Enhanced UGAL Routing Schemes for Dragonfly Networks

Ram Sharan Chaulagain and Xin Yuan
Department of Computer Science
Florida State University
Tallahassee, Florida
{chaulaga,xyuan}@cs.fsu.edu

ABSTRACT

The Dragonfly networks have been adopted in the current supercomputers, and will be deployed in future generation supercomputers and data centers. Effective routing on Dragonfly is challenging. Universal Globally Adaptive Load-balanced routing (UGAL) is the state-of-the-art routing algorithm for Dragonfly. For each packet, UGAL selects either a minimal path or a non-minimal path based on their estimated latencies. Practical UGAL makes routing decisions with local information, deriving the estimated latency for each path from the local queue occupancy and path hop count information. In this work, we develop techniques to improve the accuracy of the latency estimation for UGAL with local information, which results in more effective routing decisions. In particular, our schemes are able to proactively mitigate the potential network congestion with imbalanced network traffic. Extensive simulation experiments using synthetic traffic patterns and application workloads demonstrate that our enhanced UGAL schemes significantly improve the routing performance for many common traffic conditions.

CCS CONCEPTS

• **Computer systems organization → Interconnection architectures**.

KEYWORDS

Interconnection network, Dragonfly, UGAL routing, Local information

ACM Reference Format:
Ram Sharan Chaulagain and Xin Yuan. 2024. Enhanced UGAL Routing Schemes for Dragonfly Networks . In *Proceedings of the 38th ACM International Conference on Supercomputing (ICS '24), June 04–07, 2024, Kyoto, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3650200.3656602

1 INTRODUCTION

The Dragonfly networks [21] have been deployed in the current generation supercomputers such as the Frontier supercomputer [23]. They will be deployed in the future supercomputers and data centers [16]. The Dragonfly topology has a low diameter while maintaining high path diversity. To exploit the path diversity, the Universal Globally Adaptive Load-balanced routing (UGAL) [18, 21] has been developed for Dragonfly.

UGAL routes each packet either through a minimal path or a non-minimal path based on their estimated latencies. It has been shown that with accurate global latency information, UGAL is able to achieve high routing performance on Dragonfly [18, 21]. Practical UGAL, however, does not have the accurate global latency information, and uses local queue occupancy and hop count information to approximate the latency of a path.

Algorithm 1 shows UGAL with local information (UGAL-L) [21]. The latency for the minimal path is approximated with $q_m \times H_m$ where q_m is the queue occupancy and H_m is the hop count for the minimal path. Similarly, the latency for the non-minimal path is approximated with $q_{nm} \times H_{nm}$ where q_{nm} is the queue occupancy and H_{nm} is the hop count for the non-minimal path. The bias b in the routing allows the routing scheme to favor minimal paths or non-minimal paths, depending on its value. Clearly, such approximations may not reflect the actual latencies of the paths, especially in the case when there is congestion on links along the paths that are not directly connected to the router, which leads to sub-optimal routing decisions.

Due to the importance of UGAL with local information, many improvements have been proposed, including allowing adaptive routing decisions to be made after the source router [11, 12, 18], using a static threshold to decide whether to direct the traffic to the non-minimal path [10], using average queue occupancy to overcome "phantom" congestion [27], and dynamically adjusting the bias value based on traffic condition [9, 20]. None of the existing proposals challenge the latency approximation formula used in UGAL-L, as shown in Algorithm 1.

In this work, we show that the main reason that causes UGAL-L to make sub-optimal routing decisions with imbalanced traffic is the inaccurate approximation of minimal path latency under such conditions. To overcome this problem, we propose to change the latency approximation formula for minimal paths by introducing a contention term that reflects the effect of link congestion with imbalanced traffic. We further develop techniques to derive the contention term using information local to each router. Based on our new latency approximation method, we develop an enhanced UGAL with a local information scheme called UGAL-LE. By incorporating the contention term, UGAL-LE is able to proactively mitigate the potential network congestion for imbalanced traffic. Moreover, our technique for improving the accuracy of latency approximation is orthogonal to the techniques in the recently developed schemes that dynamically adjust the bias to achieve high performance. We combine our techniques with the Decoupled Gradient descent-based Bias global adaptive routing algorithm (DGB) [20], and obtain an enhanced DGB, which we call EDGB. The simulation experiments

ICS '24, June 04–07, 2024, Kyoto, Japan

ACM ISBN 979-8-4007-0610-3/24/06
https://doi.org/10.1145/3650200.3656602

show that UGAL-LE and EDGB significantly outperform other UGAL-based routing schemes, including DGB, for many common traffic conditions. In many cases, UGAL-LE and EDGB perform very similar to the theoretical UGAL with global information (UGAL-G) [21]. The contributions of this work include the following:

- We identify that the inaccurate latency approximation for minimal paths is the major cause of performance issues with UGAL-L when the traffic is imbalanced.
- We develop UGAL-LE with improved latency approximation for minimal paths using local information. Additionally, we develop EDGB, an enhanced DGB scheme, by combining our techniques with DGB.
- We evaluate our proposed UGAL-LE and EDGB with existing UGAL-based routing schemes using synthetic traffic patterns and real application workloads. Our techniques significantly improve the baseline routing schemes in many situations, which indicates that our techniques are robust and effective.

Algorithm 1: UGAL with local information (UGAL-L)

$q_m(q_{nm})$: queue occupancy for minimal (non-minimal) path
$H_m(H_{nm})$: hop count for minimal (non-minimal) path
b: bias or constant offset

```
if q_m × H_m ≤ q_nm × H_nm + b then
    route minimally
else
    route non-minimally
```

2 BACKGROUND

2.1 Dragonfly

Detailed description of the Dragonfly topology can be found in [21]. Here, we briefly introduce the topology for the completeness of this paper.

In a Dragonfly topology, multiple routers are interconnected to form a group where routers within a group collectively act as a virtual router. The groups are interconnected to form the topology [21]. A Dragonfly topology has four parameters: p, a, h, and g. Each router has p channels to connect to terminals, $a-1$ local channels to connect to other routers in the same group, and h global channels to connect to routers in other groups. g is the number of groups in the Dragonfly. The router radix is, thus, $k = p + h + a - 1$. A load balanced Dragonfly should have $a = 2p = 2h$ [21]. We will use notation $dfly(p, a, h, g)$ to represent such a Dragonfly topology. Each group consists of a routers interconnected via an intra-group interconnection network. In canonical Dragonfly, the intra-group topology is the fully-connected topology, although other intra-group topologies such as the 2D all-to-all structure in Cray Aries[3] are also allowed. This study focuses on canonical Dragonfly with fully-connected groups.

The maximum-sized Dragonfly has exactly one global connection between each pair of groups. Each group has ah connections to other groups, and thus there are $g = ah+1$ groups in the maximum-sized Dragonfly. An example maximum sized Dragonfly with $p = 2$, $a = 4$, $h = 2$, and $g = 9$ is shown in Figure 1. Practical Dragonfly can be constructed with multiple global links connecting each pair of groups. When the number of global links connecting each pair of groups is l, the number of groups is $g = \frac{ah}{l} + 1$. Common global connectivity for both maximum-sized and practical Dragonflies can be found in [4].

