. 2 exemplifies a minimal path consisting of two hops, so that two VCs are required to prevent deadlocks, and a non-minimal path consisting of four hops, thus applying this policy requires four VCs.

III. Problem Statement

As we have described above, adaptive routing algorithms in Slim Fly networks use non-minimal routes to alleviate the link overuse under adversarial traffic scenarios. Specifically, an adversarial traffic pattern occurs when p end nodes attached to a switch send packets through the same link (p is the switch concentration, see Table I). This situation provokes network contention for the access to that link, limiting the accepted traffic load per end node to $\frac{1}{p}$. This traffic pattern affects especially minimal-path routing because Slim Fly networks offer only one possible minimal-path between two end nodes. Thus, minimal paths have to cross through the contended links and switches.

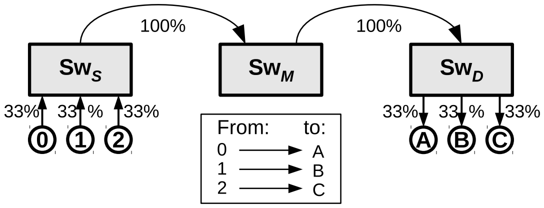


Fig. 3: Example of Adversarial Traffic in a Slim Fly network.

Fig. 3 illustrates an adversarial traffic scenario in a portion of a Slim Fly network. Sw_S has attached three end nodes 0, 1 and 2 (i.e., $p=3$), which send packets at full link speed to the three end nodes A, B and C attached to Sw_D. Because of Slim Fly connection pattern, a single minimal path exists between Sw_S and Sw_D through switch Sw_M (see Section II-A). Therefore, these traffic flows share the bandwidth of the links along the minimal path from switch Sw_S to switch Sw_D. Assuming that all links are able to deliver traffic at the same speed, both links connecting Sw_S to Sw_M and Sw_M to Sw_D operate at full speed. However, links connecting Sw_S to its nodes work at $\frac{1}{p}$ of their maximum speed because the link between Sw_S and Sw_D cannot absorb more traffic. On the other hand, links connecting Sw_D to its nodes also work at one third ($\frac{1}{p} = \frac{1}{3}$) of their maximum speed because the incoming traffic from Sw_M is distributed among them.

The effects of adversarial traffic can be mitigated by a routing algorithm that chooses different paths other than those overused, even though the length of the alternative paths is non-minimal. For this purpose, Slim Fly networks utilize Valiant routing algorithm (VAL) [6] which is able to generate non-minimal paths, used instead of minimal paths. Specifically, VAL routing selects randomly an intermediate switch Sw_I that can be traversed by means of a non-minimal path from switch Sw_S to Sw_D. This alternative route is used to balance traffic flows avoiding overused link, at the cost of doubling the path length and the amount of VCs required to avoid deadlocks (Fig. 2 shows that non-minimal routes use four VCs to prevent deadlocks). VAL routing computes paths with two, three, and four hops, depending on whether switches Sw_S, Sw_D and Sw_I are directly connected in the network. This means that, in the worst case, four VCs are required to assure deadlock prevention, no matter the average path length. Most importantly, the overuse of VCs prevent them from being used for other purposes such as congestion control or quality-of-service (QoS), among others [12].

We have identified that VAL routing applied to Slim Fly networks has several flaws that make the computing of non-minimal routes inefficient. The first one is that packets following non-minimal routes may visit the same switch twice (turning around and going back through the same route). Consequently, more VCs are required (one per additional hop) to prevent deadlocks, and latency overhead is generated to packets following this route. Furthermore, as packets suffering this problem arrive/leave the switch from/to the same port, switches should implement a crossbar able to forward a packet from a port to itself. Fig. 4 shows a routing situation when the *turn-around* problem appears.

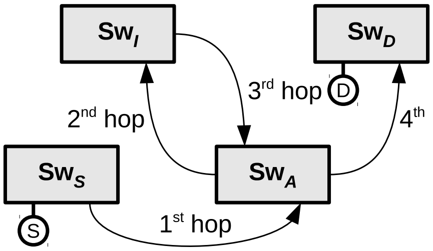


Fig. 4: Turn-around problem for packets in Slim Fly networks with VAL Routing.

