OFAR-CM: Efficient Dragonfly Networks with Simple Congestion Management

Marina García*, Enrique Vallejo†, Ramón Beivide†, Mateo Valero‡ and Germán Rodríguez*
* IBM Research - Zurich, Switzerland. {mgg, rod}@zurich.ibm.com
† University of Cantabria, Spain. {enrique.vallejo, ramon.beivide}@unican.es
‡ Universitat Politècnica de Catalunya and BSC, Spain. mateo.valero@bsc.es

***Abstract*—Dragonfly networks are appealing topologies for large-scale Datacenter and HPC networks, that provide high throughput with low diameter and moderate cost. However, they are prone to congestion under certain frequent traffic patterns that saturate specific network links. Adaptive non-minimal routing can be used to avoid such congestion. That kind of routing employs longer paths to circumvent local or global congested links. However, if a distance-based deadlock avoidance mechanism is employed, more Virtual Channels (VCs) are required, what increases design complexity and cost.**

OFAR (On-the-Fly Adaptive Routing) is a previously proposed routing that decouples VCs from deadlock avoidance, making local and global misrouting affordable. However, the severity of congestion with OFAR is higher, as it relies on an escape subnetwork with low bisection bandwidth. Additionally, OFAR allows for unlimited misroutings on the escape subnetwork, leading to unbounded paths in the network and long latencies.

In this paper we propose and evaluate OFAR-CM, a variant of OFAR combined with a simple congestion management (CM) mechanism which only relies on local information, specifically the credit count of the output ports in the local router. With simple escape subnetworks such as a Hamiltonian ring or a tree, OFAR outperforms former proposals with distance-based deadlock avoidance. Additionally, although long paths are allowed in theory, in practice packets arrive at their destination in a small number of hops. Altogether, OFAR-CM constitutes the first practicable mechanism to the date that supports both local and global misrouting in Dragonfly networks.

***Index Terms*—Dragonfly Networks; Cogestion Management; Deadlock Avoidance;**

I. Introduction

Dragonfly networks [1] 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 electrical *local links*. 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 (1D-Flattened Butterfly) in both the local and global interconnects, while the Cray XC30 (codenamed "Cascade", [3]) employs an 1D-Flattened Butterfly (FB) for the global interconnect and a 2D-FB within groups.

OFAR (On-the-Fly Adaptive Routing, [4]) is an adaptive routing mechanism for Dragonfly networks that relies on a deadlock-free escape subnetwork, [5]. In OFAR, packets can freely circulate in the canonical network. This allows for several optimizations that improve performance: local misrouting (two local hops within a group instead of the direct one) circumvents congested local links; global misrouting (sending traffic to an intermediate group) avoids saturated global links; and in-transit changing from minimal to nonminimal routing (global misrouting) increases responsiveness to traffic changes.

However, this routing freedom permits the appearance of cyclic dependencies. When these occur, packets are derived to a deadlock-free escape subnetwork, what removes the deadlock situation in the canonical network. Packets can return to the canonical network from the escape subnetwork to continue on a minimal path, or follow the escape subnetwork until they reach their destination. As packets can return to the canonical network, the system is restricted to Virtual Cut-Through switching, as discussed in [5]. The implementation in [4] employs a Hamiltonian ring with bubble flow control [6] as its deadlock-free escape subnetwork. This approach is very effective, but it introduces two main limitations. Firstly, under high traffic load, if the canonical network gets saturated, the network could respond with the throughput of the escape subnetwork (the ring), which is much lower. This is, congestion in the escape network leads to congestion in the canonical network which cannot be mitigated with the misrouting mechanisms. Secondly, under high congestion, packets can bounce from the canonical to the escape subnetwork, leading to unbounded network paths.

In this paper we introduce OFAR-CM, an overall solution which combines OFAR routing with a simple congestion management (CM) mechanism to prevent its original performance problems. Our experiments show that even a very simple CM mechanism such as injection throttling is enough to prevent any congestion issues. Furthermore, OFAR-CM obtains better performance than alternative mechanisms with higher cost in terms of number of virtual channels and design complexity. Specifically, the main contributions of this paper are:

- We introduce OFAR-CM, an overall solution for routing in Dragonflies that relies on a simple injection throttling mechanism. Using the same resources as previous proposals, OFAR-CM provides the maximum routing freedom without the congestion problems of the original OFAR.
- We evaluate two simple congestion control mechanisms, Base Congestion Management (BCM) and Escape Con-

gestion Management (ECM), that rely on local information. Overall, both are effective in preventing network congestion. However, ECM restricts injection more, leading to higher throughput but also higher latencies.

