FlexVC: Flexible Virtual Channel Management in Low-Diameter Networks

Pablo Fuentes, Enrique Vallejo, Ramón Beivide
University of Cantabria
Santander, Spain
Email: {fuentesp,vallejoe,beividej}@unican.es

Cyriel Minkenberg
Rockley Photonics Inc
Email: cyriel.minkenberg@rockleyphotonics.com

Mateo Valero
Barcelona Supercomputing Center & UPC
Barcelona, Spain
Email: mateo.valero@bsc.es

***Abstract*—Deadlock avoidance mechanisms for lossless low-distance networks typically increase the order of virtual channel (VC) index with each hop. This restricts the number of buffer resources depending on the routing mechanism and limits performance due to an inefficient use. Dynamic buffer organizations increase implementation complexity and only provide small gains in this context because a significant amount of buffering needs to be allocated statically to avoid congestion.**

We introduce *FlexVC*, a simple buffer management mechanism which permits a more flexible use of VCs. It combines statically partitioned buffers, opportunistic routing and a relaxed distance-based deadlock avoidance policy. *FlexVC* mitigates Head-of-Line blocking and reduces up to 50% the memory requirements. Simulation results in a Dragonfly network show congestion reduction and up to 37.8% throughput improvement, outperforming more complex dynamic approaches. *FlexVC* merges different flows of traffic in the same buffers, which in some cases makes more difficult to identify the traffic pattern in order to support nonminimal adaptive routing. An alternative denoted *FlexVC-minCred* improves congestion sensing for adaptive routing by tracking separately packets routed minimally and nonminimally, rising throughput up to 20.4% with 25% savings in buffer area.

***Keywords*-Buffer management, deadlock avoidance.**

I. Introduction

Low-diameter networks present low average distance between nodes and permit to reduce latency, cost and energy consumption, while achieving high scalability. The most widespread mechanism for deadlock avoidance in these networks relies on a fixed order in the use of virtual channels (VCs). Günther [1] proposed a simple mechanism in which packets follow VCs in an strictly increasing index order per hop. This deadlock avoidance policy implies that supporting long nonminimal network paths, such as those employed by Valiant [2] or in-transit adaptive routing, requires a larger number of resources compared to the base minimal routing. Likewise, avoiding protocol deadlock or supporting multiple QoS traffic classes multiplies the VC requirements (e.g., the Dragonfly network in Cray Cascade requires 8 VCs to support nonminimal routing and avoid protocol deadlock [3]). However, the deadlock avoidance mechanism prevents a free use of these buffers by imposing a strict order with just one valid VC per hop.

For practical reasons it is interesting to decouple the number and usage of VCs from deadlock avoidance. Depending on the traffic pattern, a fixed order policy can use only a subset of the available VCs, potentially increasing Head-of-Line Blocking (HoLB) and decreasing performance. A large number of VCs reduces HoLB, but increases the area required by buffers and the complexity of the router implementation (mainly muxers, demuxers, allocator and flow control logic). This problem exacerbates when deeper buffers are employed to support bursts of traffic and properly detect congestion when adaptive routing is used.

Shared buffer architectures, such as DAMQ buffers [4], can reduce per-port buffering while preserving performance. However, shared buffers are complex, increasing the access delay and the area and power required for a given buffering capacity per port. Moreover, in low-diameter direct networks DAMQ memories require a significant private buffer reservation to avoid deadlock and congestion, which in practice rules out their performance improvements, as studied in Section VI-C. Additionally, in most cases a complete order in the VC usage is not actually required to guarantee deadlock freedom. In fact, the VC employed by a given packet only needs to guarantee that there exists a safe, ascending path to its destination.

Based on these observations, we introduce a buffer management mechanism denoted *FlexVC* which removes the strict VC order proposed for deadlock avoidance, allowing to use the maximum amount of VCs per hop of the path. *FlexVC* combines statically partitioned buffers, a relaxed distance-based deadlock avoidance policy and opportunistic routing. *FlexVC* permits packet forwarding to several VCs, providing similar or better performance than shared buffers. Additionally, by relegating higher-index VCs to latter steps in the path, *FlexVC* is immune to congestion caused by excessive occupancy of a single buffer. Moreover, *FlexVC* can be implemented with less physical buffers than hops in the longest allowed nonminimal path, with buffer reductions up to 50%. Besides simplicity, the main benefits of *FlexVC* compared to a base design include reduced HoLB by using more VCs from the input ports; a lower amount of VCs required in the network, even below the longest path length; increased effective buffering per hop, better supporting bursts of traffic; and partially relaxed path

 DOI: 10.1109/IPDPS.2017.110

restrictions, increasing routing adaptivity.

Low-diameter networks typically require nonminimal adaptive routing, with the misrouting decision based on a congestion estimation. We observe that using a fixed VC per path hop aids congestion sensing. This effect is lost with *FlexVC*, since traffic from different hops can share the same VC. *FlexVC-minCred* circumvents this by accounting separately the credits corresponding to minimally- and nonminimally-routed packets. This mechanism restores the ability to identify adversarial traffic patterns and supports efficient adaptive routing.

In summary, the main contributions of this paper are:

- We introduce *FlexVC*, a simple buffer management mechanism which provides maximum flexibility in buffer use with buffer reductions up to 50%.
- We evaluate the performance of *FlexVC*. Results from our simulations show throughput increases using *FlexVC* up to 37.8% compared to the base case, better handling of traffic bursts and performance improvements compared to DAMQ with a much simpler buffer organization.
- We identify a limitation in congestion sensing caused by merging different flows of traffic in the same buffers, and propose an alternative mechanism denoted *FlexVC-minCred*. This mechanism identifies adversarial traffic properly, improving throughput up to 20% in source-adaptive routing schemes using 25% less buffer area.

The rest of this paper is organized as follows: Section II introduces the background on the topic. Section III presents *FlexVC*. Section IV introduces the evaluation framework, Section V presents the main results, Section VI discusses implementation details and Section VII considers related work. Finally, Section VIII concludes.

II. Background

This section introduces the relevant background relative to topologies, routing, distance-based deadlock avoidance and buffer management.

Topologies Multiple highly-scalable networks with diameter 2 or 3 have been proposed based on high-radix routers. Some examples are 2D or 3D Flattened-Butterflies (FB, [5], [6]); diameter-3 Dragonflies (DF, [7]) or Projective Networks (PN, [8]); and diameter-2 Slim Flies (SF, [9]), Orthogonal Fat Trees (OFT, [10]) and demi-PN [8].

Oblivious routing Multiple routing protocols have been designed for these networks. Minimal routing (MIN) requires at most as much hops as the diameter of each network. MIN is suitable for traffic patterns with uniform (UN) distribution of destinations, achieving optimal latency. Under adversarial (ADV) traffic patterns using MIN some links receive most of the traffic, leading to high congestion and poor performance. Adversarial patterns of many of these networks are studied in [7], [11]. Valiant routing (VAL, [2]) avoids in-network congestion caused by adversarial patterns by randomizing traffic: packets are sent first minimally to a random router, and then minimally to the final destination. VAL makes the traffic uniform and balances it over the network links. However, this doubles the longest network path and, therefore, halves the maximum throughput.

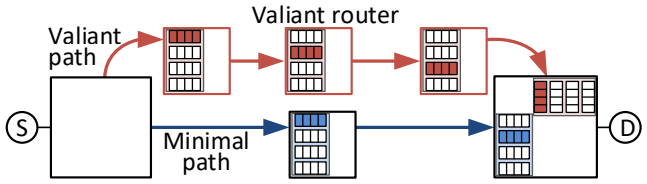


Fig. 1: Distance-based deadlock avoidance with MIN/VAL routing in a diameter-2 Slim Fly network with 4 VCs. Traffic is sent from source S to destination D. In each intermediate router (square box) the only allowed buffer is shaded.

Adaptive routing To avoid the performance disparity between MIN and VAL with different traffic patterns, nonminimal adaptive routing selects the most appropriate one by sensing network congestion. UGAL [12] selects at injection between a minimal and a Valiant path, based on their respective buffer occupancy. However, congestion often occurs in links not directly connected to the source router, what complicates sensing it. This occurs for example in Dragonflies, in which nodes are arranged in groups, with groups connected using global links. Global links are more prone to congestion, but the global link to be used by a packet is often connected to a neighbor router in the source group. *In-transit adaptive* routing mechanisms re-evaluate the routing decision in some hops of the path. Progressive Adaptive Routing (PAR, [13]) may switch from MIN to VAL after a minimal hop.

PB and source adaptive routing *Piggyback* (PB [13]) is a nonminimal *source adaptive* routing mechanism which indirectly senses congestion of remote links. Each router detects congested global ports and distributes this information to its neighbor routers. In particular, each router measures the occupancy (credits) of its global ports, and sets as 'saturated' those links that present 50% more occupancy than the average, sharing these bits with its neighbours. At injection time, each router selects a path based on both the 'saturation' status of the global link in the minimal path, and a local comparison of credits. Global ports may be set as 'saturated' by comparing the overall occupancy of all the VCs, or a single one. As studied in Section V-C, using a single VC is more efficient, since it implicitly identifies the traffic pattern in the network.

