ACOR: Adaptive Congestion-Oblivious Routing in Dragonfly networks

M. Benito, P. Fuentes, E. Vallejo, R. Beivide

Computer Architecture Group. University of Cantabria, Avda. de los Castros s/n, Santander, Spain.

*Corresponding author: Mariano Benito.

Email addresses: mariano.benito@unican.es (M. Benito), pablo.fuentes@unican.es (P. Fuentes), enrique.vallejo@unican.es (E. Vallejo), ramon.beivide@unican.es (R. Beivide)

Abstract

Low-diameter network topologies require non-minimal routing to avoid network congestion, such as Valiant routing. This increases base latency but avoids congestion issues. Optimized *restricted* variants focus on reducing path length. However, these optimizations only reduce paths for local traffic, where source and destination of each packet belong to the same partition of the network. This paper introduces *ACOR: Adaptive Congestion-Oblivious Routing*. ACOR leverages the *restricted* and *recomputation* mechanisms to reduce path length for local and global traffic, and extend it when the network conditions are adverse. ACOR relies on a sequence of misrouting policies ordered by path length. A hysteresis mechanism improves performance and avoids variability in the results. The ACOR mechanism can be combined with other non-minimal routing mechanism such as Piggyback. Results show that ACOR improves base latency in all cases, up to 28% standalone and up to 25.5% when combined with Piggyback, while requiring a simple implementation.

Keywords: Valiant routing, ACOR, Restricted Valiant, Recomputation, Piggyback, Dragonfly

2010 MSC: 68M10,

2010 MSC: 90B18,

2010 MSC: 68M14

1. Introduction

Low-diameter network topologies present low average distance between nodes in order to reduce latency. Some examples of such topologies include Flattened Butterflies [1], Dragonflies [2], Slim Flies [3] or Projective Networks [4]. These networks employ a low number of switches and links for a given network size, what leads to low power consumption and low installation costs while achieving high scalability for a given diameter [5]. To scale to a large number of nodes, minimal path diversity is restricted. In this situation, non-minimal routing is required to avoid low performance under adversarial traffic patterns.

Valiant routing [6] is a non-minimal mechanism which randomizes traffic sending packets first to a randomly selected *intermediate* switch, and then minimally to the actual destination. This balances the use of the network links and avoids congestion under traffic patterns which overload certain links, but doubles the longest path in the network.

Two simple optimizations to Valiant routing have been introduced recently in [7]: Valiant *with recomputation* recomputes the intermediate switch when the source router cannot inject traffic. With such recomputation, it avoids transient congestion, increases maximum throughput and reduces throughput variability. *Restricted* Valiant restricts the allowed set of intermediate switches for certain destinations, to reduce the length of the path and, therefore, latency. Restricted Valiant is particularly useful for *local* traffic, where both source and destination switches belong to the same network *partition*, e.g., the same group in a Dragonfly topology. In such case, restricting the intermediate switch to be in the same group avoids long paths that do not remove congestion issues.

However, restricted Valiant is not effective in the frequent case of *global* traffic where source and destination are in distant parts of the network, for example, different groups of a Dragonfly topology. This happens under pathological adversarial traffic patterns where a short non-minimal path generates network bottlenecks. The safe approach used in such case has been to employ the complete Valiant path, doubling the length of minimal routing.

This paper extends the original work in [7] by introducing *Adaptive Congestion-Oblivious Routing* (ACOR). ACOR is based on Valiant but reduces non-minimal path length to improve latency without requiring any global knowledge of the traffic pattern. It relies on two key observations: first, short non-minimal paths can be used as long as no pathological traffic pattern is present; second, when such traffic occurs, congestion is quickly propagated to the sources and hinders traffic injection, requiring multiple retries per packet. Based on these observations, ACOR leverages the optimizations from [7] to automatically adapt the path length under *global* traffic. When injecting *global* traffic, switches select misrouting policies that generate short non-minimal paths by default, selecting only *intermediate* switches adjusted to such short paths. This achieves lower latency than the original Valiant mechanism. To avoid generating network bottlenecks, switches change the misrouting policy to use longer paths when the *recomputation* of the intermediate switch is used too often. This provides safe routing under pathological traffic patterns. To provide stability, a simple hysteresis mechanism regulates the changes in the sequence of misrouting policies.

The ACOR mechanism can be also applied to adaptive routing. The reference for source-adaptive routing in a Dragonfly is Piggyback (PB, [8]). We propose *Piggyback-ACOR (PB-ACOR)*, which implements Piggyback routing selecting packet-by-packet between minimal or ACOR at injection.

In particular, the main contributions of the paper are the following:

- ACOR, an adaptive but congestion-oblivious routing mechanism that adapts the misrouting policy to avoid network congestion.
- PB-ACOR, a congestion-conscious source-adaptive routing mechanism which selects between minimal or ACOR routing at injection.
- A thorough evaluation of the proposals. Simulation results show that ACOR is competitive in throughput with Valiant while reducing base latency up to 28%. PB-ACOR improves latency at low loads (up to 25.5%), and presents high throughput at high loads under most traffic patterns.

The rest of the paper is organized as follows: Section 2 presents the required background. Section 3 summarizes the *restricted* and *recomputation* optimiza-

tions of Valiant already presented in [7]. Section 4 introduces the main proposals of this work: ACOR, the hysteresis mechanism, and PB-ACOR. Section 5 presents the evaluation methodology, and Section 6 shows the performance results. Finally, Section 7 discusses the related work and Section 8 concludes.

2. Background

2.1. Valiant routing (VAL)

The original randomization-based **Valiant routing** mechanism (VAL, [6, 9]) obtains $\mathcal{O}(\log N)$ packet delivery time for a given permutation of traffic in a hypercube network of N processors. It is based on two phases: Phase A sends each packet to a randomly selected network switch, whereas Phase B sends the packet from this *intermediate* switch to the final destination. The original implementation of Phase A in [6] follows a dimension-order process where the (only) link in each dimension of the hypercube in each current switch is traversed or not with the same random probability. This is equivalent to selecting a random intermediate destination at injection time, but requires less bookkeeping since the intermediate node is not recorded in the packet header.

Valiant routing is completely *oblivious*: the path for each packet is selected randomly, but it does not depend on the status of the network (congestion) or the destinations of the packets sent by other nodes (traffic pattern).

Valiant routing has also been proposed for other high-radix topologies to avoid congestion under adversarial traffic patterns, such as the Flattened Butterfly [1], the HyperX [10], the Dragonfly [2], the Slim Fly [3] or the Projective networks [4]. It is also the basis of multiple non-minimal adaptive routing algorithms, which select between minimal and Valiant paths (such as UGAL [11], Piggyback [12], PAR [12] or OFAR [13]).

2.2. Dragonfly topology

The Dragonfly [2] is a low-diameter topology based on high-radix switches deployed following a hierarchical direct layout. The first level comprises fully

connected groups of switches conforming a virtual high-radix switch. These groups can be connected using different second level interconnection patterns. Each group is directly connected through *local* links, while different groups are connected through *global* links. This topology can be described using 3 parameters: the number of nodes connected on each switch (p), switches on each group (a), and *global* links on each switch (h). This work assumes a *canonical dragonfly* which uses a complete graph for the second level; it also assumes a *Palmtree* global link arrangement.

2.3. Traffic patterns and routing mechanisms in the Dragonfly

This section presents different traffic patterns that may occur in the Dragonfly, and routing mechanisms that properly handle them, along with a discussion on the impact of different misrouting policies.

Under *random uniform* (UN) traffic, the destination of each packet is uniformly selected among all nodes in the network. ***Minimal* routing** (MIN, [2, 14]) is appropriate for such traffic. This routing is hierarchical and requires at most as many hops as the network diameter. Packets are sent first to the destination group making a local (L) and a global (G) hop, and then, to the destination switch making another local (L) hop. This path is denoted LGL. For some pairs of nodes, the path can be shorter since some hops are not required. For example, local traffic (where both source and destination nodes are in the same group) only requires a single local hop L.

By contrast, under *adversarial* traffic patterns (ADV), minimal routing is not suitable because it gathers all traffic into a fraction of the available links, generating bottlenecks and reducing performance. Figure 1 presents an example of *adversarial* traffic (ADV+i), in which all nodes in a given group send their traffic to the same destination group placed i groups away. The only global link between these two groups, departing from the switch denoted S_{out}, becomes the bottleneck under *minimal* routing. To avoid this bottleneck, non-minimal routing should be employed [2]. Valiant routing, explained in Section 2.1, randomizes traffic and avoids such bottlenecks. However, the additional hops from

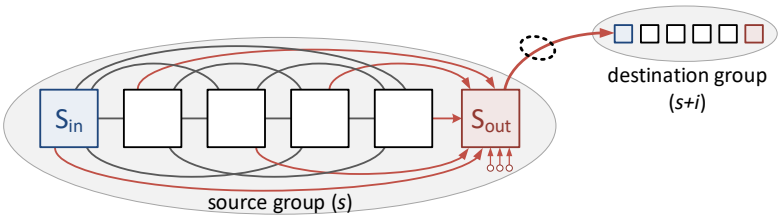


Figure 1: Adversarial traffic (ADV+i) in a Dragonfly network. Traffic from each source group s goes to group $s+i$ through a single global link.

Phase A, where packets are diverted to a random switch, double the path length ($LGL-LGL$) and result in high base latency.

For certain adversarial traffic patterns, the complete Valiant path is unnecessarily long. For example, under the *adversarial-local* (ADVl) pattern, all traffic from the p nodes of one switch goes to the p nodes in another switch within the same group, and the single local link between switches becomes a bottleneck with *minimal* routing. ***Restricted-Valiant*** (RVAL), first introduced in [7] and reviewed in Section 3.1, attacks this problem by using a Phase A with a single non-minimal local hop, resulting in paths $L-L$.

