On-the-Fly Adaptive Routing in High-Radix Hierarchical Networks

Marina García*, Enrique Vallejo*, Ramón Beivide*, Miguel Odriozola*, Cristóbal Camarero*,
Mateo Valero†, Germán Rodríguez‡, Jesús Labarta†, and Cyriel Minkenberg‡
* University of Cantabria, Spain. {garciamar, valleje, beivider, camarereoc}@unican.es
† Universitat Politècnica de Catalunya and BSC. Barcelona, Spain. {mateo.valero, jesus.labarta}@bsc.es
‡ IBM Zurich Research Laboratory, Rüschlikon, Switzerland. {sil, rod}@zurich.ibm.com

***Abstract*—Dragonfly networks have been recently proposed for the interconnection network of forthcoming exascale supercomputers. Relying on large-radix routers, they build a topology with low diameter and high throughput, divided into multiple groups of routers. While minimal routing is appropriate for uniform traffic patterns, adversarial traffic patterns can saturate inter-group links and degrade the obtained performance. Such traffic patterns occur in typical communication patterns used by many HPC applications, such as neighbor data exchanges in multi-dimensional space decompositions. Non-minimal traffic routing is employed to handle such cases. Adaptive policies have been designed to select between minimal and nonminimal routing to handle variable traffic patterns.**

However, previous papers have not taken into account the effect of saturation of intra-group (local) links. This paper studies how local link saturation can be common in these networks, and shows that it can largely reduce the performance. The solution to this problem is to use nonminimal paths that avoid those saturated local links. However, this extends the maximum path length, and since all previous routing proposals prevent deadlock by relying on an ascending order of virtual channels, it would imply unaffordable cost and complexity in the network routers.

In this paper we introduce a novel routing/flow-control scheme that decouples the routing and the deadlock avoidance mechanisms. Our model does not impose any dependencies between virtual channels, allowing for on-the-fly (in-transit) adaptive routing of packets. To prevent deadlock we employ a deadlock-free escape subnetwork based on injection restriction. Simulations show that our model obtains lower latency, higher throughput, and faster adaptation to transient traffic, because it dynamically exploits a higher path diversity to avoid saturated links. Notably, our proposal consumes traffic bursts 43% faster than previous ones.

***Index Terms*—adaptive routing; dragonfly;**

I. Introduction

Interconnection networks constitute a key subsystem in the architecture of supercomputers. Direct network topologies are those that distribute routers among nodes. Technology trends suggest the use of high-degree routers to exploit the available pin bandwidth [1]. To this extent, two-layered hierarchical networks, denoted as Dragonflies in [2], have been proposed. The IBM PERCS [3] and the forthcoming machines from the Echelon project [4] are being designed to employ a network of this type. This paper focuses on such networks.

Dragonflies are organized as groups of routers. Links between routers can be either *local* or *global*. Routers within a group are interconnected by means of a complete graph using *local* electrical wires, using one link between any pair of routers of the group. Groups are also interconnected by means of a complete graph, using one *global* optical link between any pair of groups. In the PERCS terminology the terms LL and LD are used for local links and D for global links. The main Dragonfly topological parameters, as defined in [2], 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 with no oversubscription, the equations $a = 2 \times p = 2 \times h$ must hold, [2]. For a given h, the maximum size network will be composed of $2h^2 + 1$ groups, $4h^3 + 2h$ routers and $4h^4 + 2h^2$ processing nodes. The total number of ports per router is $4h - 1$. Figure 1 represents an example of a Dragonfly with $h = 2$. A Dragonfly network built from routers with 64 ports ($h = 16$, such as the PERCS technology [3]) scales to more than 256K processing nodes, leading to multi-million core supercomputers.

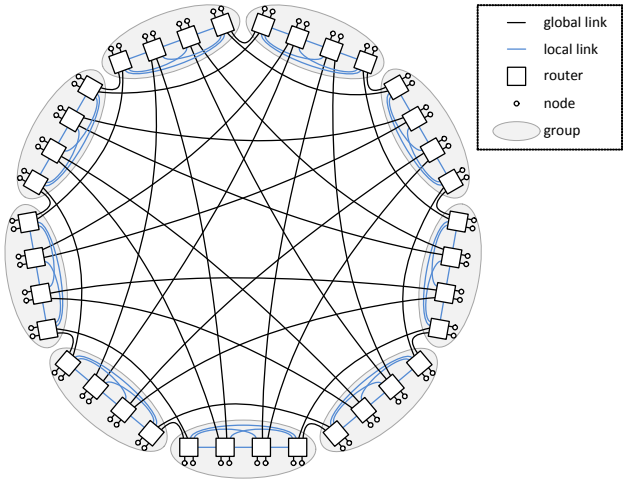


Fig. 1: Sample Dragonfly topology with h=2 (p=2, a=4), 36 routers and 72 compute nodes.

The diameter of the Dragonfly topology is 3, so any minimal path between two routers will employ at most 3 hops. With this minimal routing, a packet typically first traverses a local (l) link at the source group, then a global (g) one to reach the destination group, and finally another local link

at the destination group (path $l_1 - g_1 - l_2$). However, since there is only one global link between any pair of groups (each group comprising $2h^2$ nodes), adversarial traffic patterns contending for global links can be common. In such demanding conditions, nonminimal[1] routing can be used by randomly selecting an intermediate group to which the packet is sent before heading to its destination, [7], [2]. Under such nonminimal routing, a packet traverses at most 3 local links and 2 global ones (path $l_1 - g_1 - l_2 - g_2 - l_3$, in which the two first hops are used to arrive at the intermediate group). This traffic randomization balances the use of global links reducing contention, but doubles their average utilization halving throughput and increasing latency. Adaptive routing mechanisms select between minimal or nonminimal for each packet sent, depending on the conditions of the network.

In realistic scenarios, common HPC applications with simple near-neighbor communications easily lead to hot-spots in Dragonflies. Bhatele *et al.* study the impact of such hotspots using traces of real applications simulations [5]. They quantify the large impact of such link saturation by observing that different traces on a large simulated system are executed 1.72 to 3.06 times *faster* only by enabling nonminimal routing (Table 6 and Figure 13 of their paper, DEF vs. DFI). However, they do not consider adaptive routing mechanisms in their study. They also explore how to mitigate these hotspots by randomizing the task mapping mechanism.

