Analysis and improvement of Valiant routing in low-diameter networks

Mariano Benito*†, Pablo Fuentes*, Enrique Vallejo* and Ramón Beivide*
*University of Cantabria, Spain
Email: {*mariano.benito, pablo.fuentes, enrique.vallejo, ramon.beivide*}@unican.es
†Recore Systems BV, Enschede, The Netherlands

***Abstract*—Valiant routing randomizes network traffic to avoid pathological congestion issues by diverting traffic to a random intermediate switch. It has received significant attention in recently proposed high-radix, low-diameter topologies, which are prone to congestion issues. It has been implemented obliviously, or as the basis of some non-minimal adaptive routing algorithms.**

An analysis of the original mechanism identifies two potential improvements regarding the selection of the intermediate switch. First, when traffic is local the randomization introduced by Valiant results in unnecessarily long paths. Instead, the introduced *Restricted Valiant* routing randomizes traffic within a local partition, avoiding congestion and generating shorter paths. Second, in certain cases the path to the selected random intermediate node can be blocked; a version *with recomputation* selects a new random intermediate node as long as the associated path remains stalled.

The proposals are evaluated by simulation in a state-of-the-art Dragonfly network with different traffic patterns. Results show that Restricted Valiant is highly effective in cases of local traffic, with a small improvement under global patterns. Valiant with recomputation increases injection, further reducing average latency and increasing throughput. However, the higher injection increases congestion effects in some cases. Such problem is emphasized when more injection buffers are added, because of the increased pressure on the interconnect. Overall, the results are very relevant for routing in high-radix networks and might constitute the basis for other adaptive routing algorithms.

***Index Terms*—Valiant, routing, randomization.**

I. Introduction

Datacenter interconnection networks traditionally rely on some form of tree or Folded-Clos topology. The implementation cost and energy consumption restrictions (highly demanding for Exascale proposals) suggest the use of scalable topologies with low-diameter based on high-radix switches [1], such as Flattened Butterflies [2], Dragonflies [3] or Slim Flies [4]. Compared to the traditional Folded-Clos, these low-diameter topologies present low average distance between nodes in order to reduce latency. They also employ a lower number of switches and links for a given network size, what leads to a lower power consumption and lower installation costs while achieving high scalability for the given diameter [5].

Multiple oblivious routing protocols have been designed for these networks. *Minimal* routing achieves optimal latency under a uniform traffic pattern because the length of the path is equal to or lower than the network diameter. However, these topologies have low diversity of minimal paths, which are heavily congestion-prone under adversarial traffic patterns that gather all the traffic in only a fraction of the available links on the network. Non-minimal oblivious routing mechanisms avoid network congestion and improve performance under adversarial traffic patterns by making use of longer non-minimal routes: Valiant routing [6] randomizes the traffic sending packets first to a randomly selected *intermediate* switch, and then minimally to the actual destination. This balances the use of the network links, but doubles the longest path in the network.

This work analyzes some limitations of Valiant routing and proposes improvements. First, selecting the intermediate switch among all nodes of the network generates long paths; for local traffic (this is, depending on the destination of each packet), the set of allowed intermediate switches can be restricted without sacrificing randomness or performance. Another issue of Valiant routing is that, after an intermediate switch is assigned to a packet, the output port towards the intermediate switch may be temporarily unavailable due to stalled packets. In this paper, we propose the idea of recalculating the intermediate switch when the output port towards it is unavailable, enabling a faster dispatch of the packet. This helps frames to avoid temporarily congested areas of the network and improve the performance.

In brief, the main contributions of this paper are the following:

- A modified version of Valiant routing denoted *Restricted Valiant* (RVAL), which shortens the path of packets sent between nodes in the same network partition without compromising randomization.
- Valiant *with recomputation* of the intermediate switch, which mitigates Head-of-Line Blocking (HoLB) by selecting a new random intermediate switch when the path to the previous one is blocked.
- An evaluation of the proposed variations under different traffic patterns in a dragonfly network. Results show that under local traffic patterns, RVAL reduces latency up to 69.9% and maximizes throughput; similarly, the recomputation variant further reduces latency up to 11.4%

This work has been supported by the Spanish Ministry of Economy, Industry and Competitiveness under contract TIN2016-76635-C2-2-R (AEI/FEDER, UE), the European HiPEAC Network of Excellence and The Mont-Blanc project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671697.

and increases throughput up to 5.9%. The evaluation identifies that using a greater number of injection buffers increases network congestion and reduces throughput, except for a hot-region pattern which suffers from HoLB at injection.

The remainder of this paper is organized as follows. Section II describes the background. Section III introduces our proposed variants of the Valiant mechanism. Section IV describes the infrastructure employed in our evaluation, and Section V presents the results from our evaluation. Section VI explores related works. Finally, Section VII remarks the main conclusions from our work and presents future lines of research.