However, shortening the path in Phase A from its reference $LGL-$ introduces pathological bottlenecks under certain adversarial traffic patterns, as explained later and depicted in Figure 6 in the evaluation section. The non-minimal global hop G is clearly required to remove the bottleneck in the global link otherwise caused by adversarial traffic. Paths with a single non-minimal hop have been used in previous work [15], but they have some limitations, which are described next.

The non-minimal path is determined by the followed *misrouting policy*. This policy comprises two aspects: the *Global misrouting policy* [16] determines which global links can be used in Phase A, restricting the eligible groups for the selection of the *intermediate* switch; the *switch selection policy* determines which switch in the intermediate group can be selected as the intermediate switch. The existence of a first local hop in the reference path $LGL-$ in Phase A is determined by the global misrouting policy.

Two global misrouting policies were studied in [16], denoting ***Current Router Global*** (CRG) the policy that restricts the intermediate switch to the groups directly connected to the source switch, whereas ***Random Router Global*** (RRG) can select the intermediate switch in any group. With CRG the first local hop in the reference path in Phase A is absent, and with RRG it can be present. Similarly, the switch selection policy determines if any switch in the intermediate group can be the intermediate switch, or if it needs to be directly connected to the source group. In the former case, the second local hop in the reference path in Phase A is present; in the latter, it is missing. We refer to *misrouting* ***to switch*** or *misrouting* ***to group*** to these two cases respectively. Altogether, there are four misrouting policies: *CRG to group* ($G - LGL$) and *CRG to switch* ($GL - LGL$) without the first local hop, and *RRG to group* ($LG - LGL$) and *RRG to switch* ($LGL - LGL$) with the first hop. These policies are summarized in Table 1 from Section 4.

The original proposal for Valiant routing in the Dragonfly in [2] employs the *RRG to group* policy, saving one hop from the full Valiant path (*RRG to switch*). However, as identified in [13], the lack of the second local hop in Phase A introduces pathological performance issues under the *adversarial+h* (ADV+h) variant of the adversarial traffic (with $i = h$). Under such traffic pattern, omitting the second local hop in Phase A concentrates all the traffic from different source groups in one local link at the intermediate group, which becomes a bottleneck and bounds throughput by $1/h$.

Similarly, the *CRG* policy, which omits the first local hop, suffers under *adversarial-consecutive* traffic (ADVc, [17]). Figure 2 presents an example of ADVc, in which all nodes of a source group send traffic to nodes that belong to the consecutive h groups, concentrating traffic in a single switch in the source group and making congestion detection more difficult. Eventually, the use of the *CRG* policy translates into network unfairness, particularly when using source-adaptive routing, and higher congestion after saturation, as studied in Section 6.2.

Finally, multiple *adaptive* routing mechanisms select between *minimal* or

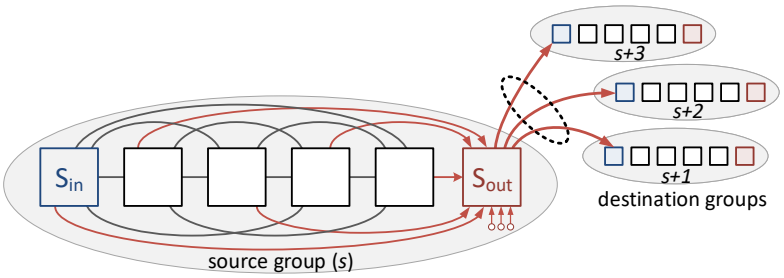


Figure 2: Adversarial-consecutive (ADVc) traffic in a Dragonfly network with $h = 3$. Traffic from each source group s goes to the next $h = 3$ consecutive groups $(s + 1, s + 2, s + 3)$ through the circled global links.

(some implementation of) *Valiant* routing based on the congestion of the network. UGAL [2] implements this selection at injection (source routing) using switch-local information. **Piggyback** (PB) [8] extends UGAL with a mechanism that detects saturated global channels, and it floods the state information of every global link to all the switches in a group. Progressive adaptive routing (PAR, [8]), and RLM/OLM [18] are examples of adaptive mechanisms that adapt the path on transit. In general, given similar performance, source routing is preferred since transit switches do not need to track congestion and reevaluate routing decisions.

3. Base proposal: Restricted Valiant with recomputation

This section reviews the proposal from [7], which consists of two optimizations to the original implementation of Valiant routing in low-diameter networks. The first modification shortens non-minimal paths under certain conditions in order to improve performance, whereas the second one increases path availability for any given packet in case of congestion.

3.1. Restricted Valiant (RVAL)

Valiant routing, described in Section 2.1, does not consider the case of hierarchical networks (Dragonfly, multilevel fat-tree) or topologies consisting of

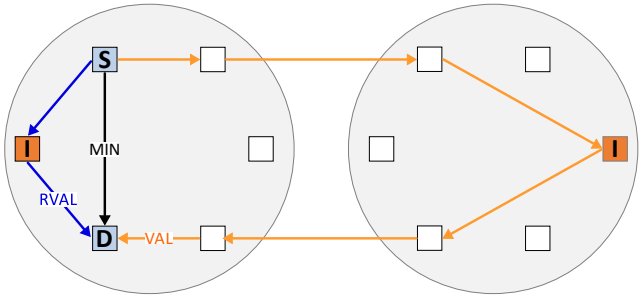


Figure 3: Example of turn-around problem in a Dragonfly with Minimal (MIN), Valiant (VAL) and Restricted Valiant (RVAL). Source (S) and destination (D) switches are in the same group; as is the intermediate switch (I) with RVAL. RVAL avoids the pathological congestion within the group with MIN, and has a shorter path than VAL.

multiple orthogonal dimensions (HyperX network), where the source and destination may be located in the same partition (group in the case of the Dragonfly, *pod* in the fat-tree) or subset of dimensions. In such cases, diverting the packets non-minimally forces them to travel outside of the original partition and back, increasing path length without contributing to congestion avoidance.

This issue represents a more general case of the *turn-around problem* identified by Yébenes *et al.* in the Slim Fly topology, where packets visit the same switch twice in their path, turning around and going back through the same route [19]. In the turn-around problem, the subsegment of the path between the two visits of the same switch can be omitted without negatively impacting the benefits of packet randomization. However, in the general case paths A (towards the intermediate switch) and B (from the intermediate switch to the destination) may not overlap while the complete route is still unnecessarily long. Figure 3 represents an example of this issue in a Dragonfly network with *global trunking*, this is, more than one global link connecting pairs of groups. Under Valiant routing (VAL) the selected random intermediate switch is in a remote group, and no switch is traversed twice because the two paths to the intermediate switch and to the destination are disjoint. However, the resultant

path is needlessly long.

***Restricted Valiant* (RVAL)** avoids this problem by limiting the selection of the non-minimal path in Phase A: if the source and destination terminals belong to the same partition, it selects an intermediate switch in that partition. In the Dragonfly network, this case corresponds to intra-group traffic, and Restricted Valiant only selects an intermediate switch from the local group, which shortens the non-minimal paths. Furthermore, it still avoids the problem associated to pathological congestion with local traffic communications, for example when all computing nodes attached to a switch communicate with nodes in the next switch, as observed with stencil workloads in [20]. As displayed in Figure 3, restricting selection of the intermediate switch to the local group (RVAL) avoids the congestion issue (in the *minimal* link) while providing a shorter path. In the general case where source and destination are located in different partitions, Restricted Valiant behaves as the original definition of Valiant.

Restricted Valiant can be also applied to the selection of the non-minimal path in source-adaptive Piggyback routing [8]. When both source and destination are in the same group, the non-minimal path considered by PB is restricted, so the packet is first sent to an intermediate switch in the local group.

3.2. Valiant with recomputation

Valiant routing randomizes the paths of the packets to balance the use of network resources. However, transient congestion may appear, which generates Head-of-Line Blocking (HoLB) and delays injection.

This issue can be mitigated with **Valiant *with recomputation***. In this variant, the selection of the intermediate destination for each packet is performed at the source router, as in the original mechanism. Whenever a packet is at the head of its injection buffer and the required output port (at the beginning of phase A) is blocked, the routing mechanism recomputes a new random intermediate destination. This recomputation may change the output port and ease packet injection, increasing throughput. Such recomputation may occur several times, until the packet is injected; once the packet is in-transit, the Valiant

destination does not change. Valiant with recomputation can be combined with the *restricted* mechanism from Section 3.1, limiting the recomputation of the intermediate switch to the subset of allowed switches.

Unlike original Valiant, Valiant with recomputation is not oblivious, because the intermediate destination is modified depending on the status of the network. According to the taxonomy in [21], Valiant with recomputation is *adaptive* but *congestion-oblivious*, because it routes based on the availability of output ports, and not on an estimation of the network congestion.

The idea of recomputing the intermediate switch can be also applied to Piggyback routing: when the required destination port for the selected intermediate switch is blocked, the switch recomputes the random intermediate switch and performs a new routing decision. With this modification, Piggyback with recomputation remains *adaptive* and *congestion-aware* as in the original PB implementation, because it routes based on the availability of output ports (due to recomputation), but also based on an estimation of the network congestion to determine if a minimal or non-minimal path should be used.

4. Adaptive Congestion-Oblivious Routing (ACOR)

This section introduces ACOR: Adaptive Congestion-Oblivious Routing. After an initial overview, the implementation is presented in Section 4.2 and the application to Piggyback adaptive routing is discussed in Section 4.3.