The Dragonfly topology induces the appearance of cyclic routing dependencies, requiring a deadlock avoidance strategy. Former proposals for routing in Dragonflies, [2], [3], rely on a set of virtual channels (VCs) that must be visited in a predefined order. The number of VCs required equals the length of the longest path: minimal routing allows for paths of length 3, while nonminimal allows for length 5. However, since local links are always used in odd hops (1, 3 and 5) and global links are used in even hops (2 and 4), it is enough to implement 3 VCs in the inputs of local links and 2 VCs in global links. Shorter paths (e.g. $l-g-g-l$ under nonminimal routing) can be rearranged to employ only the implemented VCs by skipping indexes corresponding to missing hops.

The fixed order for visiting the VCs prevents from dynamically readapting traffic when changing conditions are detected, unless additional resources are implemented: The Progressive Adaptive Routing (PAR) mechanism introduced in [6] allows for two local hops in the source group, but this requires an additional VC implemented in the local links. For any other proposed routing mechanism, the deadlock avoidance mechanism becomes a limitation as the path to be traversed, minimal or not, has to be decided at injection time. In any case, none of the proposed routing mechanisms allows the routers to misroute traffic to avoid *saturated local links*; although this has been largely ignored in previous works, we will discuss how it can be a frequent problem depending on the workload, imposing a significant penalty on the performance which grows with the network size.

In this paper we introduce **OFAR**: an **O**n the **F**ly **A**daptive **R**outing for Dragonfly networks. OFAR is a new flow-control/routing mechanism that decouples the way in which virtual channels are visited from the deadlock avoidance mechanism. This allows each router to dynamically misroute packets depending on the observed local contention. Such a dynamic mechanism improves performance as it simplifies routing and allows for faster adaptation to transient traffic over-loads. The price to be paid for such flexibility depends on the solution employed to avoid deadlocks. In general, deadlock is quite infrequent in high-degree networks using adaptive routing [8]. Therefore, a very simple solution based on a Hamiltonian ring with restricted packet injection [9] is employed as a safe escape subnetwork [10]. Our evaluations show that this model improves throughput and response time under different traffic patterns, even in the most adverse cases. The main contributions of this paper are the following:

- We identify the performance limitations of previous proposals for routing in Dragonflies. We provide the key insight that under certain traffic patterns it is the saturation of *local* links which most limits performance even when Valiant routing is used.
- We introduce *OFAR*, a novel flow-control/routing mechanism. Qualitatively, *OFAR* improves over previous proposals allowing for dynamic in-transit misrouting to prevent excessive contention, without relying on remote sensing of the network status.
- We evaluate our proposal using different traffic patterns and transient loads. Quantitatively, *OFAR* provides low latency and high throughput under different traffic patterns, while preserving a very competitive adaptation time. Under traffic bursts, *OFAR* completes traffic delivery 43 % faster than previous proposals.

We will start the paper by discussing some related prior work. We next introduce one of the motivations of the work, which is the saturation of local links under certain traffic patterns. Next, we introduce our mechanism and its evaluation.

II. Related Work

Significant details about the Dragonfly architecture and routing can be found in [2], [6], [3]. As there is a single minimum path between any pair of groups, a load-balancing mechanism that distributes traffic along nonminimal routes is required when managing adversarial traffic. Similar algorithms have been also proposed to route packets non-minimally. Hot-Potato routing [11] deflects packets to nonminimal routes when detecting collisions for output ports. A similar misrouting technique is used in the fully adaptive Chaos router, [12].

Many supercomputer networks use deadlock avoidance mechanisms based on a structured set of virtual channels (VCs) per router link, which is managed under a restrictive policy. Virtual channels regulated under a dateline policy, as proposed in [13], have been widely used for breaking cyclic dependencies in different ring-based networks. In Dragonflies,

[1]Note that different authors use the term *indirect routing* to mean either *nonminimal* routing (e.g. Valiant) [5], [3], or adaptive routing using *remote* information [2], [6]. Our proposal allows for nonminimal adaptive routing without remote information. We omit the term *indirect* to avoid confusion.

the mechanism used until now is also based on a restrictive use of VCs per each link. Günther introduces in [14] the idea of using an increasing order of buffer classes to prevent deadlock.

This strict policy of ordering virtual channels does not allow misrouting an in-transit packet. This would require to dynamically re-inject the packet into the first virtual channel, VC_0, potentially generating a cyclic dependency. The PAR mechanism [6] addresses this limitation by requiring an additional VC to prevent deadlock. Any other previous proposal requires the injection node or source router to decide whether each packet should be misrouted or not, and to select the intermediate destination. Multiple mechanisms dynamically misroute or not at injection time, [2], [6], [3]. Valiant routing, [7], always misroutes the packet to a randomly selected group. Other mechanisms first select a random intermediate group, and then determine whether to misroute the packet or not based on the occupancy information of both paths: UGAL-L, [2], relies on the occupancy of the queues at the injection router; Piggybacking (PB, [6]) relies on remote congestion information broadcast across the group; CRT, [6], measures the credit round-trip time on the local channels at the injection router. Finally, in the PERCS fabric, [3], the intermediate group can be selected using a round-robin scheme, or it can be specified by the programmer. While the latter is more flexible, it adds a significant burden to the programmer, who should be aware of the underlying topology to optimize performance.

Our proposal relies on a regulated-injection sub-network to avoid packet deadlocks, [10]. Seminal deadlock avoidance mechanisms relying on restricting packet injection can be found in [15], [16]. This policy prevents deadlocks by always keeping enough buffer space to guarantee packet movement inside potential network cycles. Restrictive packet injection policies do not need VCs which is appealing for its simplicity and scalability. Nevertheless, VCs can be added for improving performance by reducing head-of-line (HOL) blocking. The IBM BlueGene supercomputer family uses a deadlock avoidance mechanism based on restricted packet injection, [9], [17], [18], [19]. A restricted injection Hamiltonian ring was also proposed in [20] to tolerate failures on torus networks.

III. Motivation: Study of the saturation of local links

Until now, global links have been assumed as the only potential network bottleneck which would limit performance in a Dragonfly network [2], [3]. This is a reasonable approach, since the saturation of global links has the largest impact on the maximum network throughput. We will detail it next, to be able to compare this problem with the one in local links.

