Contention-based Nonminimal Adaptive Routing in High-radix Networks

Pablo Fuentes, Enrique Vallejo, Marina García, Ramón Beivide
University of Cantabria, Spain
{pablo.fuentes, enrique.vallejo, marina.garcia, ramon.beivide}@unican.es

Germán Rodríguez, Cyriel Minkenberg
IBM Zurich Research Laboratory
Switzerland,
{rod,sil}@zurich.ibm.com

Mateo Valero
Barcelona Supercomputing Center,
Spain,
mateo@bsc.es

***Abstract*—Adaptive routing is an efficient congestion avoidance mechanism for modern Datacenter and HPC networks. Congestion detection traditionally relies on the occupancy of the router queues. However, this approach can hinder performance due to coarse-grain measurements with small buffers, and potential routing oscillations with large buffers.**

We introduce an alternative mechanism, labelled Contention-Based Adaptive Routing. Our mechanism adapts routing based on an estimation of "network contention", the simultaneity of traffic flows contending for a network port. Our system employs a set of counters which track the demand for each output port. This exploits path diversity thanks to earlier detection of adversarial traffic patterns, and decouples buffer size and queue occupancy from contention detection.

We evaluate our mechanism in a Dragonfly network. Our evaluations show this mechanism achieves optimal latency under uniform traffic and similar to best previous routing mechanisms under adversarial patterns, with immediate adaptation to traffic pattern changes.

I. Introduction

High-radix routers [1] can be exploited in HPC and Datacenter networks. Such systems typically employ interconnection topologies with large path diversity to increase both the available bandwidth between pairs of routers and fault tolerance. Some examples are the folded Clos, the concentrated torus, the Flattened Butterfly [2], [3], or the Dragonfly [4] (used in Cray Cascade [5] and IBM Power 775 [6]).

By selecting one of the different paths to a given destination, adaptive routing exploits the available path diversity and avoids congested areas of the network. *Minimal* adaptive routing selects one of the different minimal paths with the same cost to the destination, which can be exploited to avoid congestion and increase performance. Meshes, torus or folded-Clos networks often exploit minimal routing, [7], [8], [9], [10]. By contrast, *nonminimal* adaptive routing selects between one or more minimal paths and one or more longer nonminimal paths. The selection of a nonminimal path makes sense to increase bandwidth between endpoints and, especially, to avoid hotspots in the minimal path. Flattened-butterflies or Dragonflies are networks that require nonminimal adaptive routing, due to the low path diversity and congestion issues when using minimal paths.

An adequate selection between one path or another is instrumental in obtaining the maximum network performance. Under minimal adaptive routing such selection can simply rely on the availability of output ports. By contrast, the selection of a nonminimal path is a critical decision because it implies a longer path for the traffic and a higher use of the network resources. We denote as *misrouting trigger* the mechanism employed to select between one preferred, minimal path, and another one (typically, nonminimal) in adaptive routing. The misrouting trigger employed in previous works has been based on an estimation of the network congestion, derived from the occupancy of the router buffers. Different variants of such mechanisms are used or have been proposed in Cray Cascade[5], UGAL [11], OFAR [12] and many other works.

Despite their wide adoption, congestion estimations based on buffer occupancy have fundamental shortcomings which limit their effectiveness: dependency on the buffer size, uncertainty, slow response and traffic oscillations. Section II will analyze these shortcomings in detail. In general, it is interesting to observe that when adaptive routing is used to prevent performance losses, congestion detection is not the *reason* that should trigger an alternative path selection, but rather the *consequence* of previous suboptimal decisions.

This paper introduces an alternative mechanism to handle routing adaptivity in interconnection networks. Rather than relying on congestion indicators such as buffer occupancy, this paper explores the use of a *network contention* metric to trigger adaptive routing. Network contention has been explored before in different contexts, such as minimal adaptive routing in NoCs [13], [14] or wireless networks [15], but never in HPC and Datacenter networks with nonminimal routing. Specifically, we introduce the idea of *contention counters*, a simple mechanism to estimate the contention of each output port. This permits an early detection of adverse network situations before they show up as fully populated buffers and performance degradation.

Three variants of the general idea have been applied to Dragonfly networks. A Dragonfly is composed of groups of high-radix routers. The few inter- and intra- group links can easily saturate under adverse traffic. A routing mechanism based on contention counters can divert traffic from contended ports to alternative nonminimal ports with less contention, relying only on local information in each router. Routers quickly adapt to traffic changes, regardless of their buffer size,

and they are not prone to routing oscillations.

Specifically, the main contributions of this paper are:

- We identify the shortcomings of using credits to trigger misrouting: when buffers are small, the uncertainty and granularity of the credit values do not allow for a proper decision; when they are large, the routing is slow in adapting to transient situations and prone to oscillations.
- We introduce a novel misrouting trigger, *contention counters*, which relies on a measure of port contention rather than the buffer occupancy, effectively decoupling the size of the buffers from the routing decisions.
- We propose three different adaptive routing implementations based on the idea of contention counters, two of them relying on local information and one specifically for Dragonfly networks which implements *Explicit Contention Notification, ECtN*.
- We evaluate the proposals by simulation in the context of Dragonfly networks. Results show that the use of contention counters provides a very fast response to traffic changes and allows for small buffers that would otherwise impede taking proper adaptive routing decisions.

The remainder of this paper is organized as follows. Section II analyzes the main shortcomings of congestion-based adaptive routing in HPC and Datacenter networks. Section III introduces the general idea of contention-based adaptive routing, and three detailed implementations based on *contention counters*. Section IV details the simulation infrastructure, including a review of the Dragonfly topology, and Section V presents the simulation results. Finally, Section VI presents a discussion about the results, Section VII introduces the related work in the field and Section VIII concludes the paper.

II. Limitations of congestion detection and misrouting trigger based on buffer occupancy

Traditional congestion detection mechanisms rely on the occupancy of a neighbor input buffer, or the credits remaining in the corresponding local output port, to detect congestion and eventually trigger misrouting. In this Section we analyze the limitations of such approach.

A. Granularity of the congestion detection

The size of the router buffers and the packet or flit size, along with the credit management mechanism determine the granularity at which the queue occupancy level can be measured. With wormhole switching the packet size is a multiple of the flit size, with a minimum resolution of one flit. As an example, the PERCS interconnect [6] employs 128-byte flits, which limits the minimum resolution. Virtual Cut-through switching with fixed-size packets exhibiting coarser granularity, or routers with small buffers, can compromise the effectiveness of the detection mechanism. For example, some Infiniband switches only admit 4 packets per input buffer, [16].