Distance-based deadlock avoidance We denote c_i the VC with index i. Distance-based deadlock avoidance assigns a VC c_i for each hop i in the path. This mechanism is clearly deadlock-free: the last VC never blocks since packets are about to be consumed, and the others may depend only on higher-index VCs, without cyclic dependencies. This policy requires in general as many VCs as hops in the longest allowed path. Figure 1 presents an example of a generic diameter-2 network (such as SF or 2D FB). To support VAL routing with path length 4, 4 VCs are required. The VC used in each hop is fixed, which means MIN does not exploit the availability of 4 VCs, since MIN paths have length 2.

Request-reply traffic and protocol deadlock The amount of required buffers increases to avoid protocol deadlock [14].

The typical solution employs separate virtual networks, as in Cascade [3]. Destination nodes receive requests and generate replies back to the original sources. Overall paths can be considered as the concatenation of the request and the reply paths in both virtual networks. Such organization provides isolation between different classes of messages but effectively doubles buffering requirements, since for each original VC now are required a *request VC* and a *reply VC*.

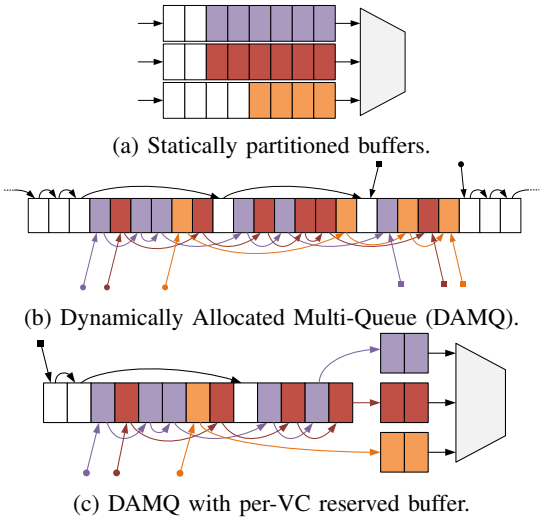
Routing or link-type restrictions In some networks, the number of VCs required for distance-based deadlock avoidance is lower than the maximum path length. This occurs when network links can be classified into different disjoint sets (e.g. X/Y/Z links in a 3D FB; local/global links in a DF; upward/downward links in an OFT), which are traversed in a fixed order (e.g. DOR routing in FB; *l-g-l* MIN paths in DF; up/down routing in OFT). In these cases, the required count of VCs in each set only depends on the amount of hops in the given set. In some cases (such as the FB), this policy trades off path diversity for VC requirements: with distance-based deadlock avoidance, adaptive routing is supported in the FB, but requires more resources. In other cases (such as the DF or OFT) this requirement is imposed by the nature of the topology. In either case, a given VC (type and index) is assigned to each hop in the path, and their order needs to be preserved to avoid deadlock. We denote such sequence a *reference path*.

The example of a diameter-3 DF with complete graphs in the local and global topologies is considered next. Minimal paths comprise a local hop in the source group, a global hop which gets to the destination group, and a local hop at the destination group. This is denoted $l_0 - g_1 - l_2$ to indicate both the type of link and VC index used in the reference path[1]. Shorter paths such as $l_0 - g_1$ or $g_1 - l_2$ are possible when missing hops are not required to reach the destination, but different order of hops such as $l_0 - l_2 - g_1$ or $g_1 - l_0 - l_2$ is never allowed nor required. Therefore, for minimal routing it is enough to implement 2 VCs in input local ports (for hops 0 and 2) and only 1 buffer in global ports (for hop 1). We denote such VC arrangement 2/1. Similarly, VAL requires 4/2 VCs[2] ($l_0 - g_1 - l_2 - l_3 - g_4 - l_5$, note the two subpaths are separated) and PAR requires an additional local VC, 5/2 VCs ($l_0 - l_1 - g_2 - l_3 - l_4 - g_5 - l_6$). Avoiding protocol deadlock doubles these requirements to 4/2, 8/4 and 10/4 respectively.

Buffer organization and cost

Different implementations can be employed in the router buffers, depending on the buffer allocation management. Statically partitioned buffers assign a fixed amount of memory per VC, whereas dynamically allocated buffers (such as Dynamically Allocated Multi-Queues, DAMQs) share a buffer pool between VCs. A constraint of current ASICs is to use limited buffers [18]. DAMQs share a single memory buffer among all the VCs within a port, for example using linked lists. Memory is allocated dynamically to each buffer on demand. Intermediate approaches employ a shared pool together with per-VC private buffering. Figure 2 shows the logical organization of these implementations. Shared buffers use is widespread to improve the utilization of the available space, for example in the SCOC design [19] and the Tianhe-2 network switches [20].

Using multiple VCs per port imposes a significant cost both in terms of area and power, and increases the router control logic. Depending on which router stage constitutes the critical path, this has an impact on router frequency of operation.



(a) Statically partitioned buffers.

(b) Dynamically Allocated Multi-Queue (DAMQ).

(c) DAMQ with per-VC reserved buffer.

Fig. 2: Logical organization of static vs dynamic buffer management with three VCs. In DAMQs a buffer pool is shared between VCs. Head and tail pointers are used to access data from each VC, and allocate free memory (black pointers).

III. *FlexVC* MECHANISM

This section presents the base *FlexVC* mechanism, applied to a Virtual Cut-Through (VCT) network without dependencies between different message classes, routing-induced path restrictions or topology-induced link-type restrictions; these are considered in subsections III-B and III-C. The impact of *FlexVC* on congestion detection mechanisms for nonminimal adaptive routing is considered in subsection III-D.

A. *Base* FlexVC

The key idea behind *FlexVC* is that, in order to guarantee deadlock freedom, packets do not need to *follow* a strictly increasing order of VCs $\{c_0, c_1, c_2, \cdots\}$; it is only required that *such increasing path exists* for every hop in a packet path, starting from the currently used VC (i.e. the index of the input buffer that contains the packet) upwards to the final destination. Such path is denoted escape path [21]. With this idea, the only restriction on the VC index to be used when a router forwards a packet by a given output port is related to whether the subsequent increasing path exists or not.

[1]VC indices in each type of port (e.g. local ports l_0 and l_2) are not assigned consecutively values to simplify the explanation. The amount of VCs is determined by the amount of different indices, not the highest value.

[2]Note that we refer to 'real' Valiant [2], denoted *Valiant Any* in [15] and *Valiant-node* in [16]. Restricted variants of Valiant in the Dragonfly can employ 3/2 VCs [7] but present pathological performance problems [17].

Using the notation in [14], we denote the set of channels by C and the set of network nodes by N. We consider an incremental routing function $R : C \times N \mapsto C$ which specifies an output VC, c_k, for each path determined by the routing protocol. A routing protocol R based on *FlexVC* specifies the *highest* VC, c_k, allowed in each hop. The router allocator and forwarding unit employ a certain *VC selection function* to select any VC, c_j, with available credits in the output port, such that $0 \leq j \leq k$. Different VC selection functions could be *highest-index*, *lowest-index*, *JSQ* (Join the Shortest Queue) or *random*. The routing protocol R considers *safe* and *opportunistic* hops.

Definition 1: Safe hops require that from the input channel c_{j0} there exists a *safe path* $\{c_{j0}, c_{j1}, \cdots, c_{jn}\}$ to the destination with increasing VC index $c_{jk} > c_{jl}\ \forall k > l \geq 0$.

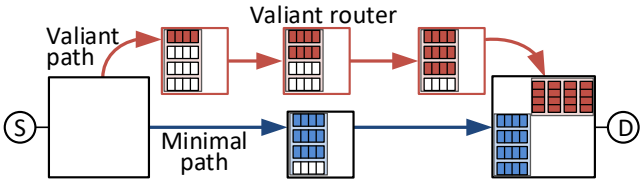
Definition 2: Opportunistic hops require that regardless of the input channel c_{j0} there exists a safe path $\{c_{j1}, c_{j2}, \cdots, c_{jn}\}$ from the next buffer c_{j1} to the destination node, with increasing VC index $c_{jk} > c_{jl}\ \forall k > l > 0$, and with $c_{j1} \geq c_{j0}$. *Opportunistic paths* contain safe hops and one or more opportunistic hops, each of them with their associated safe path as escape.

Every connected routing protocol R must provide at least one safe path for each possible destination to guarantee deadlock-freedom. *Opportunistic* paths can be used as long as the next hop buffers contain enough space for the complete packet; otherwise, packets revert to the corresponding safe path as an escape path. This routing restriction, which avoids the appearance of dependencies in the extended resource dependency graph [21], can be applied to wormhole networks as long as input buffers can hold a complete packet. Several *opportunistic* hops can appear in the same path, as long as there exists a safe escape path from each of them. Note that, as shown in Figure 3, the longest allowed safe path is a design parameter and can vary. Hence, routers with more VCs can be employed to reduce HoLB and to implement more safe paths.

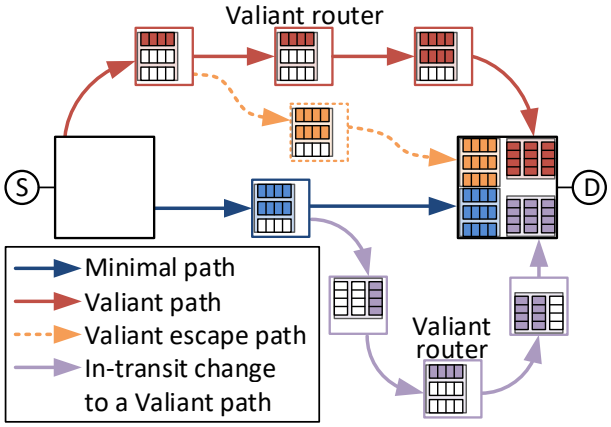
For each safe hop, the VC index determined by R equals the maximum amount of VCs minus the remaining hops to the destination router. For each opportunistic hop, the VC index determined by R equals the index determined by the shortest safe escape path associated to the opportunistic hop.