- We evaluate two low-cost options for the escape subnetwork: a Hamiltonian ring with bubble flow control, and a spanning-tree with up-down routing. Comparatively, although the tree is more likely to cause load imbalance making traffic consumption slower, it provides better latencies at low traffic loads.
- We evaluate the problems of unfairness and unbounded paths. We identify unfairness issues that can arise at the group or router levels under high loads. However, in our system packets do not follow unbounded paths.

The rest of the paper is organized as follows. Section II introduces related work in the field. Sections III and IV detail the escape subnetworks and the congestion management mechanisms evaluated in this paper. Then, Sections V and VI present and discuss the simulation results of these mechanisms, and Section VII concludes the paper.

II. Related Work

Several adaptive routing mechanisms have been proposed for Dragonfly networks [1], [7], [4]. They differ in multiple aspects, mainly whether they support certain nonminimal routing options. Global misrouting (or Valiant routing, [8]) sends traffic to an intermediate group to avoid congested global links. Local misrouting (motivated in [4]) avoids congested local links with two local hops. In-transit adaptive routing (supported in PAR, [7]) can switch a packet's path from minimal to non-minimal on the fly, adapting faster to changes in network congestion.

Regarding congestion sensing, Piggybacking (PB, [7]) floods congestion information between the routers in a group. To do so, it employs additional bits in the data packets sent between routers in the group, what reduces overhead. Other mechanisms can modify the packet path in-transit (OFAR or PAR), using information local to each router of the path.

Deadlock avoidance in multiple proposals (such as PB or PAR) relies on an increasing ordered sequence of virtual channels, based on an original result in [9]. This implies that allowing longer paths in the network (for example, to support local misrouting or in-transit adaptive routing) requires a larger number of VCs and increases the design complexity of the router. OFAR [4] employs VCs from the "original" (or canonical) Dragonfly in a fully adaptive manner, but adds a deadlock-free escape subnetwork to prevent deadlock situations. A Hamiltonian ring with bubble flow control is employed in that work as the escape subnetwork.

Multiple congestion control mechanisms have been studied and proposed for different networks. A good survey of their application in HPC can be found in [10]. Virtually every mechanism relies on injection throttling, such as the transmission window in TCP [11] or Quantized Congestion Notification (QCN, [12]) in Datacenter Bridging. The differences rely on how they detect network congestion. Several proposals rely on explicit congestion notification (ECN), such as QCN, Datacenter TCP [13] or recent Infiniband implementations [14]. In such cases, the network equipment detects congestion based on the occupancy of router queues. Alternative mechanisms derive congestion from other indicators, such as estimated round-trip time or packet loss.

Alternatively, equal-cost multipathing was proposed in [15] as a mechanism for datacenter bridges to circumvent congested network areas without reducing transmission rates. A similar approach is the one employed in adaptive routing mechanisms in Dragonflies [1], but in this case selecting between paths with different lengths or costs.

When the packets in a flow can follow different paths (to avoid congested links), additional end-to-end congestion control is not very effective since not all packets traverse the same congestion point. In this work we study local congestion control mechanisms, BCM and ECM, which apply source throttling based on the occupancy of the local queues. Similar mechanisms have been studied before in other networks, [16].

III. Escape Subnetworks

When a escape deadlock-free subnetwork is employed for deadlock avoidance, no virtual channels (VCs) are required to prevent deadlock [5], although they help mitigating Head-of-line blocking (HoLB). We will denote the number of VCs employed in local and global ports separately. For example, Piggybacking requires 3/2 VCs, meaning 3 VCs in local ports and 2 VCs in global ones.

The escape subnetwork can employ links separated to the ones in the canonical network, using extra ports in the routers and interconnecting them with additional local and global links. Another possibility is to embed the escape subnetwork. In that case it would be necessary to add at least one virtual channel to each link forming the escape subnetwork. We will denote this extra VC separately, for example 3/2(+1).

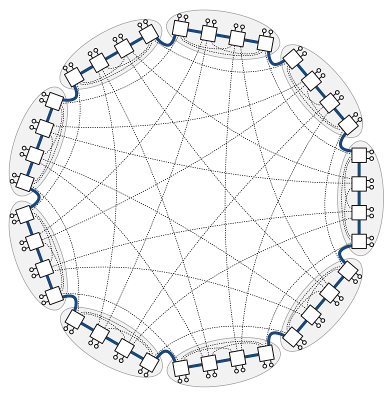
The escape subnetwork must interconnect all the routers in the network. Depending on the topology employed, the network cost and the performance results will vary. In this section we describe two such topologies, first a subnetwork based in a Hamiltonian ring like the one employed in [4] and then one based on a tree. They are depicted in Figure 1.