B. Uncertainty when using output credits

In a credit-based flow control mechanism the sender knows the buffer size of the receiver. A credit count approximates the remaining buffer space in the neighbor router. When a packet is sent, the credit count is decremented, and when an ACK packet is received (because the neighbor forwarded one packet from its input buffer) the credit count is correspondingly incremented. The bandwidth-delay product determines the minimum buffer for reliable continuous transmission.

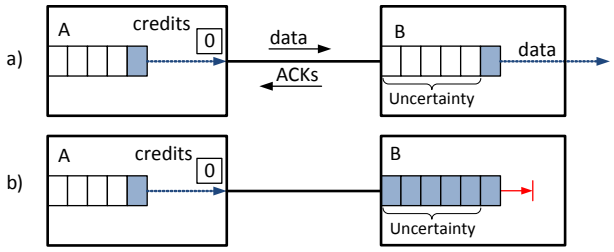


Fig. 1: Uncertainty in the use of credits. With small buffers, the continuous transmission in a) is indistinguishable from a full queue b), because all packets and credits are in-flight.

The estimation of the remaining buffer space in the neighbor node from the credit count contains an inherent uncertainty due to the data packets and ACK messages which are in-flight on the link. Figure 1 depicts the corner case in which the buffers of two consecutive routers A and B have almost the minimum capacity dictated by the link round-trip time (RTT). In both cases the credit count in the output port is 0. In case a) there is no network congestion, and router B forwards all packets as soon as they arrive. However, because of the packets and credits that are in flight, the credit counter in the output port of router A is zero. In case b) the buffer in router B is full because of congestion, so obviously the output credit count in the first router must be zero. The key point is that a null credit count cannot distinguish between the fluid case a) and the congested case b) because the sender is not aware of the packets and credits in-flight. This means that to support credit-based misrouting triggering, the buffer size should be significantly larger than the limit dictated by the RTT. Tracking the rate at which credits are returned could mitigate this problem, at a cost of higher implementation complexity, but would still be affected in the event of changes in the traffic pattern.

C. Response time on traffic changes and slow-lane traffic

Occupancy-based congestion detection mechanisms require, obviously, a high occupancy in the buffers of the current path before selecting an alternative route. However, when the traffic pattern changes to an adversarial case which generates network hotspots, a significant amount of time is required to fill the buffers in the current path before a router changes to an alternative path. Additionally, the traffic in the congested path is condemned to suffer a high latency before reaching its destination. This problem exacerbates with large buffers.

The problem is illustrated in Figure 2. After a traffic pattern change, the traffic from input ports $P_1 - P_4$ in router A should go minimally via output port P_9, but might select a nonminimal path using output ports $P_5 - P_8$, as depicted

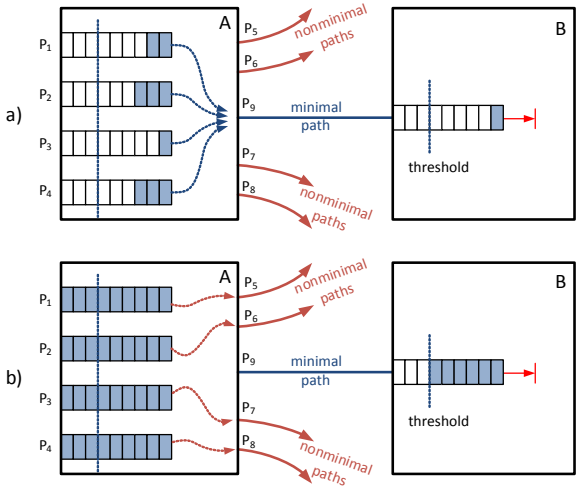


Fig. 2: Response time on traffic changes and slow-lane traffic. In a) the traffic pattern changes and multiple input ports compete for the same minimal output, which has low occupancy. When this queue gets full enough in b), the traffic is diverted nonminimally, but all the queues are full and will take a long time to drain.

in case a). Since multiple input ports in router A compete for the same output, nonminimal routing is preferable in this situation. However, before the input queue in router B reaches a significant population count, all the input queues in router A compete for the same minimal output and will send data through it. When the credits of output P_9 reach the required threshold, depicted in b), the traffic can be diverted nonminimally, but in this moment the input queues of router A will typically be quite populated. In addition to the problem of the high latency required to detect an adversarial traffic situation, packets in the minimal path will also experience a high latency during the queue drain. This is an unavoidable overhead since some traffic needs to go on the slow, congested path, in order for the routers to detect congestion.

D. Oscillations of routing

Occupancy-based congestion detection is prone to oscillations between different paths (for example, minimal and nonminimal) due to the existence of a feedback loop. When the minimal path becomes congested, traffic is diverted to nonminimal routes. Then, the buffers in the minimal path drain their packets, so traffic is moved again to the minimal path, generating a cycle. Such oscillations are especially important when the routing decision is not taken using local information, but rather relies on Explicit Congestion Notification (ECN) messages. An example of such problem will be presented in Section V-C with Piggybacking routing in Dragonflies.

III. Contention-based misrouting trigger

In this Section we first introduce the general idea behind contention-based adaptive routing, and then three specific mechanisms for high-radix routers. Two of these mechanisms are topology-agnostic, while the third one has been designed

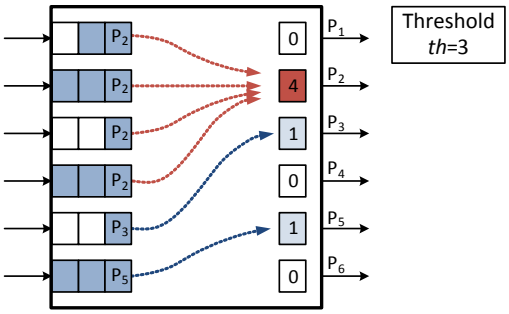


Fig. 3: *Base* contention-detection mechanism. Contention detected in port P_2 since its counter exceeds the threshold th.

for a Dragonfly network. In this Section we assume that each packet has one "preferred" minimal path, and determine the condition to select an alternative nonminimal path. Which specific path is selected among the possible options depends on the topology employed; our implementation in the Dragonfly network will be presented in Section IV-A and its application to alternative topologies is discussed in Section VI-D.

A. General idea

The idea behind the contention-based misrouting trigger is to decide the path to follow based on the contention level of each port, estimated from flows in the input queues that would proceed minimally through each output port. When many packets want to go on a given output, such output suffers from contention. In such case, packets will be diverted to alternative paths using non-minimal routing, without requiring the queues to be full. Hence, the mechanism decouples the buffer capacity from the misrouting trigger mechanism.

