TAGO: Rethinking Routing Design in High Performance Reconfigurable Networks

Min Yee Teh*, Yu-Han Hung*, George Michelogiannakis†, Shijia Yan*,
Madeleine Glick*, John Shalf† and Keren Bergman*
* Columbia University, † Lawrence Berkeley National Lab
Email: *mt3126@columbia.edu, †[mihelog, jshalf]@lbl.gov

***Abstract*—Many reconfigurable network topologies have been proposed in the past. However, efficient routing on top of these flexible interconnects still presents a challenge. In this work, we reevaluate key principles that have guided the designs of many routing protocols on static networks, and see how well those principles apply on reconfigurable network topologies. Based on a theoretical analysis of key properties that routing in a reconfigurable network should satisfy to maximize performance, we propose a topology-aware, globally-direct oblivious (TAGO) routing protocol for reconfigurable topologies. Our proposed routing protocol is simple in design and yet, when deployed in conjunction with a reconfigurable network topology, improves throughput by up to 2.2× compared to established routing protocols and even comes within 10% of the throughput of impractical adaptive routing that has instant global congestion information.**

***Index Terms*—Adaptive Routing, Oblivious Routing, Reconfigurable Networks, Bandwidth Steering**

I. Introduction

Recent trends in high performance computing (HPC) systems indicate a steady decline in the bytes-per-FLOP ratio for interconnects because the available network bandwidth does not keep pace with the massive increases in computational performance over the past decade [1, 2]. This trend is expected to continue. Over-provisioning HPC system networks to reverse this trend will be increasingly impractical because network costs are becoming a larger fraction of overall system cost [3, 4]. The continued drive towards Exascale computing will further increase computational capacity and requirements for low latency and low power networking [5], and may rapidly shift the bottleneck in many applications from computation to communication.

This challenge is further complicated by many HPC and cloud applications that exhibit non-uniform communication patterns, due to their inherent algorithm structure or system effects such as fragmentation [4, 6, 7]. These non-uniform communication patterns create "hotspots" of congestion due to uneven network utilization. This has motivated research on task placement [8, 9] and adaptive routing [10]–[13]. Over the last decade, many different reconfigurable network architectures have been proposed as an alternative method to handle skewed traffic loads [4, 6, 14]–[23]. These reconfigurable architectures aim to dynamically alter the network topology to better match expected traffic and in many cases rely on an enabling technology such as silicon photonics to make topology reconfiguration more efficient.

Past works have shown reconfigurable networks to improve the efficiency and flexibility of resource disaggregation [24], which is a promising trend for future HPC to preserve performance scaling for key scientific applications including deep learning [25, 26]. Without efficient reconfigurable networks, resource disaggregation may likely be impractical [27, 28]. Moreover, there are numerous indicators that optics are playing an increasingly major role in modern networks, given the ever growing list of optical component suppliers [29].

Therefore, continued success in network performance scaling and resource disaggregation in future HPC is contingent on making the best use of reconfigurable topologies. However, these reconfigurable topologies typically do not investigate routing algorithms in detail but instead default to deterministic minimal routing to take advantage of reconfigured paths. Thus, the impact of routing algorithms that are popular in static, uniformly-connected networks is largely unexplored for reconfigurable topologies. While the high-level goals of routing are similar across all topologies (i.e., high throughput, low hop count, etc), there is currently little understanding of the key properties a routing algorithm must satisfy to systematically achieve these goals in a reconfigurable topology.

In this work, we re-evaluate a number of common design principles for effective routing algorithms and see how well they apply in reconfigurable network settings. To our knowledge, this is the first focused study on routing for reconfigurable networks. We approach this using a theory-driven flow-level analysis from which we derive several key properties that high-performance routing schemes must satisfy when applied to reconfigurable networks. Results from our flow-level analysis indicate that under well-configured topologies (i.e. topologies that are fitted to the traffic load with a low mismatch), indirect routing through intermediate endpoints is not a requirement to maximize network throughput. At the same time, we also find that minimal-routing schemes, when naively applied, will result in increased congestion due to poor load balancing.

Based on these insights, we propose a novel routing protocol that satisfies all desirable properties of reconfigurable network routing protocols called **T**opology-**A**ware **G**lobally-direct **O**blivious (TAGO) routing. TAGO routing is a lightweight oblivious algorithm, as routers do not need to be aware of path congestion to make routing decisions. TAGO is also globally-direct, meaning that packets traverse at most

SC20, November 9-19, 2020, Is Everywhere We Are
978-1-7281-9998-6/20/31.00 ©2020 IEEE

one global link; this minimizes packet hop count and traffic interference in the inter-group links. In addition, TAGO routing is topology-aware, allowing it to load-balance the network effectively for the current network topology.

We evaluate the performance of TAGO routing against several routing schemes using system scale simulations driven by real HPC applications and datacenter traces. Our evaluations show that TAGO performs close to that of a Universal Globally-Adaptive Load balance (UGAL) with global knowledge of congestion – a theoretical ideal UGAL that is impractical to implement. Simulation results indicate that indirect routing alternatives like Valiant load balance (VLB), or even adaptive ones like UGAL-L do not perform well in reconfigurable networks.

In summary, the key contributions of this work are:

1. We reevaluate the important desirable properties of routing schemes on reconfigurable networks based on results from rigorous theoretical analysis.
2. We propose a novel routing protocol called TAGO that achieves high performance in reconfigurable networks.
3. We evaluate the performance of TAGO and other common routing schemes with extensive simulations driven by realistic HPC and data center application traces.
4. We implement TAGO on a small-scale testbed and validate the performance experimentally.

II. Background and Related work

A. Adaptive Routing

Adaptive routing algorithms aim to load balance an otherwise unbalanced (skewed) traffic pattern [30]. Unbalanced traffic can be caused by imbalances in the application's non-uniform communication pattern, or by other system factors such as fragmentation where tasks of the same application are scattered across the system non-uniformly [4, 6, 7, 31, 32].

Some popular adaptive routing algorithms use indirect routing, where traffic can take non-minimal paths by selecting a random intermediate destination. Valiant routing (VLB) [33] does this for every packet, hence transforming any traffic pattern into uniform random. Universal Globally-Aware Load balancing (UGAL) decides on a per-packet basis by estimating congestion based on buffer occupancy at neighboring routers. More recent work has proposed techniques to measure congestion that is farther downstream [11]. Other adaptive routing schemes are either tied to specific topologies [10, 12, 34], or make adaptive decisions at every hop but are restricted to minimal (direct) paths, which limits their load balancing capability [13, 30].

B. Reconfigurable Networks

Even though indirect routing can dramatically improve load balance, it also increases packet hop count and consumes additional network capacity [35]. Further, indirect routing may cause congestion in other groups through job interference [36]. This motivates reconfigurable topologies that can dynamically allocate more links directly between frequently-communicating blocks, so that more traffic may traverse direct paths without risking link congestion.

One of the pioneering works in reconfigurable networks using optical circuit switches (OCSs) is HFAST [20]. Several subsequent works have also come out of the data center community [18, 19, 22, 23, 37]. The core idea is to exploit the high bandwidth of broadband optical switches, but the high switching latency precludes fine-grained optical switching. Another approach uses electrical four- or six-port converter switches to configure the network as either a hierarchical fat tree or a flat random graph [38]. Meanwhile, many prior works would take a dual-plane approach to separate the persistent "elephant" flows which traverse the optical plane from the shorter "mice" flows which traverse the fixed electrical plane [39]–[43].

More recently in HPC, researchers have similarly extended the idea with Flexfly [6], on the Fat Tree (folded Clos) [4], and on other topologies [44]. In addition, Flexspander [21] has been proposed, which combines the principles of low-diameter expander networks [45] with reconfigurable networks. Our work is complementary and can apply to these approaches to enable lightweight, high-performance routing.

In general, reconfigurable networks would configure their connectivity based on an expected persistent traffic matrix pattern. Past work has taken different approaches to allow applications to communicate their expected traffic matrix. For example, the MPI topology directives allow applications to directly communicate their intended communication graph [46] for controlled job placement [47, 48]. Alternatively, application traffic can be predicted by monitoring the network and using neural networks [49, 50], deep learning [51], or other advanced prediction techniques.

C. Hierarchical Block-Reconfigurable Topologies

In this paper, we base our analysis on the model of block-reconfigurable network topologies. This means that only the connectivity between blocks (groups of routers) can be adjusted. Packet switches that belong to the same block are statically wired. Fig. 1 shows a model network topology considered in this work. This model fits many popular HPC topologies such as the Dragonfly topology with one or more links between groups that is similar to Cray's Aries interconnect [52], Dragonfly+ [53], and the Flexfly [6] topology. In the case of Flexfly, each block translates into a Flexfly group that connects packet switches in a full-clique. Similarly, Flexspander [21] is a block-reconfigurable topology with expander graphs for block topologies. Our model can fit any topology that can reconfigure channels between collections of routers, if we define those routers as a block. For instance, in a fat tree, a natural unit to be considered a block would be a pod (i.e., a sub-tree), whereas in a HyperX [34], a natural block unit can be routers that share a particular dimension.

