J Supercomput. manuscript No.
(will be inserted by the editor)

On-the-Fly Adaptive Routing for dragonfly interconnection networks

Marina García · Enrique Vallejo · Ramón Beivide · Cristóbal Camarero · Mateo Valero · Germán Rodríguez · Cyriel Minkenberg.

Published online: 16 December 2014

Abstract Adaptive deadlock-free routing mechanisms are required to handle variable traffic patterns in dragonfly networks. However, distance-based deadlock avoidance mechanisms typically employed in Dragonflies increase the router cost and complexity as a function of the maximum allowed path length.

This paper presents OFAR (On-the-Fly Adaptive Routing), a routing/flow-control scheme that decouples the routing and the deadlock avoidance mechanisms. OFAR allows for in-transit adaptive routing with local and global misrouting, without imposing dependencies between virtual channels, and relying on a deadlock-free escape subnetwork to avoid deadlock. This model lowers latency, increases throughput, and adapts faster to transient traffic than previously proposed mechanisms. The low capacity of the escape subnetwork makes it prone to congestion. A simple congestion management mechanism based on injection restriction is considered to avoid such issues. Finally, reliability is considered by introducing mechanisms to find multiple edge-disjoint Hamiltonian rings embedded on the dragonfly, allowing to use multiple escape subnetworks.

Keywords Interconnection network · Dragonfly network · OFAR · Adaptive routing · Deadlock avoidance

M. García, E. Vallejo, R. Beivide, C. Camarero
University of Cantabria
Tel.: +34-942-206770
E-mail: {marina.garcia, enrique.vallejo, ramon.beivide, cristobal.camarero}@unican.es

M. Valero
Barcelona Supercomputing Center and Universitat Politècnica de Catalunya
Tel.: +34-93-4137716
E-mail: mateo.valero@bsc.es

G. Rodríguez, C. Minkenberg
IBM Research Zurich
E-mail: {rod, sil}@zurich.ibm.com

1 Introduction

This paper presents ***OFAR***: an **O**n the **F**ly **A**daptive **R**outing for dragonfly networks. Dragonflies [21] have been proposed as a cost-efficient solution for large-scale interconnection networks. A dragonfly is organized in groups of routers. The interconnection between groups employs optical *global links*. Routers within groups are connected using short *local links*, typically electrical. The topologies of the local and global interconnects are typically low-diameter direct topologies that exploit high-radix routers. For example, the IBM PERCS Interconnect [2] employs an all-to-all topology (complete graph, K_x) in both the local and global interconnects, whereas the Cray XC30 (codenamed "Cascade", [8]) employs a complete graph for the global interconnect and a $K_{16} \times K_6$ topology within groups (where $\times$ represents the cartesian product of graphs). For the rest of the paper we refer to the specific case of complete graphs for both the local and global interconnect, but the results are general and can be easily extended to any dragonfly.

The main dragonfly topological parameters, as defined in [21], are the number of routers per group a, the number of processing nodes per router p and the number of global links per router h. For a well-balanced network under uniform traffic, the relations $a = 2p = 2h$ must hold, [21]. A balanced dragonfly using 64-port routers ($h = 16$) scales to more than 256K processing nodes.

The diameter of the dragonfly topology is 3, so any minimal path between two routers will employ at most 3 hops. *Minimal routing*, as proposed in [21,2] is hierarchical: a packet typically first traverses a local (l) and a global (g) link to reach the destination group, and finally another local link at the destination group (path $l_1 - g_1 - l_2$). Each of these links can easily saturate under adversarial traffic patterns, so nonminimal routing can be applied to randomize traffic and avoid saturated links. *Valiant routing* [26], as used in [21,2], diverts traffic to an intermediate group before heading the destination group. This leads to paths of up to 5 hops ($l_1 - g_1 - l_2 - g_2 - l_3$). In such paths, the first two hops corresponding to *global misrouting*, this is, a nonminimal path to an intermediate group. This global misrouting circumvents the potentially saturated global link in the minimal path. Similarly, congestion in the intermediate or destination group requires of *local misrouting*, what implies even longer paths. The different congestion scenarios and the corresponding nonminimal routing solutions are presented with more detail in Section 2.

Cyclic routing dependencies can appear in Dragonflies, requiring a deadlock avoidance strategy. Several proposals, [21,2], rely on a set of virtual channels (VCs) visited in a predefined order, based on the original mechanism in [14]. The number of VCs is bounded by the length of the longest path: minimal routing allows for paths of length 3, whereas Valiant allows for length 5. However, as local links are used in hops 1, 3 and 5 and global links are used in hops 2 and 4, it is enough to implement 3 VCs in local links and 2 VCs in global links; such VC configuration will be denoted as 3/2. Shorter paths (e.g. $l - g - l$ with Valiant) skip indexes corresponding to missing hops. Tying the allowed path length with hardware resources is undesirable, because it makes the use of longer paths (required for local misrouting or in-transit adaptive routing) more complex to implement. This problem might become much

more significant if network routers are embedded within processor chips, as several vendors have suggested in their roadmaps.

In summary, the main performance limitations in dragonfly networks come from the different congestion issues and the capability to adapt in-transit to changing traffic conditions. Additionally, the allowed paths have a direct impact on the implementation cost when a restrictive use of VCs is imposed for deadlock avoidance. The mechanism presented in this paper addresses all of these issues.

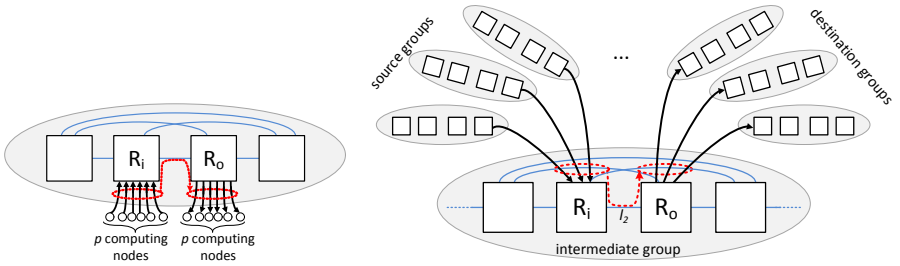
OFAR is a flow-control/routing mechanism that decouples the use of the virtual channels from the deadlock avoidance mechanism. It allows each router in the path to dynamically misroute packets depending on the observed congestion, leading to faster adaptation to transient traffic patterns and avoiding congestion in local links. An escape subnetwork is employed for deadlock avoidance. Our evaluations show that such a model improves throughput and response time under different traffic patterns, even in the most adverse cases. To prevent congestion in the escape subnetwork, we combine OFAR with a simple congestion management (CM) mechanism: injection throttling based on local information, which happens to be enough to prevent congestion in the escape subnetwork. Finally, we also consider the use of multiple disjoint escape subnetworks for fault tolerance. Two previous papers of the authors, [11,12], provide more detail on the work described here. The current paper unifies the OFAR proposal under a common framework, considers the "Valiant-any" routing from [24,26] as a reference mechanism and introduces a novel mechanism to find disjoint escape rings for fault tolerance. Evaluations with multiple escape rings with different mappings lead to the conclusion that the new ring mappings introduced in this paper provide better performance than previous alternatives thanks to a lower network congestion.

In Section 2 we study the main performance limitations of dragonfly networks caused by adversarial traffic patterns, and discuss how global and local misrouting together with in-transit misrouting efficiently avoid them. In Section 3 we detail OFAR considering two alternatives for the escape subnetwork (a Hamiltonian ring with bubble flow control, and a spanning tree with up-down routing), introduce mechanisms to find multiple disjoint Hamiltonian rings for fault tolerance, and present two simple congestion management mechanisms (denoted as BCM and ECM). Section 4 details the simulation infrastructure and Section 5 presents the performance results. Finally, Section 6 presents related work and Section 7 concludes the paper.

2 Performance limitations of dragonfly networks

In this section we survey the main performance limitations of a well-balanced dragonfly network with complete graphs in its local and global topologies; the issues are similar for other variants of such a topology.

Saturation of global links: The transfer limit of each link is 1 $phit/cycle$ (the *phit* is the amount of information transferred through a link in one cycle) and up to one global hop is required for each packet to reach its destination node with minimal routing. Using $h = p$ allows for maximum performance under uniform traffic, *UN*,



(a) Saturation under traffic local to a group

(b) Pathological saturation of the single local link, l_2, connecting R_i and R_o in the intermediate group.

Fig. 1: Saturation problems in local links

with minimal routing. However, in an adversarial traffic pattern the $2h^2$ computing nodes in one group could send traffic to the same destination group, competing for the bandwidth of a single global link. We denote this traffic as adversarial-global, *ADVG+N*, where every source node in group i selects a random destination node in group $(i+N) \pmod G$, (G is the number of groups). With minimal routing this traffic would limit the maximum bandwidth to $1/(2h^2)$ while leaving most of the global links underutilized (with $h = 16$, throughput is limited to $1/512$, less than 0.2% of its maximum). ***Global misrouting*** tries to avoid a congested global link by sending traffic to another group, possibly requiring a local hop in the source group. Valiant routing, as used in [21], employs global misrouting to equalize the use of the global links in the network. Each packet is sent to a random intermediate group, and then minimally to its destination. Because this implies two global hops for any packet to reach its destination, on average, global links will limit the maximum throughput to 0.5 phits/(node $\cdot$ cycle). The problem is not the scarcity of global links (since $p = h$), but their unbalanced use under minimal routing, caused by the traffic pattern.

