Improving Valiant Routing for Slim Fly Networks

Deyu Han*
Carnegie Mellon Univ.
Pittsburgh, PA 15213
deyuh@andrew.cmu.edu

Zhaofeng Wang
Knox College
Galesburg, IL 61401
zwang@knox.edu

David P. Bunde
Knox College
Galesburg, IL 61401
dbunde@knox.edu

Abstract—**Valiant routing, the use of a random intermediate node to distribute network traffic, has been proposed for a number of recent HPC network topologies. It is also commonly used as a bulding block for adaptive routing algorithms, which use shortest path routes when possible, but revert to Valiant routing when necessary to avoid hot spots. We show that the version of Valiant routing proposed for the Slim fly topology can cause messages to follow loops, using an edge in both directions before returning to edges of the original shortest path. Removing these loops in the UGAL-L adaptive routing algorithm is shown to provide slight improvements in average latency and also allow the network to carry up to 12% more traffic before saturation.**

1. Introduction

The continuing quest to build larger HPC systems and the availability of new technologies have rebalanced the priorities for system design. On the demand side, increasing core counts continue to require higher levels of parallelism, leading to pressure for fine-grained parallelism and the low latency communication required to support it. At the same time, energy consumption is becoming ever more important as component counts rise but total system power is capped. In terms of new technologies, new high-radix routers (e.g. [1], [2]) provide the potential for reducing the network diameter and optical interconneccts provide a relatively long-range option for low latency, power efficient data transmission.

These considerations have lead to the proposal of a variety of new topologies for high-performance systems. The most famous of these is Dragonfly [3], but other new topologies have been proposed as well [4], [5], [6]. In this paper, we focus on Slim Fly [7], which is based on the MMS graph [8]. Its structure is algebraically defined in an attempt to approach the Moore bound [9], the maximum number of vertices a graph with a given degree and diameter can have.

The higher-radix routers used in many of these new topologies mean that, in the worst case, they can greatly concentrate network traffic due to the many shortest paths that use each edge. The potential for serious hot spots has led to renewed interest in the randomized routing algorithm of Valiant [10]. This was originally developed for hypercube systems, but the idea easily generalizes; instead of a message proceeding from its source to its destination using a shortest path, it first travels to a randomly-chosen intermediate switch and then proceeds to the destination. The paths to and from the intermediate switch both use a shortest path, but the detour to the intermediate switch spreads out the traffic. The original algorithm is based on shortest path routes with the hypercube dimensions randomly ordered; Valiant [10] showed that this essentially eliminated congestion for all-to-all traffic. Subsequently, it was found that using a fixed dimension order gave the same result with a simplified analysis (e.g. [11], [12, pp. 74–78]).

The term *Valiant routing* is commonly used to mean any application of the idea of using a random intermediate switch. It is widely used in newly-proposed topologies, though not always with a fully random choice of free switch; for example, the version for Dragonfly [3] chooses a random group, effectively limiting the set from which the intermediate switch is selected to one switch from each group. Slim Fly uses the traditional idea of Valiant routing, selecting its intermediate switch uniformly at random from all switches in the system other than the message source and destination.

In this paper, we examine an inefficient aspect of Valiant routing and show how a slight improvement in the algorithm can yield improved performance. The inefficiency is that, depending on the choice of intermediate switch, Valiant routing can cause a message to route in a loop. There are three ways that this can happen in a Slim Fly network. One of these is presented in Figure 1; we return to the others in Section 3. In the depicted case, the message wants to go from switch s to switch d, to which it is directly connected. The intermediate switch i chosen for the Valiant route, however, is switch i, a switch whose own route to switch d goes through switch s. Thus, the message is first sent to switch i, then back to switch s before traversing the same link to switch d that it would have taken with minimal routing.

Our solution to the problem of looping routes is to modify the random selection of the intermediate switch so that it only selects from switches that do not cause looping. Thus, we retain the benefits of Valiant routing without this inefficiency.