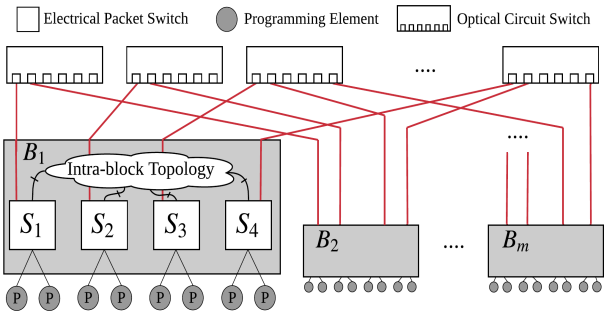


Figure 1: Model of an example reconfigurable network with m blocks. Each block wires packet switches statically in a manner defined by the intra-block topology. All blocks are interconnected in a reconfigurable manner via a layer of optical circuit switches (OCS). Changing the switch configurations of the OCS's realizes a specific inter-block logical topology.

D. Routing on Reconfigurable Networks

The majority of prior works for routing on reconfigurable topologies focuses on seggregated routing where fixed and reconfigurable network components are disjoint [54, 55]. A few works have also examined the synergy between routing and reconfiguration in a non-seggregated fashion. Unfortunately, optimal non-segregated routing on is NP-hard. Though some polynomial time heuristics can be employed, their optimality scales inversely with complexity. This makes close-to-optimal yet efficient non-segregated routing in large-scale system impractical [55]–[57]. Other related work highlights this by limiting reconfigurability to make designing efficient adaptive [58] or oblivious [59] routing tractable.

Our work decouples network reconfiguration from the routing algorithm by establishing three key properties for high-quality routing that we then implement in a simple and efficient routing algorithm.

E. Routing Challenges for Reconfigurable Networks

Routing adaptively in a reconfigurable network has to consider the topology's configuration. Otherwise, it is prone to poor load balance. For instance, Valiant routing [33] picks an intermediate destination with an equal probability for each packet. This causes an equal number of packets to each destination, but in a reconfigurable topology the amount of bandwidth available to each destination can be vastly different. This has motivated tailored Valiant routing in a SlimFly [33], as well as topology-specific adaptive routing algorithms such as topology-custom UGAL (T-UGAL) [10] and similarly OmniWAR and DimWAR that take advantage of structural regularities in a HyperX [34]. However, a general solution for any reconfigurable topology does not exist.

On the other hand, minimal-path adaptive routing algorithms are not known to achieve good load-balance due to their restricted path options except in a limited set of topologies [12, 13, 30]. Adaptive routing in a random graph has been shown to be challenging in its general form [60], as the lack of a regular structure in the network topology introduces tremendous complexity to the routing algorithm. This means that even the best state-of-the-art adaptive routing will likely yield sub-optimal performance when applied directly to reconfigurable networks.

III. Detailed Motivation Analysis

Our goal in this work is to design routing protocols well-suited to reconfigurable networks, built with principles derived from strong theoretical results and thoroughly evaluated with realistic, system-scale simulations and hardware demonstrations. To do so, we need to rigorously answer the following questions:

1) Can direct routing perform well under a wide variety of traffic-topology mismatch? **[Section III-C]**
2) Can minimal routing work sufficiently well on a reconfigurable network? **[Section III-D]**

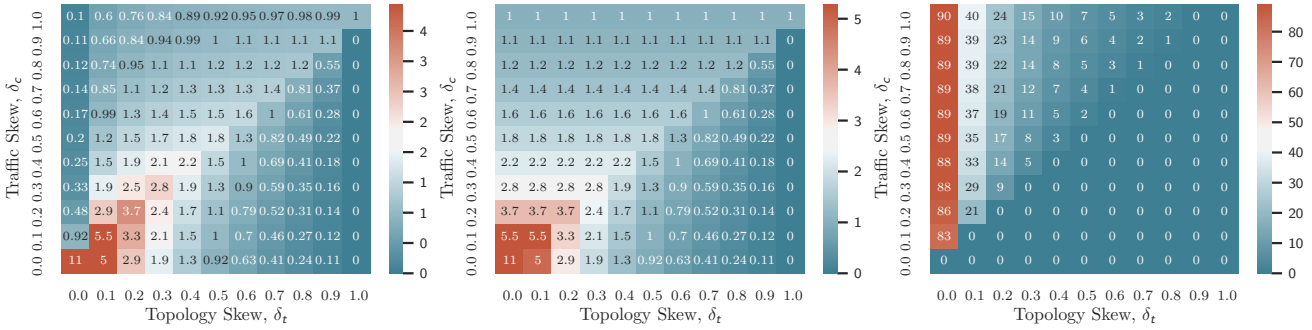
To answer these questions, we need to investigate if indirect adaptive routing can bring appreciable performance benefits to reconfigurable networks under a variety of skewed traffic loads. We introduce the idea of traffic skew in Section III-B.

A. Definitions and Mathematical Foundations

In this section we introduce the terminology used throughout this work. Global links refer to inter-block links that connect routers from different blocks. Direct routing allows packets to hop over at most *one* global link before reaching their destination. Indirect routing *requires* packets to be deflected to an intermediate block before being routed to the destination block. Minimal routing refers to routing protocols like MIN or ECMP that exclusively utilize shortest paths. Note that direct routing does not imply minimal routing, as packets that traverse only one inter-block link may still traverse non-minimal paths within a block.

Next, we briefly introduce some essential mathematical concepts. Table. I tabulates all the essential mathematical symbols used in this work. A **network graph**, $G = (V, E)$ has link capacities $c(u, v)$ for every $(u, v) \in E$. A **traffic matrix**, $T = [t_{ij}] \in \mathbb{R}^{n \times n}$, for a network graph, $G = (V, E)$, such that $|V| = n$, is a matrix such that all diagonal entries are zero. t_{ij} denotes the traffic sent from node i to node j. Without any loss of generality, we assume that the traffic matrix is ℓ_1-normalized, such that $\sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} = 1$. We take the ℓ_1-normalization of the traffic matrix to obtain the communication probability matrix, where each off-diagonal entry represents the probability of communication between two nodes.

The throughput, α, of routing traffic matrix, T, over a given topology G, is defined as a the maximum α for which αT has a feasible multi-commodity flow solution in G, with link capacity constraints and flow conservation constraints satisfied. This can be formulated as a maximum concurrent flow problem (MCFP) and solved with a linear program [61, 62] in polynomial time. We solve this linear program using Gurobi [63].



(a) Direct routes (b) Direct + Indirect routes (c) Percentage throughput improvement (%)

Figure 2: Global (inter-block) throughput as a function of different traffic and topology normalized skews, shown as heatmaps. (a) shows the throughput when routing considers only direct global paths. (b) shows the throughput when routing considers all paths, direct and indirect. (c) shows the percentage of throughput improvement when considering all paths vs. when considering only direct paths. Guide to heatmaps (a) and (b): each entry $e(\delta_t, \delta_c)$ indicates the throughput of a traffic matrix (TM) with a normalized skew δ_c in a topology with normalized skew δ_t (see Section III-C for definition). Warmer colors represent larger values.

Notation	Description
$G = (V, E)$	Network graph. V is the set of all network nodes and E is the set of all edges. **(Section III)**
$T = [t_{ij}] \in \mathbb{R}^{n \times n}$	Traffic matrix (TM) of a network with n nodes; $t_{ij} = 0$ for all $i = j$. **(Section III)**
$A = [a_{ij}] \in \mathbb{R}^{n \times n}$	Adversarial one-hot TM, with a single entry $a_{ij} = 1$ and 0 for other entries. **(Section III)**
$U = [u_{ij}] \in \mathbb{R}^{n \times n}$	Uniform TM, with $u_{ij} = \frac{1}{n(n-1)}$ for $i \neq j$ and 0 along the diagonal. **(Section III)**
$\sigma(T)$	Absolute skew of traffic matrix T. **(Section III)**
δ_c, δ_t	Normalized skew of traffic matrix and of topology, respectively. **(Section III)**
$\mathcal{S}_i$	Set of all switches in block i. **(Section IV)**
$\omega_j(s)$	Weight of routing to block j via boundary switch s. **(Section IV)**
$l_j(s)$	Number of inter-block (global) links connecting boundary switch s to block j. **(Section IV)**
d_b	Diameter of intra-block topology. **(Section IV)**

Table I: Table of mathematical notations, their descriptions, and where they appear in this paper.

B. Quantifying Traffic Skew

Intuitively, a skewed traffic matrix is one that is highly non-uniform, with some entries showing much more traffic than others. Based on this intuition, our measure of skew is the Kullback-Leibler (KL) divergence[1] between a traffic matrix T with respect to a uniform matrix.

Theorem 1. Given the uniform traffic matrix, $U = [u_{ij}] \in \mathbb{R}^{n \times n}$, the maximum attainable skew of a traffic matrix T with n^2 entries is $\log n(n-1)$.

[1]The Kullback-Leibler divergence is often interpreted as the "distance" between two probability distributions [64]. The larger the KL-divergence, the more dissimilar two probability distributions are.

Proof. By definition, the skew of T, $\sigma(T)$ is:

$$\begin{aligned} \sigma(T) &= \sum_{1 \leq i,j \leq n} t_{ij} \log(\frac{t_{ij}}{u_{ij}}) \\ &= \sum_{1 \leq i,j \leq n} t_{ij} \log t_{ij} + \log n(n-1) \\ &\leq \log n(n-1) \end{aligned} \tag{1}$$

□

Theorem 2. The one-hot traffic matrix, A, has the maximum skew of $\log n(n-1)$.

