Performance Optimization of Load Imbalanced Workloads in Large Scale Dragonfly Systems

Bogdan Prisacari, German Rodriguez, Cyriel Minkenberg
IBM Research – Zurich
Saumerstrasse 4, 8803 Ruschlikon, Switzerland

Marina Garcia, Enrique Vallejo, Ramon Beivide
University of Cantabria
Avenida de los Castros, s/n, 39005, Santander, Cantabria, Spain

***Abstract*—Dragonfly topologies are one of the most promising interconnect designs for enabling large, potentially exascale compute systems, particularly those envisioned to accommodate workloads that are sensitive to system diameter and end-to-end latency. They are cost-effective designs with a very low diameter and close to optimal performance for workloads which induce a balanced load across the network. However, these benefits are balanced by a reduced path diversity, which leaves Dragonflies vulnerable to certain adversarial traffic patterns. The performance of such workloads can be significantly improved using indirect routing approaches. However, the indirect routing approach that is most commonly used today exhibits in turn significant vulnerability to a subset of these traffic patterns for reasons that have not been, up to now entirely, understood.**

In exploring this vulnerability, we manage to provide a theoretical justification, based on inherent properties of the Dragonfly topology, of why performance degrades. Furthermore, we manage to isolate what specifically in the structure of a traffic pattern makes it a worst case in this context, and thus we are able to characterize the precise workload subset that will experience poor performance. By building upon the understanding of the interaction that causes sub-optimal behavior, we then show how simple changes to either the routing strategy or the process to node assignment can bring performance back close to ideal levels. Finally, we not only provide a theoretical justification for our performance models, but also validate them via comprehensive simulation-based studies of systems with up to 16,512 nodes.

I. Introduction and Related Work

With the advent of Big Data, a need has arisen for ever larger and more powerful systems in both the High Performance Computing and datacenter space. Furthermore, future exascale systems will not only need to be computationally powerful, but also match that power with high bandwidth and especially low latency interconnection networks [1]. Dragonfly networks, a recent cost-effective and efficient network topology, developed independently by IBM (as the interconnection fabric of the PERCS system [2]) and Kim et al. [3], is one of the most promising candidate topologies for such systems. They simultaneously offer i) low diameter, ii) high performance for uniform random traffic, iii) a hierarchical design that allows for a very advantageous performance/cost ratio, by using a combination of copper and optical wiring, iv) very good scalability (exceeding tens of thousands of nodes), as well as other advantages. Current Dragonfly-based systems include the IBM PERCS system [2] as well as Cray Cascade systems [4], most prominently Piz Daint, the largest European supercomputer [5].

However, while performing very well for applications where the communication pattern utilizes the network in an approximately uniform fashion, Dragonfly systems have been shown to be susceptible to performance degradation when dealing with certain other types of traffic. Previous works in this area [6]–[9] have identified several traffic patterns whose performance is sub-optimal on these networks and that furthermore are characteristic of relevant HPC workloads. Techniques have been proposed to reduce the performance degradation of these workloads. They range from adjusting the concurrent process placement [6] to adapting the routing strategy [3], [7] or even the topology of the network itself [7], [10]. The sought after end result of each of these approaches is to reshape network traffic into something closely resembling a uniform random pattern (which the Dragonfly network can move at peak throughput) by breaking down the regularity of adversarial workloads. However, as we will show, it is often the case with the most popular of these techniques that performance levels still fall short.

This work identifies the root cause of the sub-optimal performance. We analytically characterize non-trivial bottlenecks that emerge as a consequence of specific symmetries characteristic of a wide class of traffic patterns and provide a performance model capable of accurate predictions. While for (adversarial) linear shift patterns such bottlenecks had been empirically observed by previous work [7], a thorough analysis of the cause of the behavior has not been provided. We precisely identify necessary and sufficient traffic characteristics that make workloads prone to these bottlenecks, and exemplify them with several common practical cases: bit complement, linear shift and nearest neighbor-like traffic. Furthermore, we show that the bottlenecks can be mitigated to a large extent by employing an alternative routing strategy or process allocation. Finally, we validate our claims via a wide range of simulation-based studies of practical network configurations.