A. Hamiltonian Ring

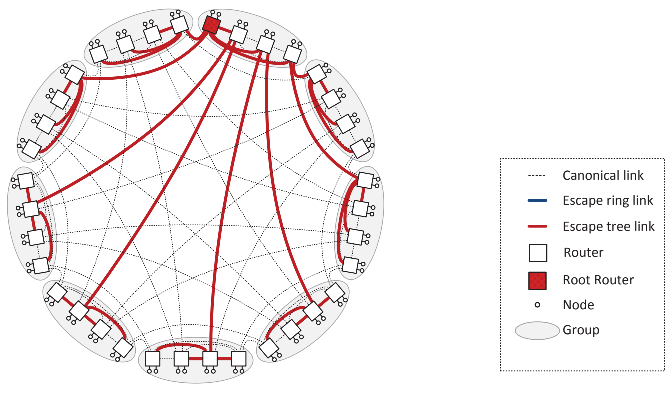
In this case, the escape subnetwork interconnects every router in the network forming a ring. With Virtual Cut-Through, the ring subnetwork is deadlock-free as long as there is space for at least one packet in one of its buffers. To assure this, bubble flow control is applied to the ring. To inject a packet in the ring, it is required to have free space for two packets in the buffer, what preserves the previous condition. On the contrary, packets can freely circulate inside the ring.

B. 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 by additional links or VCs to all the remaining routers

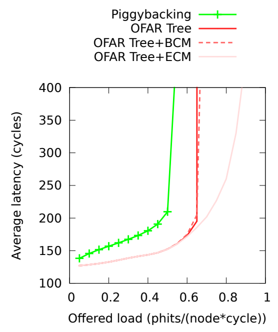


(a) Hamiltonian Ring Subnetwork

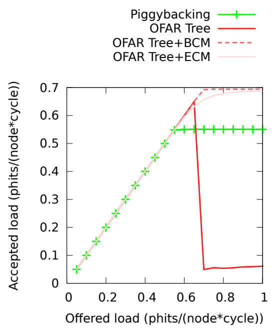


(b) Tree Subnetwork

Fig. 1: Escape subnetwork topologies for a small Dragonfly interconnection network.



(a) Latency UN.



(b) Throughput UN.

Fig. 2: Latency and throughput under uniform traffic for OFAR with a tree escape subnetwork and different CM.

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. The routing employed to advance in the tree network is up-down, what makes it deadlock-free.

IV. Congestion Management

The capacity (or bisection bandwidth) of the proposed escape subnetworks is much lower than the capacity of the canonical Dragonfly network. If all the buffers of the canonical network got full and only the escape subnetwork was used to deliver packets to their destination, the performance would drop significantly, [4]. Such situation is unlikely, but still proper congestion management (CM) mechanisms have to be applied to the network to guarantee that this never happens.

Figure 2 shows the throughput and latency results obtained for OFAR under uniform random traffic with a tree as escape subnetwork. The results of PB are shown as a reference. With OFAR, if no congestion management mechanism is applied (OFAR Tree), the throughput drops drastically when the canonical network gets congested under high load. However, a simple congestion management mechanism (BCM or ECM) prevents this throughput fall. We introduce next these two different simple CM mechanisms based on injection throttling. One of them takes into account the state of the escape subnetwork, while the other takes into account the state of the canonical network.

A. *Escape Congestion Management (ECM)*

With OFAR, the network congestion is reflected in the use of the escape subnetwork to prevent deadlock. The Escape Congestion Management (ECM) employs the occupancy of the local buffers of the escape subnetwork as an indicator of congestion. If the occupancy of all those buffers is higher than a given threshold, no packets will be injected. In such case, the local nodes will have to wait to a subsequent cycle to inject their traffic. The threshold used is chosen empirically, ranging from 0% to 100%. Regardless the threshold, buffers can still be used for in-transit traffic. Contrary to the following mechanism BCM, ECM does not take into account how occupied is the queue in which the packet should be injected.

B. *Base Congestion Management (BCM)*

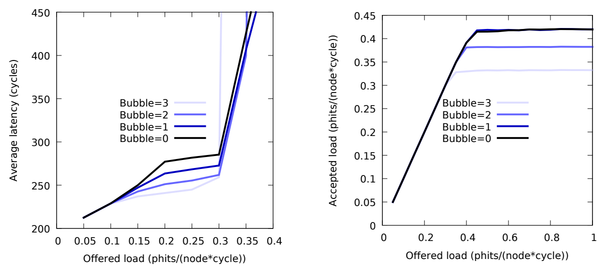
The Base Congestion Management (BCM) mechanism forbids the injection of packets when the canonical (base) network is congested. This is implemented as an extension of the bubble flow control mechanism. A certain “bubble” is required to inject packets in the next buffer, what prevents traffic injection from introducing deadlock in the canonical network. Consequently, a packet at the head of an injection queue can be injected in the network only if there is enough space in the next queue for one packet plus the bubble. Otherwise, the packet will have to wait to a subsequent cycle. The bubble size can range from 1 to the buffer size in packets minus 1, and it is chosen empirically to prevent over-throttling, as will be detailed in Section V-A. Packets are never injected directly into the escape subnetwork, so its occupancy is not relevant for the injection decision. Hence, the BCM mechanism only takes into account the state of the canonical network.