From this general idea, multiple variations of this scheme can be conceived. In this paper, we have considered two basic implementations that rely on local information. Additionally, we introduce a third mechanism, *ECtN*, which distributes contention information among the routers in the network, increasing the statistical significance of the counters.

B. Base

This *Base* mechanism employs one counter per output port, denoted *contention counter*, as depicted in Figure 3. When the header of a packet reaches its input buffer head, the routing mechanism determines its minimal output path and increases the corresponding contention counter. Alternative (nonminimal) routing is triggered only when the contention counter in the minimal path of the packet exceeds a given threshold *th*.

This contention counter remains increased until the packet is completely forwarded, even though the packet might be transmitted through a different output port. Thus, counters are decremented only when a packet tail is removed from the input buffer. We do not increase the counters when a packet enters an input buffer because, depending on the buffer size, this might allow for a single flow from one input port to

trigger misrouting. Similarly, we do not decrement the counter when a packet header starts to be forwarded. Since different ports receive packet headers in different cycles, decrementing contention upon header forwarding would lead counter values to be excessively low to provide statistical significance.

This *Base* mechanism works for high-radix routers, because there are multiple input ports which contribute to contention detection, giving statistical significance to the counters. Note that, when multiple virtual channels are used per port, each of them can concurrently increment the corresponding counter, although they can not concurrently advance to the crossbar.

C. Hybrid

Hybrid considers the contention counters and the buffer occupancy to take into account both the contention and congestion levels. In this implementation, there is one threshold for contention counters and another one for the output credits. Traffic is routed nonminimally when any of the two individual thresholds is exceeded. Both of them can be higher for the same final accuracy, avoiding the problems of excessive misrouting that can arise with a too low misrouting threshold.

D. Explicit Contention Notification (ECtN)

Explicit Congestion Notification (ECN) mechanisms send control messages to alert other routers (or the traffic sources) of a congestion situation. Analogously, the idea of **E**xplicit **C**on**t**ention **N**otification (*ECtN*) is to distribute contention information among several routers in the network, so they have more information to make an accurate routing decision.

We have applied *ECtN* to the Dragonfly network introduced in Section I and detailed later in Section IV-A. Every router maintains two arrays of global contention counters, denoted *partial* and *combined*, as seen in Figure 4. Each of them has one counter per global link of the group; if there is only one link between pairs of groups, there will be as many counters as remote groups.

The counters in the *partial* array are updated from the router input queues. When a packet is injected into a group and its destination is a remote group, the router increases the corresponding counter in its *partial* array. This occurs with local traffic at the head of injection queues, or with remote traffic being received through a global input port. As in *Base*, the *partial* array is only decremented when the packet leaves the input queue (note that it is not possible to decrement it when it leaves the group using local information).

The *combined* array is calculated by adding the counters of all the *partial* arrays. Periodically, the routers broadcast their *partial* arrays. Upon reception of a *partial* array update, routers update their *combined* arrays, as depicted in Figure 4. With this mechanism, routers have contention information for all the global ports in the group. When traffic is injected to a group and the corresponding *combined* counter exceeds a given threshold, the packet will be misrouted.

Additionally, the router also maintains one *local* counter per output port as in *Base* or *Hybrid*. They provide contention information for its own output queues, local links included, and allow for in-transit hop-by-hop routing decisions.

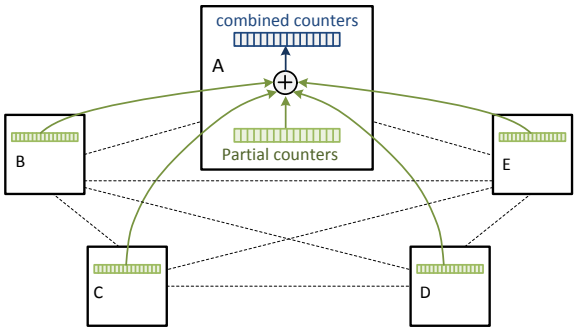


Fig. 4: Combination of *partial* counters in router A in *ECtN*.

IV. Evaluation methodology

In this Section we present the environment used to evaluate the proposals. We first present a brief overview of the Dragonfly and the implemented routing mechanisms. Next, we detail our simulation tool and the parameters employed.

A. Dragonfly topology and implemented routing mechanisms

Dragonfly networks [4] are highly scalable high-radix direct networks with a good cost-performance ratio and relatively short paths. They are considered as a promising topology to build Exascale supercomputers [17]. They are two-level hierarchical networks, where a group of routers at the first level form a virtual high-radix router. These groups of routers connect on a second-level interconnection pattern. We focus on *Canonical Dragonflies* [18] with complete graphs in both topological levels such as in PERCS [6], but our results could be similarly applied to alternative connectivity patterns.

Such network can be defined with three parameters [4]: p, the number of nodes connected to each router, a, the number of routers in each first level group, and h, the number of global links that each router uses to connect to routers in other groups.

Dragonflies are prone to network congestion under adversarial traffic patterns, both in local (intra-group, [19], [12]) and global (inter-group, [4], [6]) links. We denote *ADV+i* the adversarial pattern in which all nodes in a group send their traffic to the group i positions away. This can saturate the global link, as in *ADV+1*. The case of *ADV+h* exhibits an additional pathological case of saturation in the local links. The traffic pattern determines performance of each routing.

Minimal (MIN) routing sends traffic hierarchically to the destination, first to the destination group (using up to one *local* and one *global* link, *lg*), then minimally to the destination router using one *local* link, *l*. This is appropriate for uniform traffic (*UN*), but suffers under adversarial traffic.

Valiant (VAL) [20] sends traffic nonminimally, first to a random intermediate router (*lgl-*), then minimally to the destination (*-lgl*). This increases path diversity at the cost of longer paths. Sending traffic to an intermediate group avoids saturated global links in the minimal path, and we denote it as *global misrouting*. The two local hops in the intermediate group (*l-l*), can be seen as *local misrouting*, and avoid the pathological congestion in *ADV+h* when a single hop is used [12].

Minimal and *Valiant* are oblivious. Adaptive routing mechanisms apply misrouting depending on the network conditions. We implement two adaptive mechanisms based on congestion detection: *PB*, considered the best source-routing adaptive mechanism, and *OLM*, the best in-transit adaptive routing.

In ***PiggyBacking (PB, [21])*** each router marks its global links as saturated or not based on their credit count, and shares this data with the routers in its group, in a form of ECN. *PB* employs source routing: Valiant is applied when the minimal global link is marked as saturated, or when the occupancy of the minimal path in the source router is too congested compared to the Valiant path. Otherwise, Minimal is used.