II. Background

Dragonflies are direct, hierarchical networks with two levels. Being direct networks, every switch in the system is connected to a number of nodes, which in the case of the Dragonfly is constant across switches. The parameter p of the topology denotes this number. At the first hierarchical level, the set of switches in the network is partitioned into equal sized sets called groups. The parameter a of the topology denotes

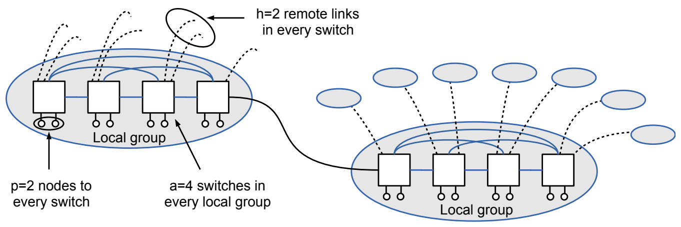


Fig. 1: Illustration of a $(p = 2, a = 4, h = 2)$ Dragonfly topology, with two groups presented in detail.

the number of switches that each group contains. Furthermore, the switches belonging to the same group are interconnected to form a typically low diameter subnetwork, such as a full mesh (such as in the PERCS interconnect) or a flattened butterfly (such as in the Cray Cascade systems). When the subnetwork is a full mesh, $a - 1$ ports per switch are necessary to create this interconnection pattern. In this work we will refer to links forming the group subnetwork as intra-group or local (L) links. Finally, each switch has a constant (across switches) number of ports that connect it to switches in other groups than its own. The topology parameter h denotes this number of ports. These ports are used to effectively interconnect the individual groups into a full mesh of groups at the second hierarchical level. In this work we will refer to links between groups as inter-group or remote (R) links.

To ensure connectivity, at least one link must connect every pair of groups. In the case of full scale Dragonfly systems, exactly one link connects every pair. For lower scale systems, multiple links can be employed to interconnect group pairs, with the benefit of larger inter-group bandwidth. In this work, we will restrict our study to the full scale case, but the derived conclusions extend naturally to the lower scale Dragonflies. Thus, at full scale, a Dragonfly network is characterized by three topological parameters, (p, a, h). The structure of a full scale $(2, 4, 2)$ Dragonfly system is illustrated in Fig. 1. Sets of $p = 2$ nodes are connected to every switch. The switches are partitioned into fully-meshed groups of size $a = 4$. Each switch has $a - 1$ ports connecting it to switches in its own group and $h = 2$ ports towards switches in other groups.

One of the most attractive features of Dragonfly networks is their very low diameter. Indeed, under the so called shortest path or direct routing, possible paths consists in a traversal of at most 3 inter-switch links. More precisely, the longest possible shortest path is made up of

- an L link traversal in the group of the source node to get to the switch that has the R link to the destination group,
- a traversal of that remote link and
- a second L link traversal in the destination group to get to the switch directly connected to the destination node.

It has been shown that, for certain topological parameter constellations (particularly those for which $p = a/2 = h$) close to ideal throughput can be achieved for uniform traffic under direct routing, which is another feature of the Dragonfly network. While shortest path routing has the advantage of low end-to-end latency, it also comes with the price of very low path diversity (at full scale, a single shortest path exists between any source-destination pair). This lack of diversity can lead to an extreme degradation in performance for certain adversarial traffic patterns. Therefore, routing algorithms have been introduced that sacrifice path length for diversity, the majority based on Valiant's algorithm [11].

The general idea of this approach is, given a message going from a certain source to a certain destination, to:

- randomly chose an intermediate switch,
- send the message along the shortest path route between the source and the intermediate switch, and finally
- send the message along the shortest path between the intermediate switch and the destination.