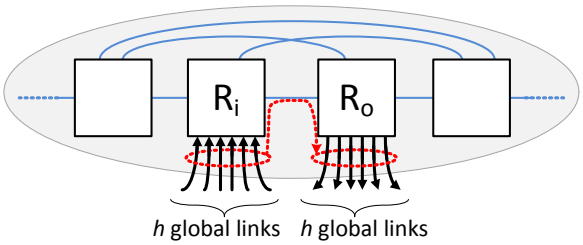
A Dragonfly network is dimensioned with as many processing nodes as outgoing global links, $h = p$. As one global hop is required for each packet with minimal routing, this allows for maximum performance under uniform traffic. However, in a worst case the $2h^2$ nodes in one group could send traffic to the same destination group, competing for the bandwidth of a single global link. This would limit the maximum bandwidth to $1/(2h^2)$ using minimal routing. In a large network with $h = 16$, this reduces throughput to less than 0.2% of its maximum, while leaving a large fraction of the global links underutilized. With Valiant routing, each packet will be sent to a random intermediate group, and then minimally to its destination. Since this implies two global jumps, on average, global links will limit the maximum throughput to $1/2$ phits/(node·cycle). Note that the initial problem is not the scarcity of global links (since $p = h$), but their unbalanced use under minimal routing, caused by the traffic pattern.

Analogously, local links also saturate when all the h compute nodes attached to a router send traffic to the nodes in a neighbor router of the same group. The single local link between these routers can transmit 1 phit/cycle, so the maximum traffic under minimal routing would be $1/h$ in this case. For the same large network $h = 16$ this would limit traffic to a 6.25% of its maximum. Again, the problem is not a lack of local links in a group, but of their unbalanced use. Valiant routing might rise this value to 50%, but it unnecessarily increases the use of global channels by sending the traffic to an intermediate group back and forth.

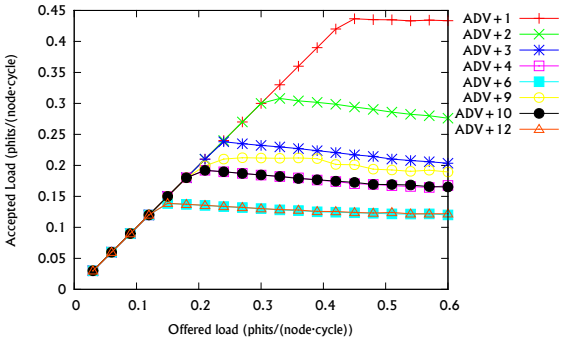
This effect should be, arguably, more frequent than the saturation of global links, since applications typically exploit the locality between neighbor processes, and those neighbor processes are typically allocated sequentially in the same group. Bhatele *et al.* study the utilization of local links for different applications [5]. Their results (Figures 5 to 8, 11 and 12, DEF mapping) confirm that *some* local links support a much larger amount of traffic than others. After our discussion, it is now obvious why their approach of randomizing the mapping of tasks to nodes inside a group achieves higher performance, since it removes the bottlenecks in local links. However, we believe that a proper solution should be applied at the network level, since randomizing the task mapping breaks the benefits of locality among neighbor tasks allocated in the same router. The use of nonminimal routing in the same work (we suspect that it is employed even for traffic internal to a given group) also leads to balanced usage of the local links.

Nevertheless, even with a nonminimal routing mechanism, there can be second-order congestion effects derived from the saturation of local links. As we will show next, there is a variable severity among different adversarial traffic patterns that, in some cases, provokes saturation on local links.

We consider patterns in which every source node in group i selects a destination node in group $i+N$, denoted as *ADV+N*, with N lower than the number of groups. We assume Valiant routing, in the general case with misrouting applied to an intermediate group different from the source and destination groups. In such case, up to 5 hops, $l_1 - g_1 - l_2 - g_2 - l_3$, are needed to get to destination. The two first hops $l_1 - g_1$ lead to the 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 the intermediate group. As argued before, Valiant routing will limit the throughput to 0.5 phits/(node·cycle). However, for certain adversarial traffic patterns, the local link l_2 will saturate, even when this leaves global links partially idle.



(a) Two routers in a group, joined with a single link.



(b) Accepted load vs. offered load in a $h = 6$ dragonfly with Valiant routing and different Adversarial Traffic Patterns

Fig. 2: Study of the throughput of adversarial traffic patterns

We consider now the adversarial traffic pattern *ADV+h*. Figure 2a shows two routers R_i and R_o of a given group G_i. Lets consider the traffic misrouted towards group G_i which is received through the h global links entering to R_i. All these packets will have to be forwarded through the h subsequent global links. As global wiring is typically consecutive (observe the topology in Figure 1), all these links happen to be in the next router, R_o. Then, all misrouted traffic received in R_i has to be forwarded to R_o through the single local link connecting both routers. This link 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·cycle). The same happens for any other *ADV+n·h* traffic pattern. When $h > 2$, it is clear that some local links (and not the global ones) will constitute the network bottleneck for this worst-case traffic. This throughput limitation will grow with the network size h, which is important as the presented problem could pass unnoticed with small-size simulations.

Depending on the connection pattern chosen for the Dragonfly, the specific worst-case traffic pattern will vary. Nevertheless, there will always be a combination of source and destination groups that will lead to severely reduced performance. Hence, the maximum throughput for nonminimal traffic will depend on the specific offset value between the source and the destination groups. Figure 2b shows how throughput notably varies depending on this offset between groups, even in a small Dragonfly with $h = 6$ under Valiant routing.

The simplest approach to avoid this saturation would be to allow for *local misrouting*, this is, diverting packets to a neighbor router to avoid a saturated local link. However, this should be allowed in any group which a packet traverses, leading to very long paths: $l - l - g - l - l - g - l - l$, or even longer if multiple nonminimal hops are allowed per group. Using the deadlock avoidance mechanisms in previous proposals, this would require 6 VCs in the local channels. By contrast, the OFAR mechanism introduced next decouples the use of VCs and the deadlock avoidance, allowing for deadlock-free routing with a reduced amount of resources.

IV. *OFAR*: A New Flow-Control/Routing for Dragonflies

The main limitations of previous routings for Dragonflies and the problems derived from saturated local links have been introduce in previous sections. In this section we introduce our mechanism *OFAR*, which differs with respect to previous ideas on several key points: *i)* adaptive in-transit misrouting, rather than determined at injection time; *ii)*random intermediate destinations dynamically determined by credits of the global ports of the current router, rather than using remote information; and *iii)* the existence of a deadlock-free sub-network to guarantee deadlock avoidance. We explore each of these issues next.