Opportunistic Local Misrouting (***OLM***, [22]) applies in-transit local and global misrouting: global misrouting can be selected at injection or after a first hop, as in *PAR*, [21], based on the credits of the current router. Nonminimal global link selection is random, according to the MM+L policy defined in [23]. Local misrouting can be used in the intermediate or destination groups to avoid saturated local links. Both cases compare the credits of the different ports, triggering misrouting when the occupancy in the nonminimal output is below a percentage of the minimal output. Both *PB* and *OLM* employ relative misrouting thresholds, rather than the simplified fixed threshold used in the explanation of Section II-C.

For contention-based adaptive routing, we implement the three models from Section III. They are adapted for in-transit adaptive routing in the Dragonfly as follows. We implement the same misrouting policy and deadlock avoidance mechanisms as OLM. The routing decision for a packet is taken when it reaches the head of an input queue. In ***Base***, when the contention counter in its minimal path exceeds the fixed misrouting threshold, a nonminimal path is selected randomly among all the available ports with a contention counter under the threshold. In ***Hybrid***, even if contention counters do not impose misrouting, traffic can be diverted based on the credits of the minimal and nonminimal paths; in this case the nonminimal path is selected randomly based on the same occupancy comparison as in *OLM*. Finally, in ***ECtN*** global misrouting can be selected at injection depending on the *combined* counters; in this case, the nonminimal path is selected randomly among those global links in the current router with a *combined* counter under the threshold. For subsequent hops, the original counters from *Base* are used.

B. Simulation infrastructure

We employ the FOGSim network simulator [24] to model input-output-buffered routers with several virtual channels to avoid deadlock and mitigate Head-of-Line blocking. Unfortunately, it is unaffordable to implement a detailed model of a tiled high-radix router [1] in a simulation of this scale, so we use a simple model of a router with a 5-cycle pipeline. We employ a separable batch allocator, with $2\times$ frequency speedup (*internal* or *crossbar* speedup) to avoid performance limitations due to Head-of-Line Blocking and suboptimal arbitration. Unless otherwise noted, the simulation parameters employed are detailed in Table I.

Parameter	**Value**
Router size	31 ports (h=8 global, p=8 injection, 15 local)
Router latency	5 cycles
Frequency speedup	$2\times$
Group size	16 routers, 128 computing nodes
System size	129 groups, 16,512 computing nodes
Global link arrangement	Palmtree [18]
Link latency	10 (local), 100 (global) cycles
Virtual Channels	2 (global ports), 3 (local and injection ports), 4 (local ports, *VAL* & *PB* to avoid deadlock)
Switching	Virtual Cut-Through
Buffer size (phits)	32 (output buffer, local input buffer per VC), 256 (global input buffer per VC)
Packet size	8 phits
Congestion thresholds	50% (OLM), 35% (Hybrid), $T = 3$ (PB)
Contention thresholds	6 (Base, ECtN), 7 (Hybrid), 10 (ECtN, *combined* counters)
partial update	100 cycles (ECtN)

TABLE I: Simulation parameters.

We model latencies of 10 and 100 cycles for both data and ACK packets in local and global links. These values are the same as in [21], which correspond to average wire length of 2 and 20 meters with a router frequency of 1 GHz. With a phit size of 10 bytes, this leads to a transmission speed of 10 GB/s. 8-phit packets comprise 80 bytes, enough for a 64-byte payload as in [5]. Higher latencies would increase the buffer requirements and the uncertainty of congestion-based adaptive routing mechanisms, which favours contention counters.

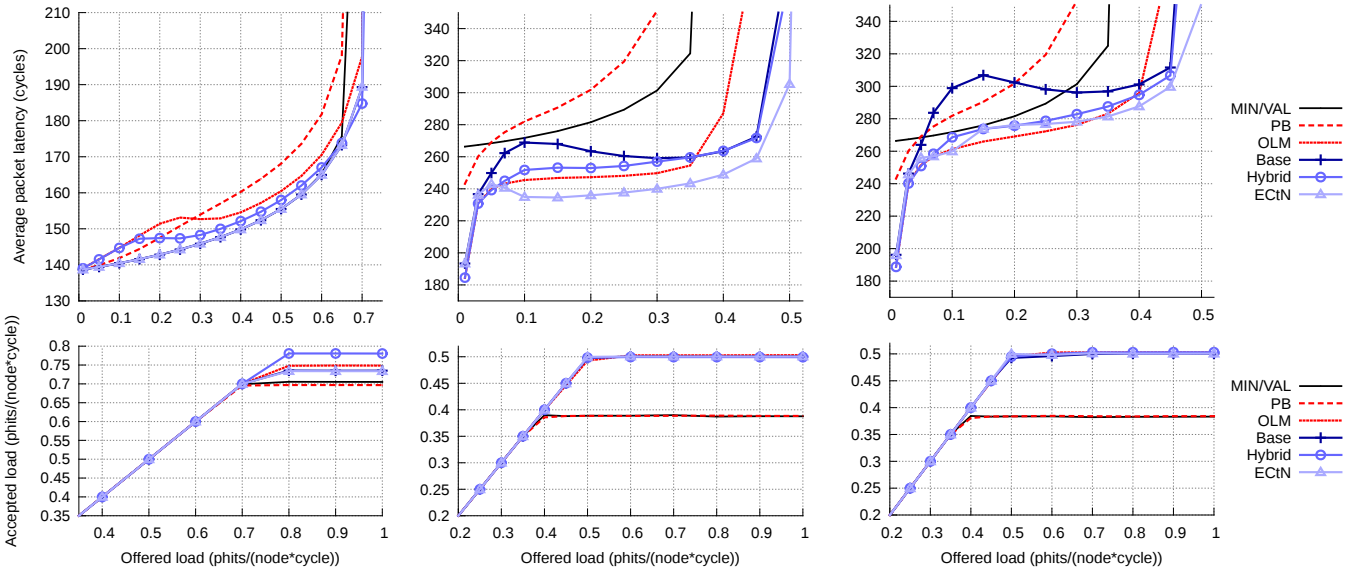
We employ synthetic traffic to evaluate performance. Each source node generates packets according to a Bernoulli process, with a controllable injection probability in phits/(node·cycle). We use the uniform (UN) and adversarial (ADV+1 and ADV+8) traffic patterns described before.

We model steady-state and transient experiments. In both cases, we first warm-up the network for a sufficient time. For steady-state experiments, we then simulate 15.000 cycles of execution during which several million packets are delivered, measuring their average latency and throughput. For transient traffic, after warm-up with a given traffic pattern, we change it to a different pattern, measuring the evolution of the latency and the percentage of globally misrouted packets. 10 simulations are averaged to obtain the figures in the paper.