II. Background

This section first presents the idea of Valiant routing. The paper evaluates different modifications to Valiant routing in a Dragonfly topology. Therefore, the Dragonfly topology and Valiant routing applied to the Dragonfly are introduced next.

A. Valiant routing

The original randomization-based Valiant routing mechanism [6], [7] obtains $\mathcal{O}(\log N)$ packet delivery time for a given permutation of traffic in a network of N processors arranged as a hypercube. It is based on two phases: Phase A sends each packet to a randomly selected network switch, whereas Phase B sends the packet from this *intermediate* switch to the final destination. The original implementation of Phase A in [6] follows a dimension-order process in which the (only) link in each dimension of the hypercube in each current switch is traversed or not randomly with the same probability; this is completely equivalent to selecting a random intermediate destination at injection time, but requires less bookkeeping since the intermediate destination is not recorded in the packet header. The mechanism is extended to *square mesh*, *d-way shuffle* and *shuffle-exchange* graphs [8], obtaining the same $\mathcal{O}(\log N)$ bound for the delivery of all packets.

Valiant routing is completely *oblivious*: the path for each packet is selected randomly, but it does not depend on the status of the network (congestion) or the destinations of the packets sent by the rest of the nodes (traffic pattern).

Valiant routing has also been proposed for other high-radix topologies to avoid pathological congestion issues, such as the Flattened Butterfly [2], the HyperX [9], the Dragonfly [3], the Slim Fly [4] or the Projective networks [10]. It is also the basis of multiple non-minimal adaptive routing algorithms, which select between minimal and Valiant paths (such as UGAL [11], PiggyBack [12], Progressive Adaptive Routing [12] or On-the-Fly Adaptive Routing [13]). The application to the Dragonfly topology is described next.

B. Dragonfly topology

The Dragonfly [3] is a low-diameter topology based on high-radix switches deployed following a hierarchical direct layout. The first level comprises fully connected groups of switches conforming a virtual high-radix switch. These groups can be connected using different second level interconnection patterns. For this work we assume a *canonical dragonfly* which uses a complete graph for this second level.

This topology can be described using the following parameters:

- p: number of nodes connected on each switch
- a: number of switches on each group
- h: number of *global* links on each switch

The best and worst traffic patterns for this topology are identified in [3]. *Uniform* random traffic (UN) sends all packets to a random destination among the whole network. *Adversarial* traffic (ADV) in this network occurs when all the nodes in one group send traffic to nodes in the following group; under *minimal* routing described next, traffic is concentrated in a single global link which joins both groups, generating congestion.

Multiple routing protocols have been designed for Dragonfly topology. A baseline *minimal* routing requires at most as much hops as the diameter of the network. Packets are sent first to the destination group, and then to the destination switch. This routing protocol is suitable for UN traffic pattern which distributes destinations uniformly. Indeed, this routing is optimal under the mentioned traffic pattern. However, as described before, it concentrates most of the traffic in a few links under *Adversarial* traffic patterns, leading to high congestion and poor performance.

Valiant routing avoids in-network congestion caused by adversarial patterns by randomizing traffic: packets are sent first minimally to a random intermediate switch and then minimally again to the final destination. Valiant makes the traffic uniform and balances it over the network links. However, this doubles the longest network path, what penalizes latency and maximum throughput in absence of congestion.

III. Restricted Valiant and Valiant with recomputation

In this section we introduce two modifications to the original Valiant routing: Restricted Valiant, which restricts the selection of the intermediate switch, and Valiant with recomputation, which determines an alternative intermediate switch when traffic cannot be injected.

A. Restricted Valiant (RVAL)

The original definition of Valiant routing [7] randomizes perfectly the paths of the packets in a hypercube network, balancing the use of the resources regardless of the traffic pattern. However, selecting a random destination among *all* the switches of the network may lead to unnecessary packet diverting and increased path length for certain topologies. For example, this may occur when the network is hierarchical (such as the Dragonfly or multilevel fat-tree) or consists of multiple orthogonal dimensions (such as a HyperX network), and both the source and destination are confined to the same partition of the hierarchy (group, *pod*) or the same subset of dimensions. In such cases, selecting a random destination outside the partition implies leaving and returning to the original partition, what increases the path length without contributing to congestion avoidance.

Restricted Valiant limits the selection of the intermediate node to the local partition when the source and destination nodes are in the same partition. If source and destination are in different partitions, Restricted Valiant behaves as the original Valiant routing. The definition of a network partition is topology-dependent.