Proof. The maximum skew for any traffic matrix (TM), T, is $\sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} \log t_{ij} = 0$. The only TM that achieves this is one with a single off-diagonal entry of 1 that maps the destination of each source to a different block, and 0 for all other entries (i.e. the *one-hot matrix*). The one-hot matrix is the most adversarial traffic to route for a block-reconfigurable topology because it maximizes the use of inter-block bandwidth. □

Given these results, we can generate traffic matrices with varying degrees of skew using the following equation:

$$T(\delta_c) = \delta_c \; A \; + \; (1 - \delta_c) \; U \tag{2}$$

Where $T(\delta_c)$ denotes the traffic matrix with skew δ_c, U denotes the uniform traffic matrix, A denotes an adversarial one-hot traffic matrix, and $\delta_c \in [0, 1]$ denotes the *normalized traffic skew*, which acts as a linear knob that controls the relative portions of uniform and adversarial traffic in T. Eqn. 2 allows us to generate traffic matrices with varying degrees of skew using just a single linear parameter.

Naturally, the definition for traffic skew can also be extended to describe normalized topology skew, δ_t. Let $X(T) = [x_{ij}] \in \mathbb{R}^{n \times n}$ be the optimal (inter-block) topology with n blocks, configured for traffic matrix, T. The normalized topology skew, δ_t describes the normalized traffic skew of T for which the topology is optimized.

C. Is Direct Routing Robust Enough?

In situations where significant uncertainty in traffic prediction exists, considering indirect paths in routing still offers a richer set of routes for load-balancing, as shown in [65]. To this end, we study how a variable traffic-topology mistmatch affects throughput performance of reconfigurable topologies when routing considers only direct paths compared to when routing considers all (direct and indirect) paths.

Each block is represented as a graph vertex, thus abstracting away the intricacies of the intra-block topology. In this experiment, we assume the network has 15 nodes (blocks). Each node has one unit of ingress and egress link capacity. A static uniform topology would thus have uniform capacity connecting every node pair, while the capacity between node pairs is made variable in a reconfigurable topology. To compute the throughput of a traffic matrix in a reconfigurable topology, we simply make the topology itself an optimization variable in the MCFP formulation. Details are omitted for brevity. Direct routing requires traffic to traverse a maximum of one inter-block link, while for indirect routing that becomes two inter-block links.

Fig. 2 shows the throughput performance of a 15-block reconfigurable topology under direct-only and direct + indirect (i.e., all available paths) routing schemes. For each heatmap, the x axis shows the normalized topology skew δ_t. The normalized topology skew denotes the skew of the traffic matrix (TM) used for topology-optimization. The y-axis shows the normalized traffic skew δ_c of the *actual* traffic matrix (TM) that ends up loading the network. The absolute difference between δ_c and δ_t can be interpreted as the traffic-topology mismatch.

For both routing schemes, the highest attainable throughput for every traffic skew, δ_c, is achieved when $\delta_c = \delta_t$ (i.e. under zero traffic-topology mismatch). This is unsurprising, as when $\delta_c = \delta_t$ indicates zero traffic-topology mismatch, which means that the configured topology is a perfect fit for the actual traffic matrix. Under these conditions, considering indirect paths in addition to direct paths when routing brings *no appreciable throughput improvements*. As the difference between δ_c and δ_t becomes greater, throughput steadily deteriorates. However, the throughput drop due to the traffic-topology mismatch is less severe when routing uses indirect paths. In practical terms, this means that when traffic prediction contains high levels of uncertainty, the topology may not be configured optimally, thus considering indirect paths in routing could improve throughput. As traffic is increasingly skewed, the maximum attainable throughput decreases, even when $\delta_c = \delta_t$. This is because as more traffic comes from a specific pair of nodes, the throughput becomes limited by the source node's total egress bandwidth, which is a constant is our experimental.

That said, the most important insight of Fig. 2(c) is that the improvements brought about by indirect routing *diminish* as the topology becomes more skewed and as the topology fits the traffic better (the traffic-topology mismatch reduces). For instance, while indirect routing can improve throughput (with respect to direct routing) by as much as 90% in a uniform topology ($\delta_t = 0$), it can only improve throughput by no more than 7% in a topology with $\delta_t = 0.5$. This means that by reconfiguring the topology to better fit an expected TM, we are also diminishing the impact of indirect routing. In many cases, the additional complexity required to implement indirect routing may even outweigh potential benefits.

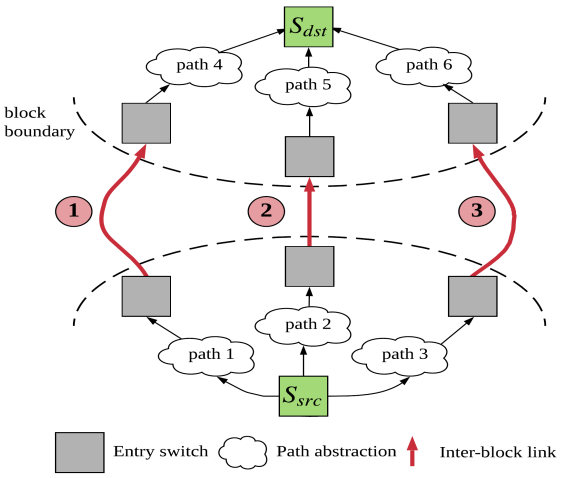


Figure 3: A scenario where minimal routing falls short. If (path 1 + path 4) has shorter path length than those of (path 2 + path 5) and (path 3 + path 6), then all traffic between S_{src} and S_{dst} will be routed on link 1 under minimal routing, while link 2 and 3 will be ignored.

We recognize that because our setup treats each block as a network node, the effects arising from intra-block routing on throughput, such as inter-group interference, are neglected. This effect, however, will likely be insignificant in groups with full connectivity and sufficient intra-group bandwidth. Further, these experiments assume that the topology is infinitesimally-reconfigurable[2], which means that the number of links between block pairs can take on fractional values.

D. Naive Minimal Routing Falls Short

Even though indirect routing may not necessarily improve throughput in low traffic-topology-mismatch conditions, defaulting to shortest-path routing can also result in poor load-balancing. This is because in block-reconfigurable topology, there may be more direct paths between blocks that are expected to communicate more intensively. However, those paths may not appear to be the same length from the viewpoint of a traffic source, depending on the intra-group topology. In that case, minimal routing will only use the shortest paths, rather than balancing traffic across all paths that the reconfigurable topology has provided. This defeats the purpose of reconfiguring. An example is shown in Fig. 3.

[2]This assumption is made to reduce the problem's complexity from exponential to polynomial.

IV. TAGO: Oblivious Routing for Reconfigurable Networks

The analysis in Section III shows that when there is little traffic–topology mismatch, oblivious direct routing can achieve maximum throughput without relying on indirect paths. Based on these observations, we surmise that a good routing solution for reconfigurable networks must satisfy the following properties:

- **Property 1:** Primarily utilize direct global paths.
- **Property 2:** Balance traffic among global channels.
- **Property 3:** Utilize intra-block path diversity.

First, to fully take advantage of the reconfiguration capability of a reconfigurable topology, the routing protocol should preferentially route traffic via direct channels that are the result of configuration such as to serve more intensive communication between certain source–destination pairs. Second, the routing protocol must effectively distribute the traffic evenly among global channels; this is to minimize overwhelming any particular global channel. Third, the routing protocol should still utilize path diversity within a block to alleviate bottlenecks. Many hierarchical topologies like the Fat Tree [66], Dragonfly, or Dragonfly+ [53] are often oversubscribed at the core (inter-group) level and possess much higher bisection bandwidth closer to the network's edge than at its core. The routing protocol must therefore take this into consideration and fully-utilize intra-group path diversity for load-balancing. Table II summarizes qualitatively the desired properties that are met by different routing protocols.

MIN routes packets exclusively along a single shortest path, therefore it fails to load balance the global links. ECMP splits traffic between *multiple* short paths, and thus can better load-balance the global links than MIN. Meanwhile, VLB relies exclusively on indirect global links, making it a poor fit for reconfigurable networks. Both UGAL and PAR may conditionally fulfill Property 1. When network is uncongested, both UGAL and PAR resorts to direct routing is low. As congestion increases, however, both UGAL and PAR will start resorting to indirect routing. In the following, we propose a routing protocol for reconfigurable networks called TAGO that satisfies all of the aforementioned properties.

Protocol	Property 1	Property 2	Property 3
MIN	✓	✗	✗
ECMP	✓	✗	✓
VLB	✗	✗	✗
UGAL	✻	✓	✗
PAR [11]	✻	✓	✗
TAGO	✓	✓	✓

Table II: Summary of the properties of different routing protocols. ✓ indicates a quality is satisfied, ✗ indicates a quality is strictly not satisfied, while ✻ indicates a quality is conditionally-satisfied, depending on specific topology classes.

A. Intra-block Routing

When routing packets between endpoints within the same block, TAGO uses ECMP routing. This ensures that the load is equally shared among multiple shortest paths. Most commercial packet switches in the market today support ECMP. To compute multiple shortest paths between network switches, one can utilize variants of the Floyd-Washall [67] or Dijkstra's algorithms. In our model topology, the intra-block topology does not change past deployment. Therefore, the forwarding table entries to routers in the same block can be programmed once prior to deployment and then kept constant. Updating these entries is only needed upon physical rewiring or a hardware failure. This effectively reduces the number of forwarding table entries required to be reprogrammed every time the topology is reconfigured, thus reducing the topology reconfiguration overhead considerably.