Before sending a packet from Sw_S to Sw_D, VAL routing, first, selects a random intermediate switch Sw_I. Then, Sw_S forwards it to Sw_I through Sw_A, a switch in the middle of the minimal route from Sw_S to Sw_I. Next, packet is routed minimally from Sw_I to Sw_D (i.e., other two hops are required), so that it visits again switch Sw_A. Although Fig. 4 exhibits the *turn-around* problem in a four-hop path, it could also happen in a three-hop path when Sw_I is directly connected both to Sw_S. In this case, as VAL routing chooses randomly an intermediate switch, it may happen that Sw_I (placed just one hop ahead) is selected as an intermediate switch. Then, the packet is sent from Sw_S to Sw_I and then, send back to Sw_S to reach finally switch Sw_D. This is a corner-case that may happen when using VAL routing in Slim Fly networks.

Another issue that may affect the efficiency of VAL routing in Slim Fly networks is the average path length of the non-minimal routes (and so the number of required VCs in average to prevent deadlocks). In some situations, VAL routing is inefficient when computing alternative, non-minimal routes. For instance, Fig. 2 shows that VAL routing needs four hops to perform a non-minimal route from Sw_S to Sw_D, but we have described that it should be possible to reach Sw_D with two hops (minimal route) and three hops. The latter option would provide shorter non-minimal routes in average, and would also save one VC, compared to four-hop routes.

Therefore, VAL routing problems must be overcome in Slim Fly networks in order to avoid unnecessary non-minimal routes that increase packet latency and the number of required VCs to prevent deadlocks. Moreover, VAL routing improvements would also benefit adaptive routing algorithms that compute non-minimal routes using VAL, such as UGAL and PAR.

IV. Improvements Description

This section details two proposals for improving non-minimal and adaptive routing algorithms in Slim Fly networks. The first proposal faces the *turn-around* problem described in Section III. We call this technique *no-turn-around Valiant* (VAL$_{nta}$). VAL$_{nta}$ routing detects this problem when a packet is moving to its intermediate switch. If the intermediate switch visited by the packet is directly connected to the destination switch, then the packet is routed immediately to that switch, without considering other alternative paths. The information required by the VAL$_{nta}$ routing technique can be included in the routing tables at switches. For instance, in the situation shown in Fig. 4, Sw_S or Sw_A can send a packet to Sw_D. If we apply VAL$_{nta}$ routing, that packet is sent directly from Sw_A to Sw_D, instead of visiting Sw_I. Then, a packet from Sw_S is routed using two hops instead of four hops, and a packet from Sw_A just requires one hop to reach Sw_D.

VAL$_{nta}$ routing can be implemented easily in commercial networks, such as those using routing tables in the switches. In these types of networks, the topology is discovered by means of control messages, so that the network manager entity is aware of the topology connection pattern and the available routes among the end nodes. Then, that entity can compute minimal and non-minimal paths and detect those paths visiting the same router twice, populating routing tables accordingly to prevent the *turn-around* problem.

The second proposal extends VAL$_{nta}$ routing with functionality to reduce the number of hops of non-minimal paths. We call this technique *three-hop Valiant* (VAL$_{3h}$). VAL$_{3h}$ reduces the number of hops in non-minimal routes to a maximum of three, and it also avoids the *turn-around* problem.

Specifically, VAL$_{3h}$ selects the intermediate switch randomly as the original Valiant (hereafter, VAL$_{ori}$) does and forwards the packet to that switch. However, instead of being forwarded to this intermediate switch in the second hop, the packet is forwarded minimally to its final destination switch. This route only has three hops, unless the current switch is directly connected to the destination switch. In this case, the packet performs two hops. Algorithm 1 describes this routing.