In the case of load imbalanced, or adversarial traffic patterns, the expectation is that, by using a different random intermediate switch for each message, the original nature of the traffic is shifted towards a uniform random traffic (which has close to ideal performance) at the expense of longer paths and of a doubling of the load under dense traffic [12]. In addition to these drawbacks, the longer paths in Valiant routing also require the use of additional virtual channels to guarantee deadlock freedom. Indeed, whereas shortest path routing only requires 2 virtual channels, general Valiant routing can require up to 4, with variants such as the one described in [3], which we will call *ValiantRestricted*, requiring at least 3.

ValiantRestricted can be described as follows: as in the general Valiant case, when a source s in group S sends a message to destination d in group D, an intermediate group I is first chosen. A minimal route (consisting of at most one L and one R hop) is taken to arrive to the first reachable switch in group I. From there, the packet follows the unique minimal route to the destination d (requiring at most two L hops and one R hop). The longest path using this Valiant variant would then have the following shape: *LR-LRL*.

The generic Valiant on the other hand, also called *ValiantAny* [8] or *VALn* [13] in other works, would randomly choose potentially any switch (not just the first reachable) in the intermediate group as the intermediate destination, thus potentially incurring an extra L hop within the intermediate group, and requiring one extra virtual channel for the L channels. In this case, the longest paths possible have the shape: *LRL-LRL*. This routing variant might negatively affect some traffic patterns (due to an increased load on the local links) while helping other patterns (providing a route around a congested intermediate group local link).

III. Imbalanced Workload Performance

While very well suited for efficiently accommodating workloads with a uniform traffic matrix, Dragonflies can experience significant performance degradation in the presence of other workloads. This is caused by the fact that the amount of bandwidth available between any pair of source-destination groups is limited, as there is typically a single remote link connecting the pair. Traffic patterns that are particularly performance challenged include the bit complement pattern,

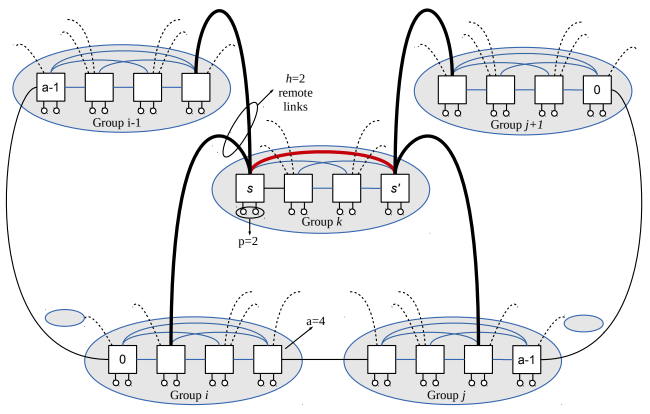


Fig. 2: Illustration of emergence of intra-group bottlenecks for bit complement traffic.

many shift permutation patterns [12] as well as the nearest neighbor exchange which occurs commonly in many scientific workloads. As a general rule, a workload will experience poor performance on a Dragonfly system under direct routing if the traffic matrix is such that the traffic originating in nodes belonging to a given group is predominantly destined to destination nodes that also share the same group. Such a pattern will induce a large amount of traffic on the link connecting the source-destination group pairs effectively causing it to become a bottleneck and limit overall throughput. In this section, we will show that for a large subset of these adversarial patterns, including the examples above, the indirect routing approach that is currently the most popular in practice, *ValiantRestricted*, is unable to increase performance significantly.

By selecting an intermediate group uniformly at random to which to route the traffic in a first phase, before actually routing towards the destination, *ValiantRestricted* is indeed able to avert the bottleneck described earlier. Indeed, the uniform choice of the intermediate group ensures that the load previously experienced by a single remote link is now distribution across all remote links originating in the source group, However, the choice is uniform only across groups, which means that once the intermediate group is reached, the messages will be immediately routed towards the destination. This leads to load being balanced only across remote links, with no guarantee for load balancing inside the individual groups. The approach is consequently unable to prevent the subsequent formation of bottlenecks in the intra-group links in the presence of certain load imbalanced traffic patterns.