4.1. Motivation and overview

This subsection motivates the use of a routing which combines short and long paths for Phase A based on the network conditions, considering the trade-off between the length of the path in Phase A and performance results under pathological traffic patterns.

As explained in Section 2.3, combining the *global misrouting policy* (CRG or RRG) and the *switch selection policy* (group or switch), four *misrouting policies* can be used to generate the path for Phase A which precedes the minimal path

Table 1: Misrouting policies. Acronyms defined in Section 2.3.

Misrouting policy	**Max. Phase A path length**	**Longest path**
RRG to switch (RRG-Sw)	3 hops	$LGL - LGL$
RRG to group (RRG-Gr)	2 hops	$LG \ - LGL$
CRG to switch (CRG-Sw)	2 hops	$GL - LGL$
CRG to group (CRG-Gr)	1 hop	$G \ - LGL$

LGL. They are presented in Table 1. The three shorter options suffer performance issues under pathological traffic patterns, as explained in Section 2.3. Usually, the trade-off has been solved using the longest path [14, 22], which avoids any pathological situation at the cost of the largest base network latency.

The goal of ACOR is to optimize the common case providing minimal latency, while supporting pathological traffic patterns with longer paths. This is implemented by adapting the selected misrouting policy to the network conditions. The implementation of the misrouting policies in Table 1 is based on the restriction mechanism in Section 3.1. Indeed, a misrouting policy can be seen as applying a restriction on the allowed intermediate switches under global traffic: *CRG to group* restricts the selection of the random intermediate destination to switches directly connected to the source switch, but belonging to different groups, *CRG to switch* restricts to any of the switches in groups directly connected to the source switch, and *RRG to group* restricts to any of the switches directly connected to the source group[1].

4.2. Adaptive Congestion-Oblivious Routing implementation

ACOR employs a *misrouting policy sequence*, which is a sequence of misrouting policies, ordered from the shortest to the longest path. The amount of policies in a given sequence is defined as the *sequence level*. Table 2 presents

[1] Note that the source group could be also selected as the intermediate destination; in such case, the hop G in Phase A is omitted.

Table 2: Misrouting policy sequences. Paths summarized in Table 1.

Name	**Label**	**Sequence**				
Two-levels (A)	2LA	CRG-Gr		——→		RRG-Sw
Two-levels (B)	2LB			CRG-Sw	→	RRG-Sw
Three-levels	3L	CRG-Gr	→	CRG-Sw	→	RRG-Sw
	Length	1		2		3

three sequences used in the current work, with two levels (2LA and 2LB) or three levels (3L). The selection of these specific policies is discussed in Section 4.2.1.

The sequence used is fixed for a given implementation. ACOR switches from one policy to another in the sequence based on the *recomputation* mechanism in Section 3.2. The *ACOR level*, or simply *level*, defines the currently selected misrouting policy from the sequence. When a packet cannot be injected because the output port is blocked, its path is recomputed. In such case, the ACOR level for the given packet can change towards longer paths before recomputing the path. When a routing is computed, the intermediate destination is selected according to the restrictions imposed by the current ACOR level.

An ACOR level can be maintained per individual packet or per switch. In the latter option, all the packets injected in each switch during a certain period of time are routed following the sequence imposed by the switch ACOR level. These two alternative implementations denoted *ACOR-Packet* and *ACOR-Switch* respectively are described in Section 4.2.2.

4.2.1. Selection of a misrouting policy sequence

Table 1 presents the available misrouting policies in the Dragonfly. Since there are policies with three different maximum path lengths, it makes sense to select sequences with two or three policies, in increasing order of path length. All the sequences need to end in the largest policy (RRG-Sw), which corresponds to the original Valiant definition and avoids any pathological congestion introduced

by shortening the path in Phase A. Short paths are used first, to reduce base latency in absence of congestion.

Two different policies with length 2 are present in Table 1: CRG-Sw ($GL-$) and RRG-Gr ($LG-$). Compared to CRG-Gr ($G-$), each of these policies solves one different pathological congestion problem, as discussed in Section 2.3. The second local hop in CRG-Sw avoids the pathological congestion in the intermediate group under ADV+h traffic, whereas the first local hop in RRG-Gr avoids the unfairness and certain congestion to a lesser extent under ADVc traffic.

The selection of which of these policies to use in a sequence is based on two arguments. First, it is relevant which of the two problems occurs at a lower load. Saturation caused by ADV+h occurs at load $1/h$ phits/node/cycle [13], for example 0.12 to 0.16 phits/node/cycle for h in $\{6-8\}$. The unfairness and congestion under ADVc occur close to saturation, near 0.5 phits/node/cycle [17]. These effects are observed in the evaluation in Section 6. Second, the problems under ADVc lie in the source group, while congestion under ADV+h occurs in a remote group (the intermediate Valiant group), which is more difficult to detect at injection time to make an accurate and early decision. For both reasons, it is reasonable to select CRG-Sw over RRG-Gr, since it solves the congestion in the remote group that would otherwise appear at low loads.

With the selection of the length-2 misrouting policy CRG-Sw, the three resulting sequences are presented in Table 2. Two sequences with two levels (2L) are considered: 2LA presents the lowest base latency by starting with CRG-Gr but switches to longer paths at lower loads; 2LB gives an intermediate latency for a wider range of loads. The three-level policy 3L presents a more complex implementation, but tries to optimize a wide range of loads.

4.2.2. ACOR level management: per-packet and per-switch.

The *ACOR level* of each packet at the head of an injection buffer indicates which misrouting policy from the sequence is used when routing the packet. The *recomputation* mechanism is called when packets cannot be injected, which is a sign of congestion issues created by the traffic pattern, the load and the

current *ACOR level*. This mechanism is leveraged to raise the level when packets get blocked repeatedly. Each ACOR level is managed independently, but the implementation details differ when the management occurs per packet or switch.

In *ACOR-Packet* all packets start with the minimum level, so the shortest path is selected by default. Before the recomputation is performed, the ACOR level of each packet is raised when the output port is unavailable. When the ACOR level reaches the longest policy (RRG-Sw, see Tables 1 and 2), it remains in such policy until the packet is injected. Since the amount of recomputation increases with the offered load and pathological congestion issues in the network, packets quickly adapt to use longer paths in presence of congestion.

In *ACOR-Switch* an *ACOR switch level* is derived from blocked output ports throughout the switch. All the packets injected in the switch during a certain period of time are routed following the sequence imposed by this level. The ACOR level of the switch is initialized with the first level of the sequence and increases and decreases according to the amount of blocked packets. Since this value adds information from blocked packets of many individual ports, it does not increase with every blocked packet. Instead, different thresholds are used, together with a hysteresis mechanism to provide stability and avoid oscillations.

For each transition in the misrouting policy sequence, ACOR-Switch employs two thresholds to increase (IT_1, IT_2) and decrease (DT_1, DT_2) the ACOR switch level. A blocked packet counter is maintained and a *hysteresis interval* (HI) is defined, with values determined empirically and presented in Section 6.2.2. The blocked packet counter is reset at the end of every hysteresis interval. The switch level increases when the number of blocked packets exceeds the current increase threshold, without waiting for the end of the hysteresis interval. By contrast, the level is decreased only when the number of blocked packets is lower than the current decrease threshold at the end of the interval.

4.3. Adaptive piggyback with ACOR

ACOR can be used as the basis of a non-minimal source-adaptive routing mechanism, such as UGAL [2] or Piggyback [8]. These mechanisms select min-

imal or non-minimal paths at injection based on an estimation of the network congestion. In an ACOR-based implementation, the non-minimal path is defined by the current level in the misrouting policy sequence.

In these mechanisms, packets may be blocked when they want to advance following minimal or non-minimal paths. Packets that are blocked when trying to follow minimal paths are not considered for the increase of the ACOR level or the hysteresis mechanism in ACOR-Switch. For this reason, appropriate thresholds may differ from the original ACOR implementation.

5. Evaluation methodology

This section introduces the simulation tool used in this work and its configuration parameters. The network simulator described in Section 5.1 is used to implement our proposal explained in Section 4 and to compare it with state of the art routing proposals as listed in Section 5.3 under a range of network loads, as described in Section 5.2.

5.1. Simulation infrastructure

We employ the FOGSim network simulator [23] to evaluate the performance of our routing proposed in Section 4. In our evaluation, we model a *canonical Dragonfly* [24] network with the configuration parameters listed in Table 3. The network forms complete graphs in both topological levels and follows a *Palmtree* global link arrangement with $p = 6$ computing nodes per switch, $a = 12$ switches per group and $h = 6$ global links per switch. These parameters lead to a network with $5,256$ terminals, which is representative of realistic HPC systems. The cycle-accurate simulator models input-output-buffered switches operating at 1GHz and employing multiple virtual channels to avoid routing deadlock. The VC index used increases in each hop, following the idea presented in [2]. Links have a bandwidth of 200 Gbps and latencies which correspond to cables of 3 and 30 meters for local and global links respectively.

Table 3 also shows configuration parameters for the *Piggyback* and *ACOR* routing mechanisms, as described in Subsection 4.2, unless otherwise stated in

the evaluation results. All the evaluation results have been obtained through a battery of several simulations.