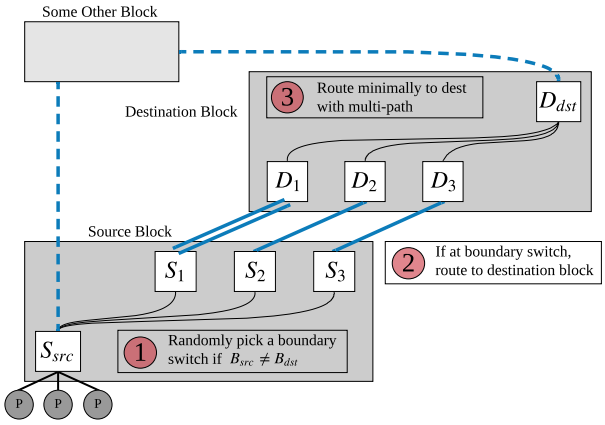


Figure 4: An illustration of TAGO's inter-block (global) routing logic. The source router, S_{src}, decides on a per-packet basis which boundary switch is used for inter-block traversal. The packet is then routed minimally to boundary switch, before being routed minimally to its destination.

B. Inter-block Routing

The inter-block routing logic is shown in Fig. 4. The source router first inspects the block ID of the destination address. If a packet has an IP address that is outside the current block, the source router will randomly select a boundary router to transit to the destination block. The block ID of the chosen boundary router to be used as a destination ID can be easily generated using schemes like address-space partitioning, VLAN, etc. This is similar to other routing schemes like UGAL and Valiant that need to encode intermediate destination IDs in packets. In order for TAGO to work, the topology reconfiguration algorithm needs to ensure that every pair of blocks is interconnected by at least one link, otherwise, some topology configurations may have no direct paths between specific block pairs. Satisfying this requirement, however, can be easily accomplished by simply adding a constraint in the topology reconfiguration algorithm to ensure that no fewer than one link connects every block-pair. After the boundary switch is selected, its ID is tagged onto the packet header. Any intermediate switch the packet traverses will simply forward

the packet to the tagged boundary switch using shortest multi-path routing.

The selection of boundary routers is governed by a set of weights. Using Fig. 4 as an example, packets from S_{src} should be twice as likely to be routed via S_1 than via S_2 or S_3, as it has twice the number of global channels connected to the destination block. To formalize this idea, the weight for selecting a boundary switch, s, to transit to a destination block, j, must be proportional to the number of inter-block channels connecting boundary switch s to the destination block; we denote this weight as $\omega_j(s)$. Let $l_j(s)$ be the number of global channels connecting packet switch s in block i to switch in block j. $\mathcal{S}_i$ represents the set of all switches in block i. Then, $\omega_j(s)$ can be computed as follows:

$$\omega_j(s) \;=\; \frac{l_j(s)}{\sum\limits_{s' \in \mathcal{S}_i} l_j(s')} \tag{3}$$

Note that while TAGO is a globally-direct routing scheme, it is non-minimal, in that it may not always route packets to their destinations via minimal paths. For instance, a source router that is a boundary router to a destination block may still send the packet to another boundary router. Even though this may increase the hop count, we find it a necessary feature for load-balancing global links (Section III-D).

C. Implementation Details and Analysis

1) Forwarding Table Complexity:
Given a network with m blocks and n switches per block, TAGO reduces the number of entries in the forwarding table from $O(m \times n)$ to $O(m + n)$. In our routing scheme, routers do not require a notion of paths to all other routers, merely to routers that belong to the same block, and which routers are boundary switches to which other blocks. So, rather than creating an entry for every router in the network, each router simply needs to contain forwarding information for 1) all peer routers in its block and 2) all other blocks and the boundary routers that have global links to said blocks. Forwarding entries to routers in other blocks simply map to a set of possible boundary routers. m entries are maintained to identify the boundary routers to every block. This can potentially bring huge savings in large scale networks. For instance, a network with 32 blocks and 31 routers per block requires only at most 63 entries instead of 992 entries. Updating a smaller forwarding table is faster, thus making topology reconfiguration also faster.

2) Worst-case Path Stretch:
While intra-block packets are routed minimally, TAGO may still route inter-block packets non-minimally in order to load-balance the global channels. For instance, even if the source router is a boundary router to the destination block, the packet may still be routed to a different boundary router to even out global link utilization. The worst-case (endpoint-to-endpoint) path stretch[3] is $\frac{2d_b+2}{3}$, where d_b is the diameter of the intra-block topology. This is still lower than the worst case stretch of UGAL and Valiant load balancing, both of which have a worst-case stretch of $d_b + \frac{2}{3}$. In a Flexfly network, where $d_b = 1$, the worst-case stretch is $\frac{4}{3}$ when using TAGO. Small control packets that are latency-sensitive may bypass non-minimal routing by having routers to send these packets exclusively along minimal paths, perhaps by dedicating a virtual channel (VC) to those packets.

[3]Path stretch refers to the ratio of the path length traversed by a packet to the path length of the shortest route.

V. System Scale Simulation

Simulator: We use Netbench, which is a discrete event packet-level simulator [68]. The simulator was developed and used recently for evaluating expander networks [69]. Our software is made publicly-available on Github to facilitate reproducibility [70].

Traffic Patterns: We use a variety of artificially-generated and realistic trace-based traffic matrices. The synthetic traffic patterns considered are the adversarial and 27-point stencil. The adversarial traffic pattern is one where each block communicates exclusively with *one* other neighboring block, which forces all inter-block traffic to traverse a small subset of global links. We also use a 27-point stencil traffic pattern for two reasons: 1) stencil codes are commonly used in many scientific computing applications, and 2) the high spatial locality due to the neighbor-intensive communication pattern is challenging for minimal routing. We also run simulations driven by real application workloads identified by the Exascale Computing Project (ECP) [71], namely Nekbone with 1024 MPI ranks and AMG with 1728 MPI ranks. Traffic traces are obtained using DUMPI [72]. We mix these applications into the system to better reflect conditions in a real system which could be running many applications in tandem at any given time. For the mixed Nekbone and AMG workloads, we use two types of task mapping schemes: 1) contiguous mapping, and 2) randomized mapping. Contiguous mapping prioritizes locality by placing workers from the same application within the same block, while randomized mapping places workers at random. The mapping also ensures that a task is mapped onto at least 1 node of every switch. We also use cloud workload traces obtained from a Hadoop cluster in one of Facebook's data centers [32].

Network Parameters: Each link has 100 Gbps bandwidth, and 30 ns propagation delay between packet switches. To prevent injection bandwidths from limiting performance, we set the injection link bandwidths to $2\times$ that of network links.

Topologies: We use three different network topologies in our simulations, namely: Dragonfly, Flexfly, and Flexspander. All topologies used for evaluations are identical in size in terms of total number of endpoints. The Dragonfly is used to test the versatility of TAGO in static networks. We then evaluate different routing schemes on Flexfly and Flexspander networks to see which routing protocol performs best in reconfigurable networks.

The simulated Dragonfly instance has 17 blocks (groups), each containing 32 electronic packet switches (EPS). Every switch has 4 inter-group links and 15 links for connecting

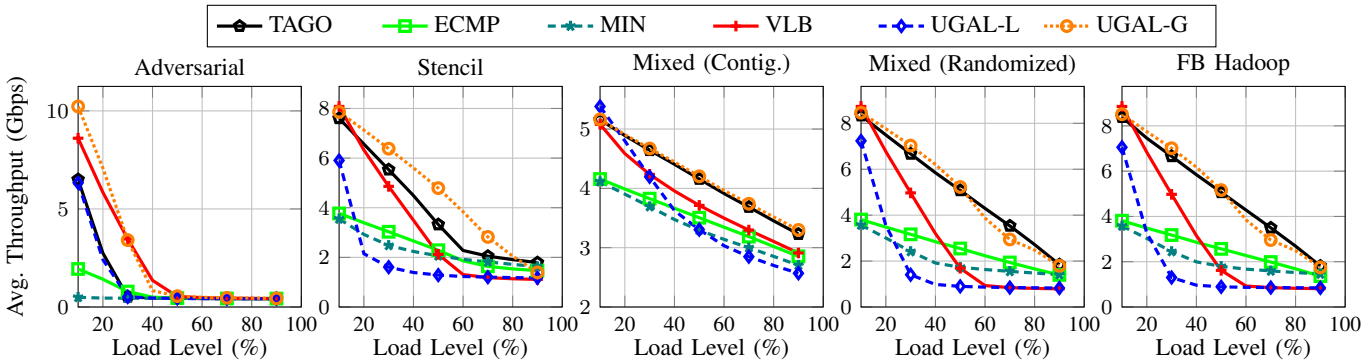


Figure 5: Average throughput performance of routing protocols under different traffic patterns on a Dragonfly topology. Higher is better.

to endpoints. Each group pair is interconnected by 8 inter-group (global) links. Modern HPC networks such as the Aries interconnect in Cray's XC40 also use multiple links per group pair. These numbers support a total of 8160 endpoints. Both Flexfly and Flexspander topologies used in simulations similarly contain 17 blocks of 32 electronic packet switches each. Blocks are physically interconnected to a layer of 17-port optical circuit switches. Changing the optical switch configurations switches the inter-block topology. EPSes within the same block are statically-interconnected using electrical links. The Flexspander intra-block topology is a random graph with sparse connectivity, as each electronic packet switch only has 20 links dedicated to intra-group connections.