Given a Dragonfly network with parameters (p, a, h)without loss of generality one can assume that:

- for a given switch, the remote links originating on that switch are assigned consecutive numbers from 0 to $h-1$,
- for a given group, the switches belonging to that group are assigned consecutive numbers from 0 to $a-1$, and
- for the entire system, the groups are assigned consecutive numbers from 0 to $a \cdot h$.

Using this numbering scheme, a typical remote link interconnection pattern (the pattern denoted "Palmtree" in [10]) is to connect switch number x in group i through its m-th remote link to a switch in group $j = (i + x \cdot h + m) \bmod (ah + 1)$. This leads to remote links on the same switch, that are consecutive according to the number scheme, interconnecting a given group to groups that are also consecutive in the same numbering scheme. In the remainder of the section, we will show that this regularity of the interconnect, coupled with certain regularities characteristic of some of the load imbalanced traffic patterns, make intra-group bottlenecks unavoidable under *ValiantRestricted* indirect routing. The traffic pattern characteristic in question is that pairs of communicating groups have the property that consecutive source groups are paired with consecutive destination groups. This is indeed the case for bit complement, shift and nearest neighbor exchanges, among others.

Let us consider that the indirect routing approach used in the system is *ValiantRestricted*. Given an arbitrary switch s of an intermediate group g, let us consider the h remote links of s. These links will connect the group g to h source groups, that will be consecutive, as a consequence of the interconnection pattern described above. When a message originating in any of the source groups is routed indirectly such that g is selected as the intermediate group, the structure of the path that the message will take will be the following. The message will be routed minimally from its source to switch x, and from there it will again be routed minimally to its destination (as explained in Sec. II). Due to the assumptions we've made on the traffic pattern, the destination groups of traffic originating in the h consecutive source groups will also be consecutive. This implies that the h destination groups will be connected to group g via consecutive links, i.e., by links x to $h-1$ on some switch s' and links 0 to $x-1$ on switch $s'' = (s' + 1) \bmod a$, for some values of x and s' that are specific to the traffic pattern. This further implies that messages incoming on the h remote links of s and taking the shortest paths to their respective destinations will necessarily have to navigate to one of the two switches s' or s'', and from there take the remote link to their respective destination groups. It necessarily follows those messages will at most use one of two possible local links within group g: either the link connecting s to s' or the link connecting s to s''. All other intra-group links originating on s will be completely unused by indirect traffic, leading to a significant intra-group load imbalance. Fig. 2 illustrates this effect for the bit complement pattern. Processes in group i send messages to processes in group j; processes in group $i-1$ send messages to processes in group $j+1$. Under indirect *ValiantRestricted* routing, due to the typical wiring of a Dragonfly, a local link in each potential intermediate group k experiences a load that is a up to factor of h higher than what it can sustain. The cause of this behavior is the *ValiantRestricted* strategy itself, which forces the taking of a minimal path towards the destination, *immediately* once the intermediate group is reached.

As an observation, for traffic patterns that exhibit a high degree of locality, i.e., for which (source,destination) group pairs have the property that the absolute difference between the source and destination group index is low (lower than h), s' and/or s'' might coincide with s, in which case traffic entering the intermediate group might exit it via the same ingress switch s, leading to the weakening of the intra-group bottleneck. Among the traffic patterns analyzed in this work, this is the case only for high locality phases of nearest neighbor exchanges and shift patterns with a very small offset.

This imbalance has an important performance impact. The traffic incoming over x of the remote links of s and the traffic incoming over the remainder $h-x$ of the remote links of s will effectively see their throughput limited to that of a single intra-group link, leading to a limitation of the effective bandwidth available to each remote link that is between $1/h$ (worst case, when $x = 0$ or $x = h$) and $2/h$ (best case, when $x = h/2$) of the L link bandwidth.