A. *Dynamic misrouting in* OFAR

In *OFAR* traffic can be sent non-minimally in each router to avoid network congestion. This misrouting can use local or global links of the current router. In principle, our mechanism allows for any number of misroutings without additional cost, but in practice it is better to set a limit. We restrict the number of times that this misrouting can be applied to prevent livelock: at most, one global non-minimal hop can be applied per packet, and one non-minimal local hop can be applied per group. We include two flags in the packet header to limit this misrouting. Therefore, when the escape subnetwork that will be detailed in section IV-C is not used, we limit the longest path to 8 hops (2 global and 6 local). In practice, such long paths hardly ever occur.

When traffic is internal to a group, only local misroute is allowed; when traffic is external to the group, both local and global misroutes are allowed. Global misroute always occurs when the packet is still in its source group, otherwise the packet would have reached the destination group and leaving it would make no sense. Each packet in each input buffer always has a 'minimal output' according to its minimal output to the destination node. Depending on the credits of the minimal output (none at all, or below a given threshold as detailed in Subsection IV-B) and the header flags, the control logic of the input unit of the router can try to misroute the packet to an output with more credits. In a group not being the source,

only local misrouting is allowed when the minimal output is a saturated local port.

We use the following policy to determine which port to use for misrouting in the source group. When the packet is still in the source group, the misroute type (local or global) will depend on the type of the input buffer that contains the packet. Those packets in injection queues (still in their source router)are misrouted by global channels. This saves the first local hop when using Valiant routing. By contrast, packets in local queues are first misrouted locally, and then globally. Although not obvious, this prevents starvation issues when the traffic pattern is adversarial. The explanation is the following: When the traffic is adversarial, there will be a single global output port in one router of the group (router R_{out}) which will be saturated. All other routers will send their packets there, so R_{out} will have $2h+1$ local input ports plus h injection ports whose minimal path is the saturated global link. If all the packets in these queues were sent non-minimally through the remaining $h-1$ global queues, they would quickly saturate, leading to starvation issues for the nodes in R_{out}.

Finally, in the evaluations of the next section we have included two *OFAR* models. The base *OFAR* model is the one described above. By contrast, the *OFAR-L* model does not allow for local misroute, same as the previously proposed routing mechanisms for the Dragonfly. This is used to dissect the specific benefits of using local misrouting in the routing mechanism.

B. Contention-aware misrouting in OFAR

Previous proposals select a random intermediate destination *before* determining if the packet should be sent minimally or not. By contrast, *OFAR* relies on the contention observed in the minimal (or Valiant) path to allow for non-minimal routing. 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 which is its corresponding minimal (or Valiant) path, along with the allowed non-minimal paths (e.g., using local or global links, since all of them are isomorphic in the topology). Depending on the measured network congestion, the allocator input unit can request the minimal path or one of the non-minimal ones.

To decide if misrouting is applied, *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 (*non-minimal queue*). 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. To determine when misrouting is allowed, we use two thresholds: Th_{min} and $Th_{non-min}$. Specifically, misrouting is allowed only when $Q_{min} >= Th_{min}$ ***and*** the minimal port is not available (it is already assigned to another input or $Q_{min} = 100\%$). When misrouting is allowed, each input unit will request a random output port among those non-minimal ports that fulfil the occupancy condition $Q_{non-min} <= Th_{non-min}$. This prevents misrouting packets to a group which is already congested. Note that always selecting the least congested output would not be appropriate, since multiple input ports could compete for the same output in case of congestion.

These two threshold values can be static, for example, $Th_{min} = 100\%$ and $Th_{non-min} = 40\%$. In such case, misroute only occurs when the minimal path has no credits left, using an output with at least the 60% of its credit count available. Alternatively, the misrouting threshold can be variable, depending on the occupancy of the minimal queue; for example, $Th_{min} = 0\%$ and $Th_{non-min} = 0.75 \times Q_{min}$. In such model, misrouting is allowed at any time if the minimal queue is not available (it has been assigned to another packet although credits can remain), but only by those queues that have less than 0.75 times the occupancy of the minimal queue. Under benign traffic the minimal queue occupancy will be typically similar to other queues in the router, so misrouting should not be frequent. In contrast, when traffic is adverse, the occupancy of the minimal queue will be much higher than in other queues, so misrouting should be frequent.

C. Deadlock-free subnetwork in OFAR

The proposed *OFAR* routing can generate cyclic dependencies that block the network. Our proposal relies on the existence of a deadlock-free subnetwork added to the original network, [10]. There are multiple alternatives for such a subnetwork that would lead to different implementation cost and performance. In this work, we consider one of the simplest solutions: we use a Hamiltonian ring with bubble flow control [9]. Packets are freely allowed to circulate in the escape ring, as long as there is space in the next buffer for the whole packet. However, when a packet is deflected to the escape ring from the canonical Dragonfly network, an extra free space for another packet is required (a bubble). In highly congested scenarios, some packets will enter this escape ring, partially increasing the length of their paths to destination. This escape subnetwork can be added either physically or virtually to the base topology. If we consider a physically added escape ring, it requires two additional ports per router and N additional wires on a N-router Dragonfly. Alternatively, a virtual embedded Hamiltonian ring maintains the same topology but requiring only an extra virtual channel in the corresponding links.

The ring is only used as the last resort to route a packet. On each hop, only if a packet cannot advance because its minimal path is congested, and misrouting is not possible (because of the misroute threshold $Th_{non-min}$ or the header flags), the corresponding escape output is requested. When a packet enters the escape ring it will try to abandon it as soon as a minimal route is available during its advancement.

It can be argued that the limited bisection bandwidth of the ring together with its high average distance would quickly saturate it, leading to overall limited throughput. Nevertheless, the main purpose of the escape network is to avoid deadlock, not to transport traffic to its final destination. As it will be shown, *OFAR* rarely uses the escape ring as deadlock is quite infrequent in our scenario. It is known that a high routing freedom as the one used by *OFAR* over large degree routers,

reduces the deadlock probability, [8]. Finally, in extremely loaded networks there could be the possibility of packet livelock. A packet could be inserted in the ring to prevent deadlock, and then return to the previous router using a minimal path. This is avoided by limiting the number of times that a packet can abandon the escape ring.