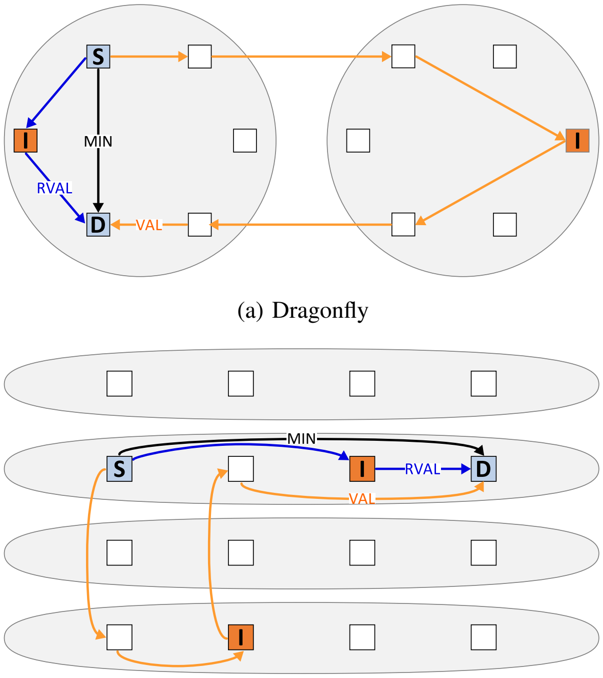
Considering a Dragonfly network, Restricted Valiant can be applied to packets with source and destination in the same group, this is, traffic which is internal to a group. Pathological congestion effects may occur with such local traffic, for example when all computing nodes attached to a switch communicate with nodes in the next switch; such problem has been observed with stencil workloads in [14]. In such case, Valiant routing avoids the congestion, but selecting an intermediate switch in a remote group implies leaving the source group (and destination) and then returning to it. By contrast, our implementation of Restricted Valiant only selects an intermediate switch within the local group, which generates shorter paths and still avoids the congestion problem.

Yébenes *et al.* identified a similar issue with Valiant routing in the Slim Fly topology, and denoted it the *turn-around problem* [15]. They defined this problem as the case in which packets visit the same switch twice, turning around and going back through the same route. In such cases, the portion of the path between the two visits of the same switch can be omitted without reducing the beneficial effect of randomization. However, in the general case the problem is slightly different, because the paths A and B may not overlap, but still be unnecessarily long. Fig. 1 depicts this issue in a Dragonfly network with *global trunking*, this is, more than one global link connecting pairs of groups, and in a 4×4 Flattened-Butterfly (HyperX). In the Dragonfly under Valiant routing (VAL) the selected random intermediate switch is in a remote group, and no switch is traversed twice because the two paths to the intermediate switch and to the destination are disjoint. However, the resultant path is unnecessarily long: restricting the selection of the intermediate switch to the local group (RVAL) avoids the congestion issue (in the *minimal* link) while providing a shorter path. A similar case occurs in the Flattened Butterfly, where RVAL avoids changing the row when both source and destination are in the same row; the same applies when both source and destination are in the same column, or in general, in a given subset of dimensions.

B. Valiant with recomputation (VAL-Recomp)

While Valiant routing randomizes the intermediate destinations, transient congestion can occur and generate Head-of-Line Blocking (HoLB), delaying injection.

Valiant with recomputation recomputes a new random intermediate destination whenever the required destination port is blocked. As in the original mechanism, a random intermediate destination is selected for each packet at the source router. When a packet is at the head of its injection buffer and the required destination port is blocked, the intermediate Valiant destination is recomputed, overwriting the previous destination. Such recomputation may occur several times, until the packet is injected; once the packet is in-transit, the Valiant destination does not change. If this mechanism is combined with the *restricted* mechanism introduced in Section III-A, the recomputation occurs among the allowed nodes.



(a) Dragonfly

(b) Flattened-Butterfly

Fig. 1: Example of Minimal (MIN), Valiant (VAL) and Restricted Valiant (RVAL) between source (S) and destination (D) nodes in the Dragonfly and Flattened-Butterfly topologies. The intermediate node Valiant paths is represented as I. In both cases, the Valiant path shows an example of turn-around problem without traversing the same switch twice. The Restricted Valiant path is shorter than VAL and avoids pathological congestion within. MIN is obviously the shortest path in every case.

The original Valiant routing is oblivious, because the intermediate destination for each packet is independent of the rest of the packets. This is no longer true when applying recomputation, because this destination is modified depending on the status of the network. According to the taxonomy in [16], Valiant with recomputation is *adaptive* but *congestion-oblivious*, because it routes based on the availability of output ports, and not on an estimation of the network congestion.

IV. Simulation Infrastructure

We employ the in-house designed FOGSim network simulator [17] to perform our experiments. The cycle-accurate simulator models input-output-buffered switches employing multiple virtual channels to avoid deadlocks. We consider a power-efficient Dragonfly network with $p = 6$ terminals per switch, $a = 12$ switches per group and $h = 6$ global links per switch, leading to 5256 terminals, which is representative of

TABLE I: Simulation parameters.