* *Work performed while a student at Knox College.*

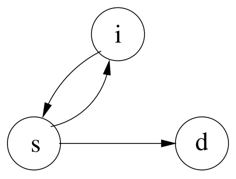

Figure 1. First case where Valiant can route messages in a loop. Switch s wants to send a message to switch d and switch i is chosen as the random intermediate switch.

Our modification to Valiant routing also has immediate implications for adaptive routing algorithms, which choose on a per-message basis whether to use minimal or Valiant routing. Switching between the algorithms rather than sticking with one is a recognition of the tradeoff between them; Valiant routing spreads the network load more evenly, but increases network load by using additional hops for each message. A common framework is to use minimal routing unless congestion is detected on the minimal route, in which case Valiant routing is used in an effort to avoid it. The decision can be made at the source switch (e.g. UGAL [3]) or later along the minimal route (e.g. PAR [13]).

Specifically, our contributions are the following:

- The new version of Valiant's algorithm that avoids the looping case described above and two others described in Section 3.
- An analysis of how often the looping patterns occur in a regular diameter-2 graph meeting the Moore bound.
- Evaluation of the effect of our improvement on both pure Valiant routing and when Valiant routing is used as a building block for the adaptive algorithm UGAL-L.

The rest of this paper is organized as follows. Section 2 formally defines the Slim Fly topology. Section 3 goes into greater depth about issues with Valiant routing as described for new topologies and defines the improved Valiant routing algorithm. Section 4 describes our experimental results. Section 5 reviews related work. Finally, Section 6 discusses future work.

2. Slim Fly

Before presenting Slim Fly networks, we first explain the Moore bound [9]. This concerns graphs with a fixed *diameter*, the farthest distance between a pair of vertices. The Moore bound states that a graph with diameter D and vertices of degree k has at most

$$1 + k \sum_{i=0}^{D-1} (k-1)^i \tag{1}$$

vertices.

To see where this comes from, start with a single vertex. It has k neighbors. Each of these has $(k-1)$ other neighbors (the degree is k, but we exclude the edge back to the starting vertex). Those are adjacent to $(k-1)^2$ other vertices. Continuing this process up to the graph diameter gives Expression 1. Note that a graph can have this number of vertices only if every vertex encountered during this process is unique; if any duplicates are encountered, the graph size will be smaller. It is not known how to construct graphs meeting the Moore bound in general.

In order to design large networks with a given diameter, it is natural to base them on the graphs known to approach the Moore bound. Each vertex is the graph becomes a switch which, in addition to the network links represented by graph edges, is connected to a collection of processing nodes. This is the design of Slim Fly networks, which are based on MMS graphs [8], which have diameter 2.

To describe the construction of MMS graphs, we mainly follow the presentation of Besta and Hoefler [7], which drew on previous descriptions [8], [14], [15].

The graph will be based on operations in the field $\mathbb{F}_q$, whose elements are the integers 0 through $(q-1)$. It supports addition and multiplication mod q.

The selection of q determines the system size, which will be $2q^2$. Conceptually, each switch is assigned a triple of coordinates, though these do not represent the physical layout of the system. The first coordinate is either 0 or 1. The other two coordinates are numbers mod q, denoted either x and y if the first coordinate is 0 or m and c if it is 1.

The value q must be selected as a prime power that can be represented as $q = 4\omega + \delta$, where $\delta \in \{-1, 0, 1\}$. Then, one must find a *primitive element* ξ of $\mathbb{F}_q$, i.e. a value such that every element of $\mathbb{F}_q$ can be expressed as ξ to some power. Then, depending on the value of δ, two sets X and X' are constructed as follows:

- If $\delta = -1$, then
$$\begin{aligned} X &= \{1, \xi^2, \xi^4, \ldots, \xi^{2\omega-2}\}, \text{ and} \\ X' &= \{\xi, \xi^3, \xi^5, \ldots, \xi^{2\omega-1}\} \end{aligned}$$
- If $\delta = 0$, then
$$\begin{aligned} X &= \{1, \xi^2, \xi^4, \ldots, \xi^{4\omega-2}\}, \text{ and} \\ X' &= \{\xi, \xi^3, \xi^5, \ldots, \xi^{4\omega-1}\} \end{aligned}$$
- If $\delta = 1$, then
$$\begin{aligned} X &= \{1, \xi^2, \xi^4, \ldots, \xi^{q-3}\}, \text{ and} \\ X' &= \{\xi, \xi^3, \xi^5, \ldots, \xi^{q-2}\} \end{aligned}$$