5.2. Traffic patterns

The size of the simulated network makes full-system or trace-based simulation impractical. In our evaluation, the network is fed with synthetic traffic, where each node injects frames according to a Bernoulli process with a variable load, alike to other network simulation experiments [25]. The communications of the synthetic traffic follow one of the following patterns:

- *Random Uniform* (UN), in which the destination of a frame is any randomly selected network node, other than the source.
- *Adversarial* (ADV+i), in which the destination of a frame is selected randomly from all the nodes in the group located i groups ahead of the source. Two values for i are considered: 1 and h (using $h = 6$), with ADV+h generating pathological congestion as described in Section 2.3.
- *Adversarial-local* (ADVl), in which the destination of a frame is selected randomly from all nodes in the consecutive switch within the same group, following a modulo sequence. This pattern has been observed in real-world evaluations [20].
- *Adversarial-consecutive* (ADVc), in which the destination of a frame is selected randomly from h destination groups. In particular, packets are sent to the h consecutive groups (+1, +2, ..., +h) after the source group, which are all connected to the same (bottleneck) switch of the source group. This traffic generates unfairness, as explained in Section 2.3.
- *Hotregion* (HOT), in which 25% of the traffic generated by each node is sent to the first 12.5% endpoints, and the remaining is distributed as Random Uniform (including the first 12.5% endpoints). This is the only considered traffic pattern which generates endpoint congestion.

Table 3: Simulation parameters.

	Parameter	**Value**
Network configuration	Total end terminals	5,256 hosts
	Topology	Dragonfly
	Groups	73 groups
	Switches per group	12 switches
	Switch degree	23 ports
	Link speed	200 Gbps
	Packet size	250 bytes
	Switch frequency	1 GHz
	Internal crossbar speedup	2×
	Switch latency	90 ns
	Local/Global link latency	15/150 ns (3/30 m)
	Injection queues size	126 KBytes
	Local/Global transit queue size	18/45 KBytes
	Local/Global/Injection virtual channels	2/1/1 (MIN), 4/2/1 (Other)
	PB routing parameters	$C_g = 120\%$, $T_g = 5$
ACOR	Switch hysteresis interval (HI)	500 ns
	First/Second increase threshold (IT_1/IT_2)	15/500
	First/Second decrease threshold (DT_1/DT_2)	5/15
PB-ACOR	Switch hysteresis interval (HI)	500 ns
	First/Second increase threshold (IT_1/IT_2)	15/50
	First/Second decrease threshold (DT_1/DT_2)	5/15

- *Random Permutation* (PERM), in which each endpoint is assigned a random destination endpoint at the start of the simulation. Each endpoint is assigned to exactly one source and vice-versa. The permutation obtained remains constant for the length of the simulation, but differs in each simulation.

5.3. Routing mechanisms

The following routing mechanisms are considered:

- *Minimal* (MIN) routing sends the packets minimally to the destination group, and then to the destination switch, employing a maximum of three hops (LGL, see Section 2.3) in a Canonical Dragonfly.

- *Valiant* (VAL) routing, as explained in Section 2.1, and following one of the misrouting policies in Table 1 for phase A (by default, RRG-Sw). Unless otherwise stated, this routing applies the two optimizations discussed in Section 3 (*Restricted Valiant* and *Valiant with recomputation*).

- *Adaptive Congestion-Oblivious Routing* (ACOR), as defined in Section 4. Different implementations consider the three different misrouting policy sequences in Table 1 and per-packet (*ACOR-Packet*) or per-switch (*ACOR-Switch*) level management.

- *Piggyback* (PB) routing, a source-adaptive congestion-aware routing mechanism originally described for the Dragonfly network in [8]. PB decides between a minimal and a non-minimal path depending on the saturation status of the queues at the first hop of each of the paths, and from saturation information from the global links of the group, distributed between the switches. A global link is tagged as saturated when its buffer occupancy exceeds a percentage C_g of the average occupancy of the global links in the same switch, plus an offset threshold, T_g. Those values are defined in Table 3. It employs the RRG-Sw misrouting policy (see Table 1) and, unless otherwise stated, the non-minimal path is selected using the *restriction* and *recomputation* optimizations as in VAL.

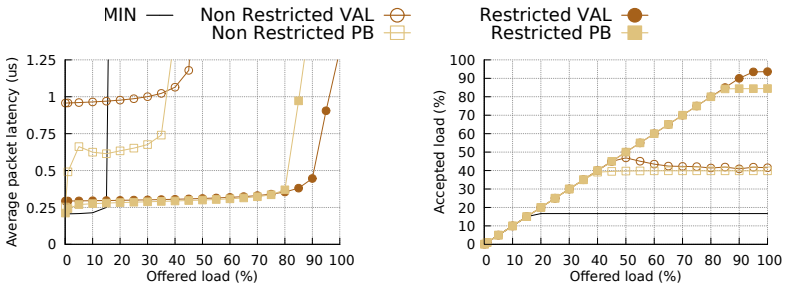


Figure 4: Latency (left) and throughput (right) of Valiant and Piggyback, using and not using the restricted mechanism under adversarial-local (ADVl) traffic pattern.

- *Piggyback with ACOR* (PB-ACOR) employs the same procedure as ACOR-Switch for the selection of the non-minimal path, but decides between the minimal and the non-minimal path following the same congestion-aware decision as PB.

6. Results

This section presents different aspects of the performance of our proposal. First, it extends the results presented in [7] implementing the mechanisms introduced on that work with Piggyback adaptive routing. Next, it presents the evaluation of the Adaptive Congestion-Oblivious Routing (ACOR), introduced on this work, under steady and transient loads and finalizes with a sensitivity analysis for the most important parameters of the implementation. Later, it presents performance results for Piggyback using ACOR for the non-minimal paths. Finally, the proposals are evaluated under alternative traffic patterns.

6.1. Restricted Valiant with Recomputation

Figure 4 presents the average latency and throughput results for *adversarial-local* (ADVl) traffic pattern using *minimal* routing (MIN), *default Valiant* implementation (VAL), *restricted Valiant* (RVAL), *adaptive Piggyback* (PB) and *restricted adaptive Piggyback* (RPB). The traffic pattern employed for this evaluation only presents intra-group traffic so, the impact of RVAL over VAL is significant and slightly significant for RPB over PB.

Using MIN the local link connecting pairs of neighbor switches becomes a bottleneck, and only $1/p = 16.6\%$ of the traffic can be delivered using the minimal routes. Before that saturation point, MIN presents optimal latency. VAL and PB with *RRG to switch* misrouting policy raise the saturation point of the accepted load near 50%. However, both RVAL and RPB accept almost 100% of the offered load, since it avoids the turn-around problem and generates short paths as depicted in Figure 3, even though the throughput obtained by PB at high loads is lower due to the use of minimal paths.

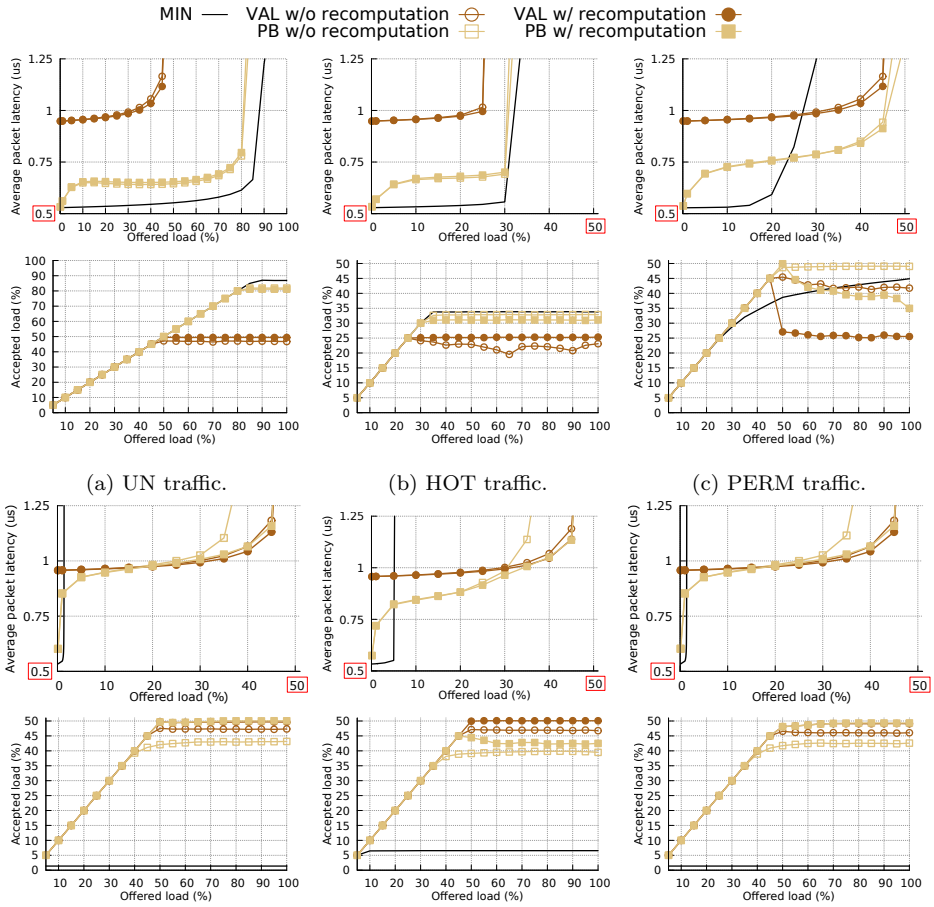
The evaluation of other traffic patterns is not presented since it was shown in previous work. As RVAL and RPB perform equally or better than VAL and PB respectively under the evaluated traffic patterns on this or previous work, the subsequent evaluations employ RVAL as VAL and RPB as PB.