Given a remote link connecting groups g and g', two main types of traffic will traverse it. The first type is made up of messages originating in group g, destined to any of the groups $g'' \notin \{g, g'\}$ and using group g' as an intermediate group. The second type is made up of messages originating in any of the groups $g'' \notin \{g, g'\}$, destined to group g' and using group g as an intermediate group. Due to the same number of (source, destination) pairs in each flow and to the way the indirect route is selected for each message, both of these flows are equal in size. Thus each flow will be able to use half of the effective bandwidth of the remote link. As this applies to any of the remote links of the network, it follows that the maximum aggregate injection throughput that the network can sustain is upper limited by half the aggregate effective throughput of all remote links. Thus, the network's performance T, expressed as the relative (to total injection bandwidth) throughput the network can sustain, is bounded by:

$$\frac{1}{2} \cdot \frac{N_r}{N} \cdot \frac{1}{h} \leq T \leq \frac{1}{2} \cdot \frac{N_r}{N} \cdot \frac{2}{h}, \qquad (1)$$

where N_r is the total number of R links and N is the total number of nodes in the system:

$$N_r = (ah + 1) \cdot a \cdot h, \qquad (2)$$

$$N = (ah + 1) \cdot a \cdot p. \qquad (3)$$

For every intermediate switch s, the number x of incoming remote links that share the same local link in the next hop takes a pattern-specific value that can be computed explicitly. Thus, for a given traffic matrix, the bottlenecks in every R link and consequently the throughput T can be computed exactly. The cumulative effect across all s is however generally well approximated by an average R limitation equal to the average of the extreme values, i.e., $3/(2h)$. Thus, from Eq. (1), (2), and (3), a reasonable estimate for the effective throughput is:

$$T = \frac{3}{4p}. \qquad (4)$$

IV. Experimental Results

In this section we validate the model introduced in Sec. III by comparing it to simulation results obtained for several traffic patterns on multiple Dragonfly topologies. We also show how addressing the two root causes of bottleneck emergence, either by altering the routing strategy (to spread traffic more evenly across the local links of the intermediate group) or the process allocation (to break the symmetries in the traffic patterns) can effectively improve performance.

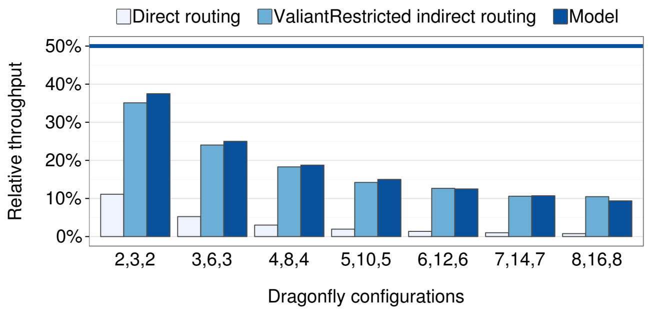
A. Framework, parameters and metrics

The results presented in this section were obtained using a simulation framework [14] that is able to accurately model and measure generic and custom networks, including Dragonflies, at a flit level. Dragonflies with as little as 42 $(2, 3, 2)$ and as many as 16,512 nodes $(8, 16, 8)$ were simulated. The switch architecture chosen was that of an input-output-buffered switch with 4 Kbytes of buffer space per port per direction per virtual channel. The links had a bandwidth of 40 Gbit/second. Credit based flow control was used and each exchanged message consisted of a single 64 byte flit. The routing algorithms used were shortest path, *ValiantRestricted* and *ValiantAny* routing with virtual channel based deadlock avoidance.

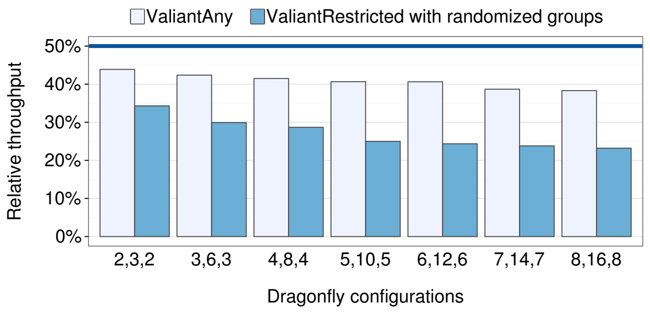
We benchmarked several traffic patterns that satisfy both the adversarial criterion (the majority of traffic originating in one group is destined to a single other group) and the consecutiveness criterion (consecutive groups send to consecutive destinations). These are:

- bit complement traffic, where each process k (where $0 \leq k < K$, K being the total number of processes) selects as the destination of every message process $K - k - 1$;
- shift traffic, where each process k selects as the destination of every message process $(k + mG) \bmod K$, G being the total number of processes hosted by an entire Dragonfly group and m being an integer larger than h (this condition is necessary to avoid the situation mentioned in Sec. III where traffic leaves the intermediate group from the switch it came in on);
- 3D nearest neighbor, particularly the most remote phase of the exchange (corresponding to the third dimension).

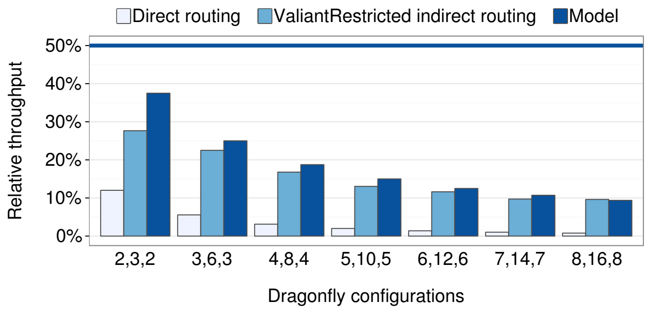
For the first two traffic patterns, messages were generated continuously at link rate throughout the simulation, while for the last pattern the total amount of data exchanged between any pair of neighbors was 256 KB. For each traffic pattern, the assignment of processes to nodes was performed either in a contiguous fashion or a randomized fashion, with a single process per node. For the contiguous placement, every process was assigned to the node that has a node index equal to the processes' index. The nodes are indexed topologically: they are numbered consecutively (starting with 0) one group at a time, and within a given group, one switch at a time. For the randomized placement, we start with a contiguous assignment, and then simply randomize the group numbering (described in Sec. III) before creating the remote link interconnection pattern. This breaks the symmetry in the workload, i.e., in



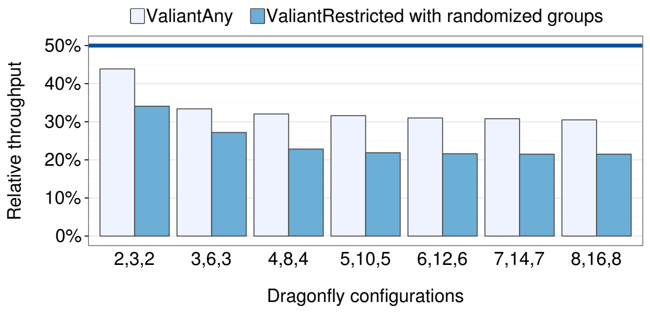
(a) Bit complement - bottleneck-limited performance



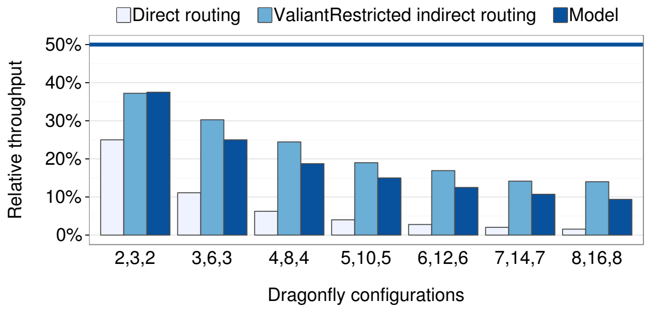
(b) Bit complement - optimized performance



