FatPaths: Routing in Supercomputers and Data Centers when Shortest Paths Fall Short

Maciej Besta[1], Marcel Schneider[1], Marek Konieczny[2], Karolina Cynk[2],
Erik Henriksson[1], Salvatore Di Girolamo[1], Ankit Singla[1], Torsten Hoefler[1]
[1]*Department of Computer Science, ETH Zurich*
[2]*Faculty of Computer Science, Electronics and Telecommunications, AGH-UST*

***Abstract*—We introduce FatPaths: a simple, generic, and robust routing architecture that enables state-of-the-art low-diameter topologies such as Slim Fly to achieve unprecedented performance. FatPaths targets Ethernet stacks in both HPC supercomputers as well as cloud data centers and clusters. FatPaths exposes and exploits the rich ("fat") diversity of both minimal and non-minimal paths for high-performance multi-pathing. Moreover, FatPaths uses a redesigned "purified" transport layer that removes virtually all TCP performance issues (e.g., the slow start), and incorporates flowlet switching, a technique used to prevent packet reordering in TCP networks, to enable very simple and effective load balancing. Our design enables recent low-diameter topologies to outperform powerful Clos designs, achieving 15% higher net throughput at 2× lower latency for comparable cost. FatPaths will significantly accelerate Ethernet clusters that form more than 50% of the Top500 list and it may become a standard routing scheme for modern topologies.**

Full paper version: https://arxiv.org/abs/1906.10885

I. Introduction

Ethernet continues to be important in the HPC landscape. While the most powerful Top500 systems use vendor-specific or Infiniband (IB) interconnects, more than half of the Top500 (cf. the Nov. 2019 issue) machines [1] are based on Ethernet, see Figure 1 (the left plot). For example, Mellanox connects 156 Ethernet systems (25 GiB and faster), which is 20% more than in Nov. 2018. The Green500 list is similar [2]. The importance of Ethernet is increased by the "convergence of HPC and Big Data", with cloud providers and data center operators aggressively aiming for high-bandwidth and low-latency fabric [3]–[5]. An example is the growing popularity of RDMA over Converged Ethernet (RoCE) [6] that facilitates deploying Remote Direct Memory Access (RDMA) [7] applications and protocols – traditionally associated with HPC and IB interconnects – on top of Ethernet.

Yet, Ethernet systems are scarce in the *highest 100* positions of Top500. For example, in November 2019, only *six* such systems were among the highest 100. Ethernet systems are also less efficient than Infiniband, custom, OmniPath, and proprietary systems, see Figure 1 (on the right). This is also the case for systems with similar sizes, injection bandwidth, and topologies, indicating overheads caused by routing. Thus, enhancing routing in HPC Ethernet clusters would improve the overall performance of ≈50% of Top500 systems and accelerate cloud infrastructure, mostly based on Ethernet [8].

Clos dominates the landscape of data centers and supercomputers [3], [4], [9]. Yet, many recent *low-diameter topologies* claim to outperform Clos in the cost-performance tradeoff. For example, Slim Fly can be ≈2× more cost- and power-efficient while having ≈25% lower latency. Similar numbers were reported for Jellyfish [10] and Xpander [3]. Thus, *modern low-diameter networks could significantly enhance compute capabilities of Ethernet clusters.*

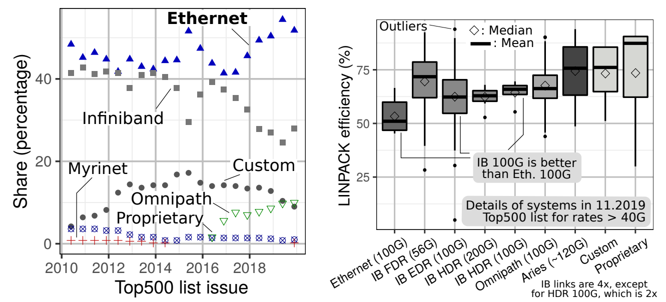

Fig. 1: The percentage of Ethernet systems in the Top500 list (on the left) and the LINPACK efficiency of Top500 systems with various networks (on the right).

However, to the best of our knowledge, no high-performance routing architecture has been proposed for low-diameter networks based on Ethernet stacks. The key issue here is that traditional routing schemes (e.g., Equal-Cost Multipath (ECMP) [11]) cannot be directly used in networks such as Slim Fly, because (as we will show) there is almost always only one shortest path between endpoint pairs (i.e., *shortest paths fall short*). Restricting traffic to these paths does not utilize such topologies' path diversity, and it is unclear how to split traffic across non-shortest paths of *unequal* lengths.