Saturation of local links: Analogously, local links also saturate when all the p computing nodes attached to a router R_i send traffic to the nodes in a neighbor router R_o of the same group, as shown in Fig. 1a. We denote this traffic as adversarial-local, *ADVL*. The p nodes attached to R_i offer a load of p phits/cycle to the only local link between these two routers. As this link has a capacity of 1 phit/cycle, under minimal routing the maximum traffic would be $1/p$ (for $p = h = 16$, this would limit traffic to 6.25% of its maximum). Again, the problem is not the number of local links in a group, but their unbalanced use caused by the traffic pattern. ***Local misrouting*** avoids a saturated local link by sending packets to an intermediate router within the group, and then to the destination router, using two local hops instead of the saturated one.

Pathological saturation of local links under adversarial-global traffic: Using global misrouting requires up to 5 hops to get to destination , $l_1 - g_1 - l_2 - g_2 - l_3$. The two first hops $l_1 - g_1$ lead to an intermediate group. The intermediate local hop l_2 is only required if the source and destination groups are not connected to the same router in

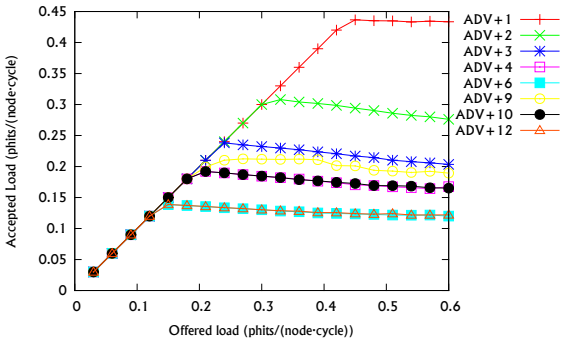


Fig. 2: Accepted vs. offered load in a $h = 6$ dragonfly with Valiant and different Adversarial Traffic Patterns

the intermediate group. For certain traffic patterns, this local link l_2 will saturate and become the network bottleneck, even when this leaves global links partially idle.

Specifically, the worst case occurs with the adversarial pattern *ADVG+h*. Fig. 1b shows two routers R_i and R_o of a given group G_i. R_i receives misrouted traffic through its h global links, which has to be forwarded through the h subsequent global links. Global wiring is typically consecutive, so all these links are in the next router, R_o. The single link connecting R_i and R_o can only convey 1 phit/cycle, so even in absence of any other throughput limit in the network, the localized saturation of certain local links will limit throughput to $1/h$ phits/(node $\cdot$ cycle). The same happens for any other *ADV+*$(n \cdot h)$ traffic pattern. For $h > 2$, the throughput limit imposed by the saturation in these local links is more restrictive than the limit imposed by global links with global misrouting, and such restriction grows with the network size.

The maximum throughput for adversarial traffic with global misrouting will depend on the specific offset between the source and destination groups. Fig. 2 shows how, with the Valiant routing used in [21], throughput varies notably depending on this offset, even in a small dragonfly with $h = 6$ (5,256 computing nodes). *ADVG+1* causes the lowest congestion, whereas *ADVG+*$(n \cdot 6)$ generates the highest. As with *ADVL* traffic, local misrouting (now, in the intermediate group) would avoid this pathological saturation. Interestingly, the original definition of Valiant routing in [26] selects a random intermediate router (not a group), which has been denoted as "Valiant Any" in [24]. This oblivious routing is roughly equivalent to performing a global misrouting followed by a local misrouting in the intermediate group.

Adapting to traffic changes. Adaptive routing mechanisms select between minimal and nonminimal paths depending on an estimation of the network congestion. Multiple adaptive mechanisms proposed for Dragonflies (such as UGAL [21], CRT or Piggybacking [18]) employ source routing, selecting the path at injection. This requires that congestion information extends throughout the network so that source routers can properly sense it. To adapt quickly to traffic changes, ***in-transit adaptive routing*** can switch from minimal to non-minimal (global misrouting) after injection

of the packet[1]. This implies that two local hops can be required in the source group, one minimal and one nonminimal (required for global misrouting).

In-transit adaptive routing supporting local misrouting in the intermediate and destination groups leads to relatively long maximum paths: $l-l-g-l-l-g-l-l$. Using the standard distance-based deadlock avoidance mechanisms described in [14], this would require at least 6 virtual channels in the local links.

3 *OFAR*: On-the-Fly Adaptive Routing

OFAR differs in several key points with respect to previous ideas : *i)* Adaptive routing is performed *in transit*, rather than being determined at injection time, with both local *and* global misrouting; *ii)* Use of local information (the credit count of the output ports of the local router) rather than remote information to select between minimal and nonminimal paths, and to select the intermediate destinations for local and global misrouting; and *iii)* The use of a deadlock-free sub-network for deadlock avoidance.

3.1 Dynamic misrouting in *OFAR*

In *OFAR* the path of each packet is not determined at injection; each router can forward traffic non-minimally to avoid network congestion, adapting the packet path. This misrouting can use either local or global links of the current router. In principle, our mechanism would allow for any number of misroutings without additional cost, but a limit is set to prevent livelock: at most, one non-minimal global hop can be applied per packet, and one non-minimal local hop per group. Two flags in the packet header are used to limit misrouting. When packets do not travel on the escape subnetwork that will be detailed in Section 3.3, the longest path is limited to 8 hops (2 global and 6 local), as discussed in Section 2.

Each packet in an input buffer of a router always has a 'minimal output' according to its minimal path to the destination. Depending on the credits of the minimal output and the header flags, the router can misroute the packet using a nonminimal output with more credits. When traffic is internal to a group, only local misroute is allowed. When the destination is a remote group, both local and global misroutes are allowed. In such case, when the packet is still in its source group only global misrouting is allowed. In a group other than the source group, only local misrouting is allowed, and only when the minimal output is a saturated local port. Both local and global misrouting are applied adaptively only when congestion is detected, so compared with the oblivious "Valiant-Any" mechanism, OFAR can save the initial global misrouting or the local misrouting in the intermediate group when there is no congestion.

Local misrouting always employs a local output port different than the minimal one. For global misrouting we use the MM+L policy from [10] to determine which

[1] We do not consider the opposite case (switching to minimal after a first nonminimal local hop which corresponds to the global misrouting) because we model the MM+L global link selection policy [10] which does not make a first local hop for global misrouting at injection. However, since OFAR decouples the router resources and the path length, it would also support that case.

port to use. Under this policy, packets still in injection queues of their source router are misrouted directly by global channels of this router. When using global misrouting this saves the first local hop, minimizing path length. By contrast, packets switching from minimal to nonminimal routing after a first minimal local hop (which are in local queues in the source group) always make a nonminimal local hop before their nonminimal global hop. This avoids starvation under adversarial traffic if the current node is the only one with a global link to the destination group.

3.2 Misrouting selection criteria

OFAR relies on the congestion observed in the minimal path to allow for non-minimal routing and to select the specific output port used. We assume an input-buffered router with a separable allocator. When a packet is in the header of an input queue, the routing subsystem will report its corresponding minimal path, along with the allowed non-minimal paths (i.e., no misrouting allowed, misroute by any local link, or misroute by any global link). Depending on the measured network congestion, the input unit of the allocator can request the minimal path or one of the non-minimal ones. If the request is not assigned by the arbiter, subsequent requests can select different output ports depending on the varying credit count.

To select between minimal or nonminimal routing, *OFAR* observes the occupancy, Q_{min}, of the queue in the minimal path (*minimal queue*), and the occupancy, $Q_{non-min}$, in any non-minimal output of the appropriate type, local or global (*non-minimal queues*). As these queues have different sizes for local and global links, we consider the percentage of buffer occupancy rather than the actual occupancy in phits. We use a misrouting threshold $Th_{non-min}$. Misrouting is allowed only when the minimal port is not available (it is already assigned to another input or $Q_{min} = 100\%$)[2]. In such case, the output port is selected randomly among those available non-minimal ports that fulfil the occupancy condition $Q_{non-min} \leq Th_{non-min}$. This prevents misrouting packets to a group which is already congested. We consider a relative misrouting threshold, which depends on the occupancy of the minimal queue: $Th_{non-min} = 0.90 \times Q_{min}$.

3.3 Deadlock avoidance based on escape subnetworks

When all minimal and nonminimal paths are unavailable, packets are diverted to a deadlock-free escape subnetwork to avoid deadlock. If packets can reach their destination through this escape subnetwork, the overall system is deadlock-free [7]. In each output assignment any available VC can be selected because this solution does not require virtual channels, although they help to mitigate Head-of-line blocking (HoLB). The capacity of the escape subnetworks proposed for OFAR is very low as compared to the *canonical dragonfly*. To avoid saturation, in each hop routers try to forward traffic from the escape subnetwork back to a canonical link (minimal or not) if possible, to reduce traffic in the escape subnetwork. Thus, regardless of the type of

[2] A minimum occupancy might be also required in the minimal queue to allow for misrouting; we have not considered such a threshold in this work.

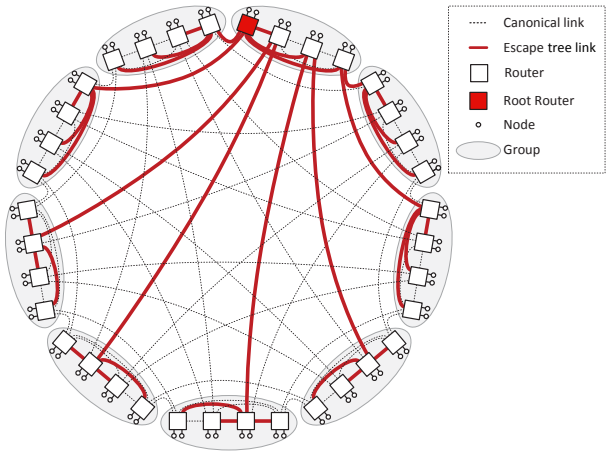


Fig. 3: dragonfly network $h = 2$ with an additional escape tree subnetwork

input port of a packet, routers will always use the following precedence order for the output selection: minimal - nonminimal - escape. Since minimal and escape routes typically follow different paths, livelock might arise; Section 5.7 studies this issue and finds that it is not a significant problem, especially when considering one of the congestion management mechanisms presented later in Section 3.4.