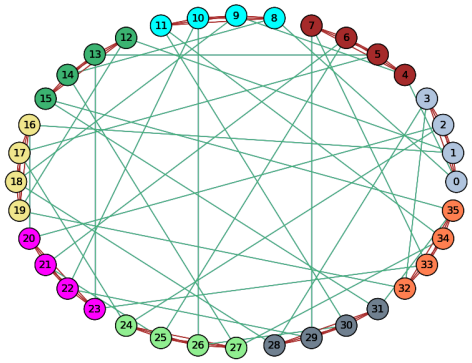

Figure 1: An example maximum sized Dragonfly topology *dfly(p = 2,a = 4,h = 2,g = 9)*

For Dragonfly topologies with a fully connected intra-group topology, the minimal path from a source router (S) to a destination router (D) has at most 3 hops: one hop in the source group to get to the router that has a global link connecting to the destination group, one hop for the global link (to the destination group), and one hop for the link in the destination group to get to the destination router. A non-minimal path from router S to router D is obtained by first randomly selecting an intermediate router (I) that is not in the source and destination groups, and combine the minimal path from S to I and then from I to D. In Figure 2, the path in the blue dashed line is a minimal path from S to D, while the path in the red solid line is a non-minimal path.

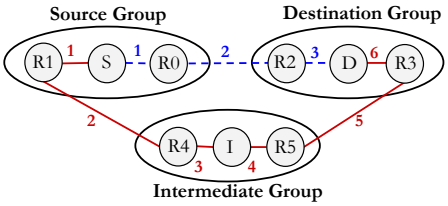

Figure 2: Minimal and non-minimal paths in Dragonfly

2.2 UGAL

The routing scheme that always sends traffic using minimal paths is the minimal routing (MIN), while the routing scheme that always sends traffic through non-minimal paths is the Valiant Load-balanced routing (VLB) [21]. MIN routing performs well for uniform traffic. However, for adversarial traffic patterns such as shift traffic patterns, it performs poorly. On the other hand, VLB routing performs well for adversarial traffic, but poorly for uniform traffic [21].

Universal Globally-Adaptive Load-balanced routing (UGAL) combines MIN and VLB in one routing system. The idea is to route packets with minimal paths when the traffic is uniform, and with non-minimal paths when the traffic is adversarial. In [21], two versions of UGAL schemes are discussed: UGAL with global information (UGAL-G) and UGAL with local information (UGAL-L). UGAL-G is a theoretical (not practical) routing scheme that assumes that the accurate global latency information is available. It was shown that UGAL-G achieves high performance on Dragonfly, while UGAL-L, depicted in Algorithm 1, performs significantly worse than UGAL-G in many situations.

3 RELATED WORK

In the seminal Dragonfly paper [21], Kim et al. discussed UGAL and suggested the random selection of intermediate groups for non-minimal routing in order to load-balance adversarial traffic patterns. Jiang et al. [18] proposed several indirect global adaptive routing schemes where adaptive routing decisions are made using information that is not directly available at the source router to approximate UGAL-G. Since then, various enhancements for the UGAL routing scheme have been developed. Garcia et al.[11] identified local congestion inside intermediate groups and improved UGAL by randomly selecting intermediate nodes for non-minimal routing. Garcia et al. [11] proposed the on the fly adaptive routing (OFAR) that can dynamically change the packet routes from minimal to non-minimal on both intra- and inter-group communication. Additional congestion management mechanisms have been studied in OFAR-CM [13]. Opportunistic local misrouting (OLM) and restricted local misrouting (RLM) [12] are other in-transit adaptive routing schemes that offer deadlock-free mechanisms that work with virtual cut-through and wormhole flow-control. Thresholds were proposed to redirect traffic away from the minimal path when the local contention and buffer usage surpass a specific limit [10]. Won et al. [27] identified phantom congestion due to the higher link latency and estimated the delay using the history window approach. Topology custom UGAL (T-UGAL) has been proposed for practical Dragonfly networks. T-UGAL [24] employs a subset of all possible VLB paths in UGAL that has a smaller average path length. Various machine learning-based routing schemes for Dragonfly were also proposed [5, 19]. Q-adaptive [19] routing leverages multi-agent reinforcement learning (MARL) to train routers. Similarly, UGAL-ML [5], a deep learning-based routing algorithm, utilizes the local router information like queue occupancy, hop count, and link-usage statistic counters to make a routing decision. The bias was initially introduced in UGAL to filter out the transient load imbalances on the minimal path [18]. UGAL can leverage bias to favor minimal or non-minimal paths. Several techniques were proposed to improve UGAL performance by adapting the bias value to the traffic condition. Traffic pattern-based adaptive routing (TPR) enhances UGAL by utilizing local router counters to infer global traffic patterns based on different operating regions and a static bias value for each operating region [9]. Decoupled Gradient descent-based Bias adaptive routing (DGB) [20] proposed a gradient descent-based method to set bias values dynamically based on network congestion. Our approach is different from all of the existing work in that we employ a mechanism to more accurately approximate the latency for the minimal path by using local information to infer the link contention on the global link in the minimal path. This reduces the number of sub-optimal routing decisions and improves the performance.

4 PROPOSED TECHNIQUES

We will first discuss the issues with the latency approximation in UGAL-L, which gives the motivation and intuition of our proposed techniques. We will then describe our techniques in detail.

4.1 Motivation and intuition

Our proposed techniques are based on some unique features of UGAL and UGAL-L. We will discuss these features, which motivate and give the intuition of our proposed techniques. In practice, UGAL considers a small number of minimal paths and a small number of non-minimal paths as candidate paths for each packet. To simplify the discussion, we will assume that only one minimal path and one non-minimal path are considered. Our techniques can be extended naturally to the case when more than one minimal and non-minimal paths are considered.

Observation 1: With a UGAL based scheme like UGAL-L, a minimal path has a much higher chance of being considered to route a packet than a non-minimal path. This is because for each pairs of source-destination routers in different groups, the number of minimal paths in a Dragonfly network is much smaller than the number of non-minimal paths. For example, in the maximum sized $dfly(4, 8, 4, 33)$, for each pair of source and destination routers in different groups, the number of minimal paths is 1 while the number of non-minimal paths is 248. In $dfly(4, 8, 4, 9)$ with 4 links between each pair of groups, for each source and destination routers in different groups, the number of minimal paths is 4, and the number of non-minimal paths is 896. Since UGAL selects one minimal and one non-minimal path from all possible paths, it inherently has a much higher chance to consider a given minimal path than a given non-minimal path.

Observation 2: Although the latency estimations of both the minimal path and non-minimal path are used in making the routing decision in UGAL-L (Algorithm 1, Line 1), the impact of the inaccuracy in latency estimation for minimal path is very different from that for non-minimal path.