Theorem 1: A routing protocol R using *FlexVC* is deadlock-free with as many VCs as the longest safe path allowed in the network.

As a sketch of the proof, note that Definition 1 requires as many VCs as the path length to consider such path as safe. With consumption assumption [22], safe paths are deadlock free by induction on the VC indices. Opportunistic paths are deadlock-free by [21] since they necessarily have an escape safe path for each opportunistic hop. Note that, as opposed to Duato's mechanism [21], *FlexVC* does not require dedicated resources for the escape path; instead, VCs in the safe path can also be used for opportunistic hops as long as an increasing VC index order can be guaranteed for the safe path, reducing the minimum number of buffers required. Note also that more VCs than the longest path can be exploited to avoid HoLB; we denote them *additional VCs*.



(a) Safe MIN/VAL paths using 4 VCs.



(b) Opportunistic Valiant and in-transit adaptive paths with 3 VCs.

Fig. 3: Sample *FlexVC* usage in a generic diameter-2 network. Allowed VCs in each hop are shaded.

TABLE I: Allowed paths using *FlexVC* in a generic diameter-2 network.

	VCs			
Routing	2	3	4	5
MIN	safe	safe	safe	safe
VAL	X	opport.	safe	safe
PAR	X	opport.	opport.	safe

Figure 3 presents two examples of paths allowed by *FlexVC* in a generic diameter-2 network (such as a SF or an adaptive FB), from a source node S to the destination D. The situation in Figure 3a employs routers with 4 VCs per input port. With this amount of VCs, minimal and Valiant paths (of length 2 and 4 respectively) are safe paths by Theorem 1, so the amount of allowed VCs per hop only depends on the remaining distance to the destination. Note that, in most hops, *FlexVC* allows to select between several VCs, compared to the base deadlock avoidance protocol which specifies a single one. This is particularly interesting to absorb transient bursts of traffic, since the effective buffer space in each hop increases without requiring DAMQ buffers.

The example in Figure 3b employs routers with 3 VCs per input port. Minimal paths are safe since they only require 2 VCs because the network has diameter 2, but Valiant paths are not safe with less than 4 VCs. However, opportunistic Valiant paths can be implemented, since for each opportunistic hop of the path there exists a safe escape path. The first two hops of the Valiant path are opportunistic; the associated escape path for the second hop is depicted, and the one for the

first hop is the MIN path. Additionally, changing from MIN to VAL in PAR [13] is also allowed with two opportunistic hops in the path: the escape path for the first hop of the Valiant path is the continuation of the minimal path; the next hop is also opportunistic, with its escape path omitted in the figure for simplicity. Note that VAL and PAR would not be deadlock-free with the 2 VCs required for MIN, since their opportunistic hops would not have an associated safe path. Table I summarizes the paths allowed using different amount of VCs in a generic diameter-2 network.

B. FlexVC *considering protocol deadlock*

Section II introduces the use of *request* and *reply* VCs to avoid protocol deadlock. *FlexVC* concatenates the request and reply paths in a single unified sequence in order to increase flexibility in buffer management, and considers the pool of VCs as a unified set rather than distinct virtual networks. The highest VC to be used differs for request and reply messages. Request packets can only employ their associated request VCs in the manner presented in Section III-A. However, reply packets can employ both reply and request VCs, increasing flexibility and further reducing HoLB.

Theorem 2: FlexVC is deadlock-free in presence of request-reply traffic.

The proof is obvious since both request and reply paths are deadlock free by considering Theorem 1 in the sub-sequence of request and reply VCs. Additionally, reply messages can employ request VCs since there exists a safe path to the destination considering the complete sequence of VCs.

Moreover, *FlexVC* can be further exploited to reduce the number of VCs required to support long paths. The set of reply VCs only needs to be dimensioned for safe minimal paths as of Theorem 1, since opportunistic reply hops following nonminimal paths can leverage lower-index request VCs. Therefore, increasing the amount of VCs can be employed to either support longer safe paths (e.g., 4+4 for safe VAL paths in a diameter-2 network) or as *additional VCs* at the start of the request sequence, which reduces HoLB in both subpaths that can use them. Both alternatives are evaluated in Section V-B.

Table II summarizes an example for a diameter-2 network. Distance-based deadlock avoidance requires 5+5=10 VCs to support safe VAL and PAR paths in both request and reply virtual networks. *FlexVC* as presented in Figure 3b supports the same paths using only 3+3=6 VCs. However, if reply VCs are dimensioned to support MIN routing only (leveraging request VCs for opportunistic VAL and PAR paths) the same set of paths would be supported with only 3+2=5 VCs, implying a reduction of 50% compared to the baseline. This case is also illustrated with the example in Figure 4, in which both minimal and Valiant paths are depicted in a network with $3+2=5$ VCs.

C. *Networks with routing or link-type restrictions*

In networks with routing or topology-induced path restrictions, *FlexVC* needs to consider the order of link types followed in reference paths.

TABLE II: Allowed paths using *FlexVC* considering protocol deadlock in a generic diameter-2 network.

	VCs (Request + Reply = overall)				
Routing	2+2=4	3+2=5	3+3=6	4+4=8	5+5=10
MIN	safe	safe	safe	safe	safe
VAL	X	opport.	opport.	safe	safe
PAR	X	opport.	opport.	opport.	safe

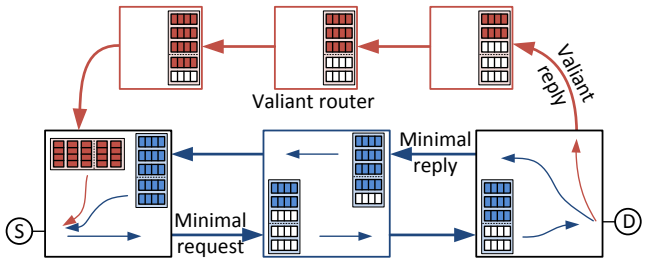


Fig. 4: Example protocol deadlock avoidance in a generic diameter-2 network with $3+2=5$ VCs using *FlexVC*.

TABLE III: Allowed paths using *FlexVC* in a diameter-3 Dragonfly network following local/global links in topology-determined order.

	Dragonfly VCs (Local/Global)				
Routing	2/1	3/1 or 2/2	3/2	4/2	5/2
MIN	safe	safe	safe	safe	safe
VAL	X	X	opport.	safe	safe
PAR	X	X	opport.	opport.	safe

FlexVC considers reference paths with different types of hops. As in Section III-A, *FlexVC* provides the highest VC (of the given type) to be used in each hop and can reduce the amount of VCs required. In this case, escape paths also need to consider the specific sequence of hops in the reference paths, not only the distance to the destination. Consider as an example a Dragonfly network, introduced in Section II. While 2/1 VCs support MIN paths $l_0 - g_1 - l_2$, adding a single VC to either global ports (2/2 VCs) or local ports (3/1 VCs) does not support opportunistic VAL: In the first case, there is no safe path after the first opportunistic local hop l_0 (there are not two additional local hops for MIN routing to the destination) and in the second there is no possible safe escape path after the first opportunistic global hop. Instead, one VC needs to be added to both local and global ports (reaching 3/2) to support VAL and PAR opportunistic paths, using the sequence $l_0 - g_1 - l_2 - g_3 - l_4$. One and two additional local VCs are required respectively to support VAL and PAR safe paths, reaching the reference paths introduced in Section II. This is summarized in Table III. *Additional VCs* of any given type are inserted at the start of the reference path.

Protocol deadlock avoidance requires in each sub-path a longer reference path which considers link types. As in the base case, the reply sub-sequence can be dimensioned for MIN routing, using opportunistic nonminimal paths that exploit the request subsequence of VCs. Results are summarized in Table IV, with 5/3 overall VCs required for opportunistic VAL and PAR paths in request and reply paths.

TABLE IV: Allowed paths using *FlexVC* considering protocol deadlock in a diameter-3 Dragonfly network. Note that 4/2 VCs allows for VAL and PAR opportunistic routing in the reply path, but not for requests since there are no safe escape paths using request VCs.

	VCs (Request + Reply = overall)			
Routing	$2 \times (2/1) = 4/2$	$3/2 + 2/1 = 5/3$	$2 \times (4/2) = 8/4$	$2 \times (5/2) = 10/4$
MIN	safe	safe	safe	safe
VAL	X / opport.	opport.	safe	safe
PAR	X / opport.	opport.	opport.	safe

D. FlexVC-minCred *and congestion sensing for nonminimal adaptive routing*

The misrouting decision for nonminimal adaptive routing relies on an estimation of network congestion, typically based on a direct or indirect measurement of the buffer occupancy of the minimal and some nonminimal path [12], [13].