V. Methodology

We have implemented the different routing proposals on an in-house developed single-cycle simulator. We model an input FIFO buffered Virtual Cut-through (VCT) router, [21]. Compared to wormhole, VCT simplifies the router architecture by not requiring virtual channel allocators. It requires input buffer space enough for a whole packet, but this is typically already provided in large-scale networks because of the flow control requirements dictated by the round-trip latency. Our measure unit is the "phit", the smallest physical unit of information that is transferred in one cycle across a physical link. In a router, one phit can be transferred through the crossbar from the head of an input buffer to any output port in each cycle.

The routing decision for a packet is taken when it reaches the head of an input buffer. This selects between the preferred minimal output or one of those non-minimal outputs allowed by the misrouting policy and the misroute thresholds. While the former is constant, different non-minimal routes can be considered as the network conditions change. Then, the routing decision is revisited every cycle as long as the packet remains in the queue head. We use 2 VCs per global link and 3 per local link and injection queues. These are the values required by previous mechanisms to avoid deadlock. VCs are not required to prevent deadlock in *OFAR* but we use them to reduce HOL blocking. We employ the same number of VCs for the escape ring for regularity, although the injection restriction already guarantees deadlock freedom. We do not model any router speedup, since it would make the design of the large-radix router even more complex. However, to prevent performance loss due to output contention, we model an iterative separable batch allocator, resembling the design in [22], with three iterative arbitration cycles. Each arbiter employs a least-recently served (LRS) policy. Note that these aspects differ from previous evaluations of the Dragonfly topology in [2], [6], so the results will not be necessarily the same.

We modeled a maximum size Dragonfly with $h = 6$. Overall, this network contains 5,256 processing nodes, requiring 2,628 global links and 4,818 local links. It is composed of 876 routers organized in 73 groups of 12 routers, with 23 ports each (25 with the physical escape ring model in *OFAR*). We use packets of 8 phits. The default network latencies are 10 cycles for local links and 100 cycles for global ones. Each local FIFO can store 32 phits, and 256 phits in the case of global FIFOs, enough for the flow control requirements dictated by round-trip latencies. The *OFAR* models employ a variable misroute threshold, $Th_{min} = 0\%$ and $Th_{non-min} = 0.9 \times Q_{min}$. As presented in Subsection IV-B, misrouting is allowed at any time if the minimal queue is not available (it has been assigned to another packet or credits have been exhausted), but only by those queues that have less than 0.9 times the occupancy of the minimal one. The selection of this policy was empirical, by simulating the network with variable threshold factors, 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.

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·cycle). The destination node is selected depending on the traffic model:

- Uniform (*UN*): The destination node is randomly selected among all the possible destinations, including the source group but not the source node itself.
- Adversarial$+N$ (*ADV+N*): The destination node is randomly selected among all nodes in the group $i+N$, where i is the source group. *ADV+1* causes the lower congestion on local links, while *ADV+n·h* generates the maximum one, as presented in Subsection III.

Finally, we have implemented the following routing mechanisms:

- Minimal (*MIN*): The packet traverses the minimal path between the source and destination.
- Valiant (*VAL*): The packet always indirectly travels to a randomly selected intermediate group, and then, travels minimally to destination.
- Piggybacking (*PB*, [6]): The injection router selects between minimal and non-minimal paths based on remote congestion information broadcast among all the routers of each group.
- *OFAR*: The base model presented in Subsection IV.
- *OFAR-L*: The same model, without allowing the misroute in local links.

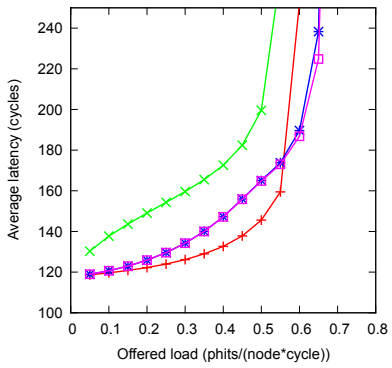
VI. Performance results

We measured the network performance in three different scenarios: steady state, transient variations and traffic bursts. We detail each of these cases next.

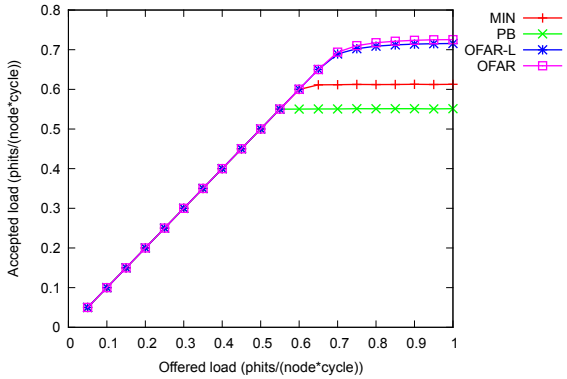
A. Steady state

On these tests we measure the average latency and throughput over a long period, after a sufficient network warm-up. Each point in the plots shows the measured value for a given offered load in phits/(node·cycle).

Figure 3a shows the average latency under uniform random traffic (*UN*). Using *MIN* as a reference, we observe that *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. Figure 3b reports throughput. The *OFAR* models improve over *MIN* and *PB*, but in either case, the use of local misrouting does not make a significant difference. As Valiant routing halves maximum

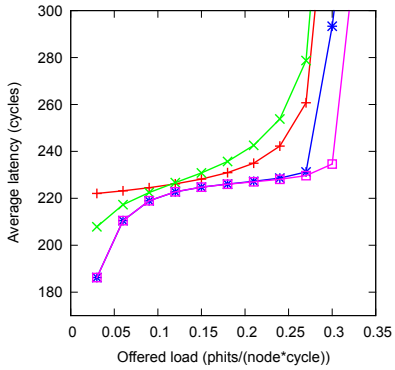


(a) Latency.

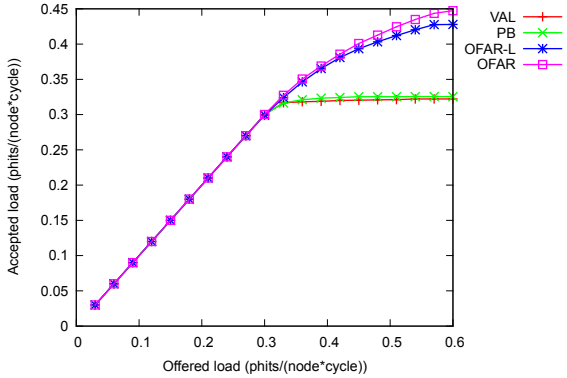


(b) Throughput.