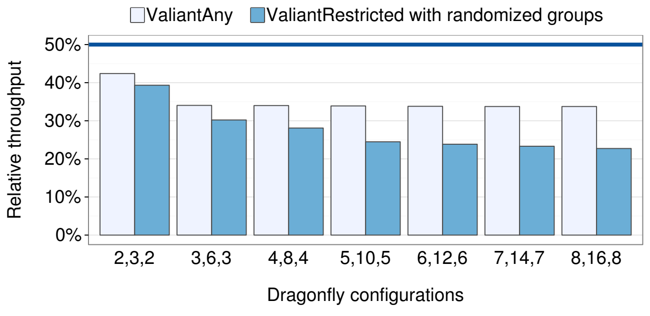
(c) Shift - bottleneck-limited performance



(d) Shift - optimized performance



(e) Nearest neighbor - bottleneck-limited performance



(f) Nearest neighbor - optimized performance

Fig. 3: Measured and predicted throughput for adversarial workloads running on balanced dragonflies of increasing size. The figures on the left show performance under i) shortest path routing, illustrating the adversarial nature of the workload ii) *ValiantRestricted* indirect routing, and iii) the local imbalance aware estimation we introduced in this work. The figures on the right show performance under i) the locally balanced *ValiantAny* indirect routing, as well as ii) the effect of group randomization on *ValiantRestricted*. The load-imbalance-oblivious theoretical maximum is shown as a horizontal line.

the resulting pattern consecutively numbered source groups no longer send to consecutively numbered destination groups.

For every experiment, the system was simulated for a fixed amount of time (10 milliseconds) chosen such that throughput is estimated within a 99% confidence interval.

B. Results

Figures 3a, 3c and 3e show the relative throughput achieved by bit complement, shift and nearest neighbor traffic respectively under shortest path and *ValiantRestricted* routing, side by side with the analytical throughput estimates. We can see that the *ValiantRestricted* indirect routing approach fails to randomize traffic sufficiently, i.e., such that it would exhibit a behavior similar to that of uniform random traffic (shown as the horizontal line in the figures). Indeed, we can see that instead, the measured performance under this approach follows closely the performance levels anticipated by the analytical model derived in Sec. III. Due to communication bottlenecks shifting to the intra-group links, and routing being unable to distribute the load within the intermediate group (as explained in detail in the same section), performance follows a descending trend as the size of the network grows. Indeed, if N is the total number of nodes in the system, the relative throughput is measured to be inversely proportional to $\sqrt[4]{N/4}$, exactly as predicted by the analytical model (Eq. (3) and (4) coupled with the balancing condition $p = a/2 = h$). Due to the specificities of each workload's traffic matrix, there are small differences between measured performance and model predictions. This is due to us using the approximate yet straightforward estimator expressed by Eq. (4). A more complex analysis can be performed that would take into account the variations in bottleneck intensity specific to each group. While more precise, such an analysis would also make it more difficult to understand whether the predicted performance level is inherent to the application class or induced by particularities of the application instance. We believe the estimator we propose reaches a good balance between genericity and precision and, as the results show, reasonably approximates measured behavior across applications.

Figures 3b, 3d and 3f on the other hand show the relative throughput achieved in the case where the routing scheme or the process to node assignment is changed precisely to address the bottlenecks we identified. That is, they show performance measured i) when using the *ValiantAny* indirect routing approach, which effectively balances not only remote traffic, but also local traffic in the indirect groups and ii) when using the *ValiantRestricted* indirect routing approach, coupled to a de-correlation between group numbering and workload numbering achieved via process to node assignment randomization. As the figures show, these measures are effective in mitigating to some extent the intra-group bottlenecks that the workloads were creating, leading to a traffic pattern that better approximates a uniform traffic scenario.

Indeed, whereas perfectly uniform traffic is expected to achieve a 50% relative throughput (indicated by a horizontal line in the figures), performance for the benchmarked traffic patterns is as follows. All traffic patterns routed with *ValiantAny* achieve a relative throughput of $33\% - 43\%$, slightly higher overall for the shift and bit complement workloads. *ValiantRestricted* on the other hand leads to progressively worse performance as the system size grows, dropping below 15% (bit complement and shift) and 20% (nearest neighbor) throughput respectively for networks with more than 512 switches. Enhancing *ValiantRestricted* with process placement randomization as an alternative to *ValiantAny* leads to a gain in performance of roughly 10%, especially for the larger network sizes.