Algorithm 1 Three-hop Valiant

```
if $Sw_{current} \neq Sw_S$ then
    Route minimally to $Sw_D$          ▷ Destination switch
else
    Route minimally to $Sw_I$          ▷ Intermediate switch
```

Fig. 5 illustrates the route followed by a packet when it performs a three-hop path selected by VAL$_{3h}$ in comparison with a four-hop path performed by VAL$_{ori}$ or VAL$_{nta}$. In this scenario, end node S connected to Sw_S is the source of the packet, end node D connected to Sw_D is the destination, and Sw_I is the intermediate switch selected randomly. We omit the information regarding the used VCs per route, since it is equal to the number of hops. Note that four-hop routes require four VCs to prevent deadlocks, while three-hop routes will require three VCs. Regardless of the path, the packet reaches Sw_A in the first hop, since it is used in the minimal route from Sw_S to Sw_I. In the second hop, however, VAL$_{3h}$ selects a minimal route to reach Sw_D through Sw_C which is minimal and it is shorter (three hops) than that selected by VAL$_{ori}$ and VAL$_{nta}$ (four hops).

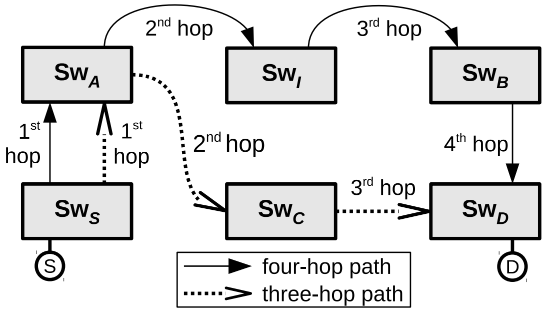


Fig. 5: A three-hop path performed by VAL$_{3h}$ and a four-hop path performed by the other VAL variants.

The implementation of VAL$_{3h}$ in commercial networks is very similar to that of VAL$_{nta}$. We only need to compute three-hop routes and populate the routing tables accordingly. The restrictions required by VAL$_{nta}$, explained before, will be also included in the computing algorithm to prevent the *turn-around* problem.

As VAL$_{3h}$ forwards packets minimally after the first hop, the non-minimal path diversity of the routing algorithm is slightly reduced. However, Slim Fly networks still offer enough path diversity to thee-hop routes, so that routing efficiency is not dramatically affected. Moreover, reducing the average path length to three hops also reduces the number of VCs required to prevent deadlocks to three. As we show in the evaluation section, it is worth the price to pay in terms of reducing routing efficiency for the gain obtained using fewer VCs to prevent deadlocks.

V. Evaluation

This section evaluates our proposals by means of simulation experiments in several Slim Fly network configurations, compared to other routing algorithms. The experiments have been performed in a simulator based on the OMNeT++ framework [13], which has been extended to include the Slim Fly topology, as well as minimal and non-minimal routing algorithms. This simulator models a pipelined switch architecture whose buffers are placed at input ports [14]. Input port buffers size is 128 KB and packet MTU is 4KB). We model credit-based flow control at VC level and virtual cut-through switching policy. Link bandwidth is configured to offer 40 Gbps. Cables length is assumed to be 5 meters and link propagation delay is 25 ns (assuming a delay of 5 ns/m). The internal crossbar has a 2.5 speedup to prevent contention within switches. Output port arbiters implement the *iSlip* [15] algorithm.

We show in Table II the Slim Fly configurations used in the experiments. These configurations have enough end node concentration to ensure full global bandwidth without overdimensioning the network.

TABLE II: Evaluated network configurations

#	Name	q	ξ	k'	p	N_s	k	N
1	*SlimFly-19_10*	13	2	19	10	338	29	3380
2	*SlimFly-29_15*	19	3	29	15	722	44	10830

To make a fair comparison with our proposals, we have evaluated several techniques. In non-minimal path algorithms, we distinguish between the policy which decides if packets follow a minimal or non-minimal path, and the policy which select the non-minimal path followed. The following combinations of routing algorithms have been evaluated:

- *Minimal-path routing (MIN)* is configured with two VCs to prevent deadlocks.
- *Non-minimal path routing (Valiant routing)*:
  - *Original VAL* (VAL$_{ori}$) is configured with four VCs to prevent deadlocks (see Section II-B).
  - *No-turn-around VAL* (VAL$_{nta}$) configured with 4 VCs to prevent deadlocks.
  - *Three-hop VAL* (VAL$_{3h}$) configured with 3 VCs to prevent deadlocks.
- The *UGAL-L* adaptive routing algorithm selects minimal or non-minimal routes, depending on the queue occupancy of the local switch when packets reach the switch in the first hop. Specifically, UGAL selects the minimal path if $Q_{min} \leq 2 \times Q_{val} + T$, where Q_{min} is the queue occupancy of the VC where the packet will be stored if the minimal path is selected; Q_{val} is the queue occupancy of the VC where the packet will be stored if the non-minimal path is chosen; and T is a constant threshold utilized to favor minimal paths [8]. If this inequality is unsatisfied, UGAL-L selects a non-minimal path using VAL. Specifically, the modeled UGAL-based routing algorithms are the following:
  - UGAL-L$_{ori}$ uses VAL$_{ori}$. It is configured with four VCs to prevent deadlocks.
  - UGAL-L$_{nta}$ uses VAL$_{nta}$. It is configured with four VCs to prevent deadlocks.
  - UGAL-L$_{3h}$ uses VAL$_{3h}$. It is configured with four VCs to prevent deadlocks.
- *Progressive Adaptive Routing* (PAR) [8] selects minimal or non-minimal path based on the queue occupancy of the local switch, as UGAL-L does. However, when packets are routed minimally, we have adapted PAR to Slim Fly topology to reevaluate the minimal-path decision when packets are in their second hop[1]. If the non-minimal path is better then the minimal one, PAR routes the packet using VAL, otherwise, the packet stays on the minimal path. This revaluation implies that packets can follow longer paths, thus and additional VC is necessary

[1] In Dragonfly networks, PAR reevaluates the path only when packets are moving within the source group.

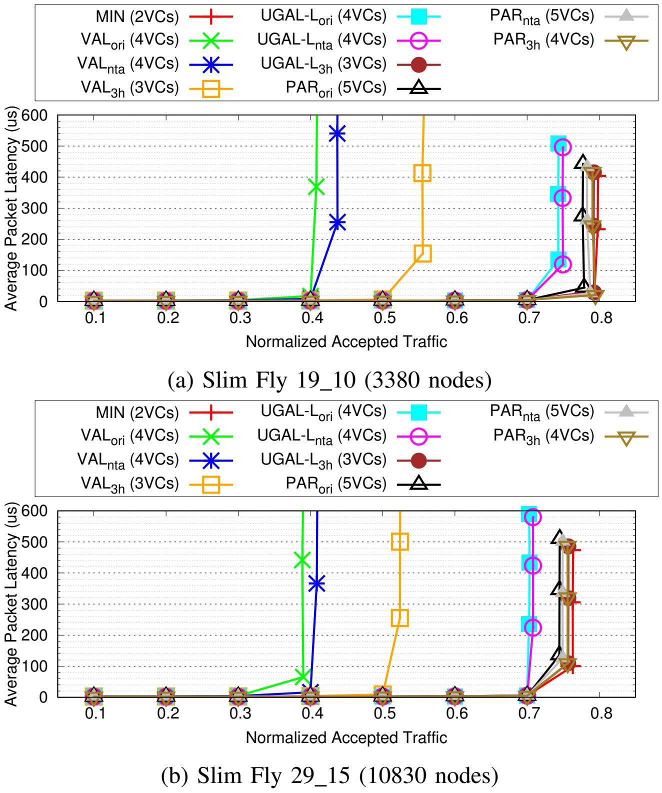
to prevent deadlocks. The non-minimal path routes are established according to the VAL variants:

- PAR$_{ori}$ uses VAL$_{ori}$. It is configured with five VCs to prevent deadlocks.
- PAR$_{nta}$ uses VAL$_{nta}$. It is configured with five VCs to avoid deadlocks.
- PAR$_{3h}$ uses VAL$_{3h}$. It is configured with five VCs to prevent deadlocks.