Finally, these sets are used to connect the switches using the following three types of edges:

(1) switches $(0, x, y)$ and $(0, x, y')$ connect iff $y - y' \in X$
(2) switches $(1, m, c)$ and $(1, m, c')$ connect iff $c - c' \in X'$
(3) switches $(0, x, y)$ and $(1, m, c)$ connect iff $y = mx + c$

Note that the first type of edges connect pairs of switches with 0 in the first coordinate, the second type connect pairs of switches with 1 in the first coordinate, and the third type connect between switches that differ in the first coordinate. Figure 2 shows some of these edges for a MMS graph with

$q = 5$. To keep the figure readable, only some edges of Type 3 are shown.

Observe that the looping situation shown in Figure 1 can happen in this graph; if the source is switch $(0, 0, 0)$, the destination is switch $(1, 0, 0)$, and the intermediate switch is switch $(0, 0, 1)$ then the message will be routed using a Type 1 edge to switch $(0, 0, 1)$ and back, followed by a type 3 edge to the destination.

3. Routing Algorithms

Now we present the remaining cases where Valiant can route messages in a loop. Recall that Figure 1 shows the first case, in which the message is routed to the intermediate switch and then back through the source switch on the way to its destination. The second case, depicted in Figure 3, is the mirror image of this. In this case, the intermediate switch i is adjacent to the destination switch d. This causes the message to be routed through the destination switch d on the way to switch i and then back to d, creating a loop as the message travels both directions along the edge between switches d and i. Unlike the others, this case can be easily avoided by having switches intercept arriving messages intended for its own nodes rather than forwarding them on; this was the behavior of our simulator even before we considered the looping cases.

The second loop case can also happen in the Slim Fly network depicted in Figure 2. Suppose the source is switch $(0, 0, 0)$, the destination is switch $(1, 0, 0)$, and the intermediate switch is switch $(1, 0, 2)$. Then the message is routed to the destination using a Type 3 edge, then to the intermediate switch and back using a Type 2 edge.

In the third and final case when Valiant sends a message around a loop, the source and destination switches are two hops apart, with another switch x between them. The intermediate switch i is a different neighbor of switch x. This causes the message to be routed through switch x on the way to switch i and then back through switch switch x on the way to the destination, creating a loop as the message traverses the edge between switches x and i in both directions. This situation is depicted in Figure 4. As with the other cases, this looping is never beneficial since the message traverses all the edges of the minimal route in addition to the loop.

To see the third loop case in the Slim Fly network depicted in Figure 2, let the source be switch $(0, 0, 1)$, the destination be switch $(1, 0, 0)$, and the intermediate switch be switch $(0, 0, 4)$. Then the message is routed through switch $(0, 0, 0)$ and to the intermediate switch using Type 1 edges, back to switch $(0, 0, 0)$ reusing one of those edges, and then on to the destination using a Type 3 edge.

Once the danger of loops is recognized, it is not hard to avoid them. Simply identify the intermediate switches that do not fall into one of the looping cases and select randomly from among those. In our simulator, we identified the non-looping intermediate switches for every (source, destination) pair during startup so the cost was insignificant for long runs. In a real system, the table of non-looping intermediates could be distributed among switches in the system.

3.1. Adaptive routing

In practice, pure Valiant routing is unlikely to be used because it performs poorly on "good" traffic patterns. In particular, if the traffic pattern is uniform random, then the randomization provided by Valiant routing does not give any benefit, but the extra hops it requires (compared to minimal routing) harms performance. Because of this, a better routing idea is to use adaptive routing, which tries to capture the good performance of minimal routing on uniform traffic and of Valiant routing on traffic with hot spots.