V. Performance results

A. Steady state

Figure 5 shows the latency and throughput obtained under steady state experiments. Figure 5a (upper graph) portrays the latency under uniform random traffic (UN). In this case, the oblivious *MIN* routing mechanism sets the lower limit in latency, because it never misroutes traffic. Both adaptive mechanisms based on credits, *PB* and *OLM*, obtain higher latency, since they occasionally send traffic nonminimally based on their measured buffer occupancy. By contrast, *Base* and *ECtN* match perfectly the optimal latency of *MIN* before congestion, which arguably is the most frequent region of operation of the network. *Hybrid* can send traffic nonminimally based on the credit count, which occasionally happens under low loads, and its latency is between *MIN* and *OLM*.



(a) UN. (b) ADV+1. (c) ADV+h.

Fig. 5: Latency and throughput under uniform (UN) and adversarial traffic (ADV+1).

By contrast, throughput shown in Figure 5a (lower graph) exhibits a different behaviour. *OLM* improves the throughput of *MIN* since it employs more VCs and, under heavy congestion, it sends some traffic nonminimally to exploit all available outputs. Such behaviour had been already observed in [22]. The throughput of *Base* and *ECtN* is close to the achieved by *OLM*, because they detect network contention faster than *OLM* does for network congestion, thus increasing the level of misrouting attempted. This behavior can be slightly improved by using a higher misrouting threshold, but at a cost of obtaining poorer performance under adversarial traffic patterns, as discussed in Section VI-A. *Hybrid* employs a threshold $th = 7$, and its throughput peaks for the studied mechanisms, thanks to the combination of network congestion and contention information.

Figure 5b depicts the response under adversarial traffic. ADV+1 traffic requires global misrouting, and *VAL* is the reference since it always misroutes packets. *PB* achieves slightly worse results, specially due to the local misrouting in the intermediate group, which is unnecessary for this traffic. The adaptive *OLM* obtains better latency and throughput than *VAL*, since it avoids local misrouting and it sends part of its traffic minimally when possible. The throughput of the Base, Hybrid and ECtN contention counters mechanisms is identical to *OLM*, reaching the Valiant limit of 0.5 phits/(node·cycle). Their latency, by contrast, shows a particular behaviour, with three different zones. Under very low loads (0.01) their latency is relatively low, because traffic is sent on the minimal path which is not congested. With low loads (around 0.05-0.10) the latency using contention counters is slightly higher than *OLM*. With these traffic loads, there are not enough packets in the input queues to increase the contention counters and provide an accurate estimation of contention, leading to minimal routing of traffic. This leads to packets accumulating in the head of the queues, until the counter eventually reaches the fixed threshold and traffic is diverted nonminimally. Interestingly, for these loads the latency only increases on the few cycles required for the accumulation of traffic that triggers misrouting. Finally, under loads up to 0.5, the latency obtained with the contention counter mechanisms is competitive with *OLM*. *ECtN* obtains the best performance, better than *OLM*, since the distribution of contention information among all the routers in the group increases the statistical significance of the measurement, allowing for misrouting at injection whenever it is required. *Hybrid* closely follows *OLM*, whereas *Base* obtains higher latency with traffic loads under 0.3.

Figure 5c shows the result under ADV+8 traffic, which requires local misrouting in the intermediate group. The response is similar to ADV+1, with the only exception of *ECtN* being slightly outperformed by *OLM* for traffic loads between 0.1 and 0.3. Contrary to ADV+1, this traffic requires local misrouting in the intermediate group, so the latency of *VAL* and *PB* (which misroute traffic to an intermediate node in our implementation, not to the intermediate group) is more competitive than in ADV+1.

Finally, Figure 6 represents the average latency obtained when the traffic pattern is a combination of ADV+1 and UN in different rates, with a load of 35%. Even in intermediate cases in which the traffic pattern is not clearly shaped, contention counters are competitive with *OLM*. Notably, *ECtN* clearly outperforms the reference *OLM*.

B. Transient traffic

Figure 7 displays the response of the adaptive mechanisms with small buffer sizes of 32 and 256 phits. After a warmup with UN traffic with load 20%, in time $t = 0$ the traffic pattern

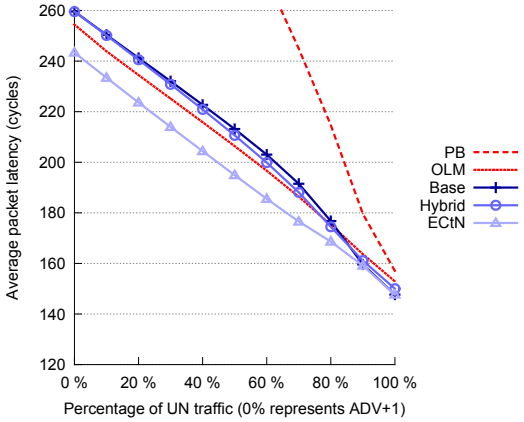
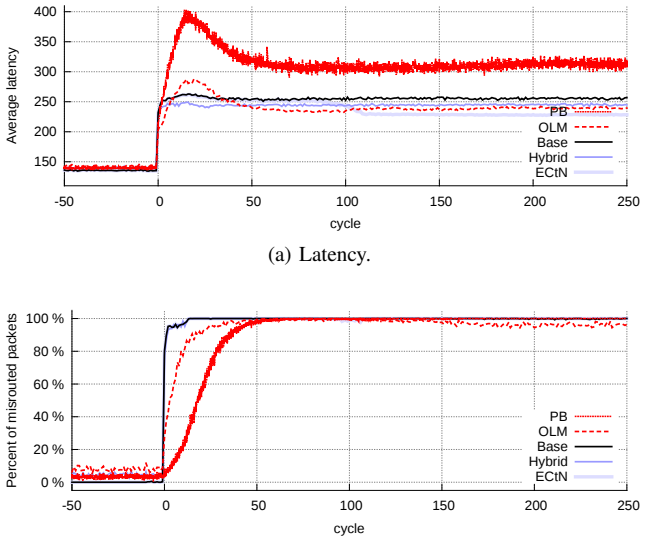


Fig. 6: Latency with mixed traffic patters. Load = 35%, divided among ADV+1 (left) and UN (right).



(a) Latency.

(b) Percentage of misrouted packets.

Fig. 7: Evolution of latency and misrouting when traffic changes from UN to ADV+1 with load 20%, with small buffers.