The escape subnetwork must interconnect all the routers in the network. It can employ links separated from the ones in the canonical network, using extra ports in the routers and interconnecting them with additional local and global links. Alternatively, it can be embedded in the canonical dragonfly, adding one extra virtual channel to each link forming the escape subnetwork. We will denote this extra VC separately, for example 3/2(+1). Depending on the topology employed, the cost and the performance will vary. We consider two alternatives next.

3.3.1 Tree

In this case one of the routers is chosen as a "root", R_{root}. We denote the group containing R_{root} as G_{root}. R_{root} is connected to all the remaining routers in G_{root}. Each router in G_{root} is globally connected with one or several remote routers, each one in a different group. Finally, each remote router is connected with the remaining routers in its group. An example is presented in Figure 3. The routing employed in the tree is up-down, which makes it deadlock-free.

3.3.2 Hamiltonian ring

This escape subnetwork interconnects every router in the network, forming a ring. With Virtual Cut-through, the ring is deadlock-free as long as there is space for at least one packet in one of its buffers. To assure this, bubble flow control is applied to the ring [5]. To inject a packet in the escape ring, free space for two packets is required

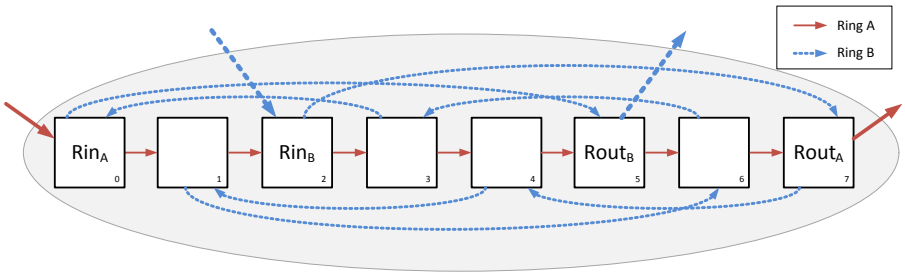


Fig. 4: Paths of two disjoint Hamiltonian rings in a group of a dragonfly network with $h = 4$. The remaining links have been omitted for simplicity

in the buffer, which preserves the previous condition. On the contrary, packets can freely circulate without restrictions once they are inside the ring. Note that, in OFAR, this only occurs when the *canonical* links are saturated, as discussed before.

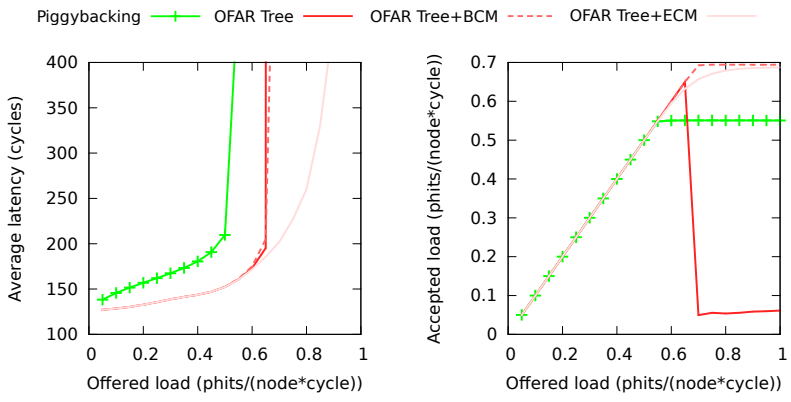
A simple Hamiltonian ring, denoted as *ring A*, is generated as follows. The $a = 2h$ routers in a group are labeled from 0 (R_0) to $2h-1$ (R_{2h-1}). Router i is connected to routers $i \pm 1 \pmod a$ in the same group, except for R_0, and R_{2h-1}. R_0 in group j is connected to R_1 in group j and R_{2h-1} in group $j-1 \pmod G$, with G the number of groups. R_{2h-1} in group j is connected to R_{2h-2} in group j and to R_0 group $j+1 \pmod G$. This ring is partially illustrated in Figure 4. The resulting Hamiltonian ring connects all the routers in the network employing a minimal number of global links equal to the number of groups. The large diameter of this Hamiltonian cycle will not be a problem, because packets only travel a few hops in the escape subnetwork, as will be studied in Subsection 5.7.

3.3.3 Fault tolerance and escape bandwidth

One or two link failures in the escape subnetwork (for a tree or ring respectively) disconnect the escape subnetwork and potentially block the system. Additionally, the bandwidth of the escape subnetwork is much lower than the one of the canonical dragonfly, which can lead to congestion as detailed in Subsection 3.4. The use of several disjoint subnetworks increases fault tolerance and total escape bandwidth.

Considering embedded subnetworks, it is impossible to find two disjoint trees since the root of one tree must share local ports with the other tree. However, there are multiple algorithms to recalculate a spanning tree in case of a failure, such as those defined for bridges in IEEE 802.1D [16]. In the case of rings, it is possible to find multiple disjoint instances, but the solution introduced next is not as obvious as the one for ring A presented in Subsection 3.3.2.

Considering a Hamiltonian escape ring, for a dragonfly with $h > 2$ it is possible to generate at least a second edge-disjoint ring B using the following methodology. On each group, we build a path traversing all the $a = 2h$ routers in the group. For $h > 2$ even, we select router $\frac{h}{2}$ as the first router R_{in} of this path, and router $2h-1-\frac{h}{2}$ as the end router R_{out}. Router i is connected to routers $i \pm (h+1) \pmod a$, except routers R_{in} (which in its group is just connected to router $\frac{h}{2}+(h+1)$), and R_{out} (which in its



(b) Latency UN. (c) Throughput UN.

Fig. 5: Latency and throughput under uniform traffic UN for *OFAR* with a tree escape subnetwork and different Congestion Management alternatives

group is just connected to router $2h-1-\frac{h}{2}-(h+1)=h-2-\frac{h}{2}$). This path employs links which increase the index in $\pm(h+1)$; for any $h>2$ this path is disjoint with the local links of ring A, which increase indexes by ± 1. Additionally, this path exists for any $h>2$ since $\frac{h}{2}+(2h-1)\cdot(h-1)\equiv 2h-1-\frac{h}{2} \pmod{a}$ and it traverses all routers in the group since $h+1$ is coprime with $2h$. Finally, every router R_{out} in group j is connected to router R_{in} in group $j+(\frac{h^2}{2}+2) \pmod{G}$, forming a ring with global links between groups. The resulting ring B connects all the routers in the network employing the same amount of local and global channels as Ring A. An example for $h=4$ is presented in Fig. 4.

For $h>3$ odd, the path of ring B is built from $R_{in}=\frac{h+1}{2}$ to $R_{out}=2h-1-\frac{h+1}{2}$ using links which increase the index by $\pm(h+2)$. Additionally, a third disjoint ring C can be built from $R_{in}=\frac{h-3}{2}$ to $R_{out}=2h-1-\frac{h-3}{2}$ using links which increase the index by $\pm(h-2)$. The proof that these two Hamiltonian rings exist for any $h>3$ is similar to the one presented for $h>2$ even.

3.4 Congestion management

In OFAR, the capacity of the escape subnetwork is lower than the capacity of the canonical dragonfly network. If all the buffers of the canonical network were completely full, and only the escape subnetwork was used to deliver packets to their destination, performance would drop significantly. Reaching such a condition should be very uncommon as it must be provoked by the occurrence of multiple concurrent deadlocks that are not alleviated in time by the escape subnetwork. As discussed in [23], deadlock is very infrequent when paths are short and there is a rich routing freedom, exactly the case of the dragonfly. Despite this situation being unlikely, proper congestion management (CM) mechanisms have to be applied to the network to guarantee that it does not happen in practice.

Fig. 5 shows the throughput and latency results obtained for *OFAR* under UN traffic with an embedded tree as escape subnetwork, using only 2/1(+1) VCs. Simulation details are presented later in Section 4 and the results of Piggybacking are shown only as a reference. With OFAR, if no congestion management mechanism is applied (*OFAR Tree*), throughput drops significantly when the load is high and the canonical network gets congested. In such case, the canonical network cannot handle the traffic that receives, which is diverted to the escape subnetwork, and the overall network throughput becomes that of the escape subnetwork. However, using a simple congestion management mechanism (BCM or ECM, described next) the throughput does not fall. Similar mechanisms have been studied before in other networks [22].

3.4.1 Escape Congestion Management (ECM)

Network congestion increases the use of the escape subnetwork to prevent deadlock. The Escape Congestion Management (ECM) mechanism employs the occupancy of the local buffers of the escape subnetwork in the current router as an indicator of congestion. No packets are injected from any of the end nodes connected to a router when the occupancy Q of all the escape buffers exceeds a threshold:

$$Q_i > Th_{ECM} \quad \forall i \in \{escape\ buffers\}$$

In such case, the nodes in the current router will have to wait for a subsequent cycle to inject their traffic. The threshold used is chosen empirically, ranging from 0% to 100%. ECM does not take into account the occupancy of the queue in which the packet should be injected, contrary to the following mechanism BCM.