We will use an example to illustrate this observation. Consider the scenario shown in Figure 3 with three active flows from one group (source group) to a destination group: $S_0 \rightarrow D_0$, $S_1 \rightarrow D_1$, and $S_2 \rightarrow D_2$. Let us assume that the three flows send packets at the same rate. In this scenario, the global link between R_3 and R_4 can be under heavy load even before the local links in R_0, R_1, and R_2 are heavily loaded: the load of the global link shared by the three flows should be three times the load on the local link. In this case,

when UGAL-L approximates the latency for the minimal path with $q_m \times H_m$, it assumes that the load on the global link is the same as the local link and significantly underestimates the latency on the minimal path, which results in sub-optimal routing decisions being made.

During the time before the back pressure from router $R3$ cause buffers in routers R_0, R_1, and R_2 to fill up, UGAL-L will push too many packets in the minimal path and congested link due to the latency under-estimation. After the link is saturated (the buffer in the link between R_3 and R_4 is full), and the situation persists, back pressure from router R_3 will result in buffers in routers R_0, R_1, and R_2 being filled up. In this situation, UGAL-L's estimation is a reasonable estimation of the total queue length along the path. However, this estimation significantly under-estimates the latency in terms of time experienced by a packet. Given that three flows share the global link, the movement of packets in the local links is three times slower compared to the packets in a normal queue (without contention in the downstream link). Consequently, the delay experienced by a packet in the local link is three times greater than that in the other links. Hence, **in the congestion situation as depicted in Figure 3, due to the under-estimation of the minimal path latency, UGAL favors minimal path more and affects the routing decision of every packet in the affected flows, causing a persistent congestion situation.**

Now, let us consider the impacts of the under-estimation of the latency of non-minimal paths. In a situation shown in Figure 3, a congested link (the link from R_3 to R_4 in the figure) can also cause the latency of a non-minimal path that uses this link to be under-estimated, which also leads to sub-optimal routing decisions. However, such sub-optimnal routing decisions will not cause persistent congestion. In fact, they will not cause significant problems in the network. This is because for each flow there are many non-minimal paths. For example, in $dfly(4, 8, 4, 33)$, the number of non-minimal paths for each source-destination node is 248; and for each routing decision, one of the non-minimal paths will be randomly selected. Thus, a congested link such as the link from R_3 to R_4 in Figure 3 will only affect $1/248 = 0.4\%$ routing decisions for an active flow.

Hence, although one should try to improve the latency estimation for both minimal and non-minimal paths, the performance of UGAL-L is likely to be significantly improved by having more accurate latency estimation for minimal paths only. We will show later that there are effective methods to obtain more accurate latency estimation for minimal paths using local information.

Observation 3: With UGAL, it is possible to have a good estimation of the number of active flows that are sharing the global link (on their minimal path) with information local to the routers in the same group. This is because when the minimal paths are under heavy loads, UGAL will route packets through VLB paths. These VLB packets will be routed to a random intermediate router and will pass other routers in the same group with an equal probability. For example, consider routing packets in flow $S_0 \rightarrow D_0$ in Figure 3. When UGAL starts to route such packets through VLB paths, those packets will have an equal probability to be routed through routers R_1, R_2, and R_3. Hence, by observing through traffic, routers R_1, R_2, and R_3 will know that there is an active flow $S_0 \rightarrow D_0$. Since the number of routers within a group is not large, routers in a group can learn all active flows in their groups quickly when UGAL starts routing packets through VLB paths.

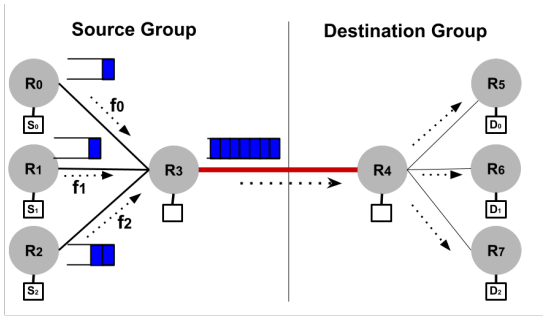

Figure 3: A motivating example

4.2 Contention factor

To more accurately approximate the latency for minimal paths, we introduce a contention term in the latency approximation formula. More specifically, we define a *contention factor* (C) for each outgoing global link in a group to be the number of routers with active flows to the destination group connected by the global link. For example, in Figure 3, the contention factor for link from R_3 to R_4 is 3. Note that the contention factor tracks the flows from router instead of from compute node to the destination group. So if there is another compute node attached to R_0 starts a flow to the destination group, the contention factor will still be 3. On the other hand, if there is another active flow from the compute node in R_3 to a compute node in the destination group, the contention factor for the link will be 4.

4.2.1 Obtaining and maintaining contention factors with local information. To obtain the contention factor for all global links of the group, each router maintains $a \times g$ counters, where a is the number of routers in the group, and g is the total number of groups. We will use $F_{i,j}$, $0 \leq i < a$ (the router id in the group) and $0 \leq j < g$ (group id) the counter that counts the number of packets from the router i proceeding to the destination group j that this router observed in a window of time (e.g. last 50 cycles).

Figure 4 illustrates the counters for $dfly(2, 4, 2, 9)$, where $a = 4$ and $g = 9$. When the router receives a packet, it checks whether the packet is originated from the same group. If the packet is originated from the same group, the router determines the source router (s) and the destination group (g), and updates the counter $F_{s,g}$. In the example in Figure 4, the router receives a packet from router 1 in the same group proceeding to destination group 5 and updates the entry $F_{1,5}$.

The routing algorithm can determine a counter threshold value (e.g. 5) for a flow to be considered active. The threshold is a parameter for the routing algorithm. For each group j, the counters F can give the number of routers with active flows to the group. We will denote this value as C_j, which is computed using equation 1.

$$C_j = \sum_{i=0}^{a-1} (if\ (F_{i,j} > threshold)\ then\ 1\ otherwise\ 0) \qquad (1)$$

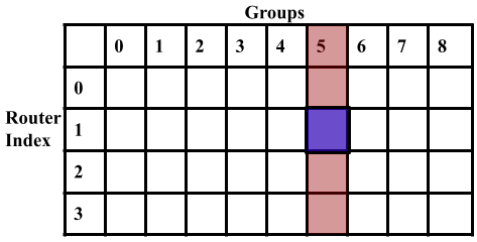

Figure 4: Example of the data structure maintained in each router to keep track of contention factors for $dfly(2, 4, 2, 9)$. The router receives a packet from router 1 in the same group to destination group 5, and updates entry $F_{1,5}$.

4.3 UGAL-LE

Our proposed UGAL-LE, an enhanced UGAL with local information, is depicted in Algorithm 2. UGAL-LE distinguishes between two types of routers. If the source router directly connects to the destination group (Lines 2 to 4), q_m is the queue occupancy on the global link. In this case, the q_m for the global link is accurate and $q_m \times H_m$ is used to approximate the path latency like the traditional UGAL-L. If the source router does not directly connect to the destination group (Lines 5 to 7), q_m is the queue occupancy on the local link. In this case, the queue occupancy on the global link should be approxmiated as $q_m \times C$: UGAL-LE adds the term $q_m \times max(C-1, 0)$ to the minimal path latency approximation to make the routing decision. Note that the contention term will make the approximated queue occupancy for the global link to be $q_m \times C$ since in the $q_m \times H_m$ term, q_m queue occupancy for the global link has already been counted.