V. Performance Results

In this section we study the behavior of OFAR with different escape subnetworks and congestion management mechanisms in a Dragonfly network, using complete graphs for the local and global interconnects. We have employed an in-house developed Dragonfly network time-driven simulator. We simulate a Dragonfly with size: $p = 6$ computing nodes and $h = 6$ global ports per router, and $a = 12$ routers per group. This network interconnects 5,256 computing nodes organized in 73 groups of 12 routers with 23 ports each. Latencies are 10 cycles for local links and 100 for global links. We model input buffered routers with FIFO queues. Taking into account the round-trip latencies, FIFO sizes are set to 32 phits for the local ones, and 256 phits for the global ones. Packet length is 8 phits. We do not model virtual output queuing (VoQ) [17] or router speedup, as they highly increase the complexity and cost of high-radix routers. We employ the MM global misrouting policy, as introduced in [18] and a misrouting threshold of 90% as presented in [4].

We evaluate OFAR with the two different subnetwork topologies, *Ring* and *Tree*, and the two congestion management mechanisms, BCM and ECM: *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 those links forming the escape subnetwork. We show the results for Piggybacking (PB) as a reference, as it is the routing based on VCs for deadlock avoidance with the best overall results in previous works, [7]. It always employs 3/2 VCs, which are required for deadlock avoidance. We carry out steady state and traffic consumption experiments. For the steady state experiments, we simulate uniform random (UN) and adversarial global (ADVG) traffic patterns. With UN traffic destination nodes randomly selected among all the possible destinations, excluding the source node itself. With the adversarial pattern ADVG+i all traffic from a group N is sent to group $N + i$. We show 2 different cases: ADVG+2 and ADVG+6. ADVG+2 requires global misrouting to obtain good performance. However, as explained in [4], due to a pathological problem of congestion in certain local links, the $ADVG + i$ traffic is more or less adversarial depending on the value of i; $ADVG + n \cdot h$ (in our case ADVG+6) generates the maximum congestion in local links, requiring of local misrouting to sustain throughput. We show results for ADVG+2 instead of ADVG+1 because the additional ring VC between consecutive groups could favor the OFAR+Ring model. The results show the average latency and throughput obtained after a network warm-up period.

For the traffic consumption experiments, each node sends 2,000 packets of a specific traffic pattern as fast as possible and we show the number of cycles required until all packets are consumed. In addition to UN, ADVG+2 and ADVG+6, we simulate one phase of an all-to-all communication. In that case each node sends one packet to each of the other nodes in the network (5,255 packets per node), randomizing the source and destination pairs to alleviate hotspots like in [19].



(a) Latency (b) Throughput

Fig. 3: Latency and throughput under ADVG+2 traffic for different BCM bubble sizes.

A. Congestion Management Parameter Selection

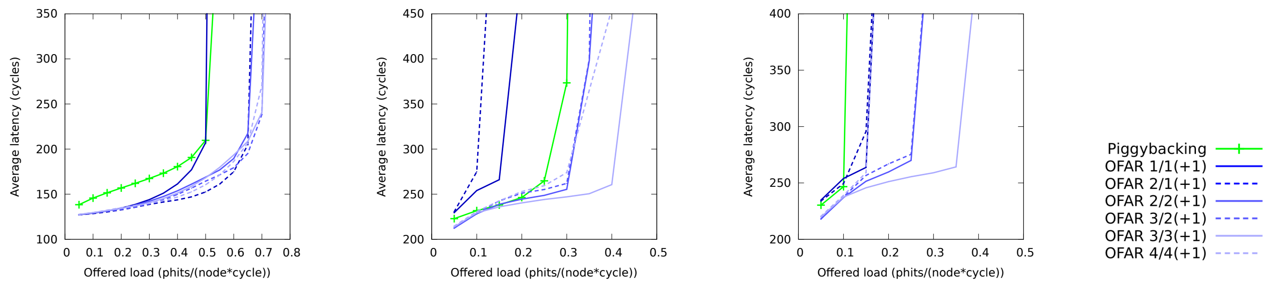
As previously explained in Section IV, the size of the bubble for BCM and the threshold for ECM have to be chosen empirically. We show next an example of how the size of the bubble affects performance results. Figure 3 shows the latency and throughput results obtained for *OFAR+Ring* with 3/2(+1) VCs, BCM, and different bubble sizes. Although a smaller bubble provides a higher throughput, it also generates higher latencies. Consequently, we choose a bubble of 2 packets for our BCM implementation, what provides a high throughput while maintaining low latencies. A similar methodology is used to choose the threshold for ECM.