3.4.2 Base Congestion Management (BCM)

The Base Congestion Management (BCM) mechanism forbids the injection of packets when the canonical (base) network is congested. This is implemented as a variant of a bubble flow control mechanism [5]. A certain "bubble" is required for computing nodes to inject packets in a buffer, which prevents traffic injection from introducing deadlock in the canonical network. A packet at an injection queue can be injected in the network only if the free space S in the destination buffer covers the packet size plus the bubble size (in phits):

$$S \geq Pkt_{Length} + Bubble \times Max(Pkt_{Length})$$

Otherwise, the packet will have to wait for a subsequent cycle. The bubble size can range from 1 to the buffer size in packets minus 1, and it is chosen empirically. The BCM mechanism does not consider the occupancy of the escape subnetwork, it only takes into account the state of the canonical network.

Table 1 Parameters of the simulated dragonfly

Parameter	Value
Ports per router	23
Global ports per router	6
Computing nodes per router	6
Inter- and intra-group topology	complete graph
Routers per group	12
Groups	73
Overall computing nodes	$5,256$
Packet size	8 phits
Latency of local/global links	$10/100$ cycles
Buffer size, local/global ports	$32/256$ phits
Number of VCs	3/2; 3/2(+1) for OFAR
Switching mechanism	Virtual Cut-through
Arbitration policy	Least-Recently Served
Router speedup	No
Allocation iterations	3
VC selection in OFAR	shortest-queue
OFAR threshold $Th_{non-min}$	$0.9 \cdot Q_{min}$
ECM threshold Th_{ECM}	20%
BCM bubble	2 packets

4 Methodology

We have implemented the different routing proposals on FOGSim [9], a single-cycle simulator developed in the University of Cantabria. Its results for existent routing mechanisms are consistent with those published in previous works. We model a FIFO input-buffered Virtual Cut-through (VCT) router, [20]. Unless otherwise noted in the text, the parameters employed in the simulations are presented in Table 1. We do not model router speedup, but employ an iterative allocator similar to [15] to compensate the lack of speedup. We model different latencies for the different links on the system, depending on them being short (local) or long (global). While we do not model the router delay, the overall latency (in absence of network congestion) includes the sum of both router delay and link latency; in our model, router delay can be considered to be subsumed within link latency values.

We employ synthetic traffic to evaluate performance. Each source node generates packets according to a Bernoulli process, with a controllable injection probability in phits/(node $\cdot$ cycle). Packet latency is measured from its generation to its complete reception in the destination node; the average latency is obtained by averaging across all packets received in the network. Such latency includes injection buffering, link latency, and additional cycles lost due to allocation or congestion issues. Throughput is measured as the amount of phits received in the network, averaged by the number of cycles and computing nodes, generating a result between 0 and 1 phit/(node $\cdot$ cycle). We average 5 simulations to generate the results in the plots.

The destination node is selected randomly depending on the traffic model. We have considered two corner cases:

– Uniform (*UN*): The destination is selected among all the network nodes, including the source group but not the source node itself.

– Adversarial-global+N (*ADVG+N*): The destination is selected among all nodes in the group $i+N$, where i is the source group. *ADVG+1* causes the lowest congestion on local links, while $ADVG+n \cdot h$ generates maximum congestion.

We evaluate than *ADVG+h* (and not *ADVL*) because it is enough to evaluate the problem of local link saturation as discussed in Section 2. Moreover, we want to ensure that implementations do not fail on pathological situations.

We have considered the following routing mechanisms:

– Minimal (*MIN*): Minimal hierarchical path between the source and destination. It only uses 2/1 VCs.
– Valiant (*VAL*): As used in [21,2], packets travel to a random intermediate group (global misrouting), and then travel minimally to its destination.
– Valiant-any (*VAL-any*): As defined in [26], and employed in [24], packets travel to a random intermediate router (roughly equivalent global and local misrouting), and then travel minimally to its destination. It employs 4/2 VCs.
– Piggybacking (*PB*, [18]): The injection router selects between minimal and Valiant paths as used in [21,2] based on remote congestion information broadcast among all the routers of each group.
– *OFAR*: The base model presented in Subsection 3.
– *OFAR-L*: The same model, without allowing misrouting in local links. This is used to expose the specific benefits of in-transit local misrouting.

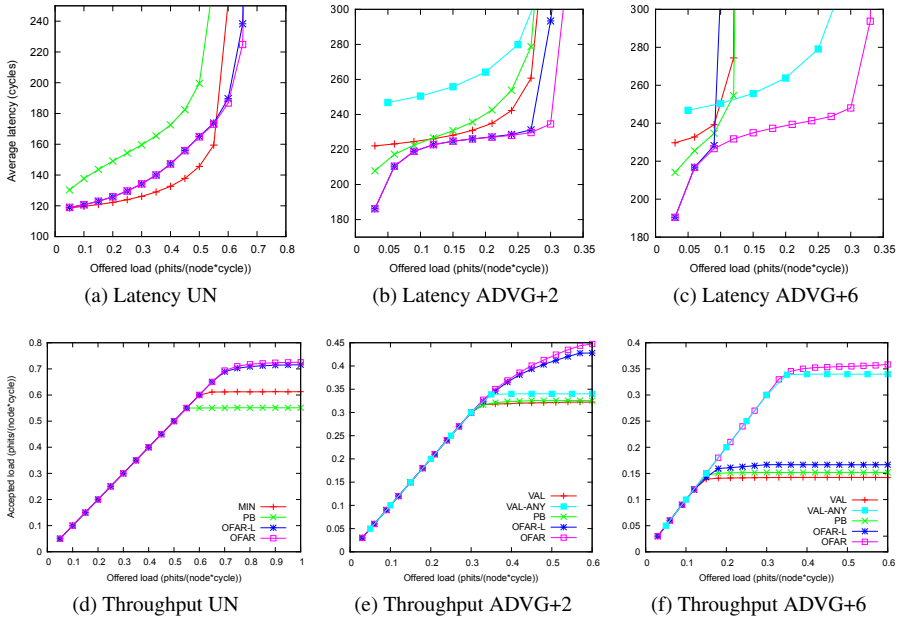
PB is used as a reference, as it is the source routing mechanism with the best overall results in previous works, [18]. It always employs 3/2 VCs, which are required for deadlock avoidance. In fact, *OFAR* is the only alternative which can employ a variable number of VCs, as a trade-off between implementation cost and performance.

As with any other adaptive routing mechanism in dragonflies, *OFAR* employs a variable misroute threshold which can be tuned to favor minimal or nonminimal routing. The selected value $Th_{non-min} = 0.9 \cdot Q_{min}$ allows misrouting if the minimal queue is not available (it is assigned to another packet or without credits), only by those queues that have less than 0.9 times the occupancy of the minimal one. This threshold was selected empirically for our network configuration, by simulating multiple threshold values and selecting a reasonable trade-off between the performance in adversarial and uniform traffic patterns. A similar study was performed for the threshold values in PB, and for the Bubble in BCM and Th_{ECM} in ECM.

We employ the escape subnetwork and congestion management mechanisms detailed in Subsection 3.4: *OFAR Ring+BCM*, *OFAR Ring+ECM*, *OFAR Tree+BCM* and *OFAR Tree+ECM*. Each subnetwork is embedded in the canonical dragonfly by adding an extra VC in the corresponding escape links. We prefer to show the results of this design, to focus in the case which is more prone to congestion. It implies a lower cost than a physical escape link using extra router ports, but in absence of congestion the performance results are similar, since it is scarcely used.

5 Performance results

We first show the results for a base model of OFAR using a Hamiltonian ring without congestion management, in three different scenarios: steady state, transient variations



(a) Latency UN (b) Latency ADVG+2 (c) Latency ADVG+6

(d) Throughput UN (e) Throughput ADVG+2 (f) Throughput ADVG+6

Fig. 6: Average latency (top) and throughput (down). Uniform (UN) and adversarial (ADVG+2 and ADVG+6) traffic, without any congestion management

and traffic bursts. Next, we focus on the impact of the escape subnetwork topology, congestion management mechanisms, unfairness issues and path length. Finally, we present a study of performance with multiple escape subnetworks.

5.1 Steady state

After 50.000 cycles of warm-up, these tests measure the average latency and throughput during a second period of 50.000 cycles, as described in Section 4. Fig. 6 shows the results under uniform random (*UN*) and adversarial-global (*ADVG*) traffic patterns: *ADVG+2* which requires global misrouting,[3] and *ADVG+6* which additionally requires local misrouting to avoid congestion. All simulations employ 3/2 VCs, except for *MIN* (2/1) and *VAL-any* (4/2), and no CM is used for *OFAR*.

For *UN*, using *MIN* as a reference, the *OFAR* models provide a competitive latency under low loads, but they saturate significantly later. The latency of the adaptive mechanism *PB*, by contrast, is significantly larger, due to a higher number of misrouted packets. Both *OFAR* models improve throughput over *MIN* and *PB*, but in either case, the use of local misrouting does not make a significant difference.

[3] We do not show results of *ADVG+1* as it could be argued that the additional ring link between the source and destination groups favors the *OFAR* models. However, since the escape network utilization is only used to avoid potential deadlock situations and not to carry traffic to the destination, the results are similar.