The following sections show the results of these routing algorithms in the network configurations defined in Table II, under uniform and adversarial traffic patterns. In all experiments we measure the average *packet latency* in ns and the *accepted traffic* normalized against the maximum bandwidth of the sum of all the end nodes maximum generation rate. The traffic load generated in the network is incremental, so that we increase the end node generation rate from 10% of the link speed to 100%, and simulate 10 load points. For each traffic load point, the network warms up for 2 ms. The performance metrics are recorded in steady state during 1 ms.

A. Results with uniform traffic

Fig. 6 shows experiment results of the routing algorithms described above for the network configurations #1 and #2 (see in Table II), under the uniform (random) traffic pattern. This well-known traffic workload consists of each end node generating packets to random destinations. Note that the variations in performance between Fig. 6a and Fig. 6b are around 5% for a network three times bigger in size.



(a) Slim Fly 19_10 (3380 nodes)

(b) Slim Fly 29_15 (10830 nodes)

Fig. 6: Random traffic scenario.

MIN routing achieves the best performance, regardless of network size, because traffic distribution is balanced among all links. In this traffic conditions, the fastest route to choose is usually the minimal one. On the contrary, the three VAL-like routing algorithms always obtain the worst performance because forwarding packets to an intermediate switch, by default, reduces the network performance when traffic is well balanced (because packets require more network resources to perform the extra hops).

Despite their low performance, there are differences among them that it is worth mentioning. VAL$_{ori}$ gets slightly worse results than VAL$_{nta}$ in both network configurations, but the difference decreases considerably as the network size grows, due to the *turn-around* problem becomes less likely for this traffic pattern as the network size grows. VAL$_{3h}$, nevertheless, always outperforms VAL$_{ori}$ and VAL$_{nta}$, and it maintains this difference in the three network sizes. VAL$_{3h}$ routes are shorter than VAL$_{ori}$ and VAL$_{nta}$, reducing the overhead introduced for the intermediate switch forwarding. Moreover, VAL$_{3h}$ requires three VCs to prevent deadlocks, while VAL$_{ori}$ and VAL$_{nta}$ require four VCs.

UGAL-L and PAR routing algorithms partially rely their performance on the path decision and, in this scenario, the fewer non-minimal paths selected, the better. UGAL-L also experiments some improvements (7%) when using our technique UGAL-L$_{3h}$. Between UGAL-L$_{ori}$ and UGAL-L$_{nta}$ the differences are small, but UGAL-L$_{3h}$ performance is always better, regardless the network size. Regarding the PAR variants, PAR$_{ori}$, PAR$_{nta}$, and PAR$_{3h}$ achieve virtually identical results among them, and also compared to UGAL-L$_{3h}$ and MIN. Note that PAR$_{3h}$ requires four VCs to prevent deadlocks, while PAR$_{ori}$, and PAR$_{nta}$ require five VCs.

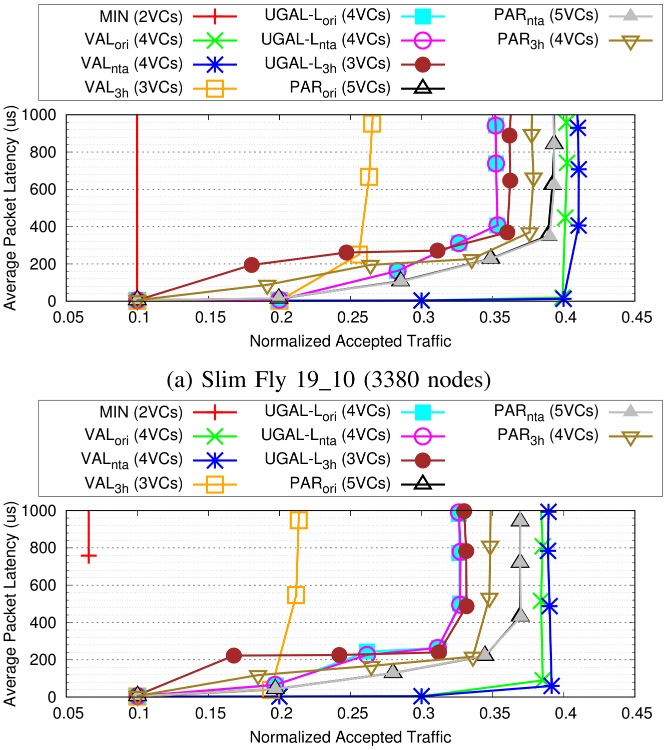
UGAL-L$_{3h}$ and MIN routing are the best options among the others under uniform traffic scenarios. MIN requires two VCs in order to prevent deadlocks, while UGAL-L$_{3h}$ requires three VCs. However, MIN routing algorithm suffers a significant performance degradation when adversarial traffic is used, as the next section explains.