Algorithm 2: UGAL-LE: an enhanced UGAL with local information

$q_m(q_{nm})$: queue occupancy for minimal (non-minimal) path
$H_m(H_{nm})$: hop count for minimal (non-minimal) path
C: contention factor for the global link in minimal path
R: source router making the routing decision
G_D: destination group

```
1 if R has a global link connecting to G_D then
2     if q_m × H_m ≤ q_nm × H_nm + b then
3         route minimally
      else
4         route non-minimally
  else
5     if q_m × H_m + q_m × max(C − 1, 0) ≤ q_nm × H_nm + b then
6         route minimally
      else
7         route non-minimally
```

4.4 Enhanced DGB (EDGB)

DGB [20] leverages both local information and global information for latency estimation. The local information (queue occupancy and hop count) is used to estimate the latency of minimal and nonminimal paths. The global information (end-to-end latency of each packet) is used to tune the bias value with an online gradient descent algorithm. Here, the global information is obtained from the feedback sent by the destination router through piggybacking. The bias value is dynamically adjusted with gradient descent, using the current bias value, the learning rate, and the gradient of latency with respect to bias. The gradient is derived using the global information. Details of DGB can be found in [20].

DGB allows the bias value to be dyanmically adjusted, its basic latency estimation is still based on local queue occupancy and hop count, like UGAL-L. Our technique, which improves the basic latency estimation for UGAL-L, is orthogonal to the techniques in DGB and can combine with DGB in a straight-forward manner. We develop the combined routing scheme and name it enhanced DGB (EDGB).

4.5 Discussion

Due to the statistical nature of UGAL to send packets over VLB paths, routers in the same group may not have the accurate information about all active flows from the group, resulting in inaccurate contention factor estimation. However, since contention factor adds a multicative term to the latency estimation for minimal paths, even when it is not accurate, it will still be much better than the latency estimation in UGAL-L. For example, consider a traffic pattern that contention factor for a global link is 5. Let us assume that a router only observes a contention factor of 2. Although the router's latency estimation is not accurate, it is still much higher and more accurate than the estimation with UGAL-L. In other words, UGAL-LE will make much less sub-optimal routing decisions than UGAL-L in such a situation.

5 EVALUATION WITH SYNTHETIC TRAFFIC PATTERNS

Extensive simulation experiments have been performed to evaluate the performance of our proposed routing schemes using both synthetic traffic patterns and HPC application workloads. Booksim2.0 [17], a cycle-accurate interconnect simulator, is used to study packet latency and saturation throughput of synthetic traffic patterns. CODES [6], an event-driven interconnect simulator, is used to evaluate the performance of the proposed routing schemes with HPC application workloads. In this section, we provide a comprehensive evaluation of our proposed routing scheme with synthetic traffic patterns using Booksim. In section 6, we will present the results for HPC workloads.

5.1 Methodology

5.1.1 Topology. We evaluated our routing schemes on topologies $dfly(4, 8, 4, 33)$ and $dfly(4, 8, 4, 9)$. More information for the topologies is summarized in the table 1. These topologies are built with 15 port switches and have the same intra-group connectivity with different numbers of groups and different numbers of global links connecting each pair of groups.

Topology	Num of nodes	No of routers	No of groups	links per group pair
$dfly(4, 8, 4, 33)$	1056	264	33	1
$dfly(4, 8, 4, 9)$	288	72	9	4

Table 1: Topologies Used in Experiments

5.1.2 Routing variations and simulator setting. We conducted a comprehensive evaluation of our enhanced UGAL schemes in diverse traffic scenarios, comparing their performance against other UGAL variants. Booksim offers a basic implementation of UGAL-L, serving as a baseline adaptive routing. We have added PAR, UGAL-G, DGB, UGAL-LE, and EDGB.

We have modified the Booksim simulator to support different sizes of dragonfly topologies and implement an absolute global links arrangement[14].

On Booksim, the routing parameters for UGAL-LE and EDGB are configured as follows. The window size for counters was set to 500 cycles and the counter threshold to decide whether a flow is active is set to 5. The counter threshold value is chosen empirically to effectively distinguish meaningful flow patterns from sporadic or random occurrences within a given window period. The learning rate for DGB is 0.1 (same as in [20]) and the learning rate for EDGB is set to 0.001. We use different learning rates for DGB and EDGB because EDGB incorporates an adjustment term for minimal path latency estimation, which acts as a negative bias that favors nonminimal paths. Setting the learning rate lower (0.001) for EDGB compared to DGB (0.1) helps stabilize the bias values, preventing abrupt changes and allowing them to stay within the optimal range for better performance.

The Booksim simulation results were collected over 10,000 cycles and the simulator is warmed up for 30,000 cycles, ensuring the steady state is reached. The router speedup is set to 2.5 to prevent the router from becoming the bottleneck. We have used 4 virtual channels for UGAL to ensure deadlock-free routing as illustrated by Won et al. [27] for all routing schemes except PAR uses 5 virtual channels. We have set local link latency to 10 cycles and global link latency to 15 cycles to mimic Cascade [8] interconnect where local link latency to global link latency is 1:1.5 ratio. Similar parameters are also used in [4, 24]. Buffer size is set to 256. We have used single flit packet in the study to avoid any potential flow control issues. The routing and simulation parameters are listed in the table 2.

5.1.3 Synthetic traffic patterns. Four different types of synthetic traffic patterns are used in the experiments with Booksim: random uniform, shift, random permutation, and mixed traffic. In a random uniform traffic pattern, each packet has an equal probability of being transmitted to any destination within the network. In the shift traffic pattern, all nodes connected to group(i) send all their traffic to group(i+1 mod g), stressing inter-group channels. In a random permutation traffic pattern, each node is assigned a unique random destination node. Mixed traffic patterns are mixed between shift/permutation and uniform traffic. We use two types of mixed traffic patterns: router-level mixed traffic where a certain percentage of routers are randomly selected and all compute nodes connected to these routers are participated in a shift/permutation pattern, and node-level mixed traffic where a certain percentage of compute nodes randomly selected to participate in the shift/permutation pattern. The rest of nodes in the mixed traffic patterns generate uniform traffic. Mixed traffic patterns help us to evaluate the behavior of routing algorithms when different parts of the network exhibit different behaviors.

Parameter	**Value**
number of virtual channels	5 for PAR 4 for other routing functions
buffer size	256
link latency	10 cycles (local) 15 cycles (global)
switch speedup	2.5
routing	UGAL-L, PAR, DGB, UGAL-LE, EDGB, UGAL-G
learning rate	0.1 - DGB, 0.001 - EDGB

Table 2: Routing and Simulation Parameters

5.2 Results and Discussion

In this section, we compare the performance of our proposed UGAL-LE and EDGB with that of UGAL-L, PAR, DGB, and UGAL-G on $dfly(4, 8, 4, 33)$ and $dfly(4, 8, 4, 9)$. The latency plot is widely used to understand the average packet latency under different offered loads. The x-axis in the latency plot represents the increasing offered load, and the y-axis represents the average packet latency. In our study, the unit of latency is a cycle. If a link latency is 10 cycles, 10 cycles are required to traverse the link.