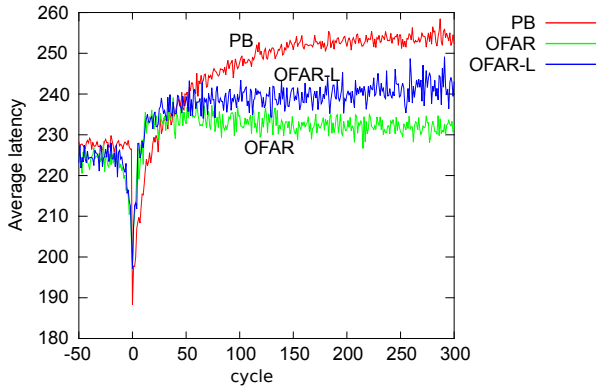


Fig. 7: Latency evolution under transient traffic. ADV+2 to ADV+6 traffic, load=0.12

For adversarial traffic patterns *ADVG*, *OFAR* provides the best latency and throughput, both with *ADVG+2* and *ADVG+6*. Under *ADVG+6* without local misrouting throughput is limited to $1/h = 1/6 = 0.166$ phits/(node $\cdot$ cycle). Fig. 6f shows that this occurs for *VAL, PB* and *OFAR-L*. *VAL-any* and *OFAR* reach around 0.36, confirming that local misrouting in the intermediate group compensates for the pathological saturation problem. However, the latency results show that *OFAR* is more efficient since it adaptively misroutes traffic only when required.

5.2 Transient traffic

The following experiments explore the response time when the traffic pattern changes. The network is warmed up with *ADVG+2* traffic with a load of 0.12 phits/(node $\cdot$ cycle), which does not cause congestion even without local misroute. Once it reaches the steady state, the traffic changes to *ADVG+6*. Fig. 7 shows the measured latency of the packets that are sent each cycle, with the adaptive routing mechanisms. *OFAR* makes the transition almost instantaneous thanks to in-transit routing decisions, while the source-routing *PB* suffers from a clear adaptation period while the congestion information is propagated.

5.3 Traffic bursts

In parallel programs, communication and computation phases are typically synchronized, so traffic bursts after barriers are common. We simulate such behaviour using packet bursts. Each node injects a fixed amount of packets (2,000) as fast as possible, with a mixture of different traffic patterns. With $h = 6$, this figure corresponds to around a million packets received. We measure the time to consume all the packets in the network. The destination of each packet is variable according to a certain distribution. We have simulated *UN*, *ADVG+2*, *ADVG+6* and three mixes of traffic with

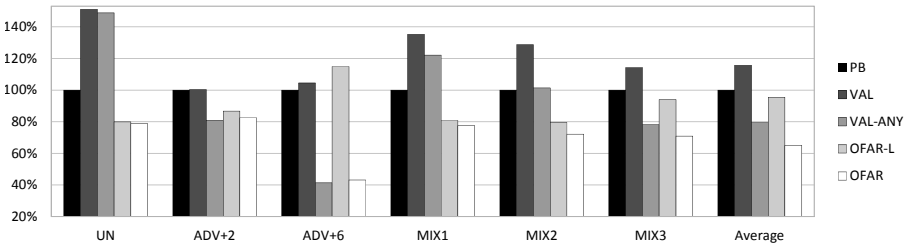


Fig. 8: Burst consumption time, normalized to *PB*, using 3/2 VCs. Lower is better

different rates of uniform and adversarial: In *MIX1* 80% of the traffic is UN, 10% is *ADVG+1* and 10% is *ADVG+6*. In *MIX2* the rates are 60-20-20 and in *MIX3* they are 20-40-40.

Fig. 8 shows the burst consumption time normalized to the result of *PB*. Compared to *PB*, the execution time of *OFAR* ranges from a 43.1% to a 81.5%. On average, the time to consume traffic for *OFAR* is 0.695 the time for *PB*, which corresponds to a speedup of 1.438×. The complete *OFAR* model always finishes sooner than all the others, including its *-L* counterpart.

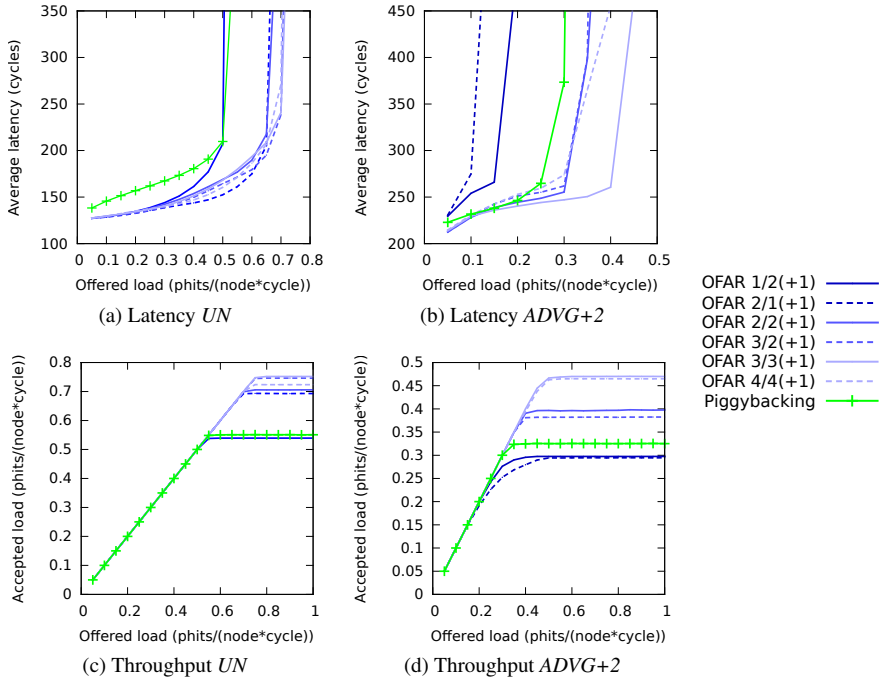
5.4 Number of virtual channels

With OFAR, VCs are not employed for deadlock avoidance, but they help mitigate Head-of-Line Blocking (HoLB). A higher number of VCs improves performance, until a point at which there is another factor that limits performance more than HoLB. Past that point, a higher number of them can degrade performance, because routers have to handle a higher number of VCs and there are more packets in the network, which can make allocation more complex and increase congestion.

As *OFAR* can be prone to congestion, especially with a low number of VCs as discussed in Section 3.4, we employ the BCM mechanism with a bubble size of 2 packets in these evaluations. BCM will be evaluated with more detail in Section 5.5. Fig. 9 shows the average latency and throughput results obtained for *OFAR Ring+BCM* with a different number of VCs under uniform (UN) and adversarial (ADVG+2) traffic. The number of VCs ranges from 1/1(+1) to 4/4(+1). PB is also shown as a reference.

In general *OFAR* always obtains better performance than PB (3/2 VCs) when using the same or more VCs. Even with a lower number of VCs, 2/2(+1), *OFAR* results are better. Only when *OFAR* employs fewer VCs, 2/1(+1) or 1/1(+1), and with ADVG+2 traffic, PB outperforms *OFAR*. When the traffic is uniform, Fig. 9a and 9c, *OFAR* 1/1(+1) obtains a result very close to that for PB. All other configurations present a better performance, very close to each other. *OFAR* with 3/3(+1) VCs is the configuration that achieves the best overall performance for all the traffic patterns. Further increase of the number of VCs does not provide a better performance: *OFAR* 4/4(+1) obtains worse results, especially for latency when the traffic is adversarial.

A problem of network unfairness appears when the VC count is low. With 2/1(+1) VCs and adversarial traffic, the average latency of *OFAR* rockets at around 0.15



(a) Latency *UN* (b) Latency *ADVG+2*

(c) Throughput *UN* (d) Throughput *ADVG+2*

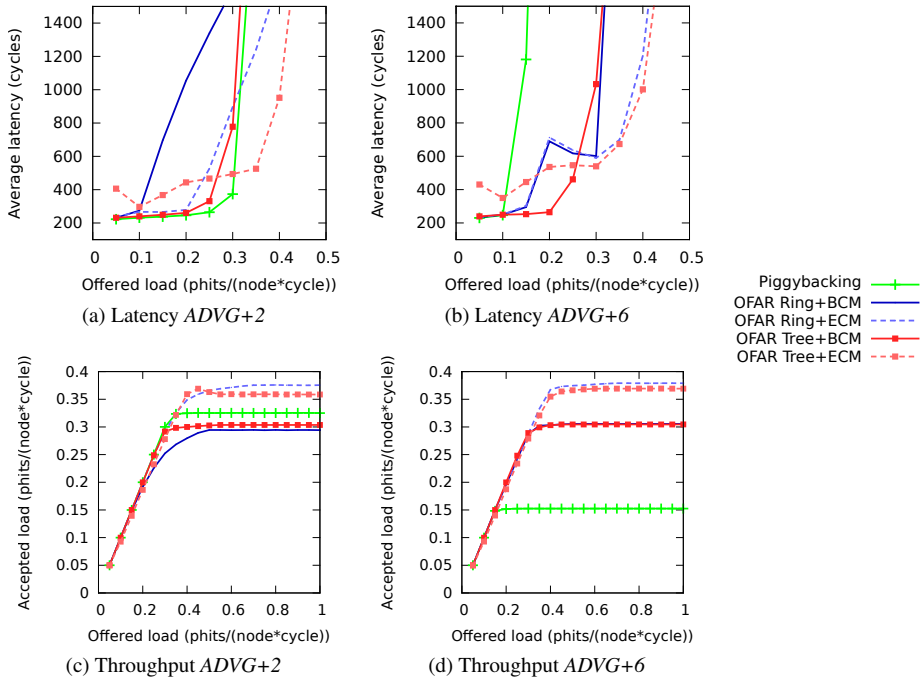
Fig. 9: Latency and throughput under uniform and adversarial traffic (ADVG+2) for PB and *OFAR* varying the number of VCs. Ring escape subnetwork with BCM congestion management

phit/(node · cycle). However, its throughput reaches 0.3 phit/(node · cycle). This effect is typical for unfairness issues, when some specific nodes of the network suffer from starvation: their latency is much higher than the rest, which increases the average latency values. In this case, the problem arises from localized congestion, as will be studied in more detail in Subsection 5.6.