B. Results with adversarial traffic

This section shows experiment results of the routing algorithms described above for the network configurations #1 and #2 (see in Table II), under adversarial traffic pattern. As described in Section III, an adversarial traffic pattern overuses some links connecting switches and provokes packet contention for their access. Every end node receives an amount of traffic that it is able to consume, in contrast with hot spot scenarios, which create network congestion.

We have evaluated the adversarial traffic defined by Algorithm 2. This algorithm, creates an adversarial traffic where every flow performs two hops, for the values of A and B. Parameters A and B depend on the Slim Fly parameters q and ξ (see Table III).

The idea behind this algorithm is that the end nodes connected to a switch send traffic to the end nodes of another switch in the other sub-graph of the Slim Fly network (see



(a) Slim Fly 19_10 (3380 nodes)

(b) Slim Fly 29_15 (10830 nodes)

Fig. 7: Bad-case traffic scenario.

Algorithm 2 Adversarial generator

```
for all $sw_{src} \in [0 \ldots N_S)$ do
    if $sw_{src} < q^2$ then                    ▷ (0,x,y) subgraph
        $sw_{dest} \leftarrow ((sw_{src} + A) \mod q^2) + q^2$
    else                                        ▷ (1,m,c) subgraph
        $sw_{dest} \leftarrow (sw_{src} + B) \mod q^2$
    for all $endNode \in [0 \ldots p)$ do
        $endNode_{src} \leftarrow sw_{src} \times p + endNode$
        $endNode_{dest} \leftarrow sw_{dest} \times p + endNode$
        $adversarial[endNode_{src}] \leftarrow endNode_{dest}$
```

TABLE III: Parameters used in Algorithm 2

q	ξ	A	B
7	3	4	3
11	2	1	10
13	2	2	11
17	3	3	14
19	3	6	13

Section II-A), i.e., end nodes from switches $(0, x, y)$ send packets towards end nodes of switches $(1, m, c)$, and vice versa. If A and B are set with other values, it creates an adversarial traffic where some flows perform two hops, but others only perform one hop.

Fig. 7 displays the network performance results under the adversarial traffic pattern defined above. MIN routing obtains the lowest performance as it is roughly affected by adversarial traffic. As Section III, MIN routing performance is limited by $\frac{1}{p}$, so that MIN performance is $\frac{1}{10} = 0.1$ (i.e., 10% of the maximum load) for network configuration #1 ($p = 10$), and $\frac{1}{15} = 0.07$ (i.e., 7% of the maximum load) for network configuration #2 ($p = 15$).

VAL$_{nta}$ and VAL$_{ori}$ obtain always the best performance regardless the network size. VAL$_{3h}$ performance, in contrast, is reduced as network size grows due to the bias to 4-hop routes reduces path diversity and also the obtained performance. On the other hand, UGAL-L$_{3h}$ performance is identical to that of UGAL-L$_{nta}$ and UGAL-L$_{ori}$, but using three VCs to prevent deadlocks instead of four VCs. PAR is better than UGAL-L when using the same Valiant variant. Overall, in this scenario the best algorithm is VAL$_{nta}$, but PAR$_{3h}$ and UGAL-L$_{3h}$ are also a valid options if we consider the uniform traffic results. In general, UGAL-L$_{3h}$ introduces less computing overhead than PAR$_{3h}$, because path reevaluation is unnecessary, and it only requires three VCs to prevent deadlocks.