Figure 5 shows the latency plot for different routing schemes with random uniform traffic on $dfly(4, 8, 4, 33)$. All of the routing schemes perform similarly with PAR slightly worse than others. This is because PAR can result in longer paths being taken for a packet, which affect the aggregate throughput. Although UGAL-LE and EDGB approximate the minimal path latency with contention factor, it does not affect their performance for uniform traffic. For this traffic pattern, the contention factor is 0. Thus, UGAL-LE is equivalent to UGAL-L and EDGB is equivalent to DGB.

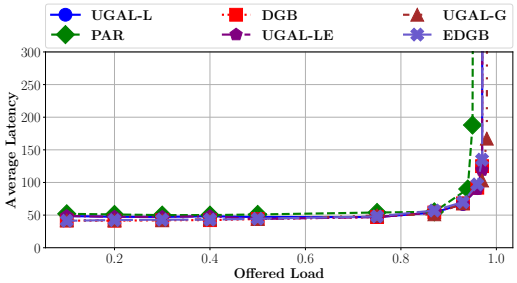

Figure 5: Uniform Random on $dfly(4, 8, 4, 33)$

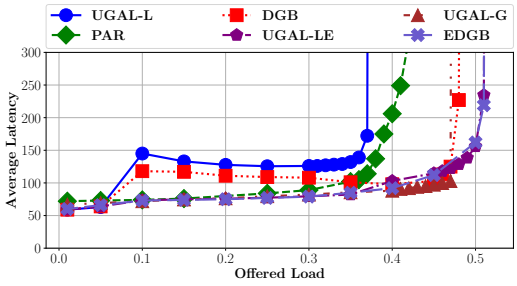

Figure 6: Adversarial shift traffic (ADV)

5.2.1 Adversarial Traffic. Figure 6 shows the latency plot of adversarial shift traffic pattern on $dfly(4, 8, 4, 33)$. UGAL-LE and EDGB improve over UGAL-L in latency for lower offered loads and achieve higher throughput than other routing schemes. Specifically, from this experiment, UGAL-L saturates at 0.37, while UGAL-LE achieves a higher saturation throughput of 0.51, reflecting a 37% increase in throughput than UGAL-L. Similarly, DGB saturates at 0.48, and EDGB attains a higher saturation throughput of 0.51, signifying a 6% improvement over DGB. PAR and UGAL-G obtain saturation throughput of 0.42 and 0.47, respectively. Several key observations underscore the effectiveness of enhanced UGAL variants. Firstly, the incorporation of the contention term significantly increases the minimum path delay estimation. This prompts UGAL-LE and EDGB to send more packets through nonminimal paths. Furthermore, since contention factor is very large for the shift pattern, UGAL-LE and EDGB effectively transforms into VLB routing, the best under such a traffic condition.

Second, UGAL-LE and EDGB have slightly higher throughput than UGAL-G in this experiment. UGAL-G has the up-to-date latency information but does not use the pattern information; UGAL-LE knows and uses the pattern information, but does not have the global information. Hence, both have advantages. In this experiment, UGAL-LE is able to direct slightly more traffic to VLB paths and achieves slightly higher throughput: at the saturated load, with UGAL-G, 6.5% of packets uses minimal paths while with UGAL-LE, 5.6% of packets uses minimal paths. Note that although UGAL-G uses current updated information to make routing decision, such information can still be outdated after the packet travels multiple hops, leading to sub-optimal routing performance.

Note that in this experiment, under medium loads, for some routing algorithms such as UGAL-L and DGB, higher offered loads (in the medium range) can result in lower packet latency. This phenomenon has also been observed in earlier studies [21]. It is due to the interaction among routing, back pressure, and buffer size as explained in [21].

Figure 7, shows the latency plot of random permutation traffic pattern on $dfly(4, 8, 4, 33)$. UGAL-LE and EDGB deliver improved performance than UGAL-L and DGB, respectively. Specifically, UGAL-L is saturated at 0.5, and UGAL-LE is saturated at 0.53, a 6% improvement. DGB saturates at 0.52 and EDGB saturates at 0.55, reflecting a 5% improvement over DGB. In addition, PAR is saturated at 0.54, and UGAL-G is saturated at 0.64. Although, enhanced UGAL variants outperform their standard UGAL counterparts, but not as much as the adversarial traffic pattern. This is due to reduced stress on inter-group channels. Moreover, EDGB avoids congested VLB paths utilizing dynamic nonmin-bias; EDGB performs better than UGAL-LE.

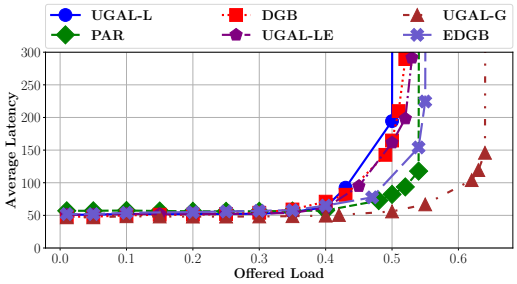

Figure 7: Random Permutation

5.2.2 Router Level Shift Mixed Traffic Pattern. Figures 8, 9, and 10 show the latency results for router-level shift mixed traffic with 25%, 50%, 75% of shift traffic on $dfly(4, 8, 4, 33)$, respectively. In Figure 8, UGAL-L saturates at 0.39, and UGAL-LE saturates at 0.53, achieving 35% improvement. DGB saturates at 0.48, and EDGB saturates at 0.55, reflecting a 14% increase in throughput. Additionally, PAR saturates at 0.54, and UGAL-G saturates at 0.58. In Figure 9, UGAL-L saturates at 0.33, and UGAL-LE saturates at 0.53 attaining a 60% improvement. DGB saturates at 0.48, and EDGB saturates at 0.53, obtaining a 10% improvement. In addition, PAR saturates at 0.51, and UGAL-G saturates at 0.56. In Figure 10, UGAL-L saturates at 0.37, and UGAL-LE saturates at 0.51, attaining 37% improvement. DGB saturates at 0.48, and EDGB saturates at 0.51, gaining by 6% in terms of throughput. Furthermore, PAR saturates at 0.44, and UGAL-G saturates at 0.5.

Note that a higher percentage of shift traffic should statistically result in lower throughput on average, it is not always the case for particular patterns since the throughput of a particular pattern depends not only on the percentage of the shift traffic, but also on how the nodes participating in the shift pattern are distributed in the network. Since Figures 8, 9, and 10 show the results of one particular pattern for each traffic distribution, the results do not follow the general trend.

In router level shift mixed traffic pattern, global link contention in the minimal path increases as the shift percentage increases. Since UGAL-LE and EDGB adds up the adjustment term proportional to global link contention, it proactively mitigates global link congestion and achieves better throughput.