With the same amount of resources, or even less, *OFAR* outperforms PB. Only under ADVG+2 traffic with a reduced number of VCs, 2/1(+1) or lower, *OFAR* cannot match PB due to HoLB issues. Next subsections will only focus on this specific OFAR configuration with few resources, 2/1(+1), to study the effects of congestion and alternative ways to cope with it.

5.5 Congestion management and escape subnetwork

This Subsection explores how the escape subnetwork and congestion management mechanisms affect the performance of *OFAR* with few resources, 2/1(+1) VCs; when more VCs are used, no significant congestion issues arise, as studied in Subsection 5.4. The performance of *OFAR Ring* and *OFAR Tree* is studied with BCM and ECM



(a) Latency *ADVG+2* (b) Latency *ADVG+6*

(c) Throughput *ADVG+2* (d) Throughput *ADVG+6*

Fig. 10: Latency (up) and throughput (down) under adversarial traffic for PB (with 3/2 VCs) and *OFAR* with few resources that lead to congestion, 2/1(+1) VCs

congestion management. Under UN traffic there are no differences, because the escape subnetwork is hardly ever used. The latency and throughput steady state results under adversarial patterns are show in Fig. 10. The variants with ECM obtain higher throughput than those with BCM. Under $ADVG+2$ traffic, the maximum throughput with BCM is slightly lower than with PB, while with ECM it is higher.

Some anomalies caused by load imbalance are present. The latency of *OFAR Ring+BCM* with 2/1(+1) in Fig. 10a above load 0.1 keeps increasing, up to reaching the throughput saturation load. Also, the throughput for *OFAR Tree+ECM* at low adversarial traffic loads is slightly lower than for the other configurations. This configuration also presents high latency values before saturation. These are also caused by load imbalance, and both will be discussed in detail next in Subsection 5.6.

Fig. 11 shows results for the traffic burst experiments for PB and the different *OFAR* configurations. An additional traffic pattern is included, in which traffic is sent according to an *All-to-all* global communication primitive. The number of messages sent in each case is similar. In every case, *OFAR Ring* consumes traffic faster than *OFAR Tree*. Both are faster when the congestion management mechanism is ECM, although, on average, the difference in time is not very large. As a result, *OFAR Ring* with ECM is the fastest *OFAR* configuration, followed very closed by *OFAR Ring* with BCM. The only case in which PB using 3/2 VCs is faster than the *OFAR*

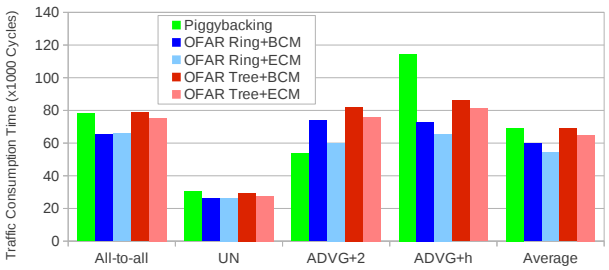


Fig. 11: Traffic consumption time for PB and *OFAR* with 2/1(+1) VCs

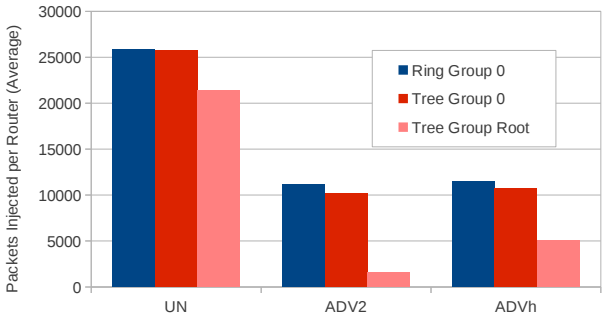


Fig. 12: Average number of packets injected per router in groups 0 and *Root* with an offered load of 1 phit/(node · cycle), when BCM and only 2/1(+1) VCs are used

configurations with 2/1(+1) is when the traffic is ADVG+2. Even in that case, the running time of OFAR Ring with ECM is very close to that of PB.

5.6 Network fairness

In the experiments in Fig. 10, *OFAR* obtains very similar throughput results with the ring or the tree escape subnetworks, regardless of the congestion management. However, as shown in Fig. 11, *OFAR Ring* consumes adversarial traffic faster than *OFAR Tree*. This behavior of *OFAR Tree* is due to a load imbalance introduced by the asymmetry of the escape subnetwork. Fig. 12, obtained with BCM, shows this effect. It depicts the total number of packets injected in the network by nodes in group G_0 and G_{root} for *OFAR Tree* and *OFAR Ring* in 50,000 cycles, after warm up, when the applied load is 1 phit/(node · cycle). Results for *UN, ADVG+2* and *ADVG+h* are presented. G_{root} is the group containing the root router R_{root} when the escape subnetwork is a tree. G_0 is a group chosen as a baseline for comparison. As there is no root in a ring, for *OFAR Ring* only results for G_0 are shown.

For *OFAR Tree*, the number of packets injected by nodes in G_{root} is significantly lower than in G_0. When a packet is injected into a tree escape subnetwork, the probability that it has to pass through the root router R_{root} is very high. As a result, R_{root} and its group G_{root} receive more traffic than the rest of the routers and groups of the network. There is a small part of the network, G_{root}, that concentrates a large part of the escape traffic. This does not happen with an escape ring, as it is a symmetric topology that balances the load among all the groups in the network. With the tree escape subnetwork, packets in the injection queues of routers in group G_{root} have to wait longer to be injected in the network. Therefore, in the same amount of cycles, nodes in G_{root} inject less packets than nodes in the rest of the groups. This explains the slightly lower throughput in Fig. 10c and 10d especially with ECM. Also, in the traffic consumption experiments, the consumption time increases because nodes in G_{root} have difficulties injecting packets.

This asymmetry of the tree escape subnetwork is also responsible for the high average latencies at low traffic loads for *OFAR Tree+ECM* when the traffic is adversarial (Figures 10a and 10b). With that configuration, *OFAR* only injects packets if the escape subnetwork is not saturated. Although at low traffic loads the escape subnetwork should not be saturated, this occurs in G_{root}, due to the concentration of traffic in that group. Routers in G_{root} detect the escape subnetwork as congested, prohibiting packet injection, and increasing average packet latency. A solution for this problem might come from using multiple escape trees, in order to distribute the traffic generated by the escape subnetwork.

Apart from load imbalance between groups, there could also exist imbalance between routers within the same group. This problem occurs for *OFAR Ring+BCM* in Figures 10a and 10b. Specifically, under an adversarial-global traffic patterns all the packets in a source group would leave it minimally through a single router. If this router also happens to have a global link of the embedded escape topology, it will receive much more traffic than other routers in the group. BCM will then prevent local traffic injection due to the congestion in the links of this router.

The experiments in this subsection show that both topologies, the tree and the Hamiltonian ring, cause load unbalance in the dragonfly network. In the case of the tree, the root group is the one in which traffic concentrates and routers in that group are not able to inject as many packets as routers in other groups. For the Hamiltonian ring, there is one specific router per group that finds difficulties to inject packets in the network. However, whereas with *OFAR Tree* the traffic unbalance appears for every traffic pattern, for *OFAR Ring* the unbalance only appears under adversarial traffic and low traffic loads.

5.7 Length of network paths

The maximum path length in the canonical dragonfly network with local and global misrouting is 8 hops (6 local and 2 global). If a packet enters the escape subnetwork, this length can increase significantly. If a packet followed the escape subnetwork up to its destination, the number of hops would be much higher if the escape subnetwork was a Hamiltonian Ring (up to $N/2$ hops, being N the total number of routers in the

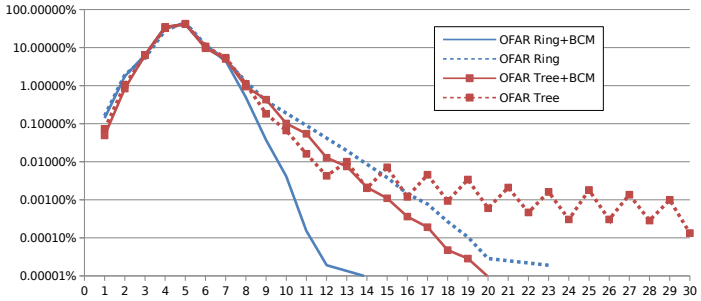


Fig. 13: Hops histogram under adversarial traffic (ADVG+h) using 2/1(+1) VCs

network), than if it was a tree (up to 6 hops). However, the escape subnetwork is used in few occasions, only to escape from potential cyclic dependencies, and packets return to the canonical network as soon as possible. Therefore, packets can enter and leave the escape subnetwork multiple times, making the maximum path length unbounded. Although it would be easy to limit the path lengths by employing a counter of the times a packet can move between the canonical and escape subnetworks, we have preferred not to include ti, to study in depth such issue. This subsection studies this concern when a congestion management mechanism is used, and observe that, in practice, unbounded paths do not happen.