Routing: We compare TAGO against 5 commonly-used routing schemes: MIN, ECMP, VLB (Valiant), UGAL-L, and UGAL-G. MIN routes all traffic along a *single* shortest path, while ECMP splits traffic equally among *multiple* shortest paths. VLB always deflects every packet to a randomly-chosen intermediate block before routing to the destination block. UGAL selects between Valiant (indirect) paths and minimal (direct) paths based on estimated congestion at the source switch. UGAL-L uses only local queue information, while UGAL-G assumes up-to-date global knowledge of congestion. Having instant global knowledge of congestion is infeasible in practice, so UGAL-G represents the idealized "oracle" performance case for UGAL.

Messages and Flows: The injected message (equivalent to a flow) size of artificial, stencil, and mixed HPC workloads is between 100 bytes and 20 Megabytes. Flows sizes for the Facebook (FB) Hadoop traces are extracted from [32].

Metrics: Our performance metrics are average flow (message) throughput and packet latency. The average flow/message's throughput is the number of bits sent per unit time in Gigabits per second (Gbps), averaged cross all traffic flows.

Packet latency is a round-trip-time (RTT) latency measured inside the network (i.e., injection queue time is not included). We also show utilization of global links.

A. Uniform Dragonfly Performance

As shown in Fig. 5, the average throughput of TAGO under adversarial traffic is comparable to that of UGAL-G at higher loads but it is lower at lower traffic loads. That is because the Dragonfly inter-group topology is non-reconfigurable and thus cannot dynamically adjust its topology to fit estimated inter-group traffic patterns. Due to the fixed inter-group topology, indirect adaptive routing can better utilize all inter-group bandwidth. However, as the network channels become saturated at higher traffic loads, the average throughput for all routing schemes converges. We notice that RTT latencies of UGAL-G and TAGO are comparable, both of which are considerably lower than the those of other routing schemes. This trend persists across all tested traffic patterns with the exception of adversarial traffic, for the same reason mentioned previously.

B. Flexfly Performance

As shown in Fig. 6, in a Flexfly the average throughput of TAGO generally matches that of UGAL-G. Under adversarial traffic, however, UGAL-G eventually outperforms TAGO as load increases. This is due to saturating inter-group bandwidth similar to a Dragonfly, though this effect is less pronounced due to Flexfly's reconfigurability. ECMP generally outperforms MIN due to its ability to split traffic among a larger set of multiple shortest paths. ECMP in the Flexfly also performs better compared to ECMP in the Dragonfly because of the presence of more short paths in a Flexfly after topology reconfiguration. That said, TAGO still consistently outperforms ECMP due to its ability to split traffic equally among the global inter-block links, while ECMP makes no conscious effort to load-balance the global links. These results show that while globally-direct routing can yield, minimal routing can still cause poor load-balancing.

Fig. 7 shows average and 99.9th percentile (tail) packet latency. TAGO generally performs close to UGAL-G except for adversarial traffic and to some degree 27-P stencil traffic. VLB and UGAL-L exhibit much higher packet latencies than other routing protocols. VLB extends average path length by exclusively routing non-minimally. This not only incurs higher latency, but also causes congestion and interference in the intermediate block. UGAL-L requires stiff back pressure and is prone to making sub-optimal global routing decisions based on inaccurate local queue information. This also causes imperfect load balance, which hurts throughput as shown previously.

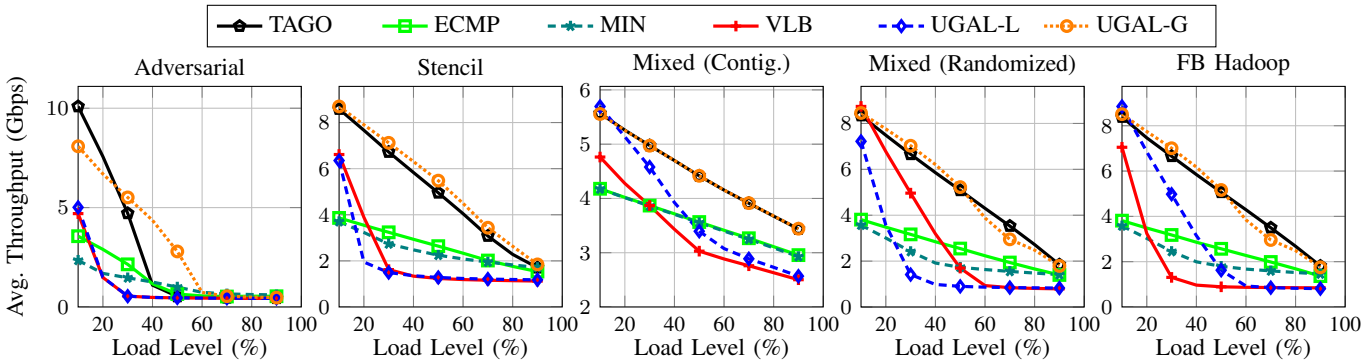


Figure 6: Average throughput performance of different routing schemes on a Flexfly topology. Higher is better.

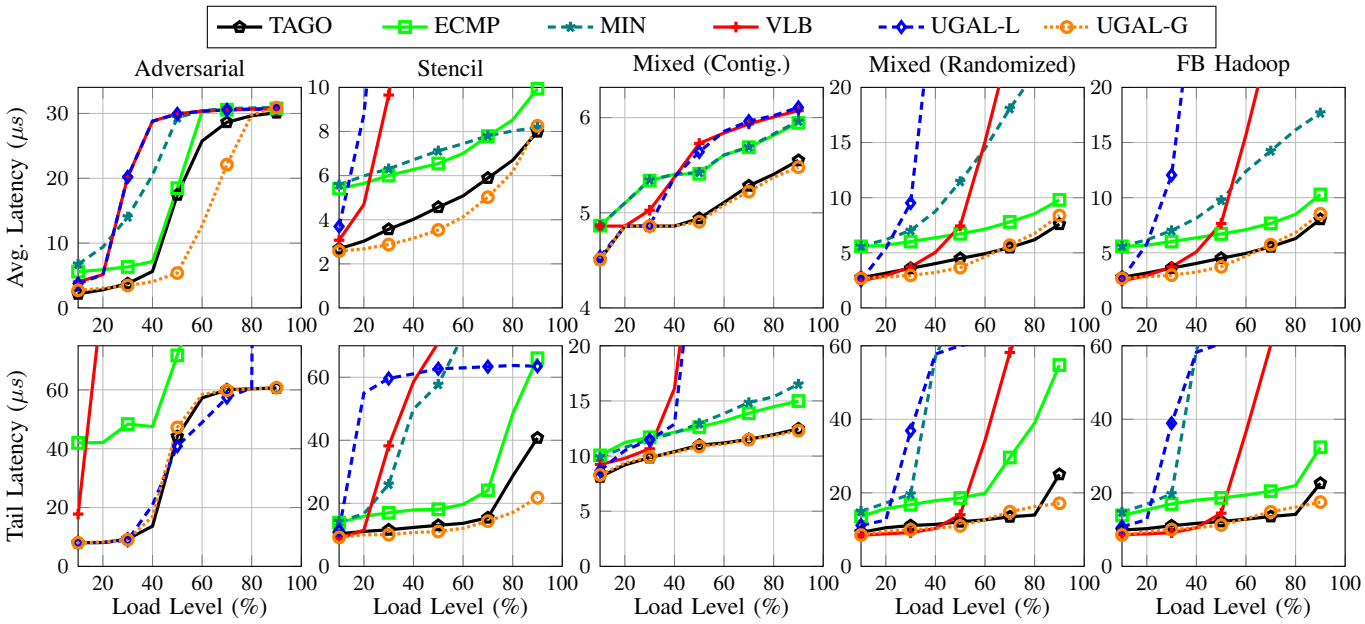


Figure 7: Packet latency of different routing protocols on a Flexfly. Top row shows the average packet latency, while the bottom row shows the tail (99.9th %-tile) packet latency. Lower is better.

C. Flexspander Performance

Fig. 8 shows throughput results for Flexspander [21]. Like Flexfly, Flexspander can also dynamically adjust its global connectivity using bandwidth steering. Unlike Flexfly, however, each Flexspander block has a random graph topology, which lacks the clear structure that a full-clique Flexfly group has. The random intra-block topology creates a more challenging environment for routing protocols to load-balance. As shown, TAGO yields throughput performance comparable to that of UGAL-G. That is because TAGO load balances well across groups even without using explicit congestion-based heuristics. TAGO also outperforms shortest-path routing like ECMP and MIN because it can more effectively spread traffic among the global links. UGAL-L makes imperfect routing decisions because it relies on local information. In fact, given Flexspander's sparser intra-group connectivity, the average intra-group hop count increases. The increase in intra-group hop count makes sensing distant inter-group link congestion from the source router more challenging. As a result, UGAL-L does not perform noticeably better than VLB, ECMP, and MIN. These insights are confirmed by packet latency results in Fig. 9.

D. Larger Scale Networks

We also verify TAGO's performance using Flexspander and Flexfly with 25 blocks of 96 EPSes each, compared to 17 blocks of 32 EPSes per block used in our previous experiments. EPSes have radix of 32, and OCSes have radix of 17. Our simulations show that TAGO's performance comes within 5% that of UGAL-G at all traffic loads. We also found that the trends observed previously are similar in the larger-scale topology, indicating that TAGO's performances scale well with increasing network sizes.