Figures 11, 12, 13 show the latency results for node-level permutation mixed traffic patterns with 25%, 50%, 75% of random permutation traffic on $dfly(4, 8, 4, 33)$, respectively. In Figure 11, all routing schemes have similar performance. In Figure 12, we observe UGAL-L achieves the saturation throughput of 0.56, and UGAL-LE saturates at 0.61, obtaining an 8% improvement. DGB saturates at 0.58, and EDGB saturates at 0.65, acquiring 12% improvement. In addition, PAR saturates at 0.64, and UGAL-G saturates at 0.66.

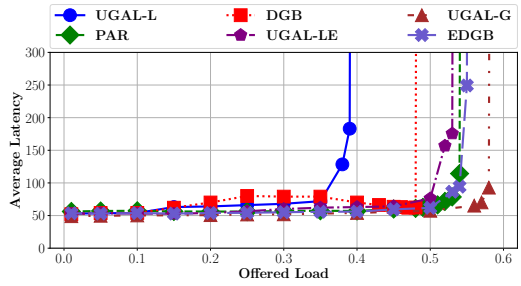

Figure 8: Router-level mixed pattern (25% shift traffic) on $dfly(4, 8, 4, 33)$

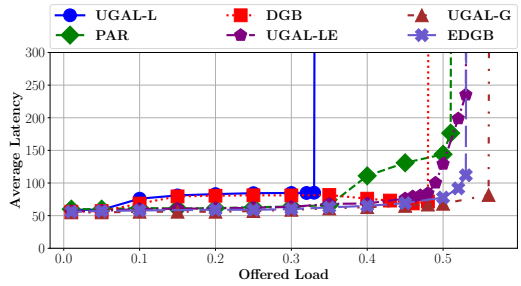

Figure 9: Router-level mixed pattern (50% shift traffic) on $dfly(4, 8, 4, 33)$

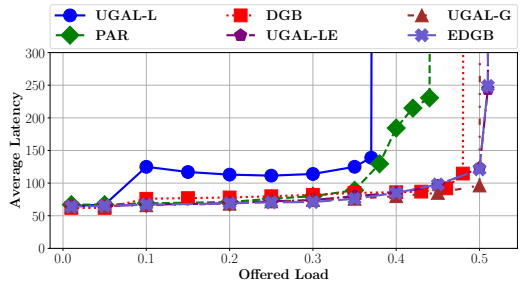

Figure 10: Router-level mixed pattern (75% shift traffic) on $dfly(4, 8, 4, 33)$

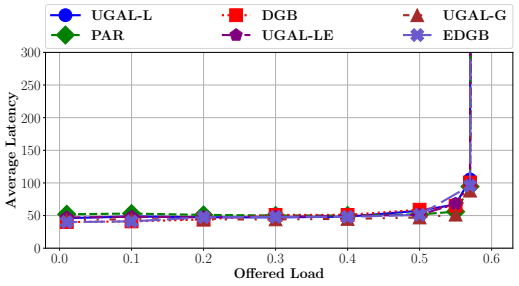

Figure 11: Node-level mixed traffic (25% random permutation) on $dfly(4, 8, 4, 33)$

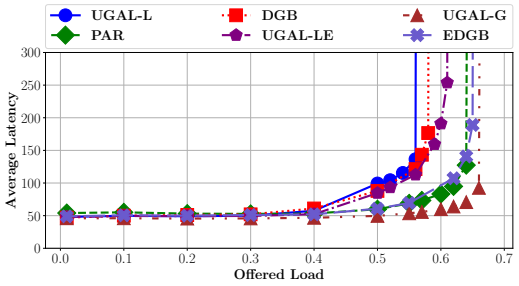

Figure 12: Node-level mixed traffic (50% random permutation) on $dfly(4, 8, 4, 33)$

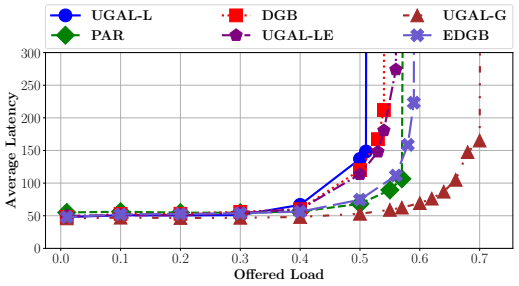

Figure 13: Node-level mixed traffic (75% random permutation) on $dfly(4, 8, 4, 33)$

Likewise, from Figure 13 UGAL-L saturates at 0.51, and UGAL-LE saturates at 0.56, reflecting a 9% improvement. DGB saturates at 0.54, and EDGB saturates at 0.59, gaining by 9% in terms of throughput. In addition, PAR saturates at 0.57, and UGAL-G saturates at 0.7. Since permutation mixed traffic causes less global link contention compared to mixed traffic with shift pattern, our technique results in less improvement. None the less, UGAL-LE and EDGB consistently improves over their corresponding baseline counterpart across many different traffic patterns, which indicates that our proposed techniques are robust and effective.

The relative performance of the routing schemes on $dfly(4, 8, 4, 9)$ is similar to that on $dfly(4, 8, 4, 33)$. We show the results for a few traffic patterns. Figure14 shows the results for shift traffic. UGAL-L saturates at 0.34, and UGAL-LE saturates at 0.54, achieving 58% higher throughput. DGB saturates at 0.48, and EDGB saturates at 0.55, gaining 14% higher throughput. Moreover, PAR is saturated at 0.46, and UGAL-G is saturated at 0.51. Figure15 shows the results for node-level mixed traffic with 50% shift traffic. UGAL-L saturates at 0.37, and UGAL-LE saturates at 0.65, reflecting a 75% increase in throughput. DGB saturates at 0.63, and EDGB saturates at 0.66, achieving 4% higher throughput. In addition, PAR is saturated at 0.63, and UGAL-G is saturated at 0.67. Figure16 shows the results for a random permutation. UGAL-L saturates at 0.76, and UGAL-LE saturates at 0.8, acquiring an 5% higher throughput.

DGB saturates at 0.78 and EDGB saturates at 0.81, signifying 3% improvement. PAR is saturated at 0.77, and UGAL-G is saturated at 0.89.