Parameter	**Value**
Topology	Dragonfly
Total end terminals	5,256 hosts
Groups	73 groups
Switches per group	12 switches
Switch degree	23 ports
Link speed	40 Gbps
Packet size	1,000 bytes
Switch frequency	1 GHz
Internal crossbar speedup	2×
Switch latency	200 ns
Local/Global link latency	40/400 ns (8/80 m)
Injection buffers	1
Injection queue size	200 KBytes
Transit queue size	100 KBytes
Virtual Channels	2/1 (MIN)
(local/global ports)	4/2 (Valiant)

realistic HPC systems. Table I shows the parameters employed. Queue size is different for injection buffers, in ports connected to endpoints, and for transit buffers, in those ports connected to other switches. Using this configuration, we have run a battery of simulations with several synthetic traffic patterns.

A. Traffic patterns

We feed the network with synthetic traffic. Each node injects frames according to a Bernoulli process with a variable load, similarly to other network simulation experiments [18]. Different traffic patterns have been considered:

- Uniform (UN), in which the destination of a frame is any other randomly selected network node.
- Adversarial (ADV), in which the destination of a frame is selected randomly from all nodes in the consecutive group. With minimal routing, ADV concentrates the traffic on a single global link between two groups, so non-minimal routing is required to obtain a good performance.
- Adversarial local (ADV-Local), in which the destination of a frame is selected randomly from all nodes in the consecutive switch within the same group, following a modulo sequence. With minimal routing, all the traffic from each switch gathers in a single local link, which becomes a bottleneck and hinders performance as observed in real-world evaluations [14].
- Hotregion (HOT), in which 25% of the traffic generated by each node is sent to the first 12.5% endpoints, and the remaining is distributed as Uniform (including the first 12.5% endpoints). This is the only considered traffic pattern which generates endpoint congestion.
- Random Permutation (PERM), in which each endpoint is assigned a random destination endpoint at the start of the simulation. Each endpoint is assigned to exactly one source and vice-versa. The permutation obtained remains constant for the length of the simulation, but differs in each simulation. The same set of permutations has been used for all routing mechanisms.

B. Routing mechanisms

Minimal routing (MIN) is included as a reference; it sends the packets minimally to the destination group, and then to the destination switch.

Valiant (VAL) selects a random intermediate switch among all switches in the network. Traffic is sent minimally to this switch, and then minimally to the destination.

The Restricted version of Valiant (RVAL) selects an intermediate switch randomly within all switches in the local group when both source and destination nodes belong to the same group; otherwise, it behaves as VAL.

Valiant with recomputation (VAL-Recomp) selects one intermediate switch like Valiant; When the minimal output in blocked at injection, it recalculates the intermediate switch and tries to inject the packet in the next allocation phase. A similar Restricted version (RVAL-Recomp) limits the candidates for intra-group traffic like RVAL.

C. Injection buffers

In the default network configuration employed in our evaluation, switches have one single injection buffer for each endpoint. However, additional injection buffers can be employed to mitigate Head-of-Line Blocking (HoLB), which should improve performance. In such case, a selection policy is needed to select between the multiple injection buffers to store each frame that arrives from an endpoint to the switch.

In our evaluations we have considered two different policies:

- Random policy (RANDOM): each frame can be assigned to any injection buffer with available space.
- Per-destination policy (DEST): all the network endpoints are split in as many sections as injection buffers, and all frames targeting an endpoint in a given section are always stored in the same injection buffer.

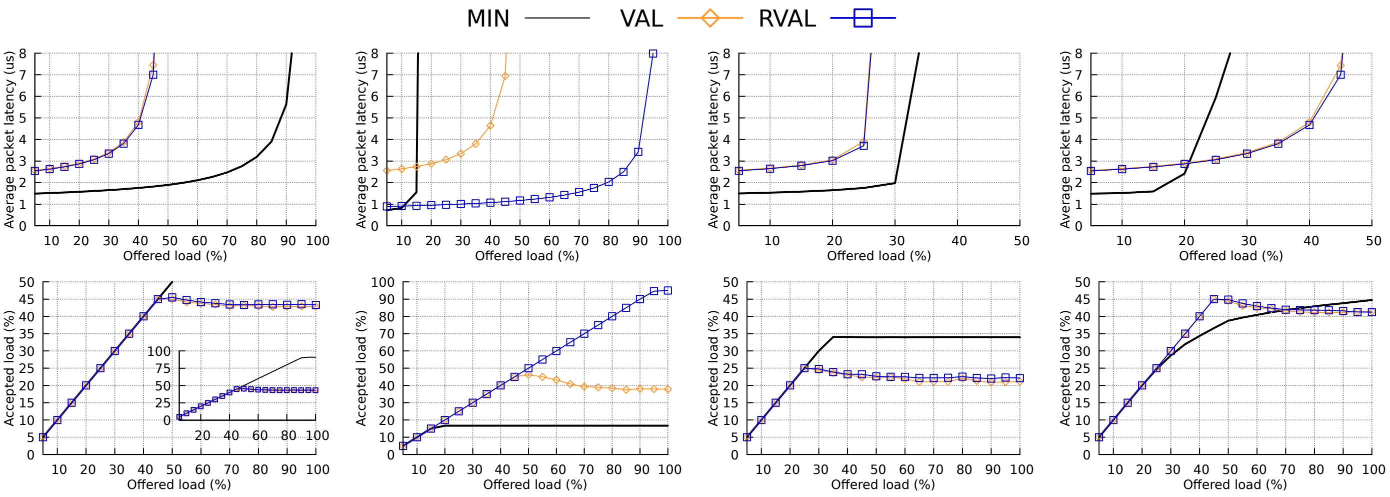
RANDOM ensures all injection buffers are used in a balanced fashion, whereas DEST tries to mitigate HoLB (particularly under MIN routing) since frames in the same buffer are more likely to follow the same path.