Figure 5 presents average latency and throughput results for different traffic patterns using VAL and PB with and without basic recomputation (VAL-Recomp and PB-Recomp), as explained in Subsection 3.2. MIN is included as a reference. Before saturation, in all traffic patterns latency improves using re-computation of the intermediate destination. This is expected, since the re-computation occurs when packets cannot be injected because of congestion in the originally selected path; recomputation mechanism selects another path, and injects traffic earlier, so packets accumulate a lower latency. Throughput results after the saturation point are different between the traffic patterns employed. In all cases except for PERM traffic, throughput is improved with re-computation.

As VAL-Recomp and PB-Recomp perform equally or better than VAL and PB respectively for all the evaluated traffic patterns except for PERM, the subsequent evaluations employ VAL-Recomp as VAL and PB-Recomp as PB. Therefore, the rest of this work employs *restricted* and *recomputation* mechanisms as a baseline.

6.2. ACOR

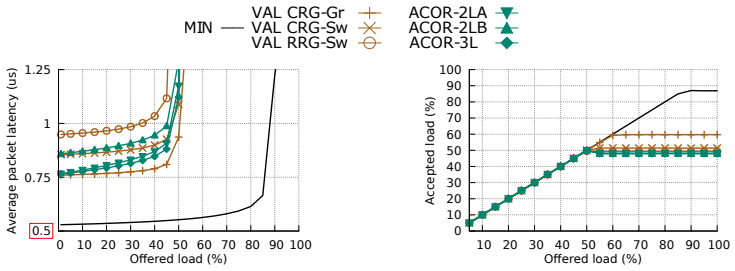
This section evaluates the performance of the *ACOR-Packet* and *ACOR-Switch*. It also discuses the behavior under transient traffic loads and analyzes



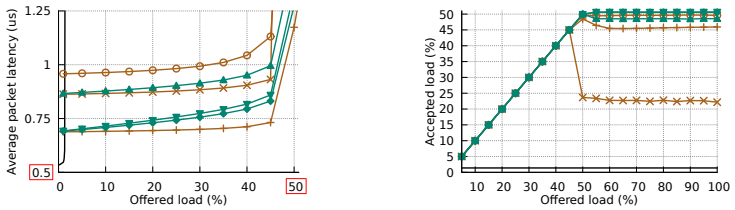
(a) UN traffic. (b) HOT traffic. (c) PERM traffic.

(d) ADV+1 traffic. (e) ADVc traffic. (f) ADV+h traffic.

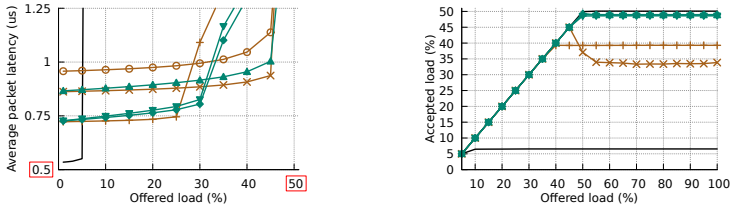
Figure 5: Latency (first and third line) and throughput (second and fourth line) of Restricted Valiant with and without recomputation under different traffic patterns.



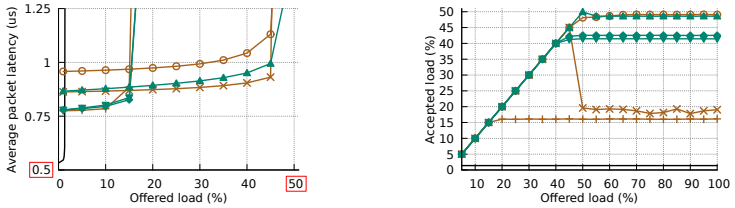
(a) UN traffic.



(b) ADV+1 traffic.



(c) ADVc traffic.



(d) ADV+h traffic.

Figure 6: Latency (left) and throughput (right) of ACOR-Packet.

the selection of configuration parameters.

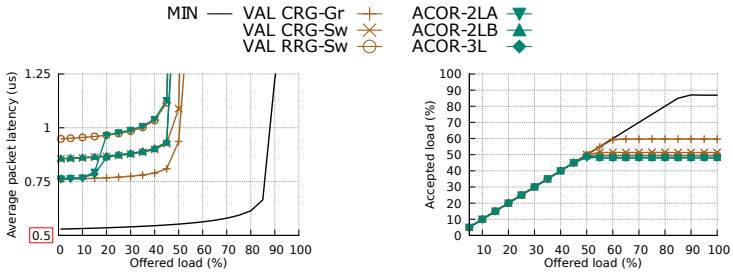
Figure 6 presents latency and throughput of ACOR-Packet under UN, ADV+*1*, ADVc and ADV+h traffic. As a reference, the result with MIN and VAL routing is also displayed; MIN is the baseline reference under UN traffic since it achieves

minimal latency. Results with VAL routing consider the 3 relevant misrouting policies in table 1: CRG-Gr, CRG-Sw and RRG-Sw. Each of these policies has a higher base latency over the previous, due to the additional local hops introduced. On the other hand, RRG-Sw presents the highest throughput under adversarial traffic patterns, since its better randomization avoids pathological congestion. CRG-Sw achieves lower latency at medium loads under ADVc and ADV+h traffic. This confirms the analysis in Section 4.2.1 where an optimal sequence in the misrouting policy employs CRG-Gr at low traffic loads and CRG-Sw at medium traffic loads to reduce latency, and RRG-Sw after the saturation point of CRG-Sw to achieve competitive throughput.

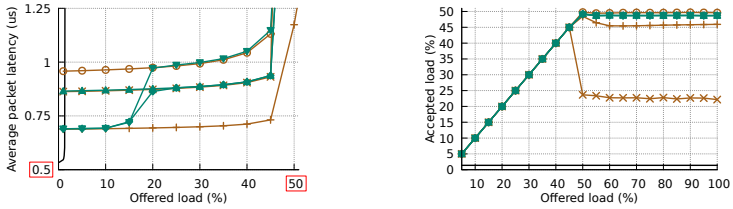
Figure 6 presents results for ACOR with the three misrouting policy sequences listed in Table 2. Since all of them employ the RRG-Sw misrouting policy at the highest level, they achieve competitive throughput under all traffic patterns; under ADV+h there is a difference in throughput between policies, with sequence 2LB outperforming the others. This occurs because under this traffic there is a pathological effect of congestion in the intermediate group which requires a non-minimal local hop, as discussed in Section 2.3. However, this effect occurs in a remote group, and ACOR relies on the congestion spreading back to the source group in order to trigger a change in the ACOR level. The 2LB sequence is able to reach higher throughput because all the levels use a misrouting policy that selects an intermediate switch instead of a group; for the same reason, the other policies suffer from high latency under medium loads.

The 2LA and 3L sequences present similar latency curves, albeit slightly lower for the 3L mechanism. This occurs because the 3L sequence has a lower proportion of packets using the highest level in the sequence (RRG-Sw policy), since two level changes are required to reach it. Therefore, the 3L variant is not effective in the per-packet implementation. The 2LB sequence has a higher base latency (due to the extra local hop at the source group) but lower latency at intermediate loads under ADVc and ADV+h traffic, because the additional local hop allows to mitigate congestion.

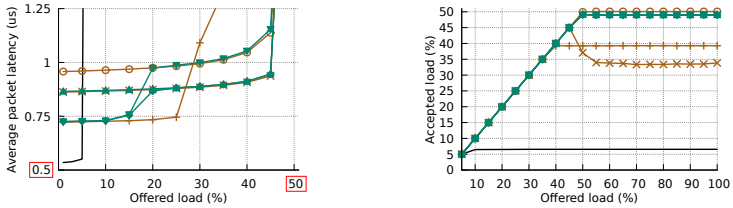
Figure 7 displays the performance results with ACOR-Switch. This mecha-



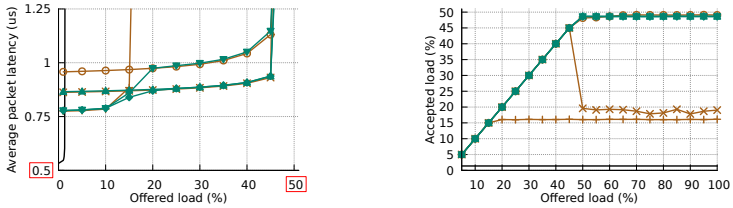
(a) UN traffic.



(b) ADV+1 traffic.



(c) ADVc traffic.



(d) ADV+h traffic.

Figure 7: Latency (left) and throughput (right) of ACOR-switch.

nism, although more complex than ACOR-Packet, is able to identify better the pathological congestion issues, using the same ACOR level for all the packets in the same switch. Throughput results are similar to those from ACOR-Packet except under ADV+h traffic, where all the sequences now achieve competi-

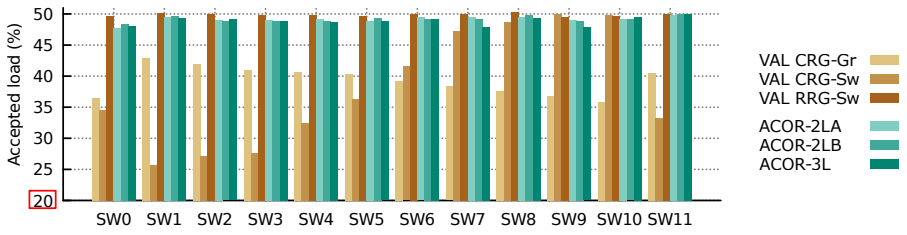


Figure 8: Averaged throughput accepted for each switch of group 0 under $ADVc$ traffic with a load of 50%.

tive performance. Managing the ACOR levels per-switch renders very different latency values, with the curves matching those from VAL with the different policies and rapid transitions between misrouting policies. Latency with the 2LB and 3L sequences is slightly higher than ACOR-Packet at intermediate loads, but stays relatively flat before the saturation point. This is particularly noticeable with the 3L sequence, which now fully transitions between levels to try to adapt to the optimal decision under all loads.