B. Network Resources

In this section we study how the number of virtual channels (VCs) affects performance. With OFAR, no VCs are needed for deadlock avoidance, but they help mitigating Head-of-Line Blocking (HoLB). As a result, a higher number of VCs will provide higher performance, until a point in which there is another factor that limits performance more than HoLB, such as the switch allocation mechanism. Passed that point, a higher number of VCs degrades performance, since there are more packets in the network what can increase congestion.

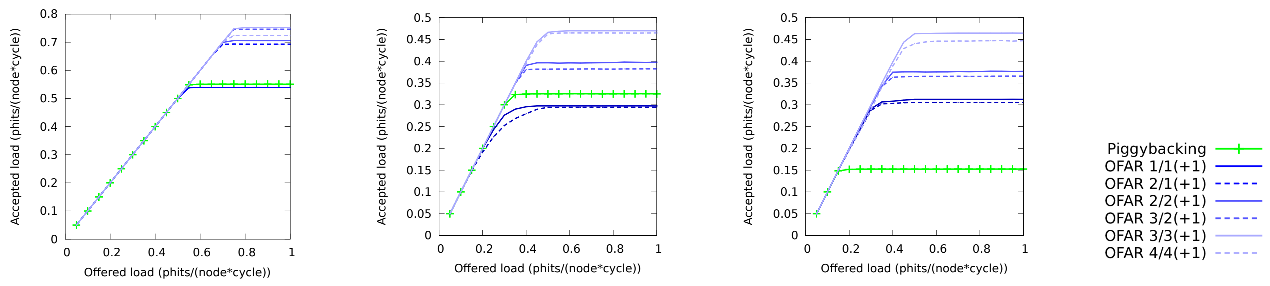
Figures 4 and 5 show the average latency and throughput results obtained for *OFAR Ring+BCM* with different number of VCs. The bubble size for our BCM implementation is 2 packets. The number of VCs ranges from 1/1(+1) to 4/4(+1). PB is also shown as a reference.

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 less VCs, 2/1(+1) or 1/1(+1), and with traffic ADVG+2, PB results are better than OFAR. When the traffic is uniform, Figures 4a and 5a, OFAR 1/1(+1) obtains a result very close to that for PB. All the rest of the 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 increasement of the number of VCs does not provide a better performance: OFAR 4/4(+1) obtains worse results, specially for the latency when the traffic is adversarial.



(a) UN. (b) ADVG+2. (c) ADVG+h.

Fig. 4: Latency under UN, ADVG+2 and ADVG+h traffic for PB and OFAR varying the number of VCs.



(a) UN. (b) ADVG+2. (c) ADVG+h.

Fig. 5: Throughput under UN, ADVG+2 and ADVG+h traffic for PB and OFAR varying the number of VCs.

From the figures we can observe a clear problem of network unfairness when the VC count is low. With 2/1(+1) VCs and adversarial traffic, the average latency of OFAR rockets at around 0.15 $phit/(node \cdot cycle)$. However, its throughput reaches 0.3 $phit/(node \cdot cycle)$. This effect is typical with unfairness issues, when some specific nodes of the network suffer from starvation: their latency is much higher than the rest, what increases the average latency values. Similar starvation problems in Dragonfly networks were studied in [18]. In this case, the problem arises from localized congestion, as will be studied in more detail in Subsection V-C.

We have seen that 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 congestion problems derived from HoLB. Thus, from now on we will only focus on this specific configuration with few resources, to study the effects of congestion and alternative ways to cope with it.

C. Congestion Management and Escape Subnetwork

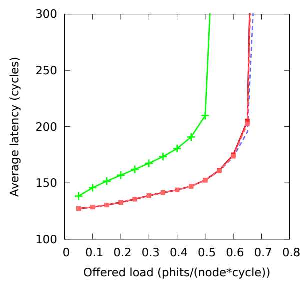
This section explores how the escape subnetwork and congestion management mechanisms affect the performance of OFAR with few resources, 2/1(+1) VCs. We study the performance of *OFAR Ring* and *OFAR Tree* with BCM and ECM congestion management. The latency and throughput steady state results are show in Figures 6 and 7. Again, PB is shown as a reference. As seen in Figures 6a and 7a, under uniform traffic no significant differences can be appreciated in the performance of the four OFAR configurations, and all of them outperform PB. Under adversarial traffic, Figures 7b and 7c, the variants with ECM obtain higher throughput than those with BCM. Under adversarial traffic $ADVG+2$, the maximum throughput of OFAR with BCM is slightly lower than with PB, while with ECM it is higher.

These figures also reflect some anomalies caused by load imbalance. Figures 6b and 6c show the same latency results for OFAR Ring+BCM with 2/1(+1) as Figures 4b and 4c, but with a larger y axis. Above load 0.1 the latencies keep 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 in Section VI.