Fig. 3: Latency and throughput under random uniform traffic (UN).



(a) Latency.



(b) Throughput.

Fig. 4: Latency and throughput under adversarial +2 traffic (ADV+2).

throughput achieved under *UN* traffic, it has been omitted in Figure 3.

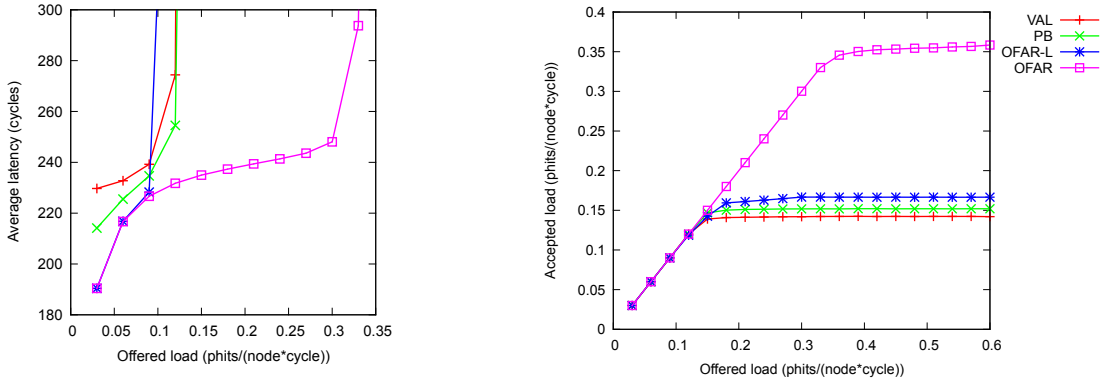
Figure 4 shows results under adversarial traffic *ADV+2*. We do not show results of *ADV+1* as it could be argued that the additional ring link between the source and destination groups favors the *OFAR* models. However, the results are similar to the case *ADV+1* and, as we will discuss later, the escape ring is hardly ever used. In this traffic pattern, the reference is *VAL*, which always misroutes traffic, instead of *MIN*, which suffers from strong congestion caused by the saturated global links.

We can observe that the *OFAR* model shows very competitive latency values. Regarding throughput, Figure 4b shows that the *OFAR* saturates at 0.45, when comparatively, *PB* saturates around 0.38. The difference comes mainly from the better performance of in-transit adaptive routing decisions with larger path diversity in *OFAR*, rather than the use of delayed information about congestion in global queues and the evaluation of a single nonminimal alternative in *PB*. Under this traffic pattern, we can observe how the complete *OFAR* model achieves better performance (especially in terms of throughput) than the *OFAR-L* model, but the difference is very low.

Finally, we evaluated network performance under the worst traffic pattern, *ADV+6*. It is presented in Figure 5. In this case, misrouted traffic can generate the largest congestion in local links as described in Subsection III. If this is not avoided, throughput would be limited to $1/h = 1/6 = 0,166$ phits/(node·cycle). Figure 5a shows that this occurs for *VAL, PB* and *OFAR-L*. *OFAR* obtains, by far, the best result. Throughput, reported in Figure 5b, confirms the significant performance difference between the base and *"-L"* models. The troughput of *OFAR* is limited to 0.36, closer to the theoretical limit of 0.5 imposed by global channels, than the 0,166 which would be imposed by the local ones without local misroute.

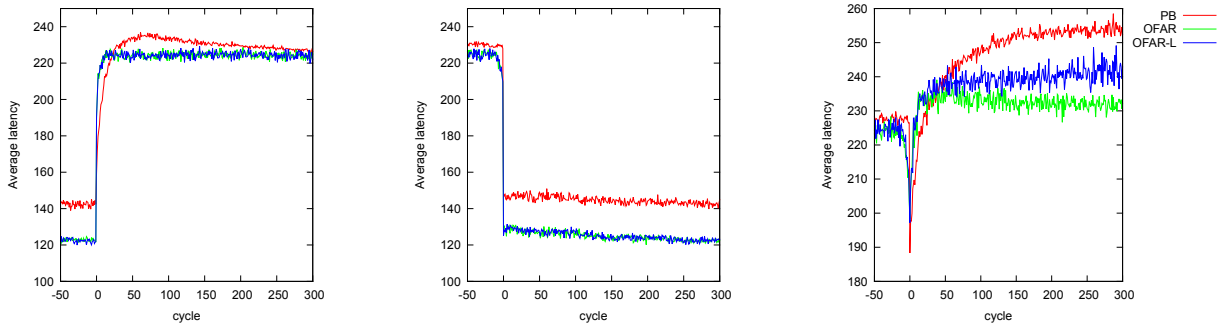
B. *Transient traffic*

These measures explore the response time when the traffic pattern changes. We warm-up the network with a given traffic pattern. Once it reaches the steady state, we change the traffic pattern, and observe how each mechanism adapts to the change. We measure the average latency of the packets that are *sent* each cycle. This is, when a packet is received, we account for the latency in the cycle that it was sent. We used



(a) Latency. (b) Throughput.

Fig. 5: Latency and throughput under adversarial +6 traffic (ADV+6).



(a) UN to ADV+2 traffic (b) ADV+2 to UN traffic (c) ADV+2 to ADV+6 traffic, load=0.12

Fig. 6: Latency evolution under transient traffic.

OFAR, *OFAR-L* and *PB*, in three different transient cases: UN to ADV+2; ADV+2 to UN and ADV+2 to ADV+6 (ADV+h). We apply a load of 0.14 phits/(node·cycle), except for the last case (ADV2 to ADV6) which would saturate the network using *PB*; we use 0.12 in that case. Figure 6 shows that for the transition of ADV+2 to UN , all the mechanisms converge very fast, since they suddenly find the required links un-congested. By contrast, in the other two cases *OFAR* makes the transition almost instantaneous, while *PB* suffers from an adaptation period.

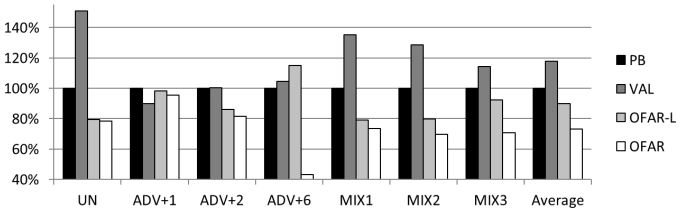


Fig. 7: Burst consumption time, normalized to *PB*. Lower is better.