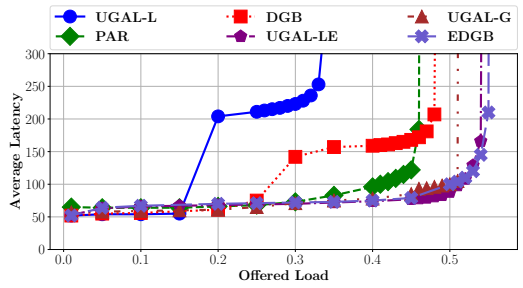

Figure 14: Shift on $dfly(4, 8, 4, 9)$

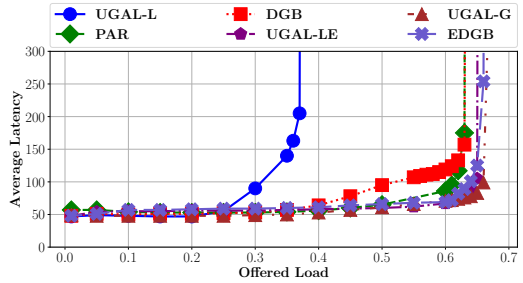

Figure 15: Node-level mixed traffic with 50% shift on $dfly(4, 8, 4, 9)$

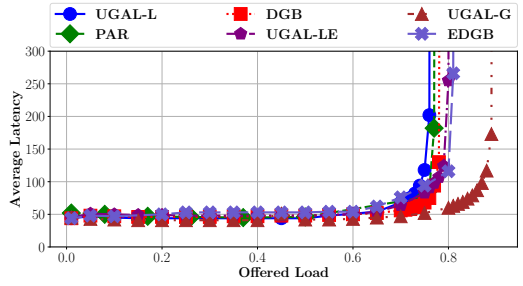

Figure 16: Random Permutation on $dfly(4, 8, 4, 9)$

We have also studied the transient behavior of different routing schemes (UGAL-L, DGB, UGAL-LE, EDGB) with the network traffic changes from the random uniform pattern with a medium load of 0.6 to a shift traffic pattern with a medium load of 0.25 on $Dfly(4, 8, 4, 33)$. To support a shift pattern with a load of 0.25, a large percentage of packets must be routed to VLB paths while for the uniform traffic with a load of 0.6, all traffic can be routed through the minimal path. In other words, the shift pattern stresses the minimal path much more than the uniform traffic. Figure 17 a

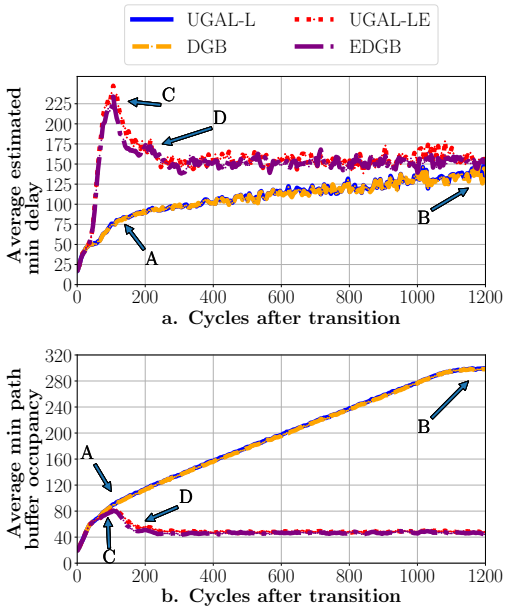

Figure 17: Transient behavior of different routing schemes

illustrates the average estimated MIN path latency after the traffic pattern switch starts. The average esimated MIN path latency is the average across all routers with active flows. UGAL-L and DGB have similar behavior while UGAL-LE and EDGB have similar behavior. This is understandable. It can be seen from the figure that our UGAL-LE reacts to the pattern change swiftly with a spike (label C in the figure) of estimated minimal path latency. This indicates that UGAL-LE detects and reacts to the new traffic pattern very quickly. At around 200 cycles after the switch (label D in the figure), UGAL-LE and EDGB reach a new steady state. On the other hand, UGAL-L and DGB also reacts to the new traffic pattern with an increasing estimated minimal path latency. However, they reaches the steady state much slower at about 1200 cycles after the switch (label B).

Figure 17 b shows the average total minimal path queue occupancy during this period of time. The average total minimal path queue occupancy is averaged across all active flows. With UGAL-LE and EDGB, the total minimal path queue occupancy increases to 90 initially and then settles at 45 at around 200 cycles after the pattern transition. With UGAL-L and DGB, the total minimal path queue occupancy continues increasing to about 300 at around 1200 cycles after the pattern transition. This indicates that UGAL-L and DGB routes many more packets to the minimal path than UGAL-LE and EDGB, which are sub-optimal for the shift traffic pattern.

6 EVALUATION WITH HPC WORKLOADS

To evaluate the performance with real HPC workloads, we extended the CODES simulation framework for online or in situ replay of workloads using the Scalable Workload Models (SWMs) [25] presented in [15] [22]. We have implemented UGAL-L and UGAL-LE within the CODES simulator. The evaluation is performed on

Application	Routing	Rank-to-Node mapping	
		Random	Continuous
MILC	UGAL-L	3.30	4.01
	UGAL-LE	3.25	3.06
Nekbone	UGAL-L	22.66	22.01
	UGAL-LE	21.47	19.18
Lammps	UGAL-L	6.029	6.19
	UGAL-LE	5.95	5.67
Stencil3D	UGAL-L	0.694	0.67
	UGAL-LE	0.681	0.54

Table 3: Communication time (in ms) of HPC applications on dfly(4,8,4,33) with different routing schemes and different mappings

the $dfly(4, 8, 4, 33)$ network topology that supports 1056 compute nodes. For UGAL-LE, the sliding window is set to 500 ns, and the counter threshold is set to 3. Routers are configured with VC buffer size of 256 packets. In the experiments, the local and global bandwidth is set up to 200Gb/s, same as the latest Slingshot system [7]. Other CODES parameters have the default values. We have evaluated routing schemes by replaying four common HPC applications, MILC [1], LAMMPS [1], Nekbone [1], and Stencil3D, as standalone jobs using their SWM code [2]. The parameters for applications have the default values, similar to prior studies [26] [22]. MILC, Nekbone, Lammps, and Stencil3D is configured with 1024, 729, 729, and 1024 ranks respectively. We replay HPC applications with continuous and random job placement policy to report the average communication times (output by CODES). The continuous job placement policy maps the compute node for each rank consecutively. Whereas, the random job placement policy assigns random nodes for each rank.

The average communication times of HPC applications with UGAL-LE and UGAL-L routing schemes on dfly(4,8,4,33) are shown in Table 3. UGAL-LE consistently out-performs UGAL-L in the experiments. The performance difference varies depending on the application and the job placement policy. Specifically, with continuous mapping, UGAL-LE achieves improvements of 23.69%, 12.85%, 8.4%, and 19.40% over UGAL-L for MILC, Nekbone, Lammps, and Stencil, respectively. Under random mapping, UGAL-LE shows improvements of 1.52% and 5.25%, 1.31%, and 1.87% for MILC, Nekbone, Lammps, and Stencil, respectively. Our technique is more effective when the network operates in the contention situation. A detailed examination of traffic patterns in these applications shows that continuous mapping results in more contention links while the traffic in random mapping is more uniform and thus less improvement with UGAL-LE.

7 CONCLUSION

We propose novel techniques to improve UGAL-based routing with local information. Unlike all of existing UGAL enhancement schemes, our schemes change the fundamental latency estimation of UGAL with local information, which results in much more accurate latency estimations for minimal paths when the network traffic is imbalanced. Our experiments demonstrate that our techniques are robust and effective: UGAL-LE and EDGB improve over their corresponding baseline counterparts significantly and consistently across diverse traffic patterns. In particular, our schemes exhibit a remarkable performance enhancement on traffic patterns that have heavy inter-group traffic, and are able to react to traffic pattern changes quickly.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation (NSF) under Grants CRI-1822737 and SHF-2007827. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. This work used the XSEDE Bridges resource at the Pittsburgh Supercomputing Center (PSC) through allocations ECS190004 and CCR200042. This work used the Bridges-2 resource at Pittsburgh Supercomputing Center (PSC) through allocation CIS230062 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #2138296.