Another benefit of the ACOR mechanism is that it is able to adapt to different needs at different switches in the group. Figure 8 shows the average accepted load for all the nodes at each switch of the first group in the network, under a 0.5 phits/node/cycle load of ADVc traffic. Under this pattern, nodes at the last switch of the group have an uneven access to the minimal global links, since they are all directly connected. This favors a higher amount of minimally-routed traffic at the bottleneck switch, but prevents them from sending traffic non-minimally when the CRG global misrouting policy is used. ACOR exhibits a similar accepted traffic load for all the switches in the group with all the sequences, even 2LA and 3L which employ CRG-Gr at the earliest level; the capability of switching between policies for each switch allows it to prevent pathological unfairness effects as those suffered with VAL routing.

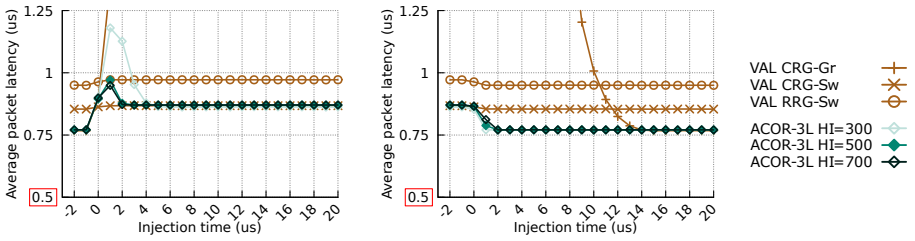


Figure 9: Average latency of ACOR-Switch 3L under ADV+h traffic with transient traffic loads. At $t = 0$, load transitions from 5% to 25% (left) and from 25% to 5% (right). Results with VAL routing are provided as a reference.

6.2.1. Performance under transient loads

A key benefit of ACOR is its ability to adapt to the network needs under different traffic loads, by changing the misrouting policy. Figure 9 illustrates the effect of the transition between levels, displaying the average latency under ADV+h traffic that changes the traffic load. Results with VAL routing are provided as a reference. When the traffic load increases (left figure), the CRG-Gr policy is no longer competitive and the latency of VAL with this policy increases significantly. ACOR transitions from using the CRG-Gr policy that provides the lowest base latency to a RRG-Sw policy which has the lowest latency at the higher load. Conversely, when the traffic load decreases (right figure), ACOR changes from the RRG-Sw policy to CRG-Gr. In both cases, ACOR matches the latency of VAL with the most suitable misrouting policy for each load, and is able to reach a steady behavior in less than $2\mu s$ in both transitions with the hysteresis interval value of 500ns used in our evaluation. Performance values with other durations are discussed later.

Figure 10 displays the current ACOR level for different switches of the same group, under the same transient traffic as in Figure 9. Certain switches change earlier to a higher level, depending on the amount of time it takes for the congestion to propagate back to them. In general, we can observe that all the switches transition to a higher level in less than $1\mu s$, and to a lower level in less than $0.5\mu s$.

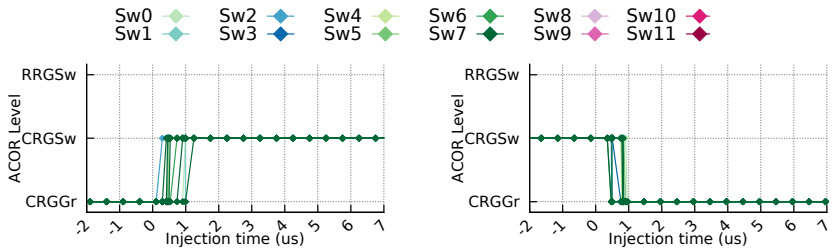
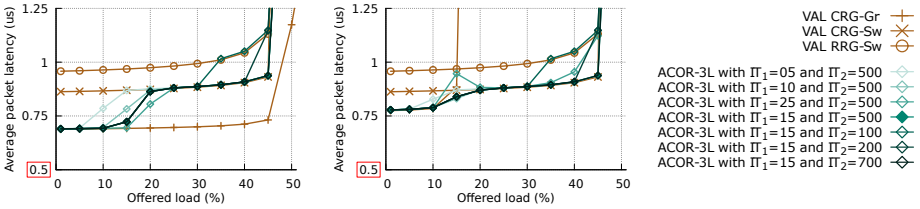


Figure 10: Evolution of the level of individual switches in a group. ACOR-Switch 3L under ADV+h traffic with transient traffic loads. At $t = 0$, load transitions from 5% to 25% (left) and from 25% to 5% (right).

6.2.2. Sensitivity analysis

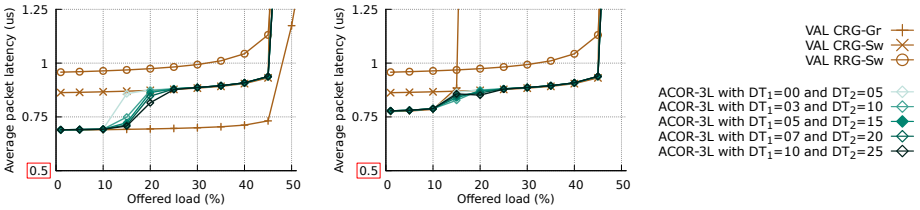
ACOR-Switch relies on several parameters to change the ACOR level from those available in the sequence. The hysteresis interval defined in Section 4.2.2 determines the amount of time that the mechanism waits before deciding to decrease a level, if the number of packets is below the threshold. Figure 9 presents different cycle durations to evaluate its impact in the sensitivity of ACOR-Switch with the 3L sequence. The behavior in the left figure shows that when short hysteresis intervals are used, longer transition time with higher peak latency occurs, which may seem counter-intuitive. A short interval resets the blocked packet statistics more often, which delays the transition to a higher level in the sequence until the network congestion becomes more severe. By contrast, a long interval is slightly detrimental when the traffic load decreases because it delays the transition to a lower level.

Two threshold values determine the decision to increase or decrease a level in the sequence, one to increase and one to decrease for every level transition. Figure 11 displays the impact of the increase threshold values with the 3L sequence, for a sweep in the traffic load under ADV+*1* and ADV+h traffic. The first threshold controls the transition from CRG-Gr to CRG-Sw, and the second threshold determines the change from CRG-Sw to RRG-Sw. Results with VAL routing are used as a reference. Lower increase threshold values make routing more eager to transit to a higher level, increasing the latency, whereas higher



(a) ADV+1 (b) ADV+h

Figure 11: Latency of ACOR-Switch 3L with different Increase Threshold values.

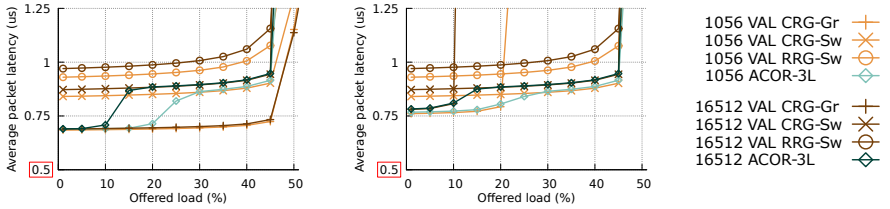


(a) ADV+1 (b) ADV+h

Figure 12: Latency of ACOR-Switch 3L with different Decrement Threshold values.

thresholds force the routing to stay in the current level for higher loads, which can lead to network congestion. The latter effect is observed in Figure 11 with $IT_1 = 25$. The selected values 15/500 in our evaluation represent a trade-off between both effects and achieve competitive performance under both patterns.

Similarly, Figure 12 shows the behavior for different decrease thresholds. In this case, the behavior is the opposite of the increase threshold: a lower value will allow ACOR to transit more easily to a lower level. Since the transition to the highest level in the sequence occurs under high traffic loads, the impact of the second threshold is rather limited. However, the first threshold has a significant impact in the latency at medium loads under ADV+*1* traffic, making the mechanism more prone to higher latency with a lower threshold: lower values allow the routing to transition more easily to the previous level and provoke oscillations in the selection of the misrouting policy. Again, the values 5/15 selected for this evaluation represent a good trade-off in the performance results.



(a) ADV+1 traffic.

(b) ADV+h traffic.

Figure 13: Latency of ACOR-Switch 3L with two different network sizes (1,056 and 16,512 hosts) under two different traffic patterns.

The behavior of ACOR-3L in the simulation results is the same for different network sizes. Figure 13 shows the behavior of ACOR-3L with a network size of 1,056 and 16,512 hosts; note that these curves can be compared with the results presented in Figure 7 for a network size of 5,256 hosts used across the paper. ACOR changes from *CRG-Gr* to *CRG-Sw* misrouting policy when VAL-CRG-Gr latency is triggered for each size under ADV+h traffic pattern and similarly jumps from *CRG-Sw* to *RRG-Sw* when VAL-CRG-Sw is saturated. The configuration parameters of ACOR-3L mechanism are different for each network size; for the network size used across the paper the simulation parameters are presented in Table 3 and $IT_1 = \{20, 10\}$ and $DT_1 = \{8, 3\}$ for 1056 and 16512 hosts respectively.