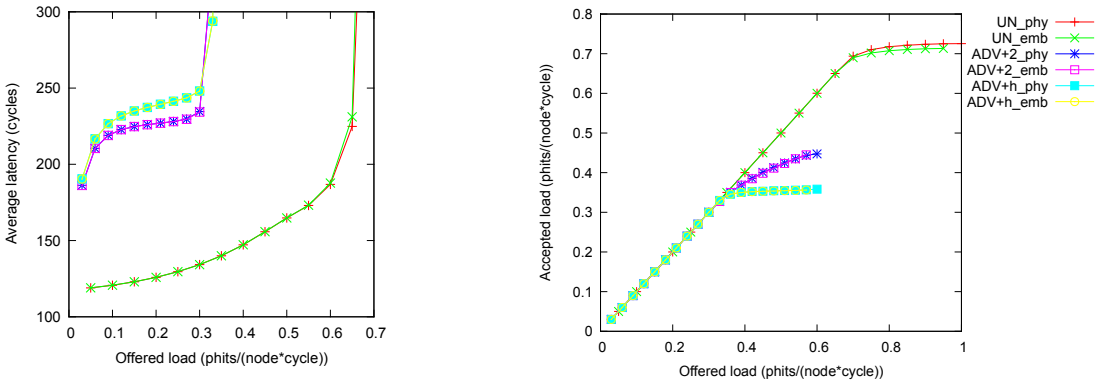
C. Traffic bursts

In parallel programs, communication and computation phases are typically synchronized, so traffic bursts after barriers are common. We simulate this 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*, *ADV+2*, *ADV+6* and three mixes of traffic with different rates of uniform and adversarial: In *MIX1* 80% of the traffic is UN, 10% is *ADV+1* and 10% is *ADV+6*. In *MIX2* the rates are 60-20-20 and in *MIX3* they are 20-40-40.

Figure 7 shows the execution time normalized to the result of *PB* on each case. The *OFAR* mechanisms always finish faster. 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 43.8%. It is noticeable that the complete *OFAR* model always finishes faster than their -L counterparts.

VII. Discussion

All the previous evaluations have been performed using a Hamiltonian physical ring and the same number of VCs as required by previous mechanisms. In this Section we will



(a) Latency (b) Throughput

Fig. 8: Latency and throughput for physical and embedded ring implementations.

address issues related to cost, tuning and reliability of *OFAR*.

The additional cost of an *OFAR* implementation based on a physical ring is easy to compute. A rough calculation shows that the proportion of added links is in the order of $2/3h$; with $h = 16$, this means 4% more wires. Nevertheless, the cost of these networks is mainly dominated by long wires. In *OFAR*, $2h^2 + 1$ are added to the $2h^4 + h^2$ original long wires. With $h = 16$, this accounts for only 0,3% more global wires. In addition, two router ports are needed to implement the ring.

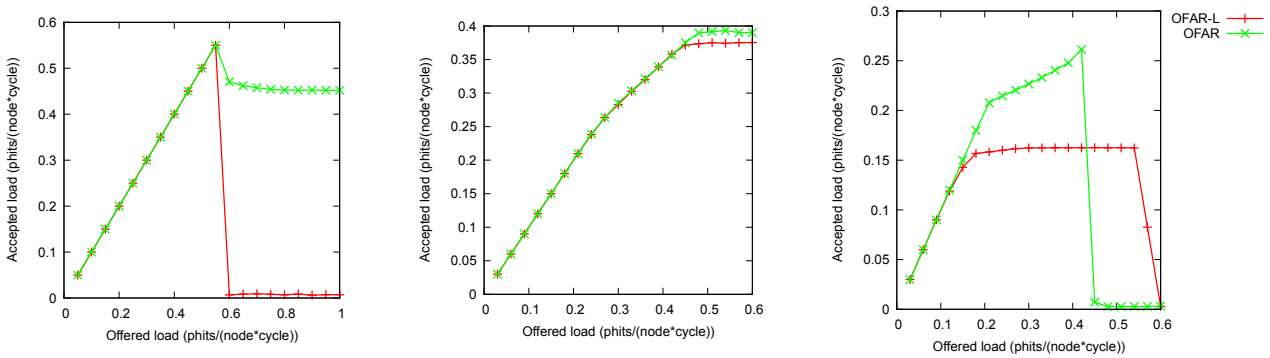
In spite of this low cost, cheaper solutions can be envisaged. For example, instead of adding a physical Hamiltonian ring, it can be virtually embedded on the original topology, connecting consecutive routers. This implementation has no added cost in terms of wires. We just use an additional virtual channel in the links that constitute the Hamiltonian embedded ring. Figure 8 compares the behavior of *OFAR* with an embedded ring and with a physical one. As it can be seen, no significant differences can be reported from the use of the physical or embedded link. This is coherent with the idea that the escape subnetwork is not used to route traffic, but only to resolve potential deadlock situations.

The previous results have shown that, even under high loads, throughput remains constant after saturation even with an embedded ring. However, in our model, the capacity of the escape network (the Hamiltonian ring) is much lower than the capacity of the canonical network (the Dragonfly). This might lead to network congestion if all the buffers of the canonical network were completely full, and only the escape ring was used to deliver packets at destination. Reaching such condition should be very uncommon as it must be provoked from the occurrence of multiple concurrent deadlocks which are not alleviated in time by the escape subnetwork. As discussed in [8], deadlock is very unfrequent when paths are short and there is a rich routing freedom, exactly the case of the Dragonfly. To verify if congestion could happen, we simulated *OFAR* with less resources: an embedded ring, and only 2 VCs for local links and 1 for global ones, without any congestion management. Figure 9 records the throughput observed under three different traffic patterns. We observe that, in some cases, throughput significantly falls as the canonical network gets completely congested. Note that the figure plots the average of several simulations, so in some points we are averaging simulations that suffered congestion with others that did not suffer it. A proper congestion control mechanism should be considered to guarantee that this case never happens in practice for a given configuration of network resources. Different congestion management alternatives for HPC can be found in [23]. Since this is typically implemented in a different level of the protocol hierarchy, its evaluation is left for future work.