E. Global Link Utilization

The utilization of global links is a good indicator of a routing scheme's load-balancing capability. We measure link

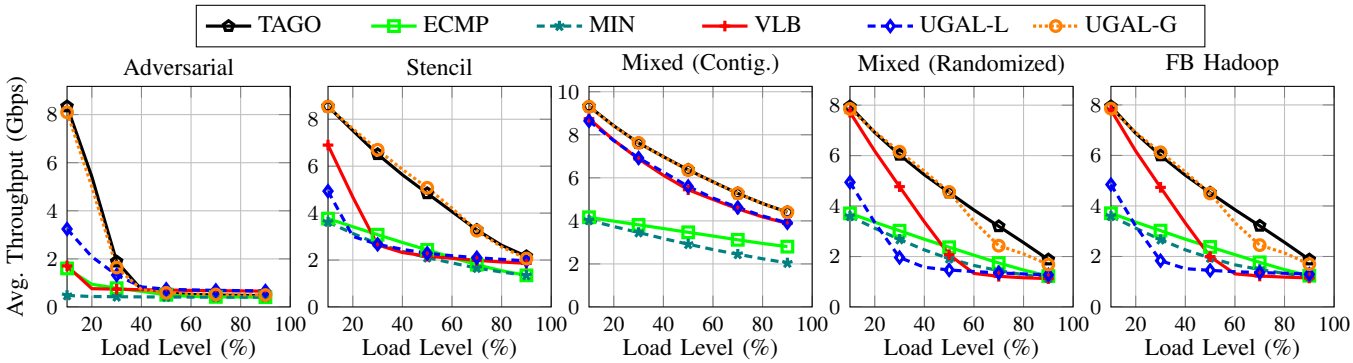


Figure 8: Average throughput for all generated flows of different routing schemes on a Flexspander topology. Higher is better.

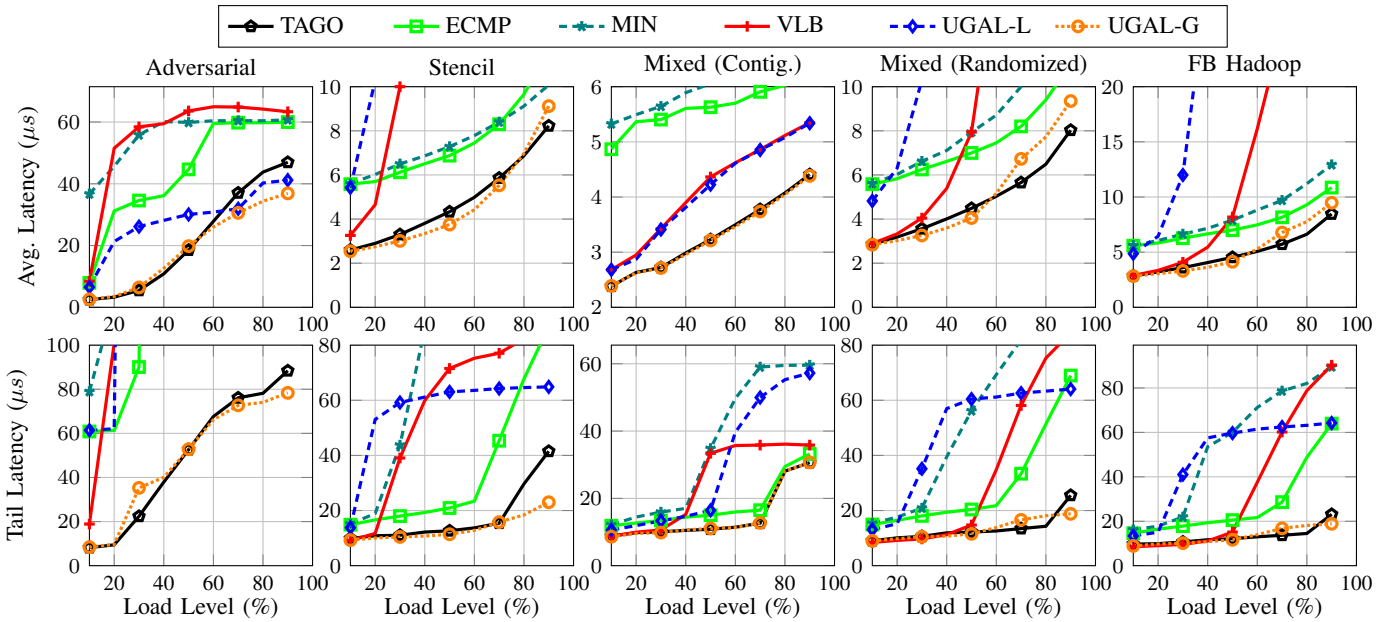


Figure 9: Packet latency of different routing protocols on a Flexspander. Top row shows the average packet latency, while the bottom row shows the tail (99.9th %-tile) packet latency. Lower is better.

utilization as the fraction of time a link spends carrying traffic over the course of the simulated time frame. A good routing scheme should evenly utilize all global channels, while still keeping the utilizations as low as possible.

Fig. 10, shows the distribution of global channel utilization of routing protocols on Flexfly at a 70% injection load. Results show that both TAGO and UGAL-G have low spreads in their link utilization distribution, indicating fairer utilization of global channels. However, UGAL-G has a higher average utilization because it uses indirect routing in some cases, whereas TAGO does not. Avoiding indirect paths reduces the number of virtual channels (VCs) required to prevent cyclic dependencies. The outlier points of VLB, ECMP, MIN, and UGAL-L show that a subset of global links experience overly high utilization of close to 100%. Links with 100% become bottlenecks that limit the throughput of all flows sharing these links. The outliers are especially pronounced in UGAL-L, even though the majority of links experience lower utilization than other routing schemes. These uneven, bimodal-like link utilization distributions are strong indicators of poor load-balancing, which partly explains UGAL-L's overall poor throughput performance.

VI. Testbed Experiments

A. HPC Testbed Description

We built a 16-node HPC testbed arranged as shown in Fig. 11(a). Our testbed consists of a data plane and a control plane. The data plane includes two blocks, each consisting of 8 servers and 4 electrical packet switches (EPSs). Each server is equipped with Intel Xeon Processors E5-2430 with 6 cores, 24 GB RAM, and a 10Gbps Network Interface Card (NIC). As shown in Fig. 11(a), for intra-block links, each EPS is attached to two servers and three other EPSs with 10G SFP+ electrical transceivers. For inter-block links, EPSs separated in the two blocks are connected using 10G SFP+ optical transceivers in C-band. Each optical transceiver is connected to a 1-meter long Single-Mode Fiber (SMF). Such arrangements using silicon-photonic switches are made to emulate reconfiguration of

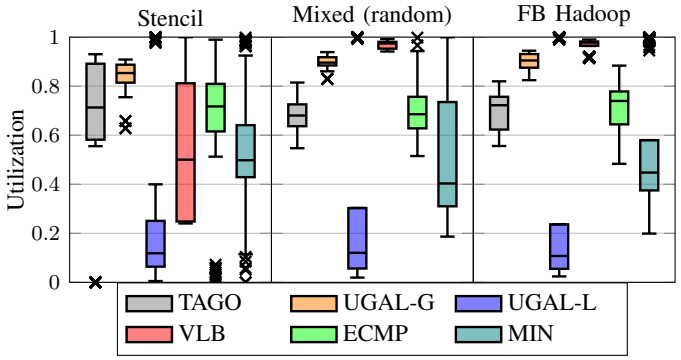


Figure 10: Distribution of global channel utilization of routing protocols in Flexfly under a) stencil, b) AMG + Nekbone (random placement), and c) FB Hadoop. Injection load is 70%. The box lines capture the first quartile (Q1), median (Q2), and third quartile (Q3). Upper and low whiskers mark the Q3 + interquartile range and Q1 - interquartile range, respectively. Outliers are marked by "×".

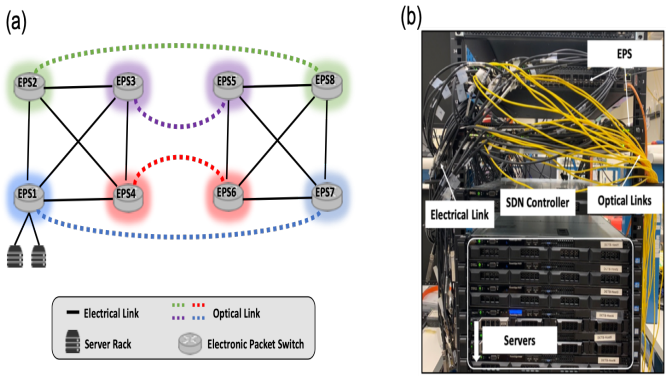


Figure 11: (a) Network topology used in our HPC testbed. (b) Photo of the testbed.

inter-block bandwidth. Fig. 11(b) shows a photo of our testbed. The control plane receives the byte counts that indicate traffic load in each link from the EPSs and implements TAGO by programming packet-handling rules into the EPS flow tables using a Ryu-based SDN controller via OpenFlow.