VI. Related Work

Interconnection network routing algorithms have been widely studied because topology performance strongly depends on them. Routing algorithms can be classified depending on whether the path is minimal or non-minimal. Another routing taxonomy classifies them into deterministic, oblivious, and adaptive. Deterministic algorithms always choose the same path between two nodes, ignoring path diversity and not balancing traffic. Oblivious algorithms choose a route without considering the network state to balance the load among network paths. They can potentially improve the performance at the cost of adding additional complexity that could lead to performance degradation. Adaptive algorithms select the path in function of the state of the network. These algorithms, theoretically, should outperform the others, if properly designed, but they introduce some overhead in the computation of additional paths.

Routing algorithms and network topologies are usually evaluated under adversarial traffic patterns in order to know the performance in extreme situations [16]. The work where Slim Fly was introduced identifies an adversarial case [5]. Moreover, another work tried to generate an adversarial traffic based on the average path length, but without success for Slim Fly networks [17]. However, as far as we know the study of Slim Fly routing algorithms is still open to explore. For the time being, Slim Fly networks have been tested under Valiant and *Universal Globally Adaptive Load-Balancing* (UGAL) [5]. Valiant's algorithm is a non-minimal oblivious routing whose aim is balance the network traffic regardless of the traffic distribution [6]. UGAL [7] is a non-minimal adaptive routing that selects a minimal or a non-minimal path based on queue occupancy, avoiding congested channels by using less-congested alternative routes. UGAL-L is the local version that considers the queue length of the local switch. UGAL-G, by contrast, is the global version that compares the queue occupancy in all switches in the network, thus it cannot implemented in real systems.

Dragonfly topology [3], in contrast with Slim Fly topology, has a wider background related to oblivious and adaptive routing. As both topologies have some similarities, the solutions proposed for the Dragonfly may be portable for the Slim Fly, although some adjustments are required. In that sense, four indirect adaptive routing are proposed [8]: Credit Round Trip (CRT), Progressive Adaptive Routing (PAR), Piggyback (PB), and Reservation (RES). Furthermore, other works suggested also the use of a history-window approach in Dragonfly networks [18]. Most of them try to select better paths based on the status of global channels, which Slim Fly lacks, thus they are not interesting for Slim Fly topology. PAR algorithm, nevertheless, posses a reevaluation mechanism that can be extrapolated to Slim Fly, but eliminating the concept of group, which is specific for Dragonfly networks.

The idea of modifying the Valiant's algorithm was also used for Dragonfly networks [18], although this proposal is specific for that topology and it cannot be extrapolated to Slim Fly. This modification only solves some adversarial situations only appearing in Dragonfly networks under specific traffic cases.

VII. Conclusion

Slim Fly networks maximize the number of end nodes in a cluster for a given diameter and switch radix, offering high performance, low latency, reduced power consumption and saving network costs. However, Slim Fly performance may be spoiled by adversarial traffic patterns. Adversarial traffic scenarios can be mitigated by a routing algorithm that balances the traffic among the available paths in the network, such as the Valiant (VAL) algorithm. This algorithm, however, presents some drawbacks when selecting non-minimal paths: the overuse of network resources (e.g., virtual channels) and the turn-around problem which causes packets visiting the same switch twice. We have proposed in this paper several improvements to non-minimal (VAL) and adaptive (UGAL and PAR) routing algorithms in order to avoid the problems mentioned above. Simulation results under uniform and adversarial traffic scenarios demonstrate that the proposed techniques are able to reduce the average length of the non-minimal routes, reducing the number of required virtual lanes to prevent deadlocks. As a future work, we plan to extend this idea to other cost-effective low-diameter topologies [19].