For our study, we use UGAL-L [7], based on a Dragonfly routing algorithm of the same name [3], to represent adaptive algorithms. The idea is to decide at the source switch whether each packet should be minimally or Valiantly routed. Several Valiant routes are generated and latency estimates are generated for each of them, as well as for the minimal route. Then the path with the lowest estimated latency is used. To create an estimate, the algorithm simply multiplies the length of the output buffer for the first hop along that route by the number of hops along the route, effectively estimating that each hop along the route will experience the same buffering delay. Obviously, this estimate can be crude, but it uses information easily available at the source switch; UGAL-L is a "local" version of UGAL-G, which does the same thing except that its estimate is the sum of the buffer lengths for each hop along the route.

Since UGAL-L uses Valiant routing as a subroutine, we easily incorporated our improvements of Valiant routing into it by using the restricted set of possible intermediate switches when selecting candidate Valiant routes.

4. Evaluation

To evaluate the impact of the non-looping version of Valiant routing, we use a simulator that models communication at the level of messages, buffers, and virtual channels. It was designed for the evaluation of new topologies, specifically Dragonfly and Slim fly. We validated it by repeating experiments reported in previous work [3], [7] and by comparison with the low-level behavior of booksim, the simulator associated with Dally and Towles [16].

For the simulations reported here, we set all the link latencies to 1, used buffers of size 32 for each VC, and consider the network saturated once the average latency reaches 100. The number of nodes per switch was set following the advice of Besta et al. [7] so that roughly two thirds of each switch's links go to other switches and roughly one third go to nodes. We ran our simulations on system sizes of between 150 and 3,042 nodes, which corresponds to values of q between 5 and 13.

In order to create a network load that would benefit from Valiant routing, we designed a pattern with a large number of hot spots distributed through the system, but also many edges that would be unused by minimal routing. The

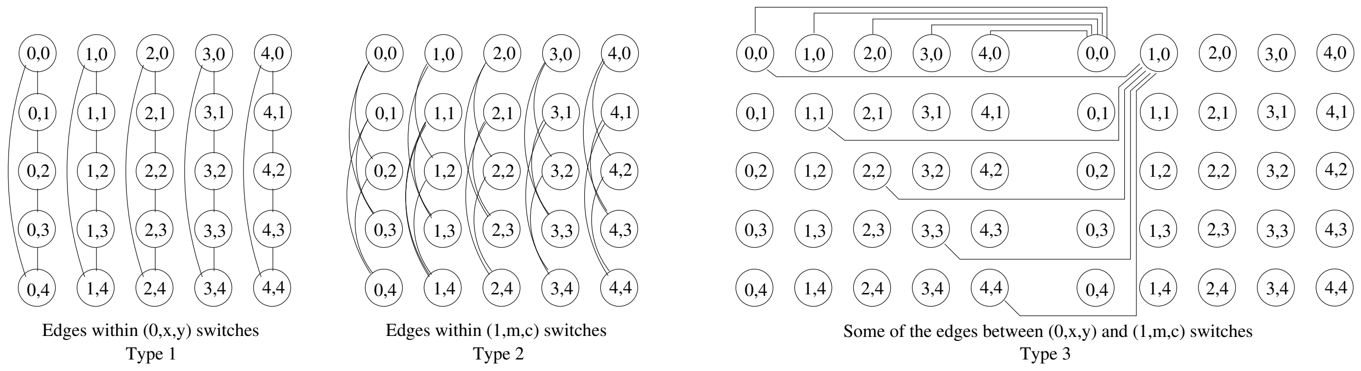
Edges within (0,x,y) switches
Type 1
Edges within (1,m,c) switches
Type 2
Some of the edges between (0,x,y) and (1,m,c) switches
Type 3

Figure 2. Depiction of edges of a Slim Fly network with $q = 5$ based on figure from [7]. The left two parts show the edges within the $(0, x, y)$ and $(1, m, c)$ switches respectively. The right two parts depict some of the edges connecting the two types of switches; many edges are omitted for clarity.