To answer this, we propose FatPaths, the first high-performance, simple, and robust routing architecture for low-diameter networks, to accelerate both HPC systems and cloud infrastructure that use Ethernet. FatPaths uses our *key research outcome*: although low-diameter networks fall short of shortest paths, they have enough "almost" shortest paths. This insight comes from our in-depth analysis of path diversity in five low-diameter topologies (contribution **#1**). Then, in our *key design outcome*, we show how to encode this rich diversity of non-minimal paths in low-diameter networks in commodity hardware (HW) using a novel routing approach called layered routing (contribution **#2**). Here, we divide network links into subsets called layers. One layer contains at least one "almost" shortest path between any two endpoints. Non-minimal multipath routing is then enabled by using more than one layer.

For higher performance in TCP environments, FatPaths uses

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

Routing Scheme (Name, Abbreviation, Reference)	Stack Layer	Supported path diversity aspect SP	NP	SM	MP	DP	ALB	AT
(1) SIMPLE ROUTING PROTOCOLS (often used as building blocks):								
Valiant load balancing (VLB) [13]	L2–L3	👎	👍	👎	👎	👎	👎	👍
Simple Spanning Tree (ST) [14]	L2	👎^{S}	👎^{S}	👎	👎	👎	👎	👍
Simple routing, e.g., OSPF [15]–[18]	L2, L3	👍	👎	👎	👎	👎	👎	👍
UGAL [19]	L2–L3	👍	👍	👎	👎	👎	👍	👍
ECMP [11], OMP [20], Pkt. Spraying [21]	L2, L3	👍	👎	👎	👍	👎	👎	👍
(2) ROUTING ARCHITECTURES:								
DCell [22]	L2–L3	👎	👍	👎	👎	👎	👎	👎
Monsoon [23]	L2, L3	(👍)	(👍)	👎	(👍)	👎	👎	👎
PortLand [9]	L2	👍	👎	👎	👍	👎	👎	👎
DRILL [24], LocalFlow [25], DRB [26]	L2	👍	👎	👎	👍	👎	👍	👎
VL2 [27]	L3	👍	👎	👎	👍	👎	(👍)	👎
Architecture by Al-Fares et al. [28]	L2–L3	👍	👎	👎	👍	👍	👍	👎
BCube [29]	L2–L3	👍	👎	👎	👍	👍	👎	👎
SEATTLE [30], others* [31]–[34]	L2	👍	👎	👎	👎	👎	👎	👍
VIRO [35]	L2–L3	👎^{S}	👎^{S}	👎	👎	👎	👎	👍
Ethernet on Air [36]	L2	👎^{S}	👎^{S}	👎	(👍)^R	👎	👎	👍
PAST [37]	L2	(👍)^S	(👍)^S	👎	👎	👍	👎	👍
MLAG, MC-LAG, others [38]	L2	(👍)	(👍)	👎	(👍)^R	👎	👎	👍
MOOSE [39]	L2	👍	👎	👎	(👍)	👎	👎	👍
MPA [40]	L3	👍	👍	👎	👍	👎	👎	👍
AMP [41]	L3	👍	👎	👎	👍	👎	👍	👍
MSTP [42], GOE [43], Viking [44]	L2	👎^{S}	👎^{S}	👎	👍	👎	👎	👍
SPB [45], TRILL [46], Shadow MACs [47]	L2	👍	👎^{R}	👎	👍	👎	👎	👍
SPAIN [48]	L2	(👍)^S	(👍)^S	(👍)^S	👍	👍	👎	👍
(3) FatPaths [This work]	L2–L3	👍	👍	👍	👍	👍	👍	👍

TABLE I: **Comparison of routing schemes, focusing on their support for path diversity. SP**, **NP**: support for arbitrary **shortest** and **non-minimal** paths, respectively. **SM**: A given scheme **simultaneously** enables minimal and non-minimal paths. **MP**: support for **multi-pathing** (between two hosts). **DP**: support for **disjoint** paths. **ALB**: support for **adaptive load balancing**. **AT**: compatible with an **arbitrary topology**. 👍, (👍), 👎: A given scheme, respectively, offers a given feature, offers it in a limited way (e.g., Monsoon [23] uses multi-pathing (ECMP) only between border and access routers), and does not offer it. ^{R}A given feature is offered *only* for resilience (*not* performance). ^{S}Minimal or non-minimal paths are offered *only* within spanning trees.

a high-performance transport design. We seamlessly combine layered routing with several TCP enhancements [4] (such as shallow buffers) and we use flowlet switching [12], a scheme that enables very simple but powerful load balancing by sending batches of packets over multiple layers (contribution **#3**).