V. Results

This section presents the results from our evaluations of the different Valiant routing variants described, using a Dragonfly network with the parameters described in Table I. The results are organized as follows: first, we analyze the impact of our Restricted-Valiant routing mechanism against the original proposal of Valiant routing. Then, the impact of recomputing the intermediate switch is evaluated. Finally, the behavior when multiple injection buffers are employed is observed.

A. Valiant in the Dragonfly

Figure 2 shows the latency (upper graphs) and throughput (lower graphs) under four different traffic patterns: UN, ADV-Local, HOT and PERM. ADV-Local only presents intra-group traffic so the impact of RVAL over VAL is significant. It is used in this evaluation instead of ADV because there is no intra-group traffic in ADV, so VAL and RVAL behave exactly the same under such traffic.



(a) UN traffic. (b) ADV-Local traffic. (c) HOT traffic. (d) PERM traffic.

Fig. 2: Latency (up) and throughput (down) of minimal, Valiant, and Restricted-Valiant routing under different traffic patterns.

Under UN, HOT and PERM the difference between VAL and RVAL is negligible. RVAL latency is slightly lower because a small proportion of the traffic takes advantage of the shorter path generated by restricted Valiant. Similarly, RVAL throughput is slightly higher. The base latency of MIN is lower than the latency of VAL and RVAL for the three aforementioned traffic patterns due to the its shorter paths.

Under hot region traffic (HOT) 25% of the traffic is sent to 12.5% of the network endpoints, and the other 75% of the traffic is UN. Using MIN routing, this implies that the links in the hot region receive 25% of the traffic plus their proportion of the UN traffic sent, limiting throughput to $\frac{12.5}{25+0.125\cdot 75} = 36.36\%$. Fig. 2c shows that MIN throughput is close to this value. With VAL routing, the hot region additionally receives the proportional part of the traffic sent non-minimally (which is 12.5%, since the selection of the intermediate switch is done randomly), further reducing maximum throughput to $\frac{12.5}{12.5+25+0.125\cdot 75} = 26.6\%$. Additionally, Valiant routing widens any congestion trees with a root in the hot region, reducing observed throughput after the saturation point. RVAL barely changes this upper bound, since the majority of the traffic going to the hot region is inter-group.

Under PERM traffic (Fig. 2d) both Valiant variants present congestion after the saturation point, with an 8% throughput drop. However, they achieve better performance than MIN between 25% and 50% traffic loads, due to unfairness present with MIN routing. This unfairness is inherent to the spatial distribution of the traffic, since certain minimal links will suffer more contention than others; using non-minimal routes as in Valiant removes this effect.

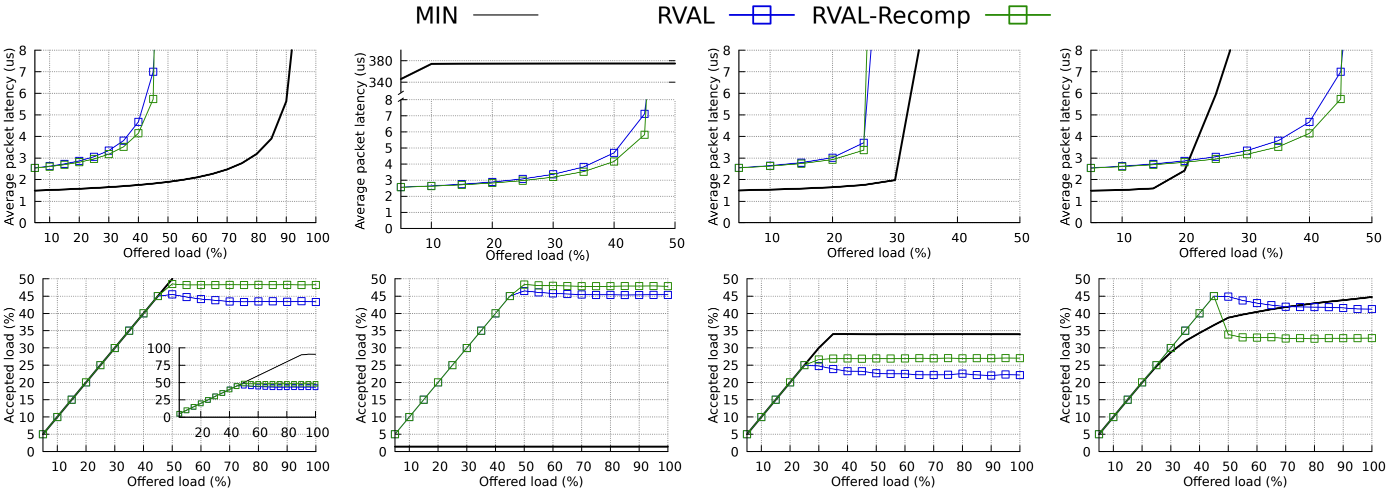
Under adversarial local traffic, the difference between VAL and RVAL is particularly significant. Using minimal routing, the local link connecting pairs of neighbour switches becomes a bottleneck, and only $\frac{1}{p} = 16.6\%$ of the traffic can be delivered using the minimal routes. The default Valiant implementation raises the saturation point compared to MIN. However, latency below the saturation point increases significantly, due to the extra cost of the two additional global hops to and from a remote intermediate group, and throughput is limited below 50%. RVAL achieves almost 100% accepted load, since it avoids the turn-around problem and generates short paths as depicted in Fig. 1. Furthermore, the latency is significantly lower than the one achieved by VAL routing, with a 69.9% reduction for a 30% load.