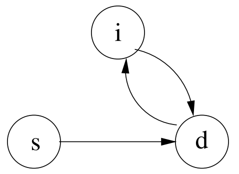

Figure 3. Second case where Valiant can route messages in a loop. Switch s wants to send a message to switch d and switch i is chosen as the random intermediate switch.

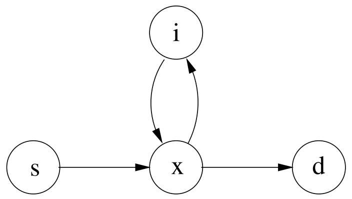

Figure 4. Third case where Valiant can route messages in a loop. Switch s wants to send a message to switch d, the minimal path to which goes through switch x. Switch i, another neighbor of switch x, is chosen as the random intermediate switch.

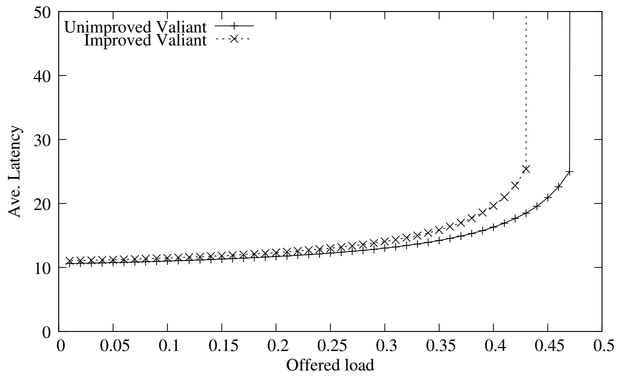

Figure 5. Latency as a function of offered load for Valiant routing with and without anti-looping improvements on a Slim fly system with $q = 13$.

		Valiant	
q	# nodes	unimp.	imp.
5	150	5.5	6.0
7	490	5.6	5.9
11	1,936	5.7	6.0
13	3,042	5.8	6.0

Figure 6. Average number of hops taken by messages for Valiant routing with and without anti-looping improvements.

switches are divided into sequences of adjacent switches. For example, the sequence S_0, S_1, S_2, …, S_m if each S_i is adjacent to S_{i-1} and S_{i+1}. Then, for each value of i, each node attached to switch S_i sends a message to a randomly chosen node of switch S_{i+1} with probability equal to the desired load. Thus, if minimal routing were chosen, only two edges from each switch would be used, one in each direction in the sequence. Since each switch is attached to multiple nodes, however, this pattern can easily saturate the edges that it does use.

4.1. Pure Valiant routing

Figure 5 shows the latency of both original Valiant routing and our improved version as a function of the offered load. The results are not encouraging; the “improved” version consistently has a higher average latency and saturates at a lower offered load than the original version. The same general behavior holds for the other system sizes we tried as well.

This behavior seems to occur because the improved Valiant is consistently giving messages longer routes, causing them to consume bandwidth on more edges. This can be seen in Figure 6, which gives the average number of hops for each version of the algorithm on different system sizes. This has been previously shown to correlate with job running time in the context of task mapping (e.g. [17], [18]). Note that the numbers of hops for Valiant routing is independent of the offered load because the Valiant algorithm does not consider the system load when choosing routes.

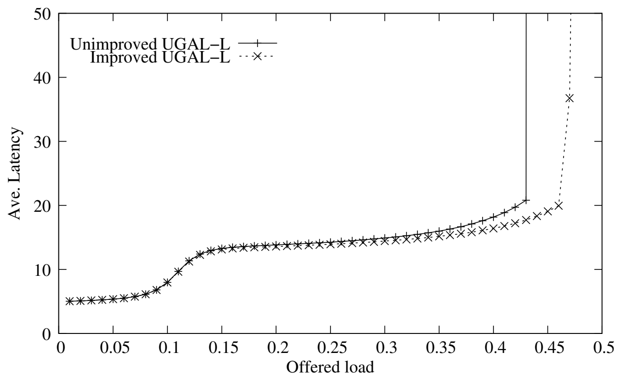

Figure 7. Latency as a function of offered load for UGAL-L routing with and without anti-looping improvements on a Slim fly system with $q = 13$.