REFERENCES

[1] [n. d.]. CORAL-2014. https://asc.llnl.gov/coral-benchmarks, Last accessed on 2023-12-20.
[2] [n. d.]. Online Workloads. https://github.com/codes-org/codes/wiki/online-workloads, Last accessed on 2023-09-25.
[3] Bob Alverson, Edwin Froese, Larry Kaplan, and Duncan Roweth. 2012. Cray XC series network. *Cray Inc., White Paper WP-Aries01-1112* (2012).
[4] Zaid Salamah A Alzaid, Saptarshi Bhowmik, Xin Yuan, and Michael Lang. 2020. Global link arrangement for practical dragonfly. In *Proceedings of the 34th ACM International Conference on Supercomputing*. 1–11.
[5] Ram Sharan Chaulagain, Fatema Tabassum Liza, Sudheer Chunduri, Xin Yuan, and Michael Lang. 2020. Achieving the Performance of Global Adaptive Routing using Local Information on Dragonfly through Deep Learning. *ACM/IEEE SC tech poster* (2020).
[6] J. Cope, N. Liu, S. Lang, P. Carns, C. Carothers, and R. Ross. 2012. Codes: Enabling co-design of multilayer exascale storage architectures. In *the Workshop on Emerging Supercomputing Technologies*.
[7] Daniele De Sensi, Salvatore Di Girolamo, Kim H McMahon, Duncan Roweth, and Torsten Hoefler. 2020. An in-depth analysis of the slingshot interconnect. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–14.
[8] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. 2012. Cray Cascade: A Scalable HPC System Based on a Dragonfly Network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Salt Lake City, Utah) *(SC '12)*. IEEE Computer Society Press, Los Alamitos, CA, USA, Article 103, 9 pages. http://dl.acm.org/citation.cfm?id=2388996.2389136
[9] Peyman Faizian, Juan Francisco Alfaro, Md Shafayat Rahman, Md Atiqul Mollah, Xin Yuan, Scott Pakin, and Michael Lang. 2018. TPR: Traffic Pattern-Based Adaptive Routing for Dragonfly Networks. *IEEE Trans. Multi-Scale Computing Systems* 4, 4 (2018), 931–943.
[10] P. Fuentes, E. Vallejo, M. Garcia, R. Beivide, G. Rodríguez, C. Minkenberg, and M. Valero. 2015. Contention-Based Nonminimal Adaptive Routing in High-Radix Networks. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*. 103–112. https://doi.org/10.1109/IPDPS.2015.78
[11] M. Garcia, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, G. Rodríguez, J. Labarta, and C. Minkenberg. 2012. On-the-Fly Adaptive Routing in High-Radix Hierarchical Networks. In *Parallel Processing (ICPP), 2012 41st International Conference on*. 279–288. https://doi.org/10.1109/ICPP.2012.46
[12] M. Garcia, E. Vallejo, R. Beivide, M. Odriozola, and M. Valero. 2013. Efficient Routing Mechanisms for Dragonfly Networks. In *Parallel Processing (ICPP), 2013 42nd International Conference on*. 582–592. https://doi.org/10.1109/ICPP.2013.72

[13] M. Garcia, E. Vallejo, R. Beivide, M. Valero, and G. Rodríguez. 2013. OFAR-CM: Efficient Dragonfly Networks with Simple Congestion Management. In *High-Performance Interconnects (HOTI), 2013 IEEE 21st Annual Symposium on*. 55–62. https://doi.org/10.1109/HOTI.2013.16

[14] E. Hastings, D. Rincon-Cruz, M. Spehlmann, S. Meyers, A. Xu, D. P. Bunde, and V. J. Leung. 2015. Comparing Global Link Arrangements for Dragonfly Networks. In *2015 IEEE International Conference on Cluster Computing*. 361–370. https://doi.org/10.1109/CLUSTER.2015.57

[15] Karl Scott Hemmert, Ray Bair, Abhinav Bhatele, Taylor Groves, Simon David Hammond, Michael J Levenhagen, Misbah Mubarak, Scott Pakin, Rob Ross, and Jeremiah J Wilke. 2019. System-level Architecture Simulation for Exascale: Challenges and Opportunities. (2019).

[16] HPE. 2019. HPC Slingshot interconnect. https://www.hpe.com/us/en/compute/hpc/slingshot-interconnect.html#. Accessed 09/30/2023.

[17] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally. 2013. A detailed and flexible cycle-accurate Network-on-Chip simulator. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 86–96. https://doi.org/10.1109/ISPASS.2013.6557149

[18] Nan Jiang, John Kim, and William J. Dally. 2009. Indirect Adaptive Routing on Large Scale Interconnection Networks. In *Proceedings of the 36th Annual International Symposium on Computer Architecture* (Austin, TX, USA) *(ISCA '09)*. ACM, New York, NY, USA, 220–231. https://doi.org/10.1145/1555754.1555783

[19] Yao Kang, Xin Wang, and Zhiling Lan. 2021. Q-adaptive: A multi-agent reinforcement learning based routing on dragonfly network. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*. 189–200.

[20] Hans Kasan, Gwangsun Kim, Yung Yi, and John Kim. 2022. Dynamic Global Adaptive Routing in High-Radix Networks. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 771,783. https://doi.org/10.1145/3470496.3527389

[21] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. 2008. Technology-Driven, Highly-Scalable Dragonfly Topology. In *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA '08)*. IEEE Computer Society, Washington, DC, USA, 77–88. https://doi.org/10.1109/ISCA.2008.19

[22] Misbah Mubarak, Neil McGlohon, Malek Musleh, Eric Borch, Robert B Ross, Ram Huggahalli, Sudheer Chunduri, Scott Parker, Christopher D Carothers, and Kalyan Kumaran. 2019. Evaluating quality of service traffic classes on the megafly network. In *High Performance Computing: 34th International Conference, ISC High Performance 2019, Frankfurt/Main, Germany, June 16–20, 2019, Proceedings 34*. Springer, 3–20.

[23] Oak Ridge National Laboratory. 2022. Frontier. https://www.olcf.ornl.gov/frontier/. Accessed 09/30/2023.

[24] Md Shafayat Rahman, Saptarshi Bhowmik, Yevgeniy Ryasnianskiy, Xin Yuan, and Michael Lang. 2019. Topology-custom UGAL Routing on Dragonfly. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, Colorado) *(SC '19)*. ACM, New York, NY, USA, Article 17, 15 pages. https://doi.org/10.1145/3295500.3356208

[25] John Thompson. 2014. Scalable workload models for system simulations. In *Workshop on Modeling & Simulation of Systems and Applications 2014*. 13–14.

[26] Xin Wang, Misbah Mubarak, Yao Kang, Robert B Ross, and Zhiling Lan. 2020. Union: An automatic workload manager for accelerating network simulation. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 821–830.

[27] J. Won, G. Kim, J. Kim, T. Jiang, M. Parker, and S. Scott. 2015. Overcoming far-end congestion in large-scale networks. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. 415–427. https://doi.org/10.1109/HPCA.2015.7056051