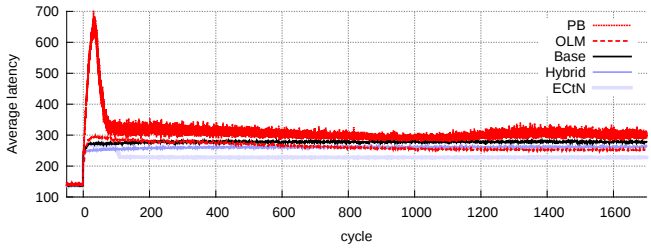


Fig. 8: Evolution of latency when traffic changes from UN to ADV+1 with load 20%, with buffers of 256 phits per VC in local ports and 2048 phits per VC in global ports.

changes to ADV+1. Other transitions are omitted for space limitations, but the response is similar. Figure 7a shows the latency evolution. The congestion-based adaptive mechanisms, *OLM* and *PB*, show a transient period of around 100 cycles while routing is adapting to the new traffic. By contrast, *Base* and *Hybrid* react almost immediately, with a response time of around 10 cycles. Finally, *ECtN* follows *Base* for the first 100 cycles, because the traffic changed exactly when the *partial* counters were being distributed (with the values from the previous traffic UN) and it relies on the local counters. At time $t = 100$ the updated *partial* counters corresponding to ADV+1 are distributed, so each router is aware of the adversarial traffic. From this moment routers misroute traffic directly from the injection queues, preventing local hops in the source group and decreasing latency and local links usage.

Figure 7b shows the amount of misrouted packets, which follows the same trend as latency in Figure 7a. It is notable that the amount of misrouted packets when using counters is very close to 0% or 100% when the routing stabilizes; this is further discussed in Section VI-C.

Figure 8 displays the response time as traffic changes from UN to ADV+1 when buffers are 256/2048 phits for input local/global ports (instead of the 32/256 used in Figure 7a). Output buffers maintain their previous size. As discussed in Section II-C, the use of large buffers delays the detection of traffic changes and the adaptation to new traffic patterns. The response time of the two credit-based mechanisms, *PB* and *OLM*, is much larger than in Figure 7a : around 1000 cycles for *OLM* and 500 for *PB*. By contrast, the mechanisms based on contention present the same response time. Additionally, in order to obtain these results we had to tune the *OLM* misrouting threshold after modifying the buffer sizes, which is unnecessary when using contention-counters.

C. Oscillations of routing

Routing mechanisms that react to congestion are prone to oscillations, because the routing control variable (congestion status) depends on the routing decisions. When congestion status is received via ECN from a remote router, this effect is amplified because the control loop is longer. PiggyBacking implements such an adaptive routing policy, with the source routing decision taken from the "saturation" information received from the neighbour routers in the group. Figure 9 shows the latency transient response to the change from UN to ADV+1 traffic in a larger timescale than Figure 7. The response of *PB* presents oscillations, around every 500 cycles. These oscillations get progressively smaller as the queue occupancy converges, but they never completely disappear.

By contrast, in *ECtN* the routing depends on the traffic contention, which is independent of the routing decision, so there is no forwarding loop. The response, after convergence, is completely flat. The 100 cycle delay caused by the period of distribution of the *partial* counters was discussed in Section V-B, and possible mechanisms to reduce this delay will be considered in Section VI-B.

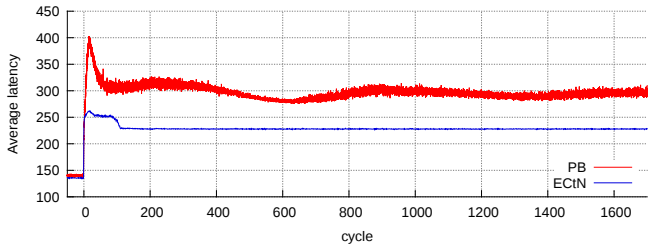


Fig. 9: Evolution of latency when traffic changes from UN to ADV+1, with small buffers and load 20%.

VI. Discussion

A. Misrouting threshold selection

Section V employed a misrouting threshold of $th = 6$. As with other adaptive routing mechanisms, threshold selection imposes a tradeoff between performance under uniform and adversarial traffic patterns. Figure 10 shows the latency and throughput obtained with different threshold values. As expected, higher threshold values provide better response under uniform traffic, and lower values improve adversarial traffic.

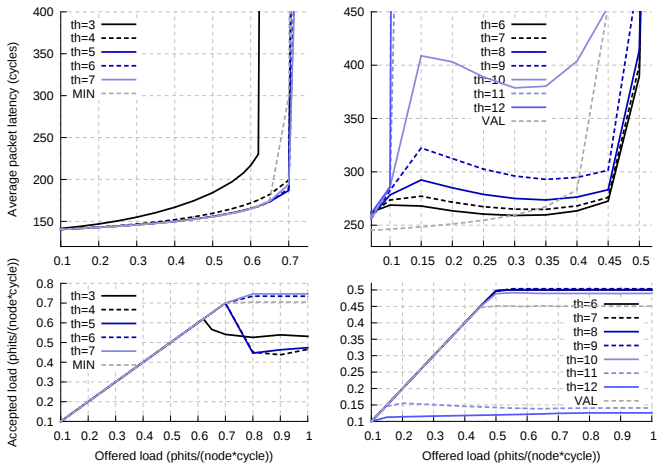
Low threshold values penalize UN traffic, as observed in Figure 10a. The threshold should be high enough to prevent false triggers under saturation so misrouting does not appear frequently. A simple analysis can be done assuming locally-random traffic and the number of VCs and ports in the router. Under saturation it is safe to assume that all input VCs will have at least one packet that will increase the value of a given counter. Thus, the average value of the contention counters will equal the average number of VCs in the input ports. In our case, with the values in Table I, the average is 2.74. A threshold doubling this value ($th \geq 6$) makes misrouting unfrequent enough so performance does not decrease.

High threshold values penalize ADV traffic, as observed in Figure 10b. In this case, the packets in all the p injection ports in a router target minimally the same destination, typically, a local link to other neighbor router with a direct global link to the destination group. In such case, the threshold must ensure that misrouting is applied at injection, what requires $th \leq p$. In practice there is more traffic in local and global input ports, so there is not an abrupt change in performance as the threshold increases, but the previous estimation appears reasonable.

Within the valid range ($6 \leq th \leq 8$ in the example), the lowest threshold should be selected to favor low latency under adversarial traffic, leading to $th = 6$. A similar study was applied to select the *combined* threshold $th = 10$ in *ECtN*. Interestingly, larger routers (such as the 48-port Aries [5] or the 56-port Torrent [25]) enlarge the range of threshold values that do not compromise neither adversarial traffic latency nor uniform traffic throughput.