6.3. PB-ACOR

This section presents results of PB-ACOR, the variant of Piggyback presented in Section 4.3 that relies on ACOR for the non-minimal path computation. Figure 14 presents results for PB-ACOR with three different misrouting policy sequences, based on the ACOR-Switch mechanism. Parameter tuning for PB-ACOR has been performed similarly to the ACOR case presented in Section 6.2.2. The parameters used are presented in Table 3, and differ from the parameters used in ACOR because the non-minimal path is not used for all packets, so the amount of blocking differs. A discussion on the effect of this parameter change is presented in Section 6.4.

Under UN traffic, the latency of the baseline PB is higher than MIN. This is explained by the system sending part of the traffic non-minimally, possibly caused by transient congestion, as explained in [26]. Note that our models do not employ the history window proposed in [26] to mitigate transient congestion. By contrast, the latency of PB-ACOR is almost optimal (similar to MIN), up to a load of 75% using sequences 2LB and L3. Apparently, the use of shorter paths provided by ACOR helps reduce the effect of transient congestion.

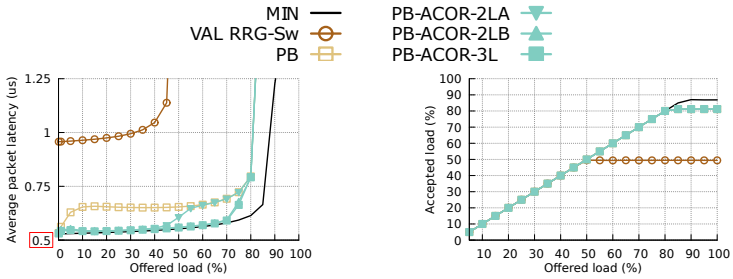
The effect in adversarial traffic is similar to ACOR. At low loads, the hop count is reduced and latency is improved. In the three adversarial patterns presented, latency at 10% load is reduced by 16.5% to 25.5%. At intermediate loads, both mechanisms start to behave similarly because both rely on RRG-Sw, and saturation throughput is equal.

6.4. ACOR with alternative traffic patterns

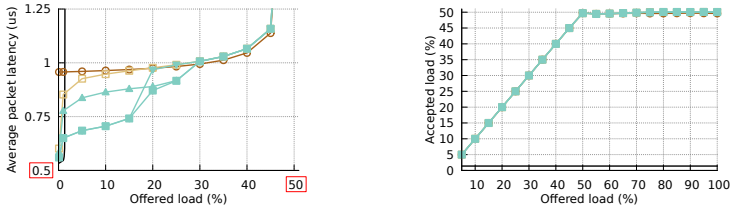
This section presents ACOR results with the two traffic patterns that do not result in adversarial traffic with significant network congestion: Hot-region and random permutations. Figures 15 and 16 present results with ACOR and PB-ACOR respectively, using per-switch level management. In all cases, MIN generates the lowest base latency because the lack of significant in-network congestion makes minimal routing profitable.

Hot-region traffic presents endpoint congestion, so shorter network paths in Phase A are better (CRG-Gr), with lower latency and higher throughput. ACOR gets close to the result of this policy for most of the range. With adaptive PB routing, the variants based on ACOR get significantly better latency, close to MIN.

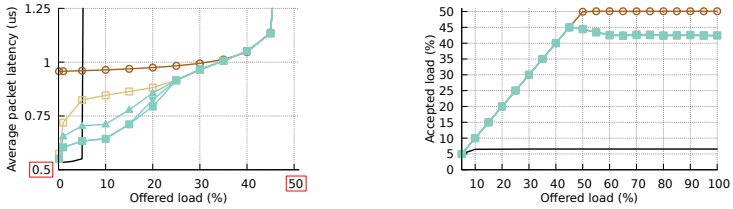
The behaviour of a random permutation differs. MIN provides again the lowest latency, but there is some unfairness before saturation, observed in the slope of the throughput curve starting around 30%. For the original Valiant variations in Figure 15, again the shortest path in CRG-Gr gives the lowest latency and throughput, but in this case two interesting facts occur: the intermediate policy CRG-Sw saturates a *lower* load than the base CRG-Gr, and



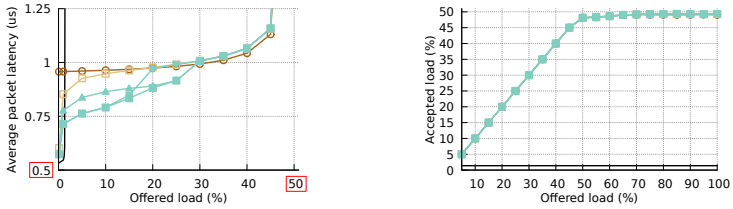
(a) UN traffic.



(b) ADV+1 traffic.



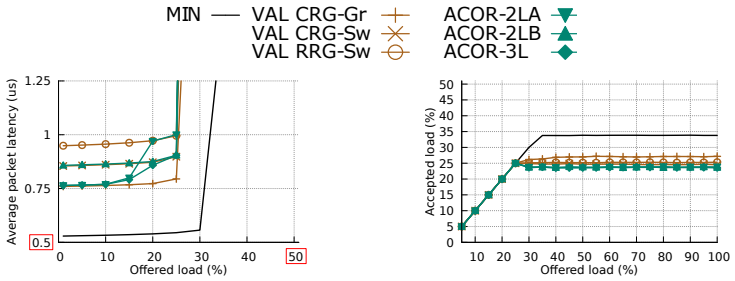
(c) ADVc traffic.



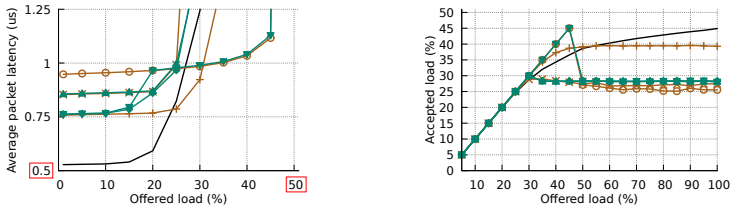
(d) ADV+h traffic.

Figure 14: Latency (left) and throughput (right) of PB-ACOR under different traffic patterns. MIN and VAL-RRG-Sw are presented as references.

both CRG-Sw and RRG-Sw get significantly lower throughput than the baseline. The ACOR implementations that employ CRG-Sw (2LB and 3L) saturate at a low load and their throughput is limited. The 2LA sequence gets competi-



(a) HOT traffic.



(b) PERM traffic.

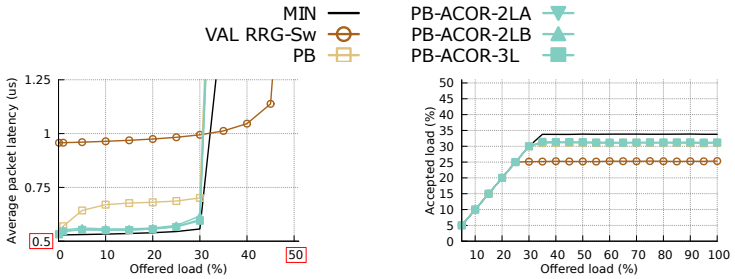
Figure 15: Latency (left) and throughput (right) of ACOR-Switch (right) under hot-region (HOT) and random permutation (PERM) traffic.

tive latency in a wide range, but again throughput degrades after the saturation point because of congestion.

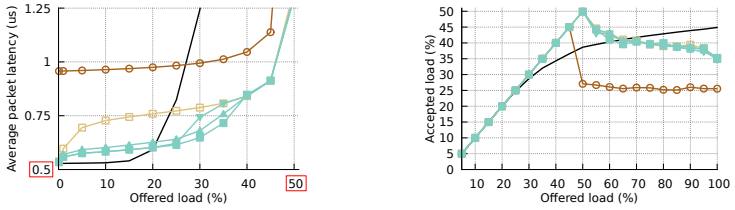
The thresholds used in ACOR and PB-ACOR differ, being *easier* to change to RRG-Sw in PB-ACOR than in ACOR. Since RRG-Sw gives better latency at intermediate loads, the latency of PB-ACOR using sequences 2LB and 3L is more competitive than 2LA. After saturation, all the adaptive mechanisms present a similar throughput.

7. Related work

Several mechanisms have proposed restricted variants of Valiant, in which the intermediate switch selection is not performed among all the switches in the network. The original proposal for Valiant in the Dragonfly by Kim *et al.* in [2] selects a random intermediate *group*, rather than a *switch*. This reduces the length of the path (and the amount of virtual channels) but has



(a) HOT traffic.



(b) PERM traffic.

Figure 16: Latency (left) and throughput (right) of PB-ACOR-Switch under hot-region (HOT) and random permutation (PERM) traffic.

shown to introduce pathological performance issues under adversarial traffic [13]. Similarly, the dragonfly implementation in [15] diverts traffic using a single global hop for the non-minimal path of Phase A, which is equivalent to select a random *group* that is directly connected to the source switch; in this case, the implementation is constrained to using commodity Ethernet hardware without bookkeeping information in the packet headers.

Different proposals implement restricted variants of Valiant routing in other topologies. The proposal for randomized routing in multidimensional square meshes in [6] does not randomize all the N dimensions, but only $N-1$; congestion is avoided assuming a single injector per switch, but this is not typically the case in current and forthcoming parallel systems. The *DAL* adaptive routing mechanism in HyperX [10] follows the idea of RVAL by misrouting only in the dimensions with offset. The Cray Cascade implements a mechanism similar to restricted Valiant, based on two different sets of tables [27]. Valiant routing in the indirect OFT and MLFM networks is restricted to switches with connected

nodes in [28]. The already-mentioned proposal in [19] identifies the turn-around problem in Slim Flies, and introduces a modified version of Valiant routing for Slim Flies which only makes one non-minimal hop in its Phase A. These works do not formally prove the absence of pathological performance issues under any traffic pattern. Our ACOR approach can be adapted to such restricted routing variants; ACOR falling back to the complete Valiant path in Phase A under high load guarantees the absence of such hypothetical pathologies.