The base use of a fixed VC per path hop aids such congestion sensing. Consider the particular case of a Dragonfly network with traditional buffer management. Under ADV traffic, part of the traffic is sent minimally using global links that connect directly to destination groups. Such traffic only employs the first VC VC_0 in the path according to Figure 1. The remaining global links forward Valiant traffic, and employ two global VCs in a balanced way. Congestion sensing can be performed by measuring buffer occupancy, either a single VC VC_0, or overall occupancy per-port. Adaptive routing attempts to balance the total traffic load uniformly across all the global links, but buffer occupancy differs from links used for minimal (only VC_0 used) or nonminimal routing (both VC_0 and VC_1). *Per-VC* sensing may discern global links used for minimal and nonminimal routing, because their occupation of VC_0 differs even when the link load is balanced. *Per-port* sensing is less efficient as it cannot discern both types of traffic, and thus does not implicitly help identify the traffic pattern.

FlexVC allows that traffic routed minimally and nonminimally share the same buffers. For this reason, *per-VC* sensing efficiency decreases with *FlexVC* under ADV; an evaluation is presented in Section V-C. *FlexVC-minCred* is a variant designed to regain the traffic pattern identification capabilities. *FlexVC-minCred* extends the base *FlexVC* accounting separately credits corresponding to minimally- and nonminimally-routed packets. Since packet headers already contained the type of routing employed and a type of credit accounting was already required, the implementation cost only requires an additional flag per credit packet to indicate the type of routing and an additional credit counter per output port. Global ports are set as 'saturated' (in PB) and buffer occupancy is compared according to the MIN credits only, either *per-VC* or *per-port*. The evaluation in Section V-C shows that *per-port* sensing obtains the best performance because it accounts for all minimally-routed packets.

IV. Evaluation infrastructure

The proposed mechanisms have been evaluated in a Dragonfly network, which presents topology-induced path restrictions; request-reply traffic has been also considered. The network has been modelled using the cycle-accurate FOGSim network simulator [23], which operates at phit level. Combined input-output buffered routers have been modelled with separate consumption ports for requests and replies. The simulated network comprises more than 2,000 routers and 16,000 nodes. Unless otherwise noted, simulations employ the parameters in Table V. *FlexVC* results are compared to a baseline statically-partitioned mechanism and a DAMQ implementation with 75% of the buffer private; the selection of this DAMQ configuration is discussed in Section VI-C. Router speedup (frequency speedup $2\times$) refers to the increased internal frequency at which router crossbars operate, compared to the frequency at which the network links operate. Frequency speedup is typically employed to compensate the effect of suboptimal allocation mechanisms [14]. Results without frequency speedup are presented in Section VI-D. In all tests we average latency and throughput in steady state within 5 simulations, for a period of 60,000 cycles after a sufficient warm-up.

TABLE V: Simulation parameters.

Parameter	**Value**
Router size	31 ports (h=8 global, p=8 injection, 15 local)
Group size	16 routers, 128 computing nodes
System size	129 groups, 2,064 routers, 16,512 computing nodes
Latency	10/100 cycles (local/global links) 5 cycles (router pipeline)
Virtual Channels	2/1 (local/global input ports) for MIN, 4/2 for VAL & PB, 3 injection buffers
Buffer size (phits)	32 (local input buffer per VC, output buffer), 256 (injection and global input buffer per VC)
DAMQ organization	25% shared, 75% private per-VC
VC selection policy	JSQ (in *FlexVC*)
Packet size	8 phits
Router speedup	$2\times$ frequency speedup
Switching	Virtual Cut-Through
Allocator	iterative input-first separable allocator
PB routing	Threshold $T = 3$, [13]

A. Routing mechanisms and buffering

The oblivious and source adaptive mechanisms presented in Section II are modeled. Oblivious mechanisms (MIN and VAL) employ the minimum number of VCs required (2/1 and 4/2, respectively). The source-adaptive mechanism PB also requires 4/2 VCs to ensure deadlock freedom. Two variants for congestion sensing are considered: in *PB per-port* global ports are set as '*saturated*' based on the sum of the remaining credits of all the VCs in the port; by contrast, *PB per-VC* considers separately the credits in the first VC of each global port. With request-reply traffic, the first VC used in each subpath is considered. In such case, *PB per-VC* distributes the *saturation* information of both VCs, which doubles the computation and communication overhead from *PB per-port*. In all cases, the amount of VCs is doubled for request-reply traffic. *FlexVC* variants exploiting a different amount of VCs

are also modeled, and indicated explicitly. DAMQ variants employ the same overall amount of memory as in each base case, with 25% of the buffer shared and the remaining 75% private, distributed among all the VCs.

B. Traffic patterns

Three synthetic traffic patterns have been considered. Two of them employ a Bernoulli process which generates packets according to a certain injection probability, with a randomly selected destination. Under *uniform* (UN) traffic, the destination of each packet is any possible node in the network, except the source; under *adversarial* (ADV) traffic, the destination of each packet is one random node in the following group. ADV traffic requires VAL routing since MIN saturates the single inter-group link in this topology.

The third pattern BURSTY-UN is a burst traffic model that employs a Markov chain with two different states (ON/OFF) [24]. Such model has been found to accurately represent Data Center traffic [25]. In the ON state, nodes generate traffic according to a Bernoulli process with UN destination selection. The destination is unchanged during each ON burst. In the OFF state, nodes do not generate traffic. Transition probabilities from ON to OFF and OFF to ON can be configured to provide different average load and average burst length; we employ an average burst length of 5 packets.

Reactive variants of the previous patterns implement two different types of packets: requests and replies, as in [3]. Nodes generate requests following one of the previous traffic patterns, and insert a reply upon the arrival of a request. The destination of a reply is always the source node of the received request.

V. Results

This section presents results of the evaluations, comparing the performance of *FlexVC* against both a base implementation with statically-partitioned buffers and DAMQ memories; both oblivious and adaptive routing mechanisms are considered, as well as the use of request-reply traffic.

A. Oblivious routing

Figure 5 shows results with oblivious routing, MIN in the case of uniform patterns (UN, BURSTY-UN) and VAL for adversarial traffic (ADV). 2/1 VCs are employed for MIN and 4/2 for VAL. *FlexVC* results with 4/2 VCs with MIN and 8/4 VCs are also presented; note that the base mechanisms cannot exploit additional VCs for deadlock avoidance restrictions. The amount of memory per VC is constant, as presented in Table V. Under UN traffic in 5a, latency curves with MIN are similar before the saturation point. Throughput reaches 0.7 phits/node/cycle for both baseline and DAMQ. *FlexVC* increases throughput up to 0.75 thanks to HoLB mitigation. Doubling the number of VCs to 4/2 (i.e. the amount of resources required for VAL) further increases throughput to 0.85. A set of 8/4 VCs with FlexVC allows to improve that figure up to 0.9 phits/node/cycle.

The saturation throughput with BURSTY-UN in Figure 5b decreases from UN, ranging from 0.47 to 0.70, again with the best performance for *FlexVC*. Interestingly, in this case average latency curves differ well below the saturation point. For example, for a load of 0.4 phits/node/cycle, DAMQ reduces average latency in 4,7% over baseline, while *FlexVC* reduces 10,1%, 19,3% and 21,7% with 2/1, 4/2 and 8/4 VCs respectively.

Finally, under ADV traffic with VAL routing the network links represent a significant bottleneck. Nevertheless, *FlexVC* represents an improvement over the baseline and DAMQ configurations in Figure 5c. Doubling the number of VCs from the minimum set of 4/2 VCs to 8/4 lets *FlexVC* achieve a throughput of 0.49 phits/node/cycle in saturation, approaching the theoretical limit for VAL routing. Latency-wise all implementations perform similarly.

Figure 6 portrays maximum throughput for constant buffer sizes per port. We have considered four total buffer capacities: 64/256, 128/512, 192/768 and 256/1024 phits per local/global port. Upper charts refer absolute throughput in phits/node/cycle, whereas lower charts display the relative increase over Baseline with the same total buffer capacity.

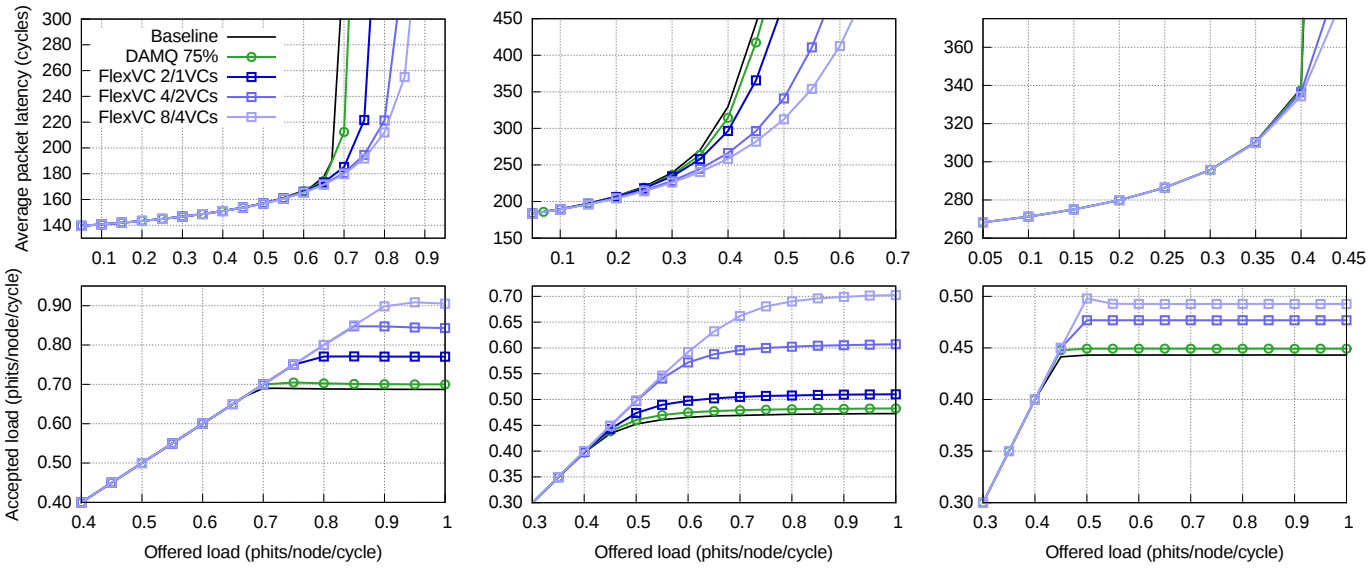
FlexVC is beneficial for all buffer sizes under all traffic patterns, increasing throughput up to 12% with the same number of VCs and up to 23% when a larger VC set is exploited (with the same memory capacity per port as the baseline configuration). The impact is higher with small buffers of 64/256 phits of total capacity per port, where the flexibility in VC use allows to overcome temporary fill-ups at certain VCs. Likewise, *FlexVC* has the highest impact under BURSTY-UN because the traffic bursts are more prone to congesting isolated VCs. Interestingly, in all the configurations it outperforms the DAMQ implementation, which achieves a very modest improvement over the baseline.