B. Experimental Results

We run the Gyrokinetic Toroidal Code (GTC) benchmark on our testbed. GTC is a 3D parallel particle-in-cell code developed to simulate turbulent transport in fusion plasmas [73]. In this experiment, we use a skeletonized GTC with its computational routines removed. MPI is used for the synchronization of rank assignments over compute nodes. We designed two different sets of flow tables, one using minimal routing and one using TAGO. The performance of each routing protocol is measured by the completion time of the skeletonized GTC application. We keep the MPI rank assignments identical across both experiments.

Figs. 12(a) and (b) show the throughput time series of all inter-block links throughout the application's execution for a) MIN, and b) TAGO routing, respectively. Fig. 12(a) shows that all inter-block traffic traverses only the link connecting EPS 1 and EPS 7 instead of spreading among the three other available inter-block links. As shown, the inter-block communication occupies the majority of the full 10 Gbps link capacity, indicating that traffic is mostly congested on just a single inter-block link. As a result, application completion time is 1250 seconds.

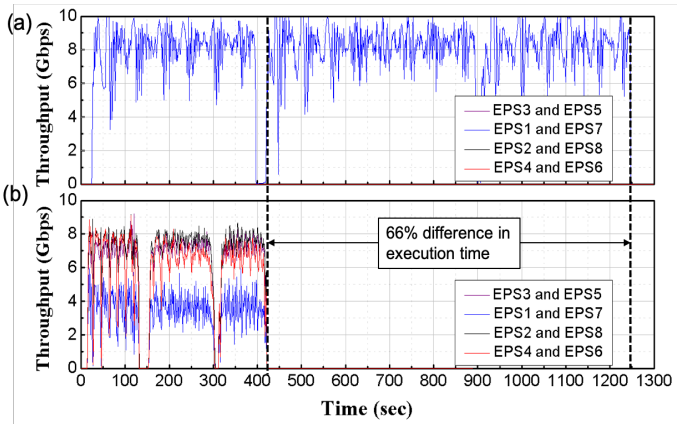


Figure 12: Testbed link throughput time series of (a) minimal routing and (b) TAGO routing for the GTC application.

We then repeat the previous experiment with a second flow table obtained using TAGO. Now inter-block traffic is distributed among all four inter-block links. As shown in Fig, 12(b), all four inter-block links now carry traffic, indicating that inter-block traffic is load-balanced well across all available resources. The overall throughput of inter-block communication becomes around 30 Gbps, which is 3× that of the first scenario (MIN routing). As a result, application completion time shows a 66% improvement from 1250 seconds in the first case to 410 seconds.

In addition to GTC, we also tested TAGO's performance using a distributed deep learning (DL) application called MobileNetV2 [74] on the CIFAR-10 data set. When routing with MIN, we observed the effective bandwidth of the DL application to be around 0.1 Gbps, and a runtime of 10643 seconds. When using TAGO, the observed throughput of DL application is approximately 1 Gbps, which is 10 times higher than the throughput of MIN. The throughput improvement over MIN leads to a 90% improvement in runtime from 10643 to 1148 seconds. Table. III summarizes the application speedup results of our testbed experiments.

Applications	Runtime (s)		Performance Improvement
	MIN	TAGO	
GTC	1250	410	66%
MobileNetV2	10643	1148	90%

Table III: Application performances on the testbed.

VII. Discussion

Our results confirm that TAGO meets the three desired properties introduced in Section IV without requiring global

knowledge, as long as the traffic-topology mismatches is low. In addition, our results indicate that those three properties are a good predictor of performance in such topologies. Our benchmarks contain a mixture of different payload sizes to reflect realistic network conditions, so minor changes to the payload sizes are unlikely to make a noticeable difference.

For our experiments, we compare against several popular practical algorithms as well as UGAL-G, which is commonly used as an high-performance albeit impractical adaptive routing algorithm to show a performance upper bound. Without assuming instantaneous global knowledge of congestion, it is highly unlikely for other more practical variants of adaptive routing algorithms to outperform UGAL-G, and by extension unlikely to be substantially better than TAGO. Such examples include, but are not limited to, progressive adaptive routing (PAR) [53] and K-shortest path (KSP) [75], which has shown promise on expander networks like Xpander and Jellyfish [45, 69, 76]. In our preliminary studies on KSP, we found that its performance can be highly sensitive to variations in the number of paths per switch-pair (i.e. the k value). Moreover, we found that KSP would at best perform equally to ECMP, despite the additional complexity required to realize KSP.

VIII. Conclusion and Future Work

In this work, we reevaluate the common design principles for efficient routing in the context of reconfigurable networks. Using rigorous flow-level analysis, we identify the essential properties that high-performance routing schemes should satisfy. Based on the identified traits, we propose a novel routing protocol called TAGO, a lightweight oblivious routing scheme optimized for reconfigurable network topologies. We then evaluate the performance of TAGO using extensive simulations driven by realistic HPC and data center workloads. Finally, we realize TAGO and validate its performance on a small scale-experimental testbed.

For future work, we will extending TAGO's capability by considering cases when significant traffic-topology mismatch is high. This is particularly important to make TAGO feasible for less agile reconfigurable topologies, which may have far less capability to react to changing traffic patterns over time. As we have shown in Fig. 2 back in Section III, incorporating indirect paths in routing may in fact be beneficial to overall throughput when the traffic-topology mismatch is high. Another topic for future work is to either limit the possible network topology configurations so that the routing algorithm may provide deadlock-free guarantees, or to grant full freedom to all possible network topology configurations while equipping the routing algorithm with deadlock-recovery mechanisms.

Acknowledgements

We are grateful to the anonymous reviewers for their constructive feedback and discussions, which have been instrumental in helping us improve the quality of this paper. This work is partially supported by the ARPA-E ENLITENED program project under the award DE-AR00000843. Min Yee is currently supported by the Wei Family Foundation Fellowship. This work was also supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE- AC02-05CH11231.

References

[1] (2018, November) The Top500 HPC list. [Online]. Available: https://www.top500.org/green500/lists/2018/11/

[2] K. Bergman, "Empowering flexible and scalable high performance architectures with embedded photonics," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2018, p. 378.

[3] G. Georgakoudis, N. Jain, T. Ono, K. Inoue, S. Miwa, and A. Bhatele, "Evaluating the impact of energy efficient networks on hpc workloads," in *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, 2019, pp. 301–310.

[4] G. Michelogiannakis, Y. Shen, M. Y. Teh, X. Meng, B. Aivazi, T. Groves, J. Shalf, M. Glick, M. Ghobadi, L. Dennison *et al.*, "Bandwidth steering in HPC using silicon nanophotonics," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)*, 2019, pp. 1–25.

[5] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *Proceedings of the 9th International Conference on High Performance Computing for Computational Science*, ser. VECPAR'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 1–25.

[6] K. Wen, P. Samadi, S. Rumley, C. P. Chen, Y. Shen, M. Bahadori, K. Bergman, and J. Wilke, "Flexfly: Enabling a reconfigurable dragonfly through silicon photonics," in *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*, 2016.

[7] F. Redaelli, M. D. Santambrogio, and D. Sciuto, "Task scheduling with configuration prefetching and anti-fragmentation techniques on dynamically reconfigurable systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2008.

[8] G. Michelogiannakis, K. Z. Ibrahim, J. Shalf, J. J. Wilke, S. Knight, and J. P. Kenny, "Aphid: Hierarchical task placement to enable a tapered fat tree topology for lower power and cost in hpc networks," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2017, pp. 228–237.

[9] E. Jeannot, G. Mercier, and F. Tessier, "Process placement in multicore clusters:algorithmic issues and practical techniques," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 993–1002, April 2014.

[10] M. S. Rahman, S. Bhowmik, Y. Ryasnianskiy, X. Yuan, and M. Lang, "Topology-custom UGAL routing on dragonfly," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)*, 2019, pp. 1–15.

[11] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," in *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3. ACM, 2009, pp. 220–231.

[12] N. McDonald, M. Isaev, A. Flores, A. Davis, and J. Kim, "Practical and efficient incremental adaptive routing for hyperx networks," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)*, 2019, pp. 1–13.

[13] P. Geoffray and T. Hoefler, "Adaptive routing strategies for modern high performance networks," in *2008 16th IEEE Symposium on High Performance Interconnects*, 2008, pp. 165–172.

[14] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "Firefly: A reconfigurable wireless data center fabric using free-space optics," in *SIGCOMM*, 2014, pp. 319–330.

[15] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "Projector: Agile reconfigurable data center interconnect," in *SIGCOMM*, 2016.

[16] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, "Mirror mirror on the ceiling: Flexible wireless links for data centers," in *SIGCOMM*, 2012.

[17] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," in *Proc. HotNets*, 2009.

[18] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," in *SIGCOMM*, 2011.

[19] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *SIGCOMM*, 2011.

[20] J. Shalf, S. Kamil, L. Oliker, and D. Skinner, "Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect," in *SC'05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*. IEEE, 2005, pp. 17–17.

[21] M. Y. Teh, Z. Wu, and K. Bergman, "Flexspander: Augmenting expander networks in high performance computing and data centers with optical bandwidth steering," *Journal of Optical Communications and Networking*, Jan 2020.

[22] K. C. Webb, A. C. Snoeren, and K. Yocum, "Topology switching for data center networks," in *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, ser. Hot-ICE'11. USA: USENIX Association, 2011, p. 14.