4.2. UGAL-L routing

Even though the anti-looping improvements harm the performance of Valiant routing, they yield an improvement when the improved Valiant algorithm is used as a building block of UGAL-L. Figure 7 shows the latency of both versions of UGAL-L as a function of offered load; this is the analog of Figure 5. Now it is the improved version that achieves a small but growing improvement in latency, leading it to saturate later (at an offered load of 0.47 vs 0.43 for the unimproved version).

The other difference visible in Figure 7 is the S-curve occurring at an offered load around 0.1. This marks the transition from UGAL-L using minimal routes nearly always (for approximately 99% of messages at offered load 0.1) to a significant amount of Valiant routing (approximately 30% of messages at offered load 0.15).

Looking at the percentage of messages that use Valiant routing, the unmodified version of UGAL-L consistently does this at a higher rate. That may be part of the explanation for the difference between the algorithms since Valiant paths are generally longer. Looping case 2 is an exception, but looping case 1 is particularly tempting for UGAL-L since the algorithm assesses a possible path by looking at the buffer length of the first edge and our communication pattern congests only one of each switch's edges. Eliminating looping case 1 from consideration may make the improved UGAL-L less likely to choose a Valiant path.

The average number of hops has more interesting behavior with UGAL-L than Valiant. For both the improved and original versions, it starts at 3 hops (two between nodes and switches, plus 1 between the source and destination switches). The value generally increases with the load, but remains below the values for Valiant routing shown in Figure 6 even at saturation. For all but one of the sizes considered, the average number of hops was the same at saturation; the exception was the $q = 5$ size when it was 5.1 hops for the version with the improved Valiant algorihtm vs 5.2 hops for the original version. The trend was that the average number of hops increased with the system size.

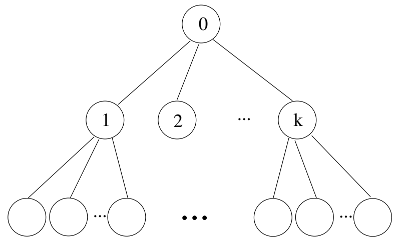
0
1
2
...
k
...
...
•••
...
k−1 unique neighbors each.
All missing edges go between nodes on the bottom level

Figure 8. Sketch of a k-regular graph meeting the Moore bound for diameter 2.

4.3. Relationship with system size

A natural question to ask about the anti-looping approach is how well it will perform on larger systems. The following theorem provides some reason for concern since it shows that in at least one circumstance, the percentage of messages affected decreases with system size:

Theorem 1. On a graph meeting the Moore bound for degree k and diameter 2, a fraction $1/(k + 1)$ of the messages sent between switches uniformly at random will fall into one of the three looping cases.

Proof: A k-regular diameter 2 graph meeting the Moore bound must have $1 + k + k(k - 1) = k^2 + 1$ switches. From a given source switch, k of these are at distance 1 and $k(k - 1)$ are at distance 2. This graph is depicted in Figure 8; the missing edges must all connect switches on the bottom level and there must be no other switches.

Our first looping case occurs when the source and destination switches are adjacent and the intermediate switch is a different neighbor of the source (e.g. the source, destination, and intermediate switch are 0, 1, and 2–k in Figure 8, respectively). The source and destination are adjacent $k/k^2 = 1/k$ of the time. When they are, the first looping case occurs for $k - 1$ of the choices of intermediate switch, once for each of the $k - 1$ neighbors of the source other than the destination. This is out of the $k^2 - 1$ switches other than the source and destination that can be selected as the intermediate. Thus, the first looping case occurs for $(1/k)((k - 1)/(k^2 - 1)) = 1/(k(k + 1))$ of the messages.

By the same reasoning, the third looping case, where the intermediate switch is a neighbor of the destination, occurs for the same fraction of messages.