Conclusion: *FlexVC* improves throughput up to 12% with the same amount of resources and 23% by exploiting more buffers (typically already provisioned to support VAL routing), in both cases outperforming DAMQ organizations. For a given buffer size per port, *FlexVC* is similar to or more efficient than DAMQ buffers, particularly when using multiple shallow queues and UN traffic patterns.

B. Request-reply traffic

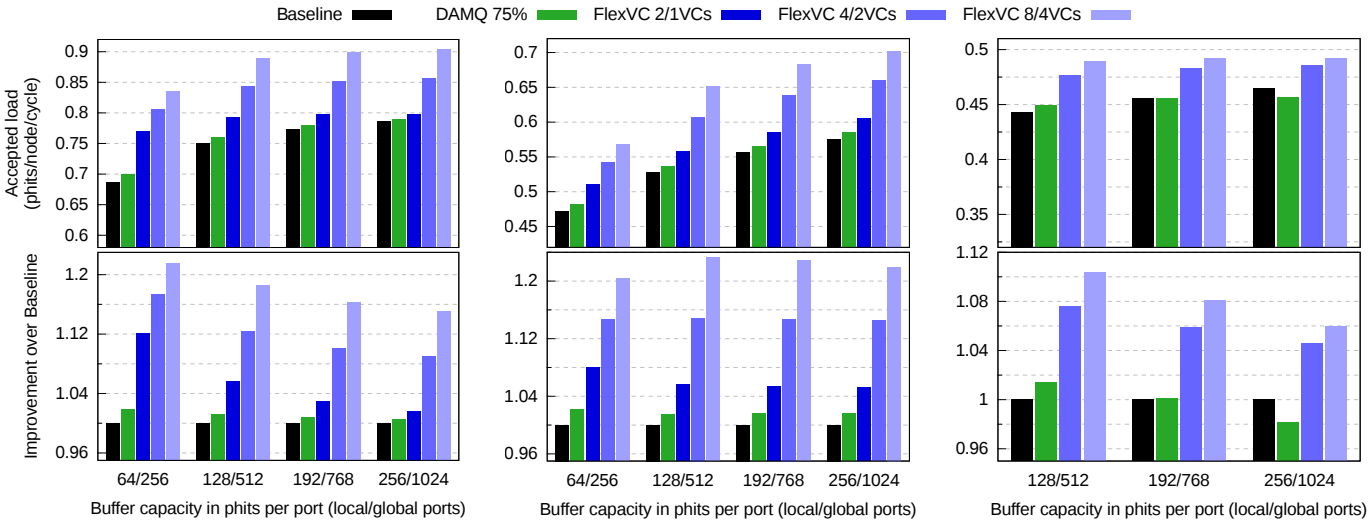
Figure 7 displays latency and throughput modeling request and reply messages as described in Section IV-B. In this case, the minimum number of VCs is 4/2 (2/1+2/1) for MIN and 8/4 for VAL. All latency curves are similar below the saturation point; however, under UN the base implementations (MIN and DAMQ) present congestion after reaching the saturation point. The use of *FlexVC* with the same 4/2 VCs presents both higher peak throughput (which grows from 0.70 to 0.75 in MIN) and a less pronounced congestion effect. Throughput at maximum load increases 24.6% from MIN and 17.9% from DAMQ.

The use of more VCs increases peak throughput and reduces congestion. *FlexVC* with 6/4 VCs arranged in 4/3+2/1 virtually removes congestion and reaches 0.85, which represents a remarkable 51.8% increase over MIN. Other configurations present intermediate results. It is noteworthy that throughput



(a) UN with MIN routing. (b) BURSTY-UN with MIN routing. (c) ADV with VAL routing.

Fig. 5: Latency and throughput under uniform (UN), uniform with bursts (BURSTY-UN) and adversarial traffic (ADV), with oblivious routing: MIN for UN/BURSTY-UN and VAL for ADV.



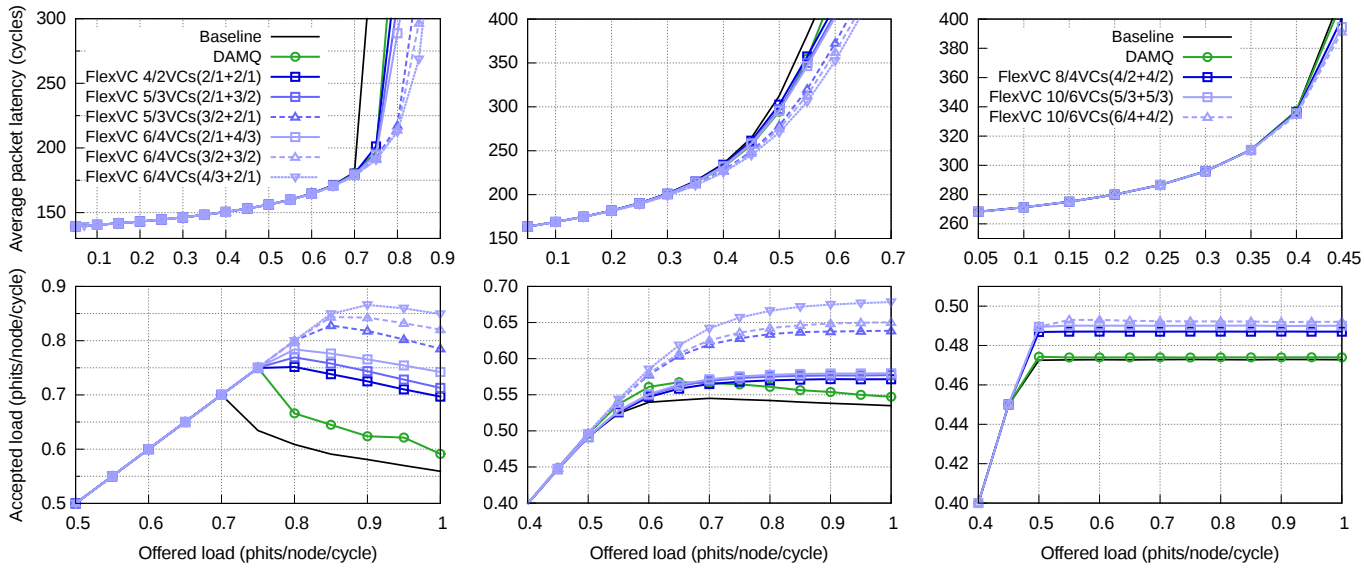
(a) UN with MIN routing. (b) BURSTY-UN with MIN routing. (c) ADV with VAL routing.

Fig. 6: Absolute and relative maximum throughput under uniform (UN), burst uniform (BURSTY-UN) and adversarial traffic (ADV), with oblivious routing: MIN for UN/BURSTY-UN and VAL for ADV.

in Figure 7a is not sorted by the overall amount of VCs, but by the amount of VCs in the request subpath (which are assigned the same line marker): The three *FlexVC* configurations with 2/1 VCs in the request subpath are in the bottom, two configurations with 3/2 are in the middle, and the best configuration employs 4/3 VCs for requests. While only 2/1 VCs are required in each subpath for MIN routing, the allocation of *additional VCs* at the start of the request subpath makes them available for both requests and replies, making a more efficient use of them.

BURSTY-UN does not present congestion except with the DAMQ implementation (and with a much smaller impact). Results are similar to those in Figure 5b, although the difference in latency between implementations is less significant here. *FlexVC* with 4/2 overall VCs is slightly better than the baseline and the DAMQ. Interestingly, the number of VCs employed in the request subpath has a bigger impact than the total number of VCs: the three configurations with 2/1 VCs for requests are at the bottom, and any of the other combinations (which assign 3/2 or 4/3 VCs to the requests) perform noticeably better.

Conclusion: *FlexVC* mitigates the congestion that occurs with long request-reply paths, increasing throughput between 3% (ADV traffic) and 24.6% (UN) with the same buffers and up to 51.8% by exploiting more resources. In such



(a) UN with MIN routing. (b) BURSTY-UN with MIN routing. (c) ADV with VAL routing.