We are not aware of any application of the re-computation of intermediate switches based on network congestion to original Valiant, but there are adaptive routing proposals that somehow resemble it. For example, the aforementioned *DAL* adaptive routing in HyperX [10] deroutes traffic in a given dimension based on the unavailability of output ports; therefore, it dynamically diverts traffic to an intermediate destination which is computed dynamically.

8. Conclusions

Low-diameter network (such as the Dragonfly) have low path diversity and require non-minimal routing to achieve good performance under adversarial traffic patterns. In the Dragonfly network, most routing mechanisms employ the Valiant algorithm to randomize the traffic and balance the use of the links, extending the length of the path and increasing the base network latency.

This work introduces Adaptive Congestion-Oblivious Routing (ACOR), which is based on Valiant routing. The goal of ACOR is to optimize the common case providing minimal latency, while supporting pathological traffic patterns with longer paths. It applies the optimizations introduced in [7] and extends restrictions in the intermediate path to global traffic. It expands the idea of path recomputation to adapt to network conditions, changing the misrouting policy following a given sequence ordered by path length. It prevents variability in the results through a simple hysteresis mechanism. The implementation is relatively simple, according to the description presented in Section 4.2. In this work, three sequences of different length have been considered. Same as Valiant,

ACOR does not send traffic minimally, so its performance under benign traffic is suboptimal. The ACOR mechanism has been coupled with a nonminimal adaptive routing, PB in the case of a Dragonfly network. PB-ACOR selects the shortest feasible non-minimal path, but only when the minimal route is congested. This mechanism maintains the benefits from ACOR for adversarial traffic and is competitive under uniform traffic.

Evaluation results show that all the ACOR variants avoid any throughput pathologies and reduce base latency by up to 28%, compared to Valiant with the restricted and recomputation optimizations. The lowest latency values are achieved with a three level (3L) sequence that exploits three different misrouting policies; however, the two level sequence 2LB presents similar results except below 15% loads, where its base latency is higher. ACOR also avoids pathological unfairness under ADVc traffic. Two management strategies for the transitions in the sequence have been considered, per-packet and per-switch. Per-switch management achieves better performance as it considers the amount of blocked packets across the whole switch; however, per-packet management with the 2LB sequence has similar performance except for higher latency at low loads, and lower implementation costs. PB-ACOR reaches high throughput and optimal latency under UN traffic, significantly outperforming Valiant, and improves base latency up to 25.5%. It also achieves rivaling throughput and significantly lower latency compared to base PB. For these reasons, PB-ACOR with the 3L misrouting policy and per-switch level management sequence results in the most competitive routing mechanism.

Acknowledgements

This work was supported by the Spanish Ministry of Economy, Industry and Competitiveness under contract TIN2016-76635-C2-2-R (AEI/FEDER, UE), the European HiPEAC Network of Excellence and The Mont-Blanc project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671697.

References

[1] J. Kim, W. J. Dally, D. Abts, Flattened butterfly: A cost-efficient topology for high-radix networks, in: International Symposium on Computer Architecture (ISCA), 2007, pp. 126–137.

[2] J. Kim, W. J. Dally, S. Scott, D. Abts, Technology-driven, highly-scalable dragonfly topology, in: International Symposium on Computer Architecture (ISCA), 2008, pp. 77–88.

[3] M. Besta, T. Hoefler, Slim Fly: A cost effective low-diameter network topology, in: IEEE/ACM Intl. Conf. on High Performance Computing, Networking, Storage and Analysis (SC14), 2014.

[4] C. Camarero, C. Martinez, E. Vallejo, R. Beivide, Projective networks: Topologies for large parallel computer systems, IEEE Transactions on Parallel & Distributed Systems 28 (7) (2017) 2003–2016.

[5] S. Rumley, M. Glick, S. D. Hammond, A. Rodrigues, K. Bergman, Design methodology for optimizing optical interconnection networks in high performance systems, in: J. M. Kunkel, T. Ludwig (Eds.), High Performance Computing, Springer International Publishing, Cham, 2015, pp. 454–471.

[6] L. G. Valiant, G. J. Brebner, Universal schemes for parallel communication, in: Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing, STOC '81, ACM, New York, NY, USA, 1981, pp. 263–277. doi:10.1145/800076.802479.
URL http://doi.acm.org/10.1145/800076.802479

[7] M. Benito, P. Fuentes, E. Vallejo, R. Beivide, Analysis and improvement of valiant routing in low-diameter networks, in: International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), IEEE, 2018, pp. 1–8. doi:10.1109/hipineb.2018.00009.

[8] N. Jiang, J. Kim, W. J. Dally, Indirect adaptive routing on large scale interconnection networks, in: Intl. Symp. on Computer Architecture (ISCA), 2009, pp. 220–231.

[9] L. Valiant, A scheme for fast parallel communication, SIAM journal on computing 11 (1982) 350.

[10] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, R. S. Schreiber, HyperX: Topology, routing, and packaging of efficient large-scale networks, in: Conference on High Performance Computing Networking, Storage and Analysis (SC '09), New York, NY, USA, 2009, pp. 41:1–41:11. doi:10.1145/1654059.1654101.
URL http://doi.acm.org/10.1145/1654059.1654101

[11] A. Singh, Load-balanced routing in interconnection networks, Ph.D. thesis, Stanford University (2005).

[12] J. Kim, W. Dally, S. Scott, D. Abts, Cost-efficient Dragonfly topology for large-scale systems, Micro, IEEE 29 (1) (2009) 33–40.

[13] M. García, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, G. Rodríguez, J. Labarta, C. Minkenberg, On-the-fly adaptive routing in high-radix hierarchical networks, in: 41st International Conference on Parallel Processing, 2012, pp. 279–288. doi:10.1109/ICPP.2012.46.

[14] B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, J. Li, et al., The PERCS high-performance interconnect, in: 18th Symposium on High Performance Interconnects, IEEE, 2010, pp. 75–82.

[15] M. Benito, E. Vallejo, R. Beivide, On the use of commodity Ethernet technology in exascale HPC systems, in: International Conference on High Performance Computing (HiPC), 2015, pp. 254–263.

[16] M. García, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, J. Labarta, G. Rodríguez, Global misrouting policies in two-level hierar-

chical networks, in: Proceedings of the 2013 Interconnection Network Architecture: On-Chip, Multi-Chip, IMA-OCMC '13, ACM, New York, NY, USA, 2013, pp. 13–16. doi:10.1145/2482759.2482763.
URL http://doi.acm.org/10.1145/2482759.2482763

[17] P. Fuentes, E. Vallejo, C. Camarero, R. Beivide, M. Valero, Network unfairness in dragonfly topologies, The Journal of Supercomputing 72 (12) (2016) 4468–4496. doi:10.1007/s11227-016-1758-z.
URL http://dx.doi.org/10.1007/s11227-016-1758-z

[18] M. García, E. Vallejo, R. Beivide, M. Odriozola, M. Valero, Efficient routing mechanisms for dragonfly networks, in: The 42nd International Conference on Parallel Processing (ICPP-42), 2013.

[19] P. Yébenes, J. Escudero-Sahuquillo, P. J. García, F. J. Quiles, T. Hoefler, Improving non-minimal and adaptive routing algorithms in slim fly networks, in: 2017 IEEE 25th Annual Symposium on High-Performance Interconnects (HOTI), 2017, pp. 1–8. doi:10.1109/HOTI.2017.11.

[20] D. J. Kerbyson, K. J. Barker, Analyzing the performance bottlenecks of the POWER7-IH network, in: International Conference on Cluster Computing, 2011, pp. 244–252. doi:10.1109/CLUSTER.2011.35.

[21] P. Gratz, B. Grot, S. W. Keckler, Regional congestion awareness for load balance in networks-on-chip, in: IEEE International Symposium on High Performance Computer Architecture, 2008, pp. 203–214. doi:10.1109/HPCA.2008.4658640.

[22] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, J. Reinhard, Cray Cascade: a scalable HPC system based on a dragonfly network, in: SC: Intl Conf on High Performance Computing, Networking, Storage and Analysis, 2012, pp. 103:1–103:9.

[23] M. García, P. Fuentes, M. Odriozola, M. Benito, E. Vallejo, R. Beivide, FOGSim interconnection network simulator (2014).
URL http://fuentesp.github.io/fogsim/

[24] C. Camarero, E. Vallejo, R. Beivide, Topological characterization of hamming and dragonfly networks and its implications on routing, ACM Trans. Archit. Code Optim. 11 (4) (2014) 39:1–39:25. doi:10.1145/2677038.
URL http://doi.acm.org/10.1145/2677038

[25] J. García-Haro, R. Marín-Sillué, J. L. Melús-Moreno, ATMSWSIM An efficient, portable and expandable ATM SWitch SIMulator tool, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994, pp. 193–212. doi:10.1007/3-540-58021-2_11.

[26] J. Won, G. Kim, J. Kim, T. Jiang, M. Parker, S. Scott, Overcoming far-end congestion in large-scale networks, in: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), 2015, pp. 415–427.

[27] M. Parker, S. Scott, A. Cheng, J. Kim, Progressive adaptive routing in a dragonfly processor interconnect network (2015).

[28] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, T. Hoefler, Cost-effective diameter-two topologies: analysis and evaluation, in: SC15: International Conference for High Performance Computing, Networking, Storage and Analysis, 2015, pp. 1–11. doi:10.1145/2807591.2807652.