The second looping case requires the source and destination switches to be at distance two, with the intermediate switch being a different neighbor of the switch x between them (e.g. source, destination, and intermediate switch are 1, 2, and 3–k in Figure 8, respectively). There are $k - 2$ neighbors of x other than the source and destination. Thus, the fraction of messages that this occurs with is $((k - 1)/k)((k - 2)/(k^2 - 1)) = (k - 2)/(k(k + 1))$.

Combining the three cases gives the desired result. □

Note that this theorem applies to graphs meeting the Moore bound. Since it is not known if these exist except

q	# nodes	Saturation point unimp.	imp.	% diff
5	150	0.62	0.66	6.4
7	490	0.48	0.53	10.4
11	1,936	0.42	0.47	11.9
13	3,042	0.43	0.47	9.3

Figure 9. Saturation point (last load for which average latency is below 100) for UGAL-L with and without the anti-looping improvements on Slim fly systems of various sizes.

for certain (small) values of k, actual networks will have fewer switches and thus have a higher fraction of messages falling into one of the looping cases.

Even so, the theorem suggests one way that our improvements could become less important with increasing system size; this trend will lead to continuing increases in the switch degree, which will reduce the number of messages benefiting from the improvements. That said, larger systems are also likely to see hotter hot spots as the number of nodes per switch increases, so this theorem is not enough to fully predict performance on larger systems. Based on our simulations, we did not see any loss of benefit; Figure 9 shows the saturation point for the original and improved UGAL-L algorithm on each of the sizes we considered. As you can see, the absolute difference between the saturating offered loads remained essentially the same.

5. Related work

Valiant [10] originally applied the idea of routing via a randomized intermediate node to all-to-all communication in a hypercube. That work showed that each message was blocked by congested links for $O(\log n)$ time steps with high probability. The original algorithm traversed the dimensions in a random order, but Valiant and Brebner [11] later simplified the analysis by assuming a fixed dimension order.

As mentioned previously, the original proposal for Valiant routing on the Dragonfly system [3] only misroutes to an intermediate group, not an individual switch. This can cause hot spots on edges within intermediate groups [19], the discovery of which led to work applying a Valiant-style misrouting locally within a group [19], [20]. Dragonfly systems have also been the target of other work on adaptive routing that uses Valiant as a building block (e.g. [13], [21]).

As a final comment on the Dragonfly topology, we note that the looping cases cannot occur on this topology as it was originally proposed, with fully connected groups and Valiant routing only to an intermediate group rather than a specific switch. It can occur, however, if the misrouting strategy is changed or in implementations of Dragonfly that do not fully connect the groups. For example, due to the way its switches are arranged into chasis, the Cray XC [22], [23] organizes the switches of each group into a 16×6 grid, with each switch connected to those sharing its row or column.

Valiant routing or variations thereof has also been proposed for other network topologies. Stacked Single-Path Trees (SSPT) [4] use a variation that restricts the choice of intermediate switch. Valiant routing was one of the alternatives proposed for HyperX [5], [24], as were adaptive strategies that decide to misroute after the source switch.

6. Discussion

We have shown that removing the loops from Valiant routes can allow UGAL-L to support a load around 12% higher on Slim fly networks. This is a significant advantage even if the latency of loads below saturation sees only modest improvements. We are interested in continuing to expore these ideas with other communication patterns and other adaptive routing algorithms. A broader question concerns the frequency of loops in other topologies and whether these kinds of ideas could be applied in other settings where Valiant routing has been used.

Acknowledgments

This work was partially supported by the National Science Foundation under grant CNS-1423413 and the Paul K. and Evalyn Elizabeth Richter Memorial Funds.

References

[1] S. Scott, D. Abts, J. Kim, and W. Dally, “The BlackWidow high-radix Clos network,” in *Proc. 33rd Ann. Intern. Symp. Comput. Arch. (ISCA)*, 2006, pp. 16–28.

[2] R. Barriuso and A. Knies, “108-port InfiniBand FDR SwitchX switch platform hardware user manual,” 2014.