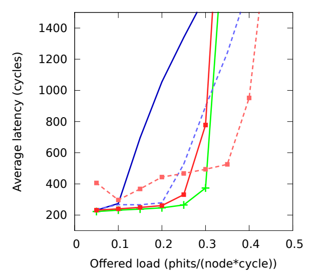
VI. Discussion

Figure 8 shows the traffic consumption times for each routing mechanism. OFAR is always faster than PB consuming traffic except when the traffic is $ADVG+2$. In general, ECM yields faster results than BCM, and the ring yields faster results than the tree. As a result, the fastest OFAR configuration is *OFAR Ring+ECM*, while the slowest is *OFAR Tree+BCM*, which needs almost the same time as PB to consume traffic.

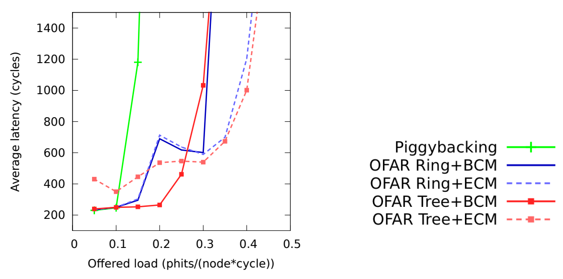
The same experiments shown in this Subsection were



(a) UN.

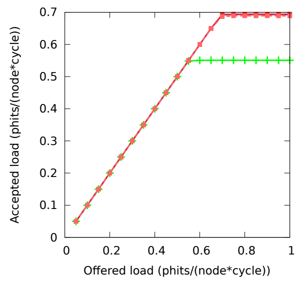


(b) ADVG+2.

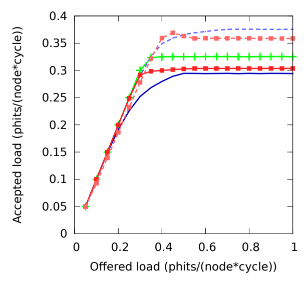


(c) ADVG+h.

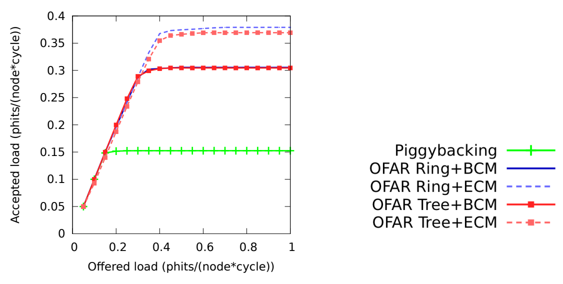
Fig. 6: Latency under UN, ADVG+2 and ADVG+h for PB and OFAR with 2/1(+1) VCs.



(a) UN.



(b) ADVG+2.



(c) ADVG+h.

Fig. 7: Throughput under UN, ADVG+2 and ADVG+h for PB and OFAR with 2/1(+1) VCs.

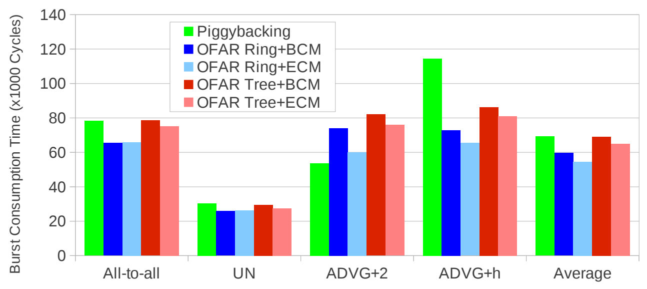


Fig. 8: Traffic consumption time for PB and OFAR 2/1(+1).

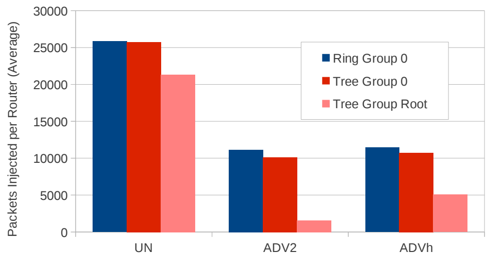
carried out for OFAR with 3/2(+1) VCs. In that case, the four OFAR configurations obtained better performance than PB in the steady state experiments, and consumed traffic in approximately 0.7 times the time of PB.