Acknowledgment

This work has been jointly supported by the Spanish MINECO and European Commission (FEDER funds) under the projects TIN2012-38341-C04 and TIN2015-66972-C5-2-R (MINECO/FEDER), and the FPI grant BES-2013-063681, and by Junta de Comunidades de Castilla- La Mancha under the project PEII-2014-028-P. Jesus Escudero-Sahuquillo is funded by the University of Castilla-La Mancha (UCLM) and the European Commission (FSE funds), with a contract for accessing the Spanish System of Science, Technology and Innovation, for the implementation of the UCLM research program (UCLM resolution date: 31/07/2014).

References

[1] Top500.org, "Top 500 List," 2017.

[2] J. Kim, J. D. Balfour, and W. J. Dally, "Flattened Butterfly Topology for On-Chip Networks," *IEEE Computer Architecture Letters*, vol. 6, pp. 37–40, Feb. 2007.

[3] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-Driven, Highly-Scalable Dragonfly Topology," in *35th International Symposium on Computer Architecture (ISCA) 2008, June 21-25, 2008, Beijing, China*, pp. 77–88, IEEE Computer Society, June 2008.

[4] R. Peñaranda, C. G. Requena, M. E. Gómez, P. López, and J. Duato, "The k-ary n-direct s-indirect family of topologies for large-scale interconnection networks," *The Journal of Supercomputing*, vol. 72, pp. 1035–1062, Mar. 2016.

[5] M. Besta and T. Hoefler, "Slim Fly: A Cost Effective Low-Diameter Network Topology," in *International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2014, New Orleans, LA, USA, November 16-21, 2014* (T. Damkroger and J. Dongarra, eds.), pp. 348–359, IEEE, Nov. 2014.

[6] L. G. Valiant, "A Scheme for Fast Parallel Communication," *SIAM J. Comput.*, vol. 11, no. 2, pp. 350–361, 1982.

[7] A. Singh, *Load-Balanced Routing in Interconnection Networks*. PhD thesis, Standford University, Mar. 2005.

[8] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," in *36th International Symposium on Computer Architecture (ISCA 2009), June 20-24, 2009, Austin, TX, USA* (S. W. Keckler and L. A. Barroso, eds.), pp. 220–231, ACM, 2009.

[9] B. D. McKay, M. Miller, and J. Širáň, "A Note on Large Graphs of Diameter Two and Given Maximum Degree," *Journal of Combinatorial Theory, Series B*, vol. 74, pp. 110–118, Sept. 1998.

[10] P. R. Hafner, "Geometric realisation of the graphs of McKay–Miller–Širáň," *Journal of Combinatorial Theory, Series B*, vol. 90, pp. 223–232, Mar. 2004.

[11] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection Networks: An Engineering Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2003.

[12] P. Yébenes, J. Escudero-Sahuquilllo, P. J. García, F.-J. Alfaro-Cortés, and F. J. Quiles, "Providing differentiated services, congestion management, and deadlock freedom in dragonfly networks with adaptive routing," *Concurrency and Computation: Practice and Experience*, 2016. cpe.4066.

[13] P. Yebenes, J. Escudero-Sahuquillo, P. J. García, and F. J. Quiles, "Towards modeling interconnection networks of exascale systems with omnet++," in *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2013, Belfast, United Kingdom, February 27 - March 1, 2013*, pp. 203–207, 2013.

[14] T. M. Pinkston and J. Duato, "Appendix F: Interconnection Networks," in *Computer Architecture: A Quantitative Approach* (Elsevier, ed.), Morgan Kaufmann, 2012.

[15] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 188–201, Apr. 1999.

[16] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[17] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla, "Measuring and understanding throughput of network topologies," in *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 761–772, Nov 2016.

[18] J. Won, G. Kim, J. Kim, T. Jiang, M. Parker, and S. Scott, "Overcoming far-end congestion in large-scale networks," in *21st IEEE International Symposium on High Performance Computer Architecture, HPCA 2015, Burlingame, CA, USA, February 7-11, 2015*, pp. 415–427, IEEE Computer Society, 2015.

[19] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. Hoefler, "Cost-Effective Diameter-Two Topologies: Analysis and Evaluation," in *In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC15)*, ACM, Nov. 2015.