In conclusion, RVAL performs equal or better than VAL under the evaluated traffic patterns. For this reason, the results in the next subsections only evaluate RVAL.

B. *Valiant with recomputation*

Fig. 3 presents average latency and throughput results for the different traffic patterns employing RVAL routing, with and without recomputing the intermediate Valiant destination. Results with minimal routing are provided as a reference. Latency results are presented in the load region before saturation. In all cases, latency is improved by recomputing the intermediate destination. This is expected, since the recomputation of the Valiant destination occurs when packets cannot be injected because of congestion in the originally selected path; the recomputation selects another path, and injects traffic earlier. In the case of ADV traffic (which typically requires Valiant routing) with a load of 40%, average latency is reduced by 11.4% when using recomputation.

Throughput results after the saturation point differ for each traffic pattern. Both UN and ADV present a similar trend, with the original Valiant mechanism presenting high variability and slightly reduced throughput. The reference with MIN routing is significantly different, with higher throughput under UN due to a lower usage of the global links, and very poor throughput under ADV because only one global link is used to carry all the traffic from every group, limiting the accepted load to $\frac{1}{a\cdot p} = 1.38\%$. Valiant with recomputation injects packets



(a) UN traffic.

(b) ADV traffic.

(c) HOT traffic.

(d) PERM traffic.

Fig. 3: Latency (up) and throughput (down) of Valiant routing under different traffic patterns.

earlier, which leads to higher throughput, with an increase between 4.3% and 5.9%.

The saturation throughput for Hotregion traffic (HOT) in Fig. 3c is lower, because the nodes in the hot region cannot accept all the received traffic. However, both Valiant routing mechanisms present the same pattern as the previous cases, with the recomputation variant presenting a flat response with higher accepted throughput.

Finally, the case of the random permutations (PERM) in Fig. 3d is particular. The use of recomputation provides a flat response with less variation, but suffering from high congestion, observed by the drop of accepted throughput after the saturation point. The oblivious version of Valiant suffers less severely from such congestion, but presents high variability. We hypothesize that the increase in congestion derives from the higher accepted load when using recomputation; a congestion control mechanism, which is not considered in our simulations, should be applied to avoid such issues.

C. Recomputation with multiple injection buffers

This section presents throughput results using multiple buffers for injection. As mentioned in section IV-C, we consider two policies to select the injection buffer for a given packet, DEST (Figure 4) and RANDOM (Figure 5). Latency results are omitted, since the impact of the number of injection buffers on average latency before saturation is negligible.

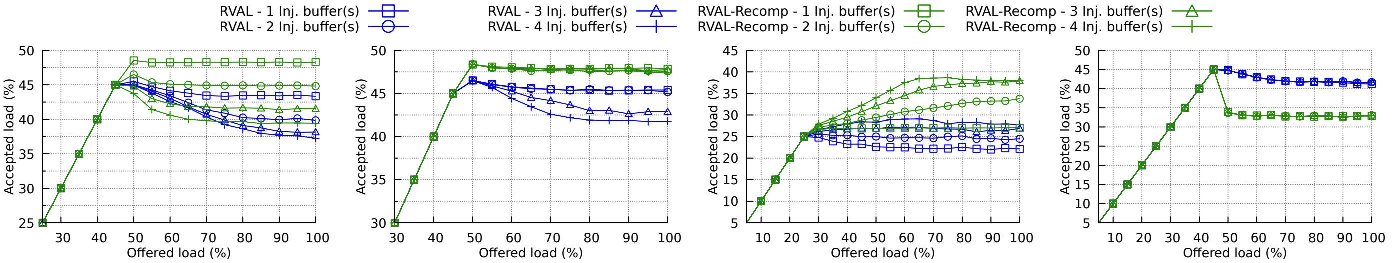
In general, the use of multiple injection buffers is detrimental to the achieved throughput. This decrease in the maximum throughput likely corresponds to an increase in congestion: using a single injection buffer can indirectly throttle injection due to HoLB, which is mitigated with the addition of more injection buffers. Using recomputation helps to alleviate the performance degradation when the congestion is not completely uniform, leveraging the less congested links to dispatch packets faster, and presents a lower overall impact from the use of multiple injection buffers than the RVAL without recomputation variant.