A. Network Fairness

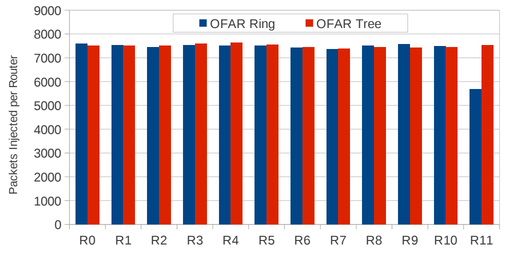
In the experiments in Figures 6 and 7, OFAR obtains very similar throughput results with the ring and the tree escape subnetworks, regardless the congestion management. However, as shown in Figure 8, *OFAR ring* consumes ADVG traffic faster than *OFAR tree*. This behavior is due to a load imbalance introduced by the asymmetry of the tree escape subnetwork. Figure 9a, obtained with BCM, shows this effect. It depicts the total number of packets injected 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 $phit/(node \cdot cycle)$. We present results for UN, ADVG+2 and ADVG+h. 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. Since there is no root in a ring, for *OFAR Ring* we just show results for G_0.

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. A small part of the network, G_{root}, concentrates great part of the 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. As a result, 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 Figures 7b and 7c especially with ECM, and the higher consumption times.

This asymmetry is also responsible for the high latencies at low traffic loads for *OFAR Tree+ECM* when the traffic is adversarial (Figures 6b and 6c). With that configuration, OFAR only injects packets if the escape subnetwork is not saturated. Although at low traffic loads the escape subnetwork should not



(a) Average number of packets injected per router in groups 0 and *Root* with and offered load of 1 $phit/(node \cdot cycle)$.



(b) Packets injected per router in group 0 with ADVG+h traffic and an offered load of 0.2 $phit/(node \cdot cycle)$.

Fig. 9: Packets injected with *OFAR Ring+BCM* in different groups of a Dragonfly network.

be saturated, in G_{root} it is, 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 disjoint escape trees, as previously suggested in [4], in order to distribute the traffic generated by the escape subnetwork. We do not explore this idea in this paper.

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 7b and 7c. Specifically, under an adversarial pattern all the traffic of a group will 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. Figure 9b shows an example with ADVG+h traffic, showing the number of packets injected by each router in group 0 for an offered traffic load of 0.2 $phit/(node \cdot cycle)$ with *OFAR Ring+BCM* and *OFAR Tree+BCM*. With *OFAR Ring+BCM*, R_{11}, through which minimal and escape traffic leave the group, injects 25% less packets than the rest of the routers in the group. On the contrary, with *OFAR Tree+BCM*, no router starves. The effect of this unbalance is reflected in Figure 7c, in which the average latency at 0.2 $phit/(node \cdot cycle)$ for *OFAR Ring+BCM* is much higher than for *OFAR Tree+BCM*.

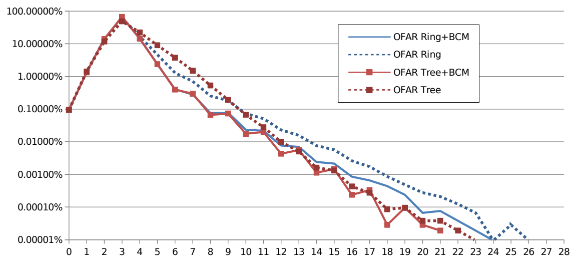
B. Length of network paths

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 routers in the network), than if it was a tree (up to 6 hops). However, the escape subnetwork is only used to escape from potential cyclic dependencies, and packets try to 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. We study next this concern when congestion management is used, and observe that, in practice, unbounded paths do not happen.

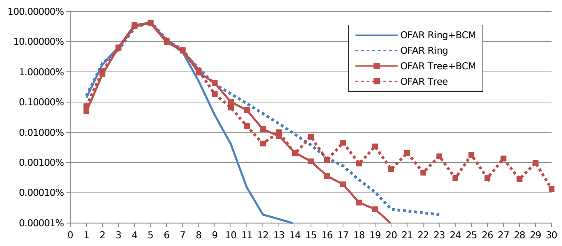
Histograms in Figures 10a and 10b show how many packets needed a certain number of hops to reach their destination nodes under UN and ADVG+h traffic respectively, with or without BCM, and with log scale in the vertical axis. Data was collected in traffic consumption experiments like those in Subsection V-C. Under UN the four OFAR configurations show a similar behavior. The maximum route lengths are slightly shorter for *OFAR Tree* than for *OFAR ring*. In all the cases, more than a 99.9% of the packets need less than 11 hops to reach the destination node.

Interestingly, under ADVG+h traffic, *OFAR Ring* provides shorter paths than *OFAR Tree*. This points out that a escape subnetwork with longer average and maximum distances does not necessarily lead to longer paths. Figure 10b shows that long paths are really infrequent when congestion management is used. In our simulations 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, while for *OFAR Ring* the longest path takes 24 hops with *OFAR Tree* one of the packets had to make 217 hops, with multiple injections to the escape subnetwork. When a packet is in the source group and feels that the canonical network is congested, it goes into the escape subnetwork. If there is congestion in the next group following the escape subnetwork, 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; until it senses congestion again and has to be injected once more in the escape subnetwork. This process will be repeated as long as congestion situation remains. As explained in Subsection VI-A, with *OFAR Tree*, group G_{root} is more prone to congestion than the rest of the groups. As a result, multiple injections are more likely. Nevertheless, this situation is not very common. For *OFAR Tree* under ADVG+h traffic, more than a 99.99% of the packets need less than 30 hops to reach its destination.