Fig. 7: Latency and throughput under uniform (UN), uniform with bursts (BURSTY-UN) and adversarial (ADV) traffic, modeling request-reply dependencies. Oblivious routing (MIN for UN and BURSTY-UN, and VAL for ADV). *FlexVC* configurations with the same amount of *request* VCs are assigned the same marker. BURSTY-UN curves share the same legend as UN curves.

configurations, it is more efficient to add *additional VCs* at the begin of the request subpath than allowing longer safe paths in both requests and replies.

C. Adaptive routing

Figure 8 presents results of source adaptive routing using Piggyback (PB) with request-reply traffic and the *-per-port* and *-per-VC* variants. Results of in-transit adaptive routing are omitted for brevity. Base configurations require 4/2+4/2=8/4 VCs, while *FlexVC* variants employ 6/3 VCs arranged as 4/2+2/1, according to the findings in Section III-C.

The impact of using *PB-per-port* or *PB-per-VC*, without *FlexVC*, is analyzed first. Figure 8a shows that PB presents some congestion under UN traffic, and Figure 8c reveals that *per-port* sensing performs worse than *per-VC* sensing under ADV traffic, as discussed in Section III-D. As discussed in Section III-D, *per-VC* implicitly identifies the traffic pattern by analysing the amount of traffic routed minimally.

Under UN traffic, all the four variants of *FlexVC* clearly outperform the baseline PB, avoiding congestion providing up to 20.4% saturation throughput increase and reduced latency. However, the two variants of *FlexVC* without differentiated credit tracking perform worse than the base VC management under ADV traffic, providing higher base latency and reduced throughput. In such case, only *FlexVC-minCred* using *per-port* sensing is competitive with the baseline. With *FlexVC* different flows employ a common set of VCs, so *FlexVC-minCred* with *per-VC* sensing provides less accuracy and lower throughput.

Conclusion: PB with *FlexVC-minCred* provides a 20.4% throughput increase and noticeable latency reductions with a 25% VC reduction, employing additional credit counters to properly identify traffic patterns.

VI. Implementation Discussion

A. VC selection function

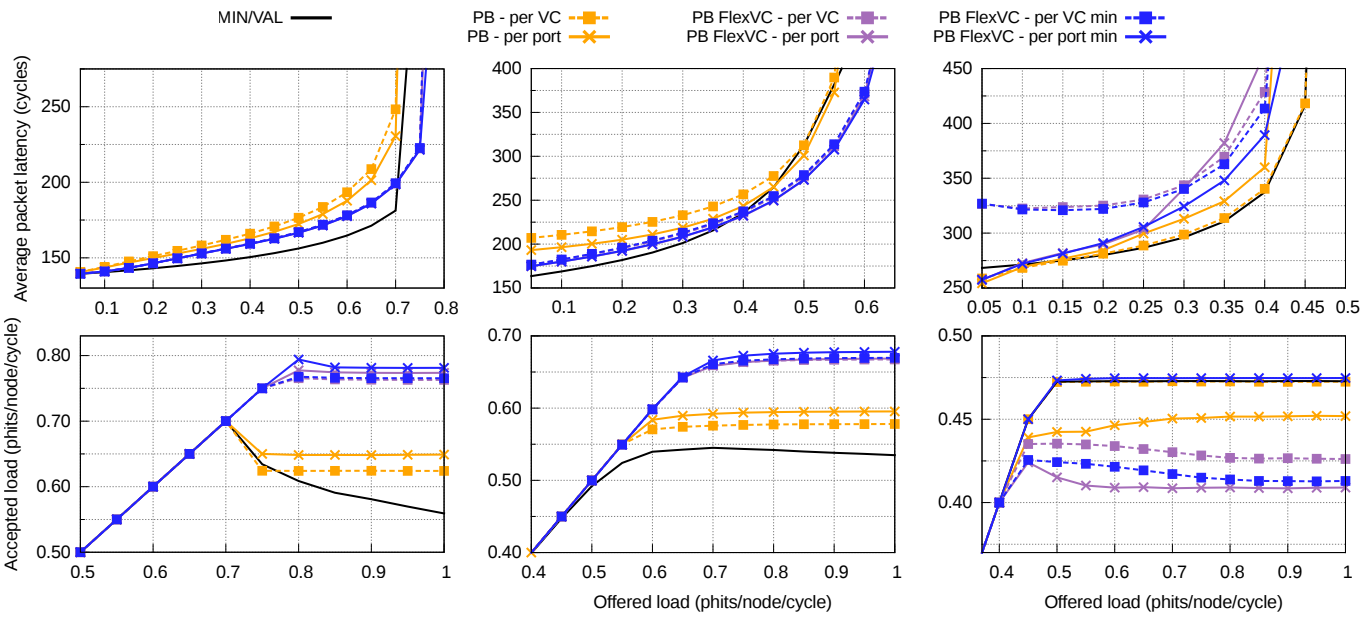
Previous results employ a *JSQ* (Join the Shortest Queue) VC selection function. Our experiments show that the function used has no appreciable impact in not request-reply traffic as evaluated in Sections V-A and V-C. Figure 9 presents results of request-reply UN traffic under maximum load and MIN routing, similar to the results in Figure 7a.

As discussed in Section V-B, the amount of VCs in the request sub-path is the factor that most determines performance; the VC selection function has a small impact. For each configuration, *JSQ* provides the best performance on average, since it balances the utilization of all VCs, being closely followed by *highest-VC*. Interestingly, a *Random* policy is also competitive in many configurations. *Lowest-VC* consistently provides the lowest performance. *Lowest-VC* tends to saturate lower-index VCs, more used in the first hops of requests, what eventually restricts injection. A side effect of this restriction (not observed in this figure) is that *Lowest-VC* also presents lower peak throughput. In any case, the difference between policies under maximum load varies less than 3.4% in average.

Conclusion: The VC selection function has small impact on performance. The best policies are *JSQ* and *highest-VC*.

B. Cost and complexity of FIFO and DAMQ

Amount of memory FIFO buffers are typically implemented through circular buffers using SRAM [26]. DAMQs also rely on SRAM to store data, but need to store pointers for linked lists [4] or alternative control structures (discussed in Section VII). Buffer overhead for DAMQs is small but not negligible. For a 4KB DAMQ with 8-byte phits (512 phits per DAMQ), pointers need to be 9 bits long and the overhead



(a) Uniform (UN). (b) Uniform with bursts (BURSTY-UN). (c) Adversarial (ADV).

Fig. 8: Latency and throughput under request-reply traffic, using Piggyback source adaptive routing. MIN and VAL are the reference for UN/BURSTY-UN and ADV, respectively. 4/2+4/2 VCs are used in baseline PB and VAL, 4/2+2/1 in *FlexVC* PB and 2/1+2/1 in MIN.

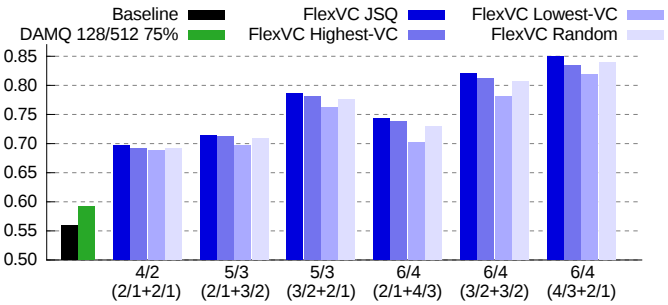


Fig. 9: Throughput under UN request-reply traffic at 100% load, with multiple VC selection functions and amount of VCs. MIN routing.

is roughly 576 bytes (14% increase). Considering per-packet pointers (as in [27]) with 8-phit packets, this overhead shrinks to 1.6%, but flexibility with variable packet size is reduced.

Access latency The indirections required in DAMQs increase access latency. The implementation in [28] adds three cycles to read or write access latency. Choi *et al.* measure in [27] slowdowns in packet access time ranging 59-77% for different DAMQ implementations. Note that no DAMQ slowdown has been considered in our simulations, presenting optimistic results for DAMQs.

Flow control Credit management in DAMQs is more complex than using FIFOs because senders need to track both per-port and per-VC occupancy, particularly when using reservations per VC and adaptive routing. *FlexVC* with oblivious routing relies on the original and simpler per-VC credit management. *FlexVC-minCred* increases the number of counters, being closer to DAMQs.

Router complexity The complexity and latency of (de)muxers and allocators grows with the number of buffers (shared or individual) implemented per port. While more buffers are useful to avoid HoLB, they make these elements more complex. *FlexVC* can exploit buffers which would be required anyhow for deadlock avoidance when using long paths, so in practice it does not need to increase the number of buffers. *FlexVC* requires a VC selection function. Results in Section VI-A show that JSQ is typically the most performant. Such function can be easily implemented with a stage of comparators which select the VC with the highest credit count.

Conclusion: The implementation of *FlexVC* is simpler than DAMQs, considering both the buffer organization and latency, and the complexity of the other elements of the router.

C. Impact of reserved space in DAMQs

Figure 10 portrays throughput achieved with DAMQ buffers under UN traffic, varying the private buffer size per VC. The system employs MIN routing, 1 global VC and 2 local VCs, with 512 phits in global ports and 128 phits shared among the VCs in local ports. Each line represents a different amount of reserved private buffering per VC, in packets; since packets are 8 phits, possible values range from 0 (all the DAMQ buffer is shared) to 8 (equivalent to statically partitioned buffers: 64 phits are private per VC, no shared buffering).