[3] J. Kim, W. Dally, S. Scott, and D. Abts, “Technology-driven, highly-scalable dragonfly topology,” in *Proc. 35th Ann. Intern. Symp. Comput. Arch. (ISCA)*, 2008, pp. 77–78.

[4] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. Hoefler, “Cost-effective diameter-two topologies: Analysis and evaluation,” in *Proc. Conf. High Performance Computing, Networking, Storage and Analysis (SC)*, 2015.

[5] J. Ahn, N. Binkert, A. Davis, M. McLaren, and R. Schreiber, “HyperX: Topology, routing, and packaging of efficient large-scale networks,” in *Proc. ACM/IEEE Conf. High Performance Computing (SC)*, 2009.

[6] M. Koibuchi, H. Matsutani, H. Amano, D. Hsu, and H. Casanova, “A case for random shortcut topologies for HPC interconnects,” in *Proc. 39th Ann. Intern. Symp. Comput. Arch. (ISCA)*, 2012, pp. 177–188.

[7] M. Besta and T. Hoefler, “Slim fly: A cost effective low-diameter network topology,” in *Proc. Conf. High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.

[8] B. McKay, M. Miller, and J. Širán, “A note on large graphs of diameter two and given maximum degree,” *J. Combinatorial Theory, Series B*, vol. 74, no. 1, pp. 110–118, 1998.

[9] M. Miller and J. Širán, “Moore graphs and beyond: A survey of the degree/diameter problem,” *Electronic J. Combinatorics*, vol. 61, pp. 1–63, 2005.

[10] L. Valiant, “A scheme for fast parallel communication,” *SIAM J. Computing*, vol. 11, no. 2, pp. 350–361, 1982.

[11] L. Valiant and G. Brebner, “Universal schemes for parallel communication,” in *Proc. 13th ACM Symp. on Theory of Computing (STOC)*, 1981, pp. 263–277.

[12] R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge University Press, 1995.

[13] N. Jiang, J. Kim, and W. Dally, "Indirect adaptive routing on large scale interconnection networks," in *Proc. 36th Ann. Intern. Symp. Comput. Arch. (ISCA)*, 2009, pp. 220–231.

[14] J. Šiagiová, "A note on the McKay-miller-širán graphs," *J. Combinatorial Theory, Series B*, vol. 81, pp. 205–208, 2001.

[15] P. Hafner, "Geometric realisation of the graphs of mckay-miller-Širáň," *J. Combinatorial Theory, Series B*, vol. 90, pp. 223–232, 2004.

[16] W. Dally and B. Towles, *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.

[17] M. Deveci, S. Rajamanickam, V. Leung, K. Pedretti, S. Olivier, D. Bunde, Ü. Çatalyürek, and K. Devine, "Exploiting geometric partitioning in task mapping for parallel computers," in *Proc. 28th IEEE Intern. Parallel and Distributed Processing Symp. (IPDPS)*, 2014.

[18] T. Hoefler and M. Snir, "Generic topology mapping strategies for large-scale parallel architectures," in *Proc. 25rd ACM Intern. Conf. Supercomputing (ICS)*, 2011.

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

[20] M. García, E. Vallejo, R. Beivide, M. Odriozola, and M. Valero, "Efficient routing mechanisms for dragonfly networks," in *Proc. 42nd Intern. Conf. Parallel Processing (ICPP)*, 2013, pp. 582–592.

[21] P. Fuentes, E. Vallejo, M. García, R. Beivide, G. Rodríguez, C. Minkenberg, and M. Valero, "Contention-based nonminimal adaptive routing in high-radix networks," in *Proc. 29th IEEE Intern. Parallel and Distributed Processing Symp. (IPDPS)*, 2015.

[22] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, "Cray XC series network," Cray, Inc., White paper, 2012.

[23] 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 *Proc. Conf. High Performance Computing, Networking, Storage and Analysis (SC)*, 2012, p. 103.

[24] L. Bhuyan and D. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Comput.*, vol. C-33, no. 4, pp. 323–333, 1984.