With respect to reliability, *OFAR* could block the system with more than a single failure in its Hamiltonian ring. There are several ways to address this issue. One would be based on the use of several embedded Hamiltonian rings. Preliminary studies, not presented here, show that up to h edge-disjoint Hamiltonian rings could be embedded on this topology. This value is bounded by the number of local links in each group $(h \times (2h^2 - 1))$ and the number of local hops used by a Hamiltonian path on each group ($2h - 1$, since there are $2h$ routers). The system could maintain functionality as long as one of the Hamiltonians had less than two failures. Other solutions, such as an escape subnetwork with higher degree, or a hierarchical escape network, could be explored.

VIII. Conclusions

This paper has introduced *OFAR*, a flow-control/routing mechanism that addresses some of the main performance limitations of Dragonfly topologies, namely the saturation of local links and the poor efficiency of the misrouting decision process in existing mechanisms. We have presented an efficient alternative that allows for flexible on the fly misrouting of packets. *OFAR* dynamically selects the misroute port based on a dynamic misrouting threshold, rather than at injection time. This adaptive misrouting is enabled by employing a escape subnetwork to prevent deadlock, rather than a fixed order in the virtual channels.



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

Fig. 9: Congestion experienced with a reduced number of VCs: 2 VCs in local links an 1 VC in global links.

We have studied a misrouting policy for for *OFAR* which balances the traffic among the global links of the group, and obtains good values for latency, burst consumption and response time for transient traffic. Comparatively, the local misroute within a group has been proven as an efficient technique to avoid hot-spots and increase throughput.

Ongoing work includes the use of congestion avoidance mechanisms and the design of alternative router architectures and escape subnetworks. With respect to routers, we are dealing with links that totally avoid the use of virtual channels. As *OFAR* does not rely on VCs to avoid deadlock, input buffers with 2 or 3 read ports could provide a more scalable and efficient design. With respect to the escape subnetwork, embedding multi-Hamiltonian rings and exploring hierarchical restricted injection mechanisms are being considered.

Acknowledgements

This work has been supported by the Spanish Ministerio de Ciencia e Innovación, under project TIN2010-21291-C02-02, The HiPEAC Network of Excelence, the Consolider Project CSD2007-00050 "Supercomputación y e-Ciencia" and the Spanish Ministerio de Educación, grant AP2010-4900.

References

[1] J. Kim, W. Dally, B. Towles, and A. Gupta, "Microarchitecture of a high-radix router," in *ACM SIGARCH Computer Architecture News*, vol. 33, no. 2. IEEE Computer Society, 2005, pp. 420–431.

[2] J. Kim, W. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*. IEEE Computer Society, 2008, pp. 77–88.

[3] 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 *2010 18th IEEE Symposium on High Performance Interconnects*. IEEE, 2010, pp. 75–82.

[4] W. Dally, "GPU computing: To exascale and beyond. Invited talk." *Supercomputing, New Orleans.*, 2010.

[5] A. Bhatele, W. D. Gropp, N. Jain, and L. V. Kale, "Avoiding hot-spots on two-level direct networks," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, nov. 2011, pp. 1 –11.

[6] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," in *International Symposium on Computer Architecture*, 2009, pp. 220–231.

[7] L. Valiant, "A scheme for fast parallel communication," *SIAM journal on computing*, vol. 11, p. 350, 1982.

[8] T. Pinkston, "Deadlock characterization and resolution in interconnection networks," *Deadlock Resolution in Computer-Integrated Systems*, pp. 445–492, 2004.

[9] C. Carrion, R. Beivide, J. Gregorio, and F. Vallejo, "A flow control mechanism to avoid message deadlock in k-ary n-cube networks," in *International Conference on High-Performance Computing*, dec 1997, pp. 322 –329.

[10] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 4, no. 12, pp. 1320 –1331, dec 1993.

[11] A. Greenberg and B. Hajek, "Deflection routing in hypercube networks," *Communications, IEEE Transactions on*, vol. 40, no. 6, pp. 1070 –1081, jun 1992.

[12] S. Konstantinidou and L. Snyder, "The chaos router," *Computers, IEEE Transactions on*, vol. 43, no. 12, pp. 1386 –1397, dec 1994.

[13] W. J. Dally, "Virtual-channel flow control," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 3, no. 2, pp. 194 –205, mar 1992.

[14] K. Gunther, "Prevention of deadlocks in packet-switched data transport systems," *Communications, IEEE Transactions on*, vol. 29, no. 4, pp. 512 – 524, apr 1981.

[15] S. Brookes and A. Roscoe, "Deadlock analysis in networks of communicating processes," *Distributed Computing*, vol. 4, no. 4, pp. 209–230, 1991.

[16] I. Cidon and Y. Ofek, "Metaring-a full-duplex ring with fairness and spatial reuse," *Communications, IEEE Transactions on*, vol. 41, no. 1, pp. 110–120, 1993.

[17] V. Puente, C. Izu, R. Beivide, J. Gregorio, F. Vallejo, and J. Prellezo, "The adaptive bubble router," *Journal of Parallel and Distributed Computing*, vol. 61, no. 9, pp. 1180–1208, 2001.

[18] M. Blumrich, D. Chen, P. Coteus, A. Gara, M. Giampapa, P. Heidelberger, S. Singh, B. Steinmacher-Burow, T. Takken, and P. Vranas, "Design and analysis of the BlueGene/L torus interconnection network," *IBM Research Report RC23025 (W0312-022)*, vol. 3, 2003.

[19] D. Chen, N. Eisley, P. Heidelberger, R. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. Satterfield, B. Steinmacher-Burow, and J. Parker, "The IBM Blue Gene/Q interconnection network and message unit," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE, 2011, pp. 1–10.

[20] V. Puente, J. Gregorio, F. Vallejo, and R. Beivide, "Immunet: a cheap and robust fault-tolerant packet routing mechanism," in *International Symposium on Computer Architecture*, june 2004, pp. 198 – 209.

[21] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks (1976)*, vol. 3, no. 4, pp. 267–286, 1979.

[22] P. Gupta and N. McKeown, "Designing and implementing a fast crossbar scheduler," *Micro, IEEE*, vol. 19, no. 1, pp. 20–28, 1999.

[23] P. J. García, "Congestion management in HPC interconnection networks." *HPC Advisory Council European Workshop, Hamburg*, 2011.