B. Complexity of the implementation

The complexity of the *Base* and *Hybrid* mechanisms is very low: several parallel counters [26] need to be updated and compared for every packet being sent, similar to the ordinary routing actions. Additionally, the update of contention counters does not need to be in the critical path, since a slight delay does not significantly harm performance. By contrast, the cost of *ECtN* can be significant. *ECtN* requires two additional sets of counters (*partial* and *combined*) plus the required memory to hold the *partial* values received from other routers. In terms of traffic, we have assumed in our simulations that the full *partial* counters are spread every 100 cycles, without simulating the corresponding overhead. *Partial* arrays contain 128 counters (for the 128 global links per group) and each of them requires 4 bits, which are enough to saturate the misrouting threshold $10 \leq 2^4$. With the 10-byte phits considered in Section IV-B, this would require around 6 phits, or a 6% overhead.



(a) UN. (b) ADV+1.

Fig. 10: Sensitivity of *Base* to the misrouting threshold.

Alternative mechanisms can be used to reduce the traffic load of *ECtN*. The simplest case would be to send only nonempty values. In such case, a 7-bit identifier needs to be included to identify the corresponding counter among the 128, making $7 + 4 = 11$ bits per counter. Up to 40 counters can be active at a time (since we consider 8 global ports with 2 VCs and 8 injection ports with 3 VCs), so the overall data of this alternative would be similar to sending the full *partial* array. However, two simple improvements can be applied in this case: a) incremental updates, which build on the last sent version of the *partial* array, and b) asynchronous updates, which increase the ordinary dissemination period, but can send the counters which are detected to change abruptly.

C. Use of the minimal paths under adversarial traffic

The implemented models employ a fixed misrouting threshold. Under heavy adversarial traffic load, this can lead to all of the traffic being diverted nonminimally because the contention counters are high. Meanwhile, the minimal path might remain completely empty. In a real system this would typically not happen because not all traffic can be sent adaptively (e.g. in Cascade [5] minimal routing is used for packets that need to preserve in-order delivery). Alternatively, a statistical misrouting trigger can be considered. When the corresponding

contention counter exceeds a threshold, the probability of routing nonminimally grows with the counter value, but the minimal path is still used in a certain proportion. We have not explored this model in this paper.

D. Alternative topologies

In this work we have evaluated contention counters with Dragonfly networks based on complete graphs in the local and global topologies. Such network is amenable, since there is only one minimal path which identifies the contention counter to use for misrouting trigger. A similar case occurs with Flattened-Butterflies using Dimension Order Routing.

However, many network topologies have multiple minimal paths and multiple non-minimal paths, such as Dragonfly networks with parallel links between groups, Folded-Clos or Torus. In a different context, it has been shown how contention information can be used to select between multiple minimal paths [13]. The application of contention counters to select between the multiple minimal or nonminimal paths is very dependent on the characteristics of each particular topology, and out of the scope of the current paper.

VII. Related work

The design of large-radix routers has been studied in multiple works, such as [1], [27], [28]. Large-radix routers allow for interconnection networks which scale to large number of nodes and they are assumed to optimally exploit the available pin bandwidth of current chips. Some examples of topologies based on large-radix routers are folded-Clos, Flattened Butterfly [2] or Dragonfly [4] networks.

Valiant routing [20] avoids network hotspots by sending all packets minimally to a random intermediate router, and then minimally to destination. The impact of using an intermediate group in the Dragonfly, instead of an intermediate router, was evaluated in [29]. Different variants of nonminimal adaptive routing have been proposed for multiple network topologies, such as folded-Clos [9], Flattened Butterflies [2], [3] or Dragonflies [4], [21], [12]. The problem of oscillations of adaptive routing has been known for a long time, [30], [31]. In all of these cases, the misrouting trigger relies on a congestion detection scheme based on buffer occupancy.

Congestion detection mechanisms in WAN and lossy networks have been typically indirect, based on collisions, packet drops or jitter. Random Early Detection (RED [32]) mechanisms analyze the buffer occupancy to determine the congestion status. When routers detect congestion, the sources can be notified indirectly (i.e., by dropping packets) or explicitly (ECN: Explicit Congestion Notification). ECN is used in many technologies, such as the FECN and BECN messages in Frame Relay, the EFCI bit in ATM cells, the ECN bits in IP [33], the Quantized Congestion Notification in Datacenter Ethernet (802.1Qau) [34] or the congestion control in Infiniband [35].

Most congestion-control implementations react by throttling injection, [36], [37]. For example, focusing on HPC and Datacenter networks, the Datacenter TCP protocol [38] uses the IP ECN bits to restrict the transmission window of the sources, relying on an estimation of the amount of congestion. There exist alternative mechanisms that use adaptive routing to circumvent congested network areas. Such routing was proposed for lossless Datacenter Ethernet networks in [39], while Piggybacking and Credit Round-Trip Time (PB and CRT, [21]) behave as ECN mechanisms to support adaptive source routing in Dragonfly networks. Alternative mechanisms to cope with congestion such as RECN [40] alleviate the impact of congestion by using separate buffers for congested traffic, but require additional hardware in the router logic.

Contention indicators have been employed to drive routing in alternative contexts. Elwhishi *et al.* introduce their use in the context of shared-medium mobile wireless networks [15]. In the context of mesh-based networks-on-chip, Regional Congestion Awareness (RCA) [13] explores the use of contention information for minimal adaptive routing. It shows that contention information can be effectively employed to select between different minimal paths. RCA relies on the evolution of crossbar demand (i.e. allocator requests) for the output ports, whereas our contention counters track the minimal output port of each packet, regardless of its actual followed path. Although they could be similar under uniform traffic, their behaviour could differ with adversarial traffic: crossbar demand could oscillate between alternative paths, whereas contention counters not. In the same context, Chang *et al.* [14] consider the rate of change in the buffer levels to predict congestion, what avoids uncertainty issues with small buffers. In the context of interconnection networks, Dynamic Routing Control [41] detects hotspots in Omega networks with oblivious routing by counting the packets in each input queue with the same destination, and prioritizes traffic not targeting the hotspot, without adapting routing.

VIII. Conclusions

This paper has introduced the idea of *Contention-based adaptive routing* which mitigates the main shortcomings of congestion-based adaptive routing. Our proposal is independent of the buffer size, does not suffer from oscillations in routing, and has fast adaptation to changes in the traffic pattern. This idea can be implemented in high-radix routers relying on a low-cost set of contention counters. We have modelled the mechanism for large-scale Dragonfly networks.