V. Conclusions

In this work we have provided a detailed analysis of bottlenecks induced by certain adversarial communication patterns in the internal groups of Dragonfly networks under a specific Valiant routing approach: *ValiantRestricted*. Although enabling shorter indirect paths and being more cost-effective (by requiring less switch resources than other Valiant routings), we have shown that, on-par with observations from other works, this indirect routing approach is unable to distribute load in the network such that performance levels approach those of uniform traffic. Furthermore, we identified and characterized several workloads for which *ValiantRestricted* is prone to creating severe intra-group load imbalances that induce a significant degradation of the expected performance, degradation that increases with the scale of the network. In this work we provide a theoretical explanation for this behavior and an analytical model that accurately predicts performance.

In support of these theoretical results, we equally provided simulation-based experimental measurements that showed our conclusions to hold in practical system configurations with as many as 16,512 endpoints. Finally, by means of similar experiments, we also showed that relative throughput can be increased by roughly 10% by using process to node assignment randomization on top of *ValiantRestricted*, and that an even larger improvement can be obtained by switching to an alternative indirect routing approach, *ValiantAny*, requiring one extra virtual channel on each link and increasing by one the length of indirect paths. This latter approach is able to better balance load both inter and intra-group, significantly improving the similarity of the resulting traffic to uniform random traffic, and consequently improving performance to roughly 65% to 85% of the optimum.

Acknowledgments

This work has been supported by the Spanish Science and Technology Commission (CICYT) under contract TIN2013-46957-C2-2-P, the European HiPEAC Network of Excellence and the JSA no. 2013-119 as part of the IBM/BSC Technology Center for Supercomputing agreement.

References

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

[2] B. Arimilli *et al.*, "The PERCS high-performance interconnect," *Proceedings of the 2010 18th IEEE Symposium on High Performance Interconnects*, pp. 75–82, 2010.

[3] J. Kim *et al.*, "Technology-driven, highly-scalable dragonfly topology," *SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 77–88, Jun. 2008.

[4] G. Faanes *et al.*, "Cray Cascade: A scalable HPC system based on a Dragonfly network," *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 103:1–103:9, 2012.

[5] "Top500," http://www.top500.org/list/2014/11/, November 2014, accessed: 2015-03-01.

[6] A. Bhatele *et al.*, "Avoiding hot-spots on two-level direct networks," *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 76:1–76:11, 2011.

[7] M. García *et al.*, "On-the-fly adaptive routing in high-radix hierarchical networks," in *The 41st International Conference on Parallel Processing (ICPP)*, 09 2012.

[8] B. Prisacari *et al.*, "Randomizing task placement and route selection do not randomize traffic (enough)," *Design Automation for Embedded Systems*, pp. 1–12, 2014.

[9] M. Alvanos *et al.*, "Improving performance of all-to-all communication through loop scheduling in PGAS environments," *Proceedings of the 27th international ACM conference on International conference on supercomputing*, pp. 457–458, 2013.

[10] C. Camarero *et al.*, "Topological characterization of hamming and dragonfly networks and its implications on routing," *ACM Trans. Archit. Code Optim.*, vol. 11, no. 4, pp. 39:1–39:25, Dec. 2014.

[11] L. G. Valiant, "A scheme for fast parallel communication," *SIAM J. Comput.*, vol. 11, no. 2, pp. 350–361, 1982.

[12] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.

[13] J. Won *et al.*, "Overcoming far-end congestion in large-scale networks," *IEEE 21st International Symposium on High Performance Computer Architecture*, pp. 415–427, Feb 2015.

[14] C. Minkenberg *et al.*, "End-to-end modeling and simulation of high-performance computing systems," *Springer Proceedings in Physics: Use Cases of Discrete Event Simulation: Appliance and Research*, p. 201, 2012.