Although in practice unbounded paths do not occur when using congestion management, a simple mechanism could limit the number of subnetwork injections and bound path lengths. Every packet would need a counter, incremented



(a) UN.



(b) ADVG+h.

Fig. 10: Hops histogram under uniform (UN) and adversarial traffic (ADVG+h).

each time it is injected in the escape subnetwork. Once that counter saturates (for example, 15 injections for a 4-bit counter), the packet is forced to continue through the escape subnetwork until reaching its destination. In our experiments with congestion management, no packet was injected to the escape subnetwork more than 12 times, so this mechanism would not have a significant impact on performance.

VII. Conclusions

In this paper we have introduced and evaluated a novel alternative that solves the main limitations of former routing mechanisms in Dragonfly networks; routing freedom, resource cost, congestion problems and unbounded path lengths. Our mechanism combines OFAR with simple injection throttling. Compared to alternative proposals, our mechanism only relies on local information, supports local and global misrouting without increasing the number of VCs, and achieves higher performance thanks to the higher routing freedom.

With similar cost (in terms of VCs), our proposal clearly outperforms alternatives such as PB. Implementations with lower cost might suffer unfairness issues. In such case, we have evaluated two congestion management mechanisms, BCM and ECM, and two escape subnetwork topologies, a Hamiltonian ring and a tree. The congestion management mechanisms avoid network saturation that could lead to a performance drop. We have analyzed how the topology of the escape subnetwork affects network load imbalance and performance. Finally, this work shows that, despite path lengths with OFAR are unbounded in theory, they are relatively short in practice.

Acknowledgment

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. ERC-2012-Adg-321253-RoMoL, the Spanish Ministry of Science under contracts TIN2010-21291-C02-02, TIN2012-34557, and by the European HiPEAC Network of Excellence. M. García participated in this work while affiliated with the University of Cantabria.

References

[1] J. Kim, W. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *35th International Symposium on Computer Architecture (ISCA'08)*, 2008, pp. 77–88.

[2] 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.

[3] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard, "Cray Cascade: a scalable HPC system based on a Dragonfly network," in *Intl Conf on High Performance Computing, Networking, Storage and Analysis*, 2012.

[4] M. García, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, G. Rodríguez, J. Labarta, and C. Minkenberg, "On-the-fly adaptive routing in high-radix hierarchical networks," in *The 41st International Conference on Parallel Processing (ICPP)*, 09 2012.

[5] J. Duato and T. M. Pinkston, "A general theory for deadlock-free adaptive routing using a mixed set of resources," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 12, pp. 1219–1235, Dec. 2001.

[6] C. Carrion, R. Beivide, J. Gregorio, and F. Vallejo, "A flow control mechanism to avoid message deadlock in k-ary n-cube networks," in *4th Intl. Conf. on High-Performance Computing*, 1997, pp. 322 –329.

[7] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," in *36th Intl. Symposium on Computer Architecture (ISCA '09)*, 2009, pp. 220–231.

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

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

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

[11] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, 1988, pp. 314–329.

[12] *"IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks - Amendment: 10: Congestion Notification," 802.1Qau*, IEEE Std., April 2010.

[13] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *ACM SIGCOMM Conference 2010*, pp. 63–74.

[14] E. Gran, M. Eimot, S.-A. Reinemo, T. Skeie, O. Lysne, L. Huse, and G. Shainer, "First experiences with congestion control in InfiniBand hardware," in *IEEE Intl. Symp. on Parallel Distributed Processing*, 2010.

[15] C. Minkenberg, M. Gusat, and G. Rodriguez, "Adaptive routing in data center bridges," *Symposium on High-Performance Interconnects*, 2009.

[16] S. Lam and M. Reiser, "Congestion control of store-and-forward networks by input buffer limits–an analysis," *Communications, IEEE Transactions on*, vol. 27, no. 1, pp. 127–134, 1979.

[17] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *Communications, IEEE Transactions on*, vol. 35, no. 12, pp. 1347–1356, 1987.

[18] M. García, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, G. Rodríguez, and J. Labarta, "Global misrouting policies in two-level hierarchical networks," in *7th Intl Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip (INA-OCMC)*, 2013.

[19] R. Rajamony, M. Stephenson, and W. Speight, "The Power 775 architecture at scale," in *27th Intl. Conference on Supercomputing*, 2013.