Fig. 13 shows a histogram of the path lengths in traffic consumption experiments as shown in Subsection 5.5 under *ADVG+h* traffic, with and without BCM, and with log scale on the vertical axis. Interestingly, under *ADVG+h* traffic, *OFAR Ring* provides shorter paths than *OFAR Tree*, pointing out that an escape subnetwork with longer average and maximum distances does not necessarily lead to longer paths.

Fig. 13 shows that long paths are really infrequent when congestion management is used. In the simulations carried out, the longest path with *OFAR Tree+BCM* was 20 hops, and with *OFAR Ring+BCM* only 14 hops. Bigger differences appear when there is no congestion management. In that case, in the simulations carried out for this work, whereas for *OFAR Ring* the longest path took 24 hops with *OFAR Tree* one of the packets had to make 217 hops, many of them in the escape subnetwork. *OFAR Ring+BCM* presents the overall best performance.

Multiple injections can occur in the escape subnetwork. When a packet is in the source group and congestion is detected in the canonical network, it goes into the escape subnetwork. If the next group following the escape subnetwork is also congested, the packet will have to wait long to advance to it. As a result, whenever there is space, the packet will return to the canonical network; then, congestion is detected again and the packet is injected once more in the escape subnetwork. This process will be repeated as long as congestion remains. As explained in Subsection 5.6, with the tree escape subnetwork, group G_{root} is more prone to congestion than the rest of the groups. Therefore, in this case multiple injections in the escape subnetwork are more likely.

Livelock might occur if packets bounced consecutively from the escape to the canonical subnetworks. Nevertheless, this situation is not so common as to pose a significant problem. In the worst configuration of *OFAR Tree* under ADVG+h traffic in Fig. 13, more than a 99.99% of the packets need less than 30 hops to reach their destination. Additionally, a traditional alternative to mitigate the potential livelock issue is to employ a counter of injections into the escape path; after the counter exceeds a given threshold (for example, 15 for a 4-bit counter) the packet is not allowed to leave the escape subnetwork again. Since the amount of packets with such a high number of injections is very low, this would not have a significant impact in performance.

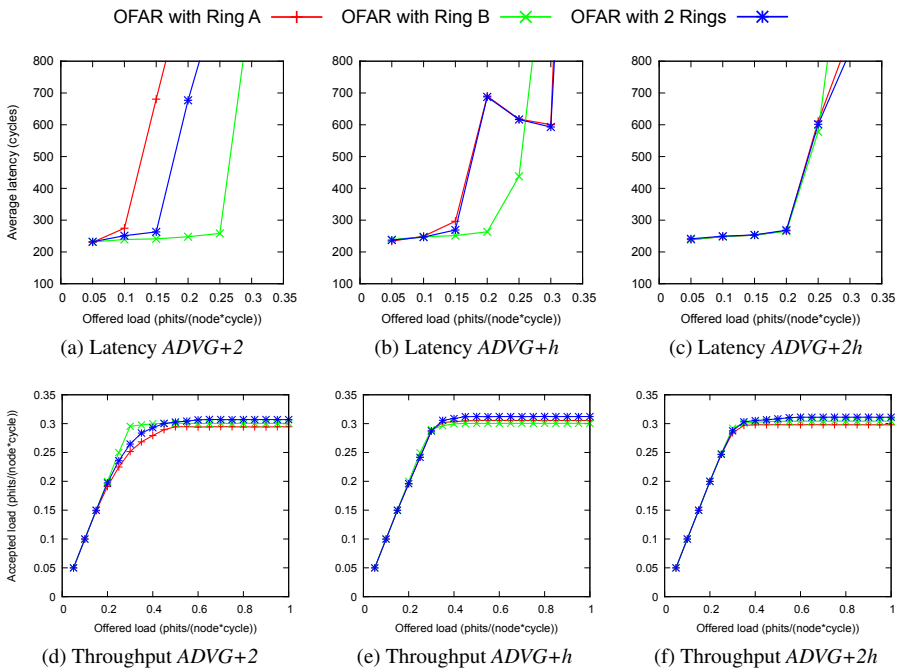
5.8 Single vs. multiple escape subnetworks

This subsection studies the use of multiple escape subnetworks, specifically two disjoint Hamiltonian rings added to a dragonfly of size $h = 6$. Both rings are embedded employing one extra virtual channel per subnetwork. The first ring, *ring A* is the one described in Section 3.3.2, with $R_{in} = 0$ and $R_{out} = 11$. *Ring B* is the second ring described in Section 3.3.3 with $R_{in} = 3$ and $R_{out} = 8$. Ring B connects router i with router $i+7 \pmod a$ within the same group, where $a = 2h = 12$ is the number of routers per group. Each group j is connected to group $j+20 \pmod G$, where $G = 73$ is the total number of groups in this network.

In our implementation, when a packet is generated, it is assigned one of the two escape rings. This means that a packet can only employ that assigned ring for deadlock avoidance. Although not studied in this work, in case of a link failure in a ring, all packets should be automatically assigned the remaining ring.

OFAR performance is compared when employing one and two escape rings using 2/1(+1) VCs. In addition, the different performance results depending on how the single Hamiltonian ring is mapped in the canonical dragonfly are studied. Fig. 14 shows the latency and throughput results for *OFAR* with BCM when the escape subnetwork is the single ring A (*OFAR* with Ring A), the single ring B (*OFAR* with ring B), and when both Hamiltonian rings are employed (*OFAR* with 2 rings). Three adversarial traffic patterns are considered: ADVG+2, ADVG+h and ADVG+2h. In the last case, the global link which minimal paths use to leave the group does not contain a virtual channel for any escape ring.

Each of the three *OFAR* configurations achieves a similar maximum throughput of $\backsimeq 0.3$ phits/(node $\cdot$ cycle). However, Fig. 14a and 14b show that the latency for OFAR with ring A and OFAR with two Rings rises earlier than for OFAR with ring B. OFAR with Ring A presents the same latency curve as in Subsection 5.5, saturating at around 0.1 phits/(node $\cdot$ cycle) when the traffic is ADVG+2, and 0.15 phits/(node $\cdot$ cycle) when the traffic is ADVG+h ($h = 6$). This effect is due to starvation of some of the routers in the network, as explained in Subsection 5.6. This pathology does not show up when the ring is mapped in the canonical dragonfly in a different way: *OFAR* with Ring B. For that configuration, latency rises at approximately 0.27 phits/(node $\cdot$ cycle) in both cases, a value close to the maximum throughput achieved. For adversarial traffic ADVG+2, *OFAR* with two Rings improves the latency results of *OFAR*



(a) Latency *ADVG+2* (b) Latency *ADVG+h* (c) Latency *ADVG+2h*

(d) Throughput *ADVG+2* (e) Throughput *ADVG+h* (f) Throughput *ADVG+2h*

Fig. 14: Latency (up) and throughput (down) under different adversarial traffic patterns (ADVG+2, ADVG+h and ADVG+2h) for *OFAR* with different Hamiltonian ring escape subnetworks when applying BCM and using 2/1(+1) VCs

with Ring A, although it cannot reach those for *OFAR* with Ring B, due to the load unbalance caused by Ring A. However, when the traffic is ADVG+h, a more adversarial traffic for Ring A, *OFAR* with the two rings presents the same behavior as *OFAR* with ring A. The second ring in that configuration, Ring B, is not able to mitigate the load unbalance caused by Ring A.

When the traffic is adversarial ADV+12 (ADV+2h), R_{out} in *OFAR* with Ring A (R_{11}) is not the router through which packets leave the group when using the minimal path (R_{10}). As a result, starvation in R_{out} disappears. Fig. 14c and 14f show the performance results.

The results in this subsection show that employing more than one embedded Hamiltonian ring does not provide performance improvements. On the contrary, the way in which the ring is mapped in the canonical dragonfly does make a difference, as some congestion situations due to certain traffic patterns can be avoided. Arguably, the alternative rings introduced in Subsection 3.3.3 are better options, since it is more unlikely that adversarial traffic coincides with the global links of such rings due to their higher hop in terms of group indexes. In any case, the use of several rings instead of a single one still improves fault tolerance.

6 Related work

Significant details about the dragonfly architecture and routing can be found in [21, 18, 2]. The problem of saturation of global links was introduced in [21, 2]. Local link saturation with inter-group traffic was first evidenced in [19] and the pathological saturation in local links was introduced in [11]. Saturation in local links could be, arguably, more frequent than the saturation of global links, since the applications typically try to exploit the locality between neighbor processes usually allocated sequentially in the same group. Bhatele *et al.* study how mapping choices can impact performance in [3], suggesting random task placement. With such placement local links do not get saturated, at the cost of locality loss.

The simplest routing mechanisms are oblivious to the network status. Minimal routing hierarchically follows the shortest path between each pair of nodes. Valiant routing, as used for the dragonfly in [21], applies global misrouting to each packet regardless of the network status, by selecting an intermediate group. PERCS allows the programmer to specify the intermediate group for each packet [2]. This differs from the original definition from Valiant [26], which would select an intermediate router. This is roughly equivalent to global misrouting followed by local misrouting in the intermediate group. Such a definition is the one employed in the Cray Cascade [8] to avoid pathological congestion issues similar to the case with our *ADVG+h* traffic, and it has been evaluated as "Valiant-any" in [24].

Adaptive routing mechanisms select the path of each packet depending on the network conditions. Two types of adaptive routing mechanisms have been proposed: source and in-transit (or on-the-fly) routing. Source routing mechanisms determine the path of each packet at injection time. Examples of source-routing mechanisms are UGAL, Piggybacking (PB) or CRT [18]. These mechanisms need to estimate the congestion in the global links of the group to select between a minimal or non-minimal routing for each packet. This estimation typically relies on indirect information (for example, the credit count in the outputs of other routers in the group). For this reason, they are relatively complex and slow in adapting to traffic changes. Progressive Adaptive Routing (PAR, [18]) introduced in-transit adaptive routing for Dragonflies.