Our *Base* mechanism obtains optimal latency under uniform traffic, competitive throughput when compared to the best state-of-the-art adaptive routing mechanisms, and immediate adaptation to traffic changes.

Two alternative variations have been studied. First, a *Hybrid* version which combines contention and congestion information improves throughput, but provides worse latency under uniform traffic. Second, the *ECtN* version which disseminates contention information. This mechanism provides the best latency (or close to) in all scenarios and can be applied to low-radix routers, but entails a higher implementation cost, both in area and communication requirements.

ACKNOWLEDGMENT

This work has been supported by the Spanish Ministry of Education, FPU grant FPU13/00337, the Spanish Science and Technology Commission (CICYT) under contracts TIN2012-34557 and TIN2013-46957-C2-2-P, the European Union FP7 under Agreement ICT-288777 (Mont-Blanc) and ERC-321253 (RoMoL), the European HiPEAC Network of Excellence, and the JSA no. 2013-119 as part of the IBM/BSC Technology Center for Supercomputing agreement.

REFERENCES

[1] 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. J. Dally, and D. Abts, "Flattened Butterfly: a cost-efficient topology for high-radix networks," in *ISCA: Intl. Symposium on Computer architecture*, 2007, pp. 126–137.

[3] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "HyperX: topology, routing, and packaging of efficient large-scale networks," in *SC '09: Conf. on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 41:1–41:11.

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

[5] 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 *SC: Intl Conf on High Performance Computing, Networking, Storage and Analysis*, 2012, pp. 103:1–103:9.

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

[7] 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.

[8] 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 *SC: Intl. Conf. for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–10.

[9] J. Kim, W. J. Dally, and D. Abts, "Adaptive routing in high-radix Clos network," in *SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, 2006.

[10] D. Roweth and T. Jones, "QsNetIII, an adaptively routed network for high performance computing," in *IEEE Symposium on High Performance Interconnects (HOTI)*, 2008, pp. 157–164.

[11] A. Singh, "Load-balanced routing in interconnection networks," Ph.D. dissertation, Stanford University, 2005.

[12] M. Garcia, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, G. Rodriguez, J. Labarta, and C. Minkenberg, "On-the-fly adaptive routing in high-radix hierarchical networks," in *41st International Conference on Parallel Processing (ICPP)*, 2012, pp. 279–288.

[13] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *HPCA'08: IEEE 14th Intl. Symp. on High Performance Computer Architecture.*, 2008, pp. 203–214.

[14] E.-J. Chang, H.-K. Hsin, S.-Y. Lin, and A.-Y. Wu, "Path-congestion-aware adaptive routing with a contention prediction scheme for network-on-chip systems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 33, no. 1, pp. 113–126, 2014.

[15] A. Elwhishi, P.-H. Ho, K. Naik, and B. Shihada, "Self-adaptive contention aware routing protocol for intermittently connected mobile networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1422–1435, Jul. 2013.

[16] J. Santos, Y. Turner, and G. Janakiraman, "End-to-end congestion control for InfiniBand," in *INFOCOM 2003. 22nd Annual Joint Conference of the IEEE Computer and Communications.*, vol. 2. IEEE Societies, 2003, pp. 1123–1133.

[17] K. Bergman and et al., "Exascale computing study: Technology challenges in achieving exascale systems," 2008.

[18] C. Camarero, E. Vallejo, and R. Beivide, "Topological characterization of hamming and dragonfly networks and its implications on routing," *ACM Trans. Archit. Code Optim.*, vol. 11, no. 4, pp. 39:1–39:25, 2014.

[19] D. J. Kerbyson and K. J. Barker, "Analyzing the performance bottlenecks of the POWER7-IH network." in *CLUSTER*. IEEE, 2011, pp. 244–252.

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

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

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

[23] M. García, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, J. Labarta, and G. Rodríguez, "Global misrouting policies in two-level hierarchical networks," in *INA-OCMC: Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip*, 2013, pp. 13–16.

[24] M. García, P. Fuentes, M. Odriozola, E. Vallejo, and R. Beivide. (2014) FOGSim Interconnection Network Simulator. University of Cantabria. [Online]. Available: https://code.google.com/p/fogsim/

[25] B. Arimilli, S. Baumgartner, S. Clark, D. Dreps, D. Siljenberg, and A. Maki, "The IBM POWER7 hub module: A terabyte interconnect switch for high-performance computer systems," in *Hot Chips*, 2010.

[26] E. E. Swartzlander, "Parallel counters," *IEEE Trans. Comput.*, vol. 22, no. 11, pp. 1021–1024, Nov. 1973.

[27] J. H. Ahn, Y. H. Son, and J. Kim, "Scalable high-radix router microarchitecture using a network switch organization," *ACM Trans. Archit. Code Optim.*, vol. 10, no. 3, pp. 17:1–17:25, Sep. 2008.

[28] G. Passas, "VLSI micro-architectures for high-radix crossbars," Ph.D. dissertation, FORTH-ICS, April 2012.

[29] B. Prisacari, G. Rodriguez, M. Garcia, E. Vallejo, R. Beivide, and C. Minkenberg, "Performance implications of remote-only load balancing under adversarial traffic in dragonflies," in *INA-OCMC: Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip*. ACM, 2014, pp. 5:1–5:4.

[30] A. Khanna and J. Zinky, "The revised ARPANET routing metric," in *Symp. on Communications Architectures & Protocols*, ser. SIGCOMM '89, 1989, pp. 45–56.

[31] Z. Wang and J. Crowcroft, "Analysis of shortest-path routing algorithms in a dynamic network environment," *SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 2, pp. 63–71, Apr. 1992.

[32] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[33] K. Ramakrishnan, S. Floyd, and D. Black, *RFC 3168: The Addition of Explicit Congestion Notification (ECN) to IP*, Std., 2001.

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

[35] 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.

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

[37] V. Jacobson, "Congestion avoidance and control," in *SIGCOMM '88: Communications architectures and protocols*. ACM, 1988, pp. 314–329.

[38] 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.

[39] C. Minkenberg, M. Gusat, and G. Rodriguez, "Adaptive routing in data center bridges," in *17th IEEE Symposium on High Performance Interconnects (HOTI'09)*. IEEE Computer Society, 2009, pp. 33–41.

[40] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo, "A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks," in *HPCA-11: Intl. Symp. on High-Performance Computer Architecture.*, 2005, pp. 108–119.

[41] J.-K. Peir and Y.-H. Lee, "Improving multistage network performance under uniform and hot-spot traffics," in *2nd IEEE Symposium on Parallel and Distributed Processing*, dec 1990, pp. 548 –551.