[23] G. Yuan, R. Proietti, X. Liu, A. Castro, D. Zang, N. Sun, C. Liu, Z. Cao, and S. J. B. Yoo, "ARON: Application-driven reconfigurable optical networking for hpc data centers," in *ECOC 2016; 42nd European Conference on Optical Communication*, 2016, pp. 1–3.

[24] B. Abali, R. J. Eickemeyer, H. Franke, C. Li, and M. Taubenblatt, "Disaggregated and optically interconnected memory: when will it be cost effective?" *CoRR*, vol. abs/1503.01416, 2015. [Online]. Available: http://arxiv.org/abs/1503.01416

[25] M. Bielski, C. Pinto, D. Raho, and R. Pacalet, "Survey on memory and devices disaggregation solutions for hpc systems," in *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, 2016, pp. 197–204.

[26] Y. Zhu, W. Yu, B. Jiao, K. Mohror, A. Moody, and F. Chowdhury, "Efficient user-level storage disaggregation for deep learning," in *IEEE International Conference on Cluster Computing (CLUSTER)*, 2019.

[27] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker, "Network requirements for resource disaggregation," in *OSDI*, 2016.

[28] H. Meyer, J. C. Sancho, J. V. Quiroga, F. Zyulkyarov, D. Roca, and M. Nemirovsky, "Disaggregated computing. an evaluation of current trends for datacentres," *Procedia Computer Science"*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050917306968

[29] "Optical networking for adaptive networks," *Enterprise Networking*, May 2019.

[30] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[31] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J. Sexton, and R. Walkup, "Optimizing task layout on the blue gene/l supercomputer," *IBM Journal on Research and Development*, vol. 49, March/May 2005.

[32] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *SIGCOMM*, 2015.

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

[34] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: topology, routing, and packaging of efficient large-scale networks," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)*, 2009.

[35] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, "Expanding across time to deliver bandwidth efficiency and low latency," in *NSDI*, 2020, pp. 1–18.

[36] M. Mubarak, P. Carns, J. Jenkins, J. K. Li, N. Jain, S. Snyder, R. Ross, C. D. Carothers, A. Bhatele, and K.-L. Ma, "Quantifying i/o and communication traffic interference on dragonfly networks equipped with burst buffers," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2017, pp. 204–215.

[37] K.-T. Foerster and S. Schmid, "Survey of reconfigurable data center networks: Enablers, algorithms, complexity," *SIGACT News*, vol. 50, no. 2, p. 62–79, Jul. 2019. [Online]. Available: https://doi.org/10.1145/3351452.3351464

[38] Y. Xia, X. S. Sun, S. Dzinamarira, D. Wu, X. S. Huang, and T. S. E. Ng, "A tale of two topologies: Exploring convertible data center network architectures with flat-tree," in *SIGCOMM*, 2017, pp. 295–308.

[39] H. Eberle and N. Gura, "Separated high-bandwidth and low-latency communication in the cluster interconnect clint," in *Proceedings of the IEEE Conference on Supercomputing*, 2002.

[40] K. J. Barker, A. Benner, R. Hoare, A. Hoisie, A. K. Jones, D. K. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker, "On the feasibility of optical circuit switching for high performance computing systems," in *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, Nov 2005, pp. 16–16.

[41] G. Wang, D. G. Andersen, M. Kaminsky, M. Kozuch, T. S. E. Ng, K. Papagiannaki, M. Glick, and L. B. Mummert, "Your data center is a router: The case for reconfigurable optical circuit switched paths." in *HotNets*, 2009.

[42] Y. Tang, H. Guo, and J. Wu, "Ocbridge: An efficient topology reconfiguration strategy in optical data center network," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, March 2018, pp. 1–3.

[43] Y. Tarutani, Y. Ohsita, and M. Murata, "Virtual network reconfiguration for reducing energy consumption in optical data centers," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 6, no. 10, pp. 925–942, Oct 2014.

[44] Z. Zhao, B. Guo, Y. Shang, and S. Huang, "Hierarchical and reconfigurable optical/electrical interconnection network for high-performance computing," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 12, no. 3, pp. 50–61, 2020.

[45] A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira, "Xpander: Towards optimal-performance datacenters," in *CoNEXT*, 2016.

[46] T. Hoefler, R. Rabenseifner, H. Ritzdorf, B. R. de Supinski, R. Thakur, and L. Jesper, "The scalable process topology interface of MPI 2.2," *Concurrency and Computation: Practice and Experience*, Mar 2011.

[47] A. H. Abdel-Gawad, M. Thottethodi, and A. Bhatele, "Rahtm: Routing algorithm aware hierarchical task mapping," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2014, pp. 325–335.

[48] G. Almási, C. Archer, J. G. Castaños, C. C. Erway, P. Heidelberger, X. Martorell, J. E. Moreira, K. Pinnow, J. Ratterman, N. Smeds, B. Steinmacher-burow, W. Gropp, and B. Toonen, "Implementing MPI on the BlueGene/L Supercomputer," in *Euro-Par 2004 Parallel Processing*, M. Danelutto, M. Vanneschi, and D. Laforenza, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 833–845.

[49] Y. Li, H. Liu, W. Yang, D. Hu, and W. Xu, "Inter-data-center network traffic prediction with elephant flows," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 206–213.

[50] A. Azzouni and G. Pujolle, "Neutm: A neural network-based framework for traffic matrix prediction in sdn," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018, pp. 1–5.

[51] L. Nie, D. Jiang, L. Guo, S. Yu, and H. Song, "Traffic matrix prediction and estimation based on deep learning for data center networks," in *2016 IEEE Globecom Workshops (GC Wkshps)*, Dec 2016, pp. 1–6.

[52] 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 *International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2012.

[53] A. Shpiner, Z. Haramaty, S. Eliad, V. Zdornov, B. Gafni, and E. Zahavi, "Dragonfly+: Low cost topology for scaling datacenters," in *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*. IEEE, 2017, pp. 1–8.

[54] Z. Yang, Y. Cui, S. Xiao, X. Wang, M. Li, C. Li, and Y. Liu, "Achieving efficient routing in reconfigurable dcns," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, Dec. 2019. [Online]. Available: https://doi.org/10.1145/3366695

[55] T. Fenz, K. Foerster, S. Schmid, and A. Villedieu, "Efficient non-segregated routing for reconfigurable demand-aware networks," in *2019 IFIP Networking Conference (IFIP Networking)*, 2019, pp. 1–9.

[56] K. Pan, H. Li, W. Liu, Z. Zhu, and B. Zhu, "On the optimal routing for reconfigurable network architecture," in *9th International Conference on Communications and Networking in China*, 2014, pp. 154–159.

[57] K.-T. Foerster, M. Pacut, and S. Schmid, "On the complexity of non-segregated routing in reconfigurable data center architectures," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 2, p. 2–8, May 2019.

[58] Z. Qian, P. Bogdan, G. Wei, C.-y. Tsui, and R. Marculescu, "A traffic-aware adaptive routing algorithm on a highly reconfigurable network-

on-chip architecture," in *IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (CODES+ISSS)*, Oct 2012.

[59] M. H. Cho, M. Lis, K. S. Shim, M. Kinsy, T. Wen, and S. Devadas, "Oblivious routing in on-chip bandwidth-adaptive networks," in *2009 18th International Conference on Parallel Architectures and Compilation Techniques*, 2009, pp. 181–190.

[60] P. Basu, A. Bar-Noy, R. Ramanathan, and M. P. Johnson, "Modeling and analysis of time-varying graphs," *CoRR*, vol. abs/1012.0260, 2010. [Online]. Available: http://arxiv.org/abs/1012.0260

[61] S. A. Jyothi, A. Singla, P. Godfrey, and A. Kolla, "Measuring and understanding throughput of network topologies," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2016.

[62] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *Journal of the ACM (JACM)*, vol. 37, no. 2, pp. 318–334, 1990.

[63] L. Gurobi Optimization, "Gurobi optimizer reference manual," in *"http://www.gurobi.com"*, 2019.

[64] O. M. Abusaid and F. M. Salem, "Kullback-leibler divergence minimization for competitive learning of self-organizing maps," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[65] M. Y. Teh, S. Zhao, and K. Bergman, "Metteor: Robust multi-traffic topology engineering for commercial data center networks," in *arXiv*, 2020.

[66] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE transactions on Computers*, vol. 100, no. 10, pp. 892–901, 1985.

[67] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[68] Netbench, https://github.com/ndal-eth/netbench.

[69] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *SIGCOMM*. New York, NY, USA: ACM, 2017, pp. 281–294.

[70] M. Y. Teh, "Tago: Software simulator," Jun 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3870995

[71] Characterization of the DOE mini-apps. Accessed: 2019-02-16. [Online]. Available: https://portal.nersc.gov/project/CAL/doe-miniapps.htm

[72] H. Adalsteinsson, S. Cranford, D. A. Evensky, J. P. Kenny, J. Mayo, A. Pinar, and C. L. Janssen, "A simulator for large-scale parallel computer architectures," *Int. J. Distrib. Syst. Technol.*, vol. 1, no. 2, pp. 57–73, Apr. 2010.

[73] S. Ethier, W. M. Tang, and Z. Lin, "Gyrokinetic particle-in-cell simulations of plasma microturbulence on advanced computing platforms," *Journal of Physics: Conference Series*, vol. 16, pp. 1–15, 2005.

[74] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[75] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[76] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers, randomly," in *NSDI*, 2012.