With no private reservation (0 private phits), the system presents deadlock: when VC_0 is assigned all the memory in several ports, packets cannot advance to VC_1 in the next buffer, creating a cyclic dependency between VCs. Deadlock is only observed at saturation loads but may occur for any traffic load. With 16 private phits per local port (25% of the overall space is private) we observe congestion. An analysis of the simulation data shows that most of the buffering space is again

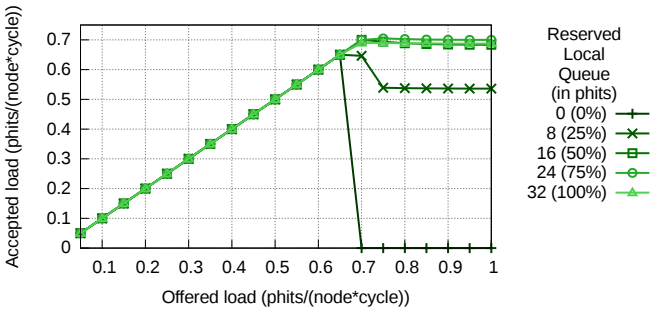


Fig. 10: Throughput under UN traffic with MIN routing, using DAMQ buffers with different buffer reservation per VC.

dynamically assigned to VC_0, not leaving enough amount of buffer in VC_1 to cover link round trip time. The optimal result is obtained with 75% of private buffering, which is only slightly better than statically partitioned buffers, as already observed in Figure 6a. Analogous tests were conducted with other buffer sizes, obtaining in all cases the best performance with around 75% of the overall space being private.

Conclusion: DAMQ buffers require most memory private per VC to avoid congestion with distance-based deadlock avoidance, what reduces their effectiveness.

D. Impact of router speedup

Evaluations from Section V-A have been repeated using routers without speedup. Latency and throughput results are omitted for brevity. Maximum throughput for different buffer sizes is presented in Figure 11. Base throughput is significantly lower without speedup because of HoLB. DAMQs show little benefit from the base case with fixed-order VC management under uniform patterns, and its use is detrimental under ADV for every configuration. Under ADV traffic the impact of buffer organization with the same number of buffers is very low, less than 3%. Doubling the size of the VC set yields a higher improvement of more than 7%. Under both uniform patterns, *FlexVC* performs consistently better than the baseline and the DAMQ. *FlexVC* with the same number of VCs improves throughput less than 7.2%. Increasing the number of VCs targets the main problem (HoLB) and improves throughput up to 37.8% from the base case.

Conclusion: *FlexVC* is more efficient in systems without speedup because they suffer more HoLB, presenting throughput increases up to 37.8% with the same buffer space.

E. FlexVC with other topologies

This paper only evaluates *FlexVC* in a Dragonfly topology, which has link-type restrictions as discussed in Sections II and III-C. Because of such restrictions, Dragonflies only require a maximum of 2 VCs for MIN routing. This equals the requirement for diameter-2 networks, such as Slim Flies or demi-Proyective Networks which do not have any link-type restrictions. *FlexVC* buffer requirements for such networks has been already considered in Tables I and II. For FB's two alternatives exist: DOR routing without buffer requirements for deadlock-freedom, or adaptive routing with the base distance-based deadlock-avoidance. In the latter case, it requires as many VCs as the number of dimensions; *FlexVC* can be used to improve performance similarly to the results presented.

The applicability of *FlexVC-minCred* to support nonminimal adaptive routing in alternative topologies has not been explored yet, and is left for future work.

VII. Related work

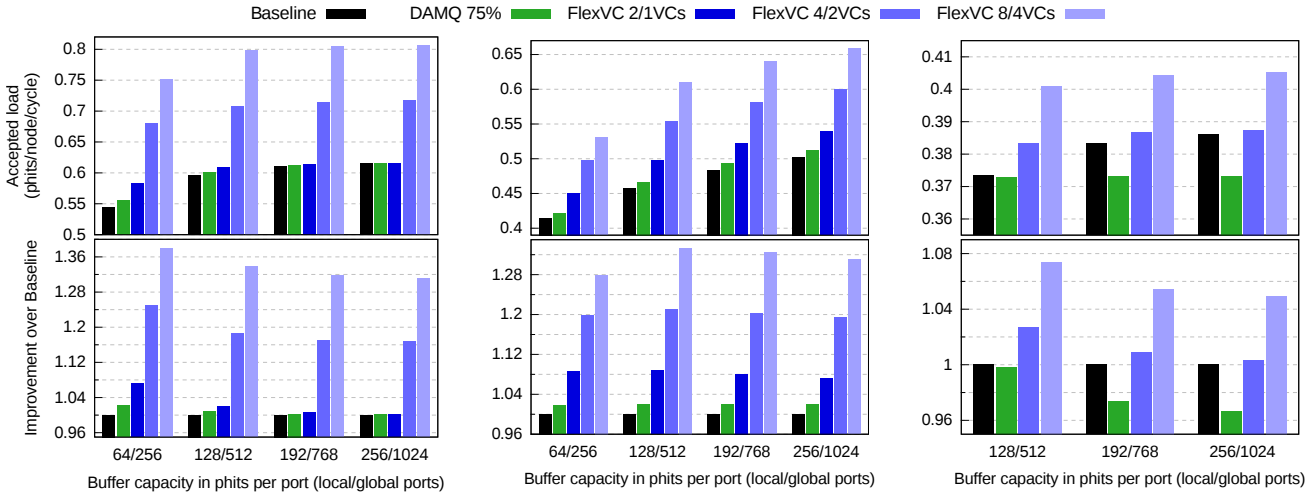
Distance-based deadlock avoidance: Seminal works on distance-based deadlock avoidance in store-and-forward networks were introduced by Günther [1] and Gopal [29]. Several current systems employ such mechanisms (or a variation of them), such as IBM PERCS [30] or Cray Cascade [3], and have been extended to commodity InfiniBand [31]. In these proposals, the amount of required VCs increases with the maximum path length. Therefore, supporting nonminimal paths, in-transit adaptive routing and avoiding protocol deadlock significantly increases buffer requirements.

Opportunistic Local Misrouting (OLM, [32]) violates the base order of increasing VC index only for certain local hops in Dragonflies. However, it does not fully exploit the available VCs to reduce HoLB, does not consider protocol-deadlock and is not exploited to simplify buffer management.

Alternative routing mechanisms: Distance-based deadlock avoidance allows to deal separately with deadlock avoidance and routing. However, this mechanism is not supported in all network technologies. For example, Infiniband switches select the output VC (denoted Virtual Lane, VL) based on the input and output ports and the packet service level (which does not change during the path). For this reason, most routing mechanisms in Infiniband (such as LASH [33], SSSP [34] and DF-SSSP [35]) assign a single VC to a complete path from source to destination. These routing protocols typically calculate sets of paths with a reduced amount of cyclic dependencies, so that the VL assignment phase result fits in a low amount of VLs. NUE routing [36] provides better results by combining path computing and VL assignment in a single calculation, but still assigns VLs to complete paths. Schneider *et al* extend distance-based deadlock avoidance to Infiniband in [31], but still determine a single fixed output VC per packet.

Avoiding Protocol deadlock: The typical mechanism employed to avoid protocol deadlock in lossless networks relies on two virtual networks, one for requests and other for replies (e.g., as implemented in Alpha 21364). This doubles the buffering requirements. In [37] the authors introduce a bubble-based deadlock avoidance protocol for on-chip networks which does not employ separate networks. Instead, their mechanism shares router buffers but employs a separate bubble for each type of message. Similarly, our FlexVC proposal avoids a strict separation in different virtual networks, but making sure that replies have exclusive buffers to avoid deadlock.

Buffer sizing and organization: Buffers need to cover the link round-trip latency to permit lossless flow control. This is the minimum size per VC considered in our simulations. However, larger buffers are typically implemented to properly



(a) UN with MIN routing. (b) BURSTY-UN with MIN routing. (c) ADV with VAL routing.

Fig. 11: Absolute and relative maximum throughput figures under uniform, burst uniform and adversarial traffic patterns without router speedup, using oblivious routing (MIN in the case of UN/BURSTY-UN, VAL under ADV traffic).

deal with congestion bursts of traffic. Yébenes *et al.* [38] reduce HoLB in Dragonflies by implementing multiple buffers per VC; however, this multiplies the complexity and buffer requirements, whereas our proposal allows to exploit the buffers which would be already available for longer paths.

Dynamically allocated shared buffer implementations (DAMQ [4]) employ a single memory shared between the different VCs of a port. This improves performance by sharing the complete amount of buffering on demand and better tolerating micro-bursts of traffic. Different designs for dynamically shared buffers include linked-lists [4], self-compacting buffers [39] or Fully-connected circular buffers [40]. Alternative designs allow for a variable number of VCs, what helps reduce HoLB particularly under adaptive routing [27], [41].

These dynamically allocated buffers present two main drawbacks. First, implementations are more complex, leading to an increase in area, power and delay. Second, dynamically shared buffers suffer congestion when a single VC occupies the complete buffer space, as studied in Section VI-C. Different implementations with reserved space per VC have been presented in [42], [43]. In particular, DAMQ buffers with reserved space per VC are employed in the Tianhe-2 network switch [20]. Alternative approaches suggest to extend flow-control to detect such congestion and regulate buffer usage [44], [45], but increase buffer complexity even further.