Günther introduces in [14] the use of an increasing order of buffer classes to prevent deadlock, which is employed in multiple previous routing mechanisms for Dragonflies. This strict policy of ordering virtual channels does not permit in-transit misrouting, which would require re-injecting the packet into the first virtual channel, VC_0, potentially generating a cyclic dependency. The PAR mechanism [18] addresses this limitation by implementing an additional VC.

Our proposal relies on a deadlock-free subnetwork to avoid packet deadlocks, [7]. Based on this idea, Silla and Duato proposed a general deadlock-free mechanism in [25] based on duplicating the number of virtual channels in the network. Their mechanism is not specific to the Dragonfly network and their escape network topology equals the original topology, what would require a larger number of VCs in the case of the Dragonfly. In our proposal we study two different escape topologies: a tree, and a Hamiltonian ring to which injection restriction is applied to avoid deadlock. Seminal deadlock avoidance mechanisms relying on restricting packet injection can be found in [4, 6].

Multiple congestion control mechanisms have been studied and proposed for different networks. A survey of their application in HPC can be found in [13]. Virtually every mechanism relies on injection throttling, such as the transmission window in TCP [17] or Quantized Congestion Notification (QCN, [1]) in Datacenter Bridging. The differences rely on how they detect network congestion.

In this work we study two open-loop local congestion control mechanisms, BCM and ECM, which apply source throttling based on the occupancy of the local queues. Similar mechanisms have been studied in other networks, [22].

7 Conclusions

This paper has identified and addressed the key performance limitations of dragonfly networks. These limitations are mainly the saturation of local and global links and the inefficiency of source routing. We have presented *OFAR* routing, which is an efficient alternative which selects the misroute output port in-transit, rather than at injection time. This adaptive misrouting is enabled by employing an escape subnetwork to prevent deadlock, rather than a fixed order of visiting virtual channels. To avoid congestion, the combination of OFAR with simple injection throttling has been evaluated. This alternative solves the main limitations of the base *OFAR* model: congestion problems and the appearance in practice of very long paths. To tolerate faults, a mechanism that finds multiple disjoint escape rings has been presented. Eventually, such a mechanism can also help avoid network unfairness by avoiding the coincidence of traffic flows with the escape ring paths.

Compared to alternative proposals, *OFAR* only relies on local information, achieves higher performance thanks to its support of local and global misrouting without increasing the number of VCs, and adapts faster to traffic changes. When the cost in terms of VCs is similar, *OFAR* clearly outperforms alternatives such as PB which employ source routing (so they are slower adapting to changes) and do not support local misrouting.

Implementations with a very low number of VCs suffer from congestion and unfairness issues. In such cases, two congestion management mechanisms, BCM and ECM, and two escape subnetwork topologies, a Hamiltonian ring and a tree, have been evaluated. Congestion management avoids escape network saturation that could lead to a severe performance drop. The effect of the topology of the escape subnetwork on the network load imbalance and performance has been analyzed. Despite path lengths with *OFAR* could be unbounded in theory, results have shown they are relatively short in practice and that the use of a congestion management mechanism reduces them. Finally, the use of multiple escape subnetworks helps improve fault-tolerance, but not performance; our evaluations highlight that escape subnetworks should be designed so they do not overlap with frequent traffic patterns. Thus, the alternative escape rings introduced in this work appear as the best option for the escape subnetwork.

Acknowledgements

This work has been supported by the Spanish Ministry of Education, FPU grant AP2010-4900; the Spanish Science and Technology Commission (CICYT) under contracts TIN2010-21291-C02-02, TIN2012-34557 and TIN2013-46957-C2-2-P; the European Union FP7 under Agreements ICT-288777 (Mont-Blanc) and ERC-321253 (RoMoL); the European HiPEAC Network of Excellence and the JSA no. 2013-119 as part of the IBM/BSC Technology Center for Supercomputing agreement.

References

1. IEEE standard for local and metropolitan area networks - virtual bridged local area networks - amendment: 10: Congestion notification, 802.1Qau (2010)
2. Arimilli, B., Arimilli, R., Chung, V., Clark, S., Denzel, W., Drerup, B., Hoefler, T., Joyner, J., Lewis, J., Li, J., et al.: The PERCS high-performance interconnect. In: 2010 18th IEEE Symposium on High Performance Interconnects, pp. 75–82. IEEE (2010)
3. Bhatele, A., Gropp, W.D., Jain, N., Kale, L.V.: Avoiding hot-spots on two-level direct networks. In: High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for, pp. 1 –11 (2011)
4. Brookes, S., Roscoe, A.: Deadlock analysis in networks of communicating processes. Distributed Computing **4**(4), 209–230 (1991)
5. Carrion, C., Beivide, R., Gregorio, J., Vallejo, F.: A flow control mechanism to avoid message deadlock in k-ary n-cube networks. In: High-Performance Computing, 1997. Proceedings. Fourth International Conference on, pp. 322 –329 (1997). DOI 10.1109/HIPC.1997.634510
6. Cidon, I., Ofek, Y.: Metaring-a full-duplex ring with fairness and spatial reuse. Communications, IEEE Transactions on **41**(1), 110–120 (1993). DOI 10.1109/26.212370
7. Duato, J.: A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. Parallel and Distributed Systems, IEEE Transactions on **6**(10), 1055–1067 (1995)
8. Faanes, G., Bataineh, A., Roweth, D., Court, T., Froese, E., Alverson, B., Johnson, T., Kopnick, J., Higgins, M., Reinhard, J.: Cray cascade: a scalable HPC system based on a dragonfly network. In: Intl Conf on High Performance Computing, Networking, Storage and Analysis, SC '12, pp. 103:1–103:9. IEEE Computer Society Press, Los Alamitos, CA, USA (2012)
9. García, M., Fuentes, P., Odriozola, M., Vallejo, E., Beivide, R.: FOGSim Interconnection Network Simulator. University of Cantabria (2014). URL https://code.google.com/p/fogsim/
10. García, M., Vallejo, E., Beivide, R., Odriozola, M., Camarero, C., Valero, M., Labarta, J., Rodríguez, G.: Global misrouting policies in two-level hierarchical networks. In: Interconnection Network Architecture: On-Chip, Multi-Chip, pp. 13–16 (2013)
11. García, M., Vallejo, E., Beivide, R., Odriozola, M., Camarero, C., Valero, M., Rodríguez, G., Labarta, J., Minkenberg, C.: On-the-fly adaptive routing in high-radix hierarchical networks. In: Intl. Conference on Parallel Processing (ICPP) (2012)
12. García, M., Vallejo, E., Beivide, R., Valero, M., Rodríguez, G.: OFAR-CM: Efficient dragonfly networks with simple congestion management. In: High-Performance Interconnects (HOTI), 2013 IEEE 21st Annual Symposium on, pp. 55–62 (2013). DOI 10.1109/HOTI.2013.16
13. Garcia, P.J.: Congestion management in HPC interconnection networks. HPC Advisory Council European Workshop (2011)
14. Gunther, K.: Prevention of deadlocks in packet-switched data transport systems. Communications, IEEE Transactions on **29**(4), 512 – 524 (1981). DOI 10.1109/TCOM.1981.1095021
15. Gupta, P., McKeown, N.: Designing and implementing a fast crossbar scheduler. Micro, IEEE **19**(1), 20–28 (1999)
16. IEEE 802 LAN/MAN Standards Committee: IEEE 802.1d-2004 MAC bridges (2004)
17. Jacobson, V.: Congestion avoidance and control. In: ACM SIGCOMM Computer Communication Review, vol. 18, pp. 314–329 (1988)
18. Jiang, N., Kim, J., Dally, W.J.: Indirect adaptive routing on large scale interconnection networks. In: ISCA '09: 36th International Symposium on Computer Architecture (2009)

19. Kerbyson, D.J., Barker, K.J.: Analyzing the performance bottlenecks of the POWER7-IH network. In: CLUSTER, pp. 244–252. IEEE (2011)
20. Kermani, P., Kleinrock, L.: Virtual cut-through: A new computer communication switching technique. Computer Networks (1976) **3**(4), 267–286 (1979)
21. Kim, J., Dally, W., Scott, S., Abts, D.: Technology-driven, highly-scalable dragonfly topology. In: Proceedings of the 35th Annual International Symposium on Computer Architecture, pp. 77–88. IEEE Computer Society (2008)
22. Lam, S., Reiser, M.: Congestion control of store-and-forward networks by input buffer limits–an analysis. Communications, IEEE Transactions on **27**(1), 127–134 (1979). DOI 10.1109/TCOM.1979.1094280
23. Pinkston, T.: Deadlock characterization and resolution in interconnection networks. Deadlock Resolution in Computer-Integrated Systems pp. 445–492 (2004)
24. Prisacari, B., Rodriguez, G., Garcia, M., Vallejo, E., Beivide, R., Minkenberg, C.: Performance implications of remote-only load balancing under adversarial traffic in dragonflies. In: 8th International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip, INA-OCMC '14 (2014). DOI 10.1145/2556857.2556860
25. Silla, F., Duato, J.: High-performance routing in networks of workstations with irregular topology. IEEE Trans. Parallel Distrib. Syst. **11**(7), 699–719 (2000). DOI 10.1109/71.877816. URL http://dx.doi.org/10.1109/71.877816
26. Valiant, L.: A scheme for fast parallel communication. SIAM journal on computing **11**, 350 (1982)