The impact of the number of injection buffers with Valiant with recomputation is lower under ADV traffic (Figs. 4b and 5b). Under this traffic pattern, all the frames from a given source group target nodes in the same destination group, which makes them likely to select the same injection buffer. A similar case can be seen under PERM traffic with the DEST policy (Fig. 4d), because all the nodes always send their frames to the same destination along the whole simulation, and therefore only 1 VC is effectively used. With the RANDOM policy (Fig. 5d), congestion with RVAL without recomputation increases significantly, achieving lower throughput than RVAL-Recomp at high loads.

Results under hot region traffic with the DEST policy (Fig. 4c) present a pathological behavior: with more injection buffers, throughput increases after the saturation point, even surpassing the theoretical limit of 26.6% described in Section V-A. This occurs because frames going to the hot region are placed in different injection buffers from those going somewhere else in the network. This mitigates HoLB between the two traffic flows and allows nodes to inject traffic outside the hot region at a higher rate, effectively changing the proportion of the traffic that goes to the hot region and increasing the overall throughput. This effect is easier to observe with recomputation because the increase in throughput is higher than without recomputation.

VI. Related work

Several mechanisms have proposed restricted variants of Valiant, in which the intermediate switch selection is not performed among all the switches in the network. The original proposal for Valiant in the Dragonfly by Kim *et al.* in [3] selects a random intermediate *group*, rather than a *switch*. This reduces the length of the path (and the amount of virtual channels) but has shown to introduce pathological performance



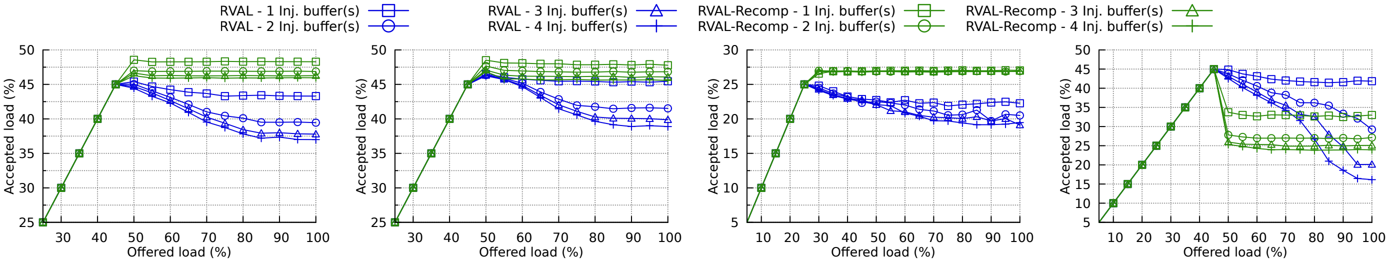
(a) UN traffic.

(b) ADV traffic.

(c) HOT traffic.

(d) PERM traffic.

Fig. 4: Throughput with different number on injection buffers under different traffic patterns, using DEST policy.



(a) UN traffic.

(b) ADV traffic.

(c) HOT traffic.

(d) PERM traffic.

Fig. 5: Throughput with different number on injection buffers under different traffic patterns, using RANDOM policy.

issues under adversarial traffic [13]. Similarly, the dragonfly implementation in [19] diverts traffic using a single global hop for the non-minimal path A, which is equivalent to selecting a random *group* that is directly connected to the source switch; in this case, the implementation is constrained to using commodity Ethernet hardware without bookkeeping information in the packet headers.

Different proposals implement restricted variants of Valiant routing in alternative topologies. The proposal for randomized routing in multidimensional square meshes in [6] does not randomize all the N dimensions, but only $N-1$; congestion is avoided assuming a single injector per switch, but this is not typically the case in current and forthcoming parallel systems. The *DAL* adaptive routing mechanism in HyperX [9] follows the idea of RVAL by misrouting only in the dimensions with offset. Valiant routing in the indirect OFT and MLFM networks is restricted to switches with connected nodes in [20]. The already-mentioned proposal in [15] identifies the turn-around problem in Slim Flies, and introduces a modified version of Valiant routing for Slim Flies which only makes one non-minimal hop in its phase A.

Regarding the recomputation of intermediate switches based on network congestion, we are not aware of its application to the original oblivious Valiant but there are adaptive routing proposals that somehow resemble it. For example, the aforementioned *DAL* adaptive routing in HyperX [9] deroutes traffic in a given dimension based on the unavailability of output ports; therefore, it dynamically diverts traffic to an intermediate destination which is computed dynamically.