VIII. Conclusions

This work has identified some of the main limitations of distance-based deadlock avoidance in low-diameter networks, mainly HoLB and inefficient utilization of router buffers. DAMQ buffers increase design complexity and only partially mitigate the previous limitations. By contrast, *FlexVC* relies on a simple design with statically partitioned buffers and on opportunistic routing, relaxing VC usage restrictions. *FlexVC* partially decouples the amount of VCs from deadlock avoidance, allowing for longer network paths with a low amount of resources, with up to 50% memory reductions.

Evaluations in a Dragonfly network show *FlexVC* improves throughput 12% compared to a base oblivious case with the same buffering, and reduces latency under traffic bursts. Furthermore, *FlexVC* leverages additional VCs typically provisioned to support VAL routing, reaching 23% improvements. Without router speedup, this figure rises up to 37.8%. Increasing the amount of VCs mitigates the congestion that appears with long oblivious paths used to avoid protocol deadlock.

However, reusing VCs for packets in different hops of their paths complicates the identification of adversarial traffic patterns for adaptive routing decisions. *FlexVC-minCred* handles credits separately for packets traveling minimally/nonminimally, properly identifying the communication pattern for adaptive routing and outperforming alternative designs. With 25% buffer reduction, this mechanism provides a 20.4% throughput increase and noticeable latency reductions over the base PB implementation. Overall, *FlexVC* is a simple design that maximizes buffer utilization and outperforms state-of-the-art and more complex alternatives.

Acknowledgment

This work has been supported by the Spanish Government (grant SEV2015-0493 of the Severo Ochoa Program), the Spanish Ministry of Economy, Industry and Competitiveness (contracts TIN2015-65316), the Spanish Research Agency (AEI/FEDER, UE - TIN2016-76635-C2-2-R), the Spanish Ministry of Education (FPU grant FPU13/00337), the Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272), the European Union FP7 programme (RoMoL ERC Advanced Grant GA 321253), the European HiPEAC Network of Excellence and the European Union's Horizon 2020 research and innovation programme (Mont-Blanc project under grant agreement No 671697).

REFERENCES

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

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

[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 High Performance Computing, Networking, Storage and Analysis*, 2012.

[4] Y. Tamir and G. L. Frazier, "Dynamically-allocated multi-queue buffers for VLSI communication switches," *IEEE Trans. Comput.*, vol. 41, no. 6, pp. 725–737, Jun. 1992.

[5] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *International Symposium on Computer Architecture*. ACM, 2007, pp. 126–137.

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

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

[8] C. Camarero, C. Martínez, E. Vallejo, and R. Beivide, "Projective networks: Topologies for large parallel computer systems," *IEEE Transactions on Parallel and Distributed Systems (To appear)*, 2017.

[9] M. Besta and T. Hoefler, "Slim Fly: A cost effective low-diameter network topology," in *Int. Conf. High Performance Computing, Networking, Storage and Analysis*. Piscataway, NJ, USA: IEEE Press, 2014, pp. 348–359.

[10] M. Valerio, L. E. Moser, and P. M. Melliar-Smith, "Using fat-trees to maximize the number of processors in a massively parallel computer," in *Intl Conf on Parallel and Distributed Systems*, 1993, pp. 128–134.

[11] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. Hoefler, "Cost-effective diameter-two topologies: Analysis and evaluation," in *Intl Conf High Performance Computing, Networking, Storage and Analysis*, New York, NY, USA, 2015, pp. 36:1–36:11.

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

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

[14] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[15] 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 *Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip (INA-OCMC)*, 2014.

[16] J. Won, G. Kim, J. Kim, T. Jiang, M. Parker, and S. Scott, "Overcoming far-end congestion in large-scale networks," in *Intl Symposium on High Performance Computer Architecture (HPCA)*, Feb 2015, pp. 415–427.

[17] 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.

[18] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The BlackWidow high-radix Clos network," in *Intl Symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 16–28.

[19] N. Chrysos, C. Minkenberg, M. Rudquist, C. Basso, and B. Vanderpool, "SCOC: High-radix switches made of bufferless clos networks," in *Intl Symp. on High Performance Computer Architecture*, 2015, pp. 402–414.

[20] X.-K. Liao, Z.-B. Pang, K.-F. Wang, Y.-T. Lu, M. Xie, J. Xia, D.-Z. Dong, and G. Suo, "High performance interconnect network for tianhe system," *Journal of Computer Science and Technology*, vol. 30, no. 2, pp. 259–272, 2015.

[21] J. Duato, "A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 8, pp. 841–854, Aug 1996.

[22] Y. Ho Song and T. M. Pinkston, "A progressive approach to handling message-dependent deadlock in parallel computer systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 3, pp. 259–275, Mar. 2003.

[23] M. García, P. Fuentes, M. Odriozola, E. Vallejo, and R. Beivide. (2014) FOGSim Interconnection Network Simulator. University of Cantabria. [Online]. Available: http://fuentesp.github.io/fogsim/

[24] A. Adas, "Traffic models in broadband networks," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 82–89, Jul 1997.

[25] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 92–99, Jan. 2010.

[26] Texas Instruments, *FIFO Architecture, Functions, and Applications*, 1999.

[27] Y. Choi and T. M. Pinkston, "Evaluation of queue designs for true fully adaptive routers," *J. Parallel Distrib. Comput.*, vol. 64, no. 5, pp. 606–616, May 2004.

[28] G. L. Frazier and Y. Tamir, "The design and implementation of a multiqueue buffer for vlsi communication switches," in *Intl Conference on Computer Design*, Oct 1989, pp. 466–471.

[29] I. S. Gopal, "Interconnection networks for high-performance parallel computers," I. D. Scherson and A. S. Youssef, Eds. Los Alamitos, CA, USA: IEEE Computer Society Press, 1994, ch. Prevention of Store-and-forward Deadlock in Computer Networks, pp. 338–344.

[30] 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.

[31] T. Schneider, O. Bibartiu, and T. Hoefler, "Ensuring deadlock-freedom in low-diameter InfiniBand networks," in *IEEE Hot Interconnects*, 2016.

[32] 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)*, 10 2013.

[33] T. Skeie, O. Lysne, and I. Theiss, "Layered shortest path (LASH) routing in irregular system area networks," in *Intl Parallel and Distributed Processing Symposium*, Washington, DC, USA, 2002, pp. 194–.

[34] T. Hoefler, T. Schneider, and A. Lumsdaine, "Optimized routing for large-scale InfiniBand networks," in *2009 17th IEEE Symposium on High Performance Interconnects*, Aug 2009, pp. 103–111.

[35] J. Domke, T. Hoefler, and W. E. Nagel, "Deadlock-free oblivious routing for arbitrary topologies," in *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, May 2011, pp. 616–627.

[36] J. Domke, T. Hoefler, and S. Matsuoka, "Routing on the Dependency Graph: A New Approach to Deadlock-Free High-Performance Routing," in *Symposium on High-Performance Parallel and Distributed Computing (HPDC'16)*, Jun. 2016.

[37] R. Wang, L. Chen, and T. M. Pinkston, "Bubble coloring: Avoiding routing- and protocol-induced deadlocks with minimal virtual channel requirement," in *International Conference on Supercomputing*, 2013, pp. 193–202.

[38] P. Yébenes, J. Escudero-Sahuquillo, P. J. García, and F. J. Quiles, "Straightforward solutions to reduce HoL blocking in different Dragonfly fully-connected interconnection patterns," *The Journal of Supercomputing*, pp. 1–23, 2016.

[39] J. Park, B. O'Krafka, S. Vassiliadis, and J. Delgado-Frias, "Design and evaluation of a DAMQ multiprocessor network with self-compacting buffers," in *Supercomputing '94., Proceedings*, Nov 1994, pp. 713–722.

[40] N. Ni, M. Pirvu, and L. Bhuyan, "Circular buffered switch design with wormhole routing and virtual channels," in *Intl Conf on Computer Design*, Oct 1998, pp. 466–473.

[41] C. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. Yousif, and C. Das, "ViChaR: A dynamic virtual channel regulator for network-on-chip routers," in *Intl Symp. on Microarchitecture*, 2006, pp. 333–346.

[42] J. Liu and J. G. Delgado-Frias, "A shared self-compacting buffer for network-on-chip systems," in *2006 49th IEEE International Midwest Symposium on Circuits and Systems*, vol. 2, Aug 2006, pp. 26–30.

[43] H. Zhang, K. Wang, J. Zhang, N. Wu, and Y. Dai, "A fast and fair shared buffer for high-radix router," *Journal of Circuits, Systems and Computers*, vol. 23, no. 01, p. 1450012, 2014.

[44] D. U. Becker, "Adaptive backpressure: Efficient buffer management for on-chip networks," in *Intl Conf. on Computer Design*. Washington, DC, USA: IEEE Computer Society, 2012, pp. 419–426.

[45] H. Zhang, K. Wang, Z. Pang, L. Xiao, Q. Dou, and Y. Yuan, "An area-efficient DAMQ buffer with congestion control support," *Journal of Circuits, Systems and Computers*, vol. 25, no. 10, p. 1650125, 2016.