VII. Conclusions and future work

This work introduces two improvements to Valiant routing, targeting high-radix networks. Restricted Valiant improves performance for traffic with locality, selecting the intermediate router in the same network partition as the source and destination. Examples of such partitions have been introduced for Dragonfly and Flattened Butterfly networks. Valiant with recomputation avoids Head-of-Line blocking at injection by selecting an alternative random intermediate router when the output port is stalled.

Both approaches are evaluated using a Dragonfly network. RVAL shows significant improvements for intra-group traffic with up to 69.9% latency reduction. The recomputation variant improves throughput by increasing injection, although in certain cases (permutations) this also increases congestion. Using multiple injection buffers further increase congestion, and the optimal buffer selection policy depends on the traffic pattern.

In future work, we aim to evaluate the impact of our proposal when used for the selection of the nonminimal route in adaptive routing mechanisms, which may mitigate the observed congestion. Also, Valiant routing is algorithmic, but most current designs are based on tables; we plan to evaluate implementations suited for such environment. Finally, the current paper suggests how to build different partitions for Dragonflies and Flattened-Butterflies, but it is not proven that no other traffic pattern introduces pathological congestion with this routing. Indeed, the locality for these networks is quite direct, based on groups or dimensions, but how to apply RVAL to other topologies without suffering congestion under any traffic pattern is left open.

References

[1] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a high radix router," in *32nd International Symposium on Computer Architecture (ISCA'05)*, June 2005, pp. 420–431.

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

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

[4] M. Besta and T. Hoefler, "Slim Fly: A cost effective low-diameter network topology," in *IEEE/ACM Intl. Conf. on High Performance Computing, Networking, Storage and Analysis (SC14)*, 2014.

[5] S. Rumley, M. Glick, S. D. Hammond, A. Rodrigues, and K. Bergman, "Design methodology for optimizing optical interconnection networks in high performance systems," in *High Performance Computing*, J. M. Kunkel and T. Ludwig, Eds. Cham: Springer International Publishing, 2015, pp. 454–471.

[6] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '81. New York, NY, USA: ACM, 1981, pp. 263–277. [Online]. Available: http://doi.acm.org/10.1145/800076.802479

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

[8] L. G. Valiant, "Optimality of a two-phase strategy for routing in interconnection networks," *IEEE Trans. Comput.*, vol. 32, no. 9, pp. 861–863, Sep. 1983. [Online]. Available: https://doi.org/10.1109/TC.1983.1676335

[9] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "HyperX: Topology, routing, and packaging of efficient large-scale networks," in *Conference on High Performance Computing Networking, Storage and Analysis (SC '09)*, New York, NY, USA, 2009, pp. 41:1–41:11. [Online]. Available: http://doi.acm.org/10.1145/1654059.1654101

[10] C. Camarero, C. Martinez, E. Vallejo, and R. Beivide, "Projective networks: Topologies for large parallel computer systems," *IEEE Transactions on Parallel & Distributed Systems*, vol. 28, no. 7, pp. 2003–2016, 2017.

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

[12] J. Kim, W. Dally, S. Scott, and D. Abts, "Cost-efficient Dragonfly topology for large-scale systems," *Micro, IEEE*, vol. 29, no. 1, pp. 33–40, 2009.

[13] 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 *41st International Conference on Parallel Processing*, Sept 2012, pp. 279–288.

[14] D. J. Kerbyson and K. J. Barker, "Analyzing the performance bottlenecks of the POWER7-IH network," in *International Conference on Cluster Computing*, Sept 2011, pp. 244–252.

[15] P. Yébenes, J. Escudero-Sahuquillo, P. J. García, F. J. Quiles, and T. Hoefler, "Improving non-minimal and adaptive routing algorithms in slim fly networks," in *2017 IEEE 25th Annual Symposium on High-Performance Interconnects (HOTI)*, Aug 2017, pp. 1–8.

[16] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *IEEE International Symposium on High Performance Computer Architecture*, Feb 2008, pp. 203–214.

[17] M. García, P. Fuentes, M. Odriozola, M. Benito, E. Vallejo, and R. Beivide. (2014) FOGSim interconnection network simulator. [Online]. Available: http://fuentesp.github.io/fogsim/

[18] J. García-Haro, R. Marín-Sillué, and J. L. Melús-Moreno, *ATMSWSIM An efficient, portable and expandable ATM SWitch SIMulator tool*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 193–212.

[19] M. Benito, E. Vallejo, and R. Beivide, "On the use of commodity Ethernet technology in exascale HPC systems," in *IEEE 22nd International Conference on High Performance Computing (HiPC)*, 2015, pp. 254–263.

[20] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. Hoefler, "Cost-effective diameter-two topologies: analysis and evaluation," in *SC15: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2015, pp. 1–11.