We exhaustively compare FatPaths to other routing schemes in Table I (contribution **#4**). FatPaths is the only scheme that simultaneously (1) enables multi-pathing using both (2) shortest and (3) non-shortest paths, (4) explicitly considers disjoint paths, (5) offers adaptive load balancing, and (6) is applicable across topologies. Table I focuses on path diversity, because, as topologies lower their diameter and reduce link count, path diversity – key to high-performance routing – becomes a scarce resource demanding careful examination.

We conduct extensive, large-scale packet-level simulations (contribution **#5**), and a comprehensive theoretical analysis (contribution **#6**). We simulate topologies with up to $\approx$*1 million endpoints*. We motivate FatPaths in Figure 2. Slim Fly and Xpander equipped with FatPaths ensure $\approx$15% higher throughput and $\approx 2\times$ lower latency than similar-cost fat trees.

We stress that FatPaths outperforms the bleeding-edge Clos proposals based on per-packet load balancing (these schemes account for packet-reordering) and novel transport mechanisms, that achieve 3-4$\times$ smaller tail flow[1] completion time (FCT) than Clos based on ECMP [4], [49]. Consequently, our work illustrates that *low-diameter networks can continue to claim an improvement in the cost-performance tradeoff against the new, superior Clos baselines* (contribution **#7**).

[1]In performance analyses, we use the term "flow", which is equivalent to a "message".

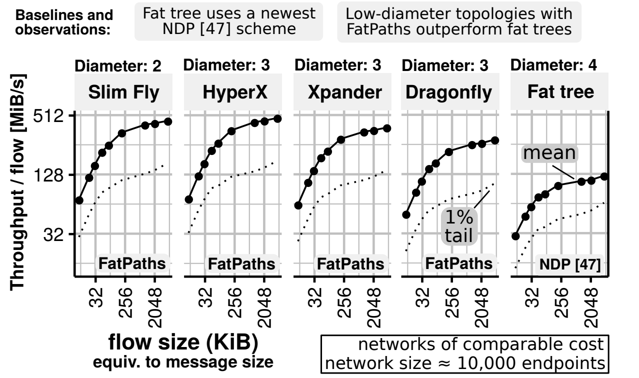

Fig. 2: Example **performance advantages of low-diameter topologies that use FatPaths over fat trees** equipped with NDP (very recent routing architecture by Handley et al. [4]). Evaluation methodology is discussed in detail in § VII.

Towards the above goals, we contribute:

- A high-performance, simple, and robust **routing architecture, FatPaths**, that enables modern low-diameter topologies such as Slim Fly to achieve unprecedented performance.
- A novel routing approach called **layered routing** that is a key ingredient of FatPaths and facilitates using diversity of non-minimal paths in modern low-diameter networks.
- The first **detailed analysis of path diversity** in five modern low-diameter network topologies, and the identification of the diversity of non-minimal paths as a key resource for their high performance.
- A **novel path diversity metric**, Path Interference, that captures bandwidth loss between specific pairs of routers.
- A comprehensive analysis of existing routing schemes in terms of their support for path diversity.
- A **theoretical analysis** showing FatPaths' advantages.
- Extensive, large-scale packet-level simulations (up to $\approx$**one million endpoints**) to demonstrate the advantages of low-diameter network topologies equipped with FatPaths over very recent Clos designs, achieving 15% higher net throughput at 2$\times$ lower latency for comparable cost.

II. Notation, Background, Concepts

We first outline basic concepts; Table II compiles the notation.

Network model	V, E	Sets of vertices/edges (routers/links, $V = \{0, \dots, N_r - 1\}$).
	N, N_r	#endpoints and #routers in the network ($N_r = \lvert V \rvert$).
	p, k'	#endpoints attached to a router, #channels to other routers.
	D, d	Network diameter and the average path length.
Paths (§ IV)	$x \in V$	Different routers used in § IV ($x \in \{s, t, a, b, c, d\}$).
	$c_l(A, B)$	*Count of (at most l-hop) disjoint paths* between router sets A, B.
	$c_{\text{min}}(s,t), l_{\text{min}}(s,t)$	*Diversity* and *lenghts of minimal paths* between routers s and t.
	$I_{ac,bd}$	*Path interference* between pairs of routers a, b and c, d.
Layers (§ V)	n	The total number of layers in FatPaths routing.
	σ_i	A layer, defined by its forwarding function, $i \in \{1, \dots, n\}$.
	ρ	Fraction of edges used in routing.

TABLE II: The **most important symbols** used in this work.

A. Network Model

We model an interconnection network as an undirected graph $G = (V, E)$; V and E are sets of routers[2] ($\lvert V \rvert = N_r$) and full-duplex inter-router physical links. Endpoints are *not* modeled explicitly. There are N endpoints, p endpoints are attached to each router (*concentration*) and k' channels from each router

[2]We abstract away HW details and use a term "router" for L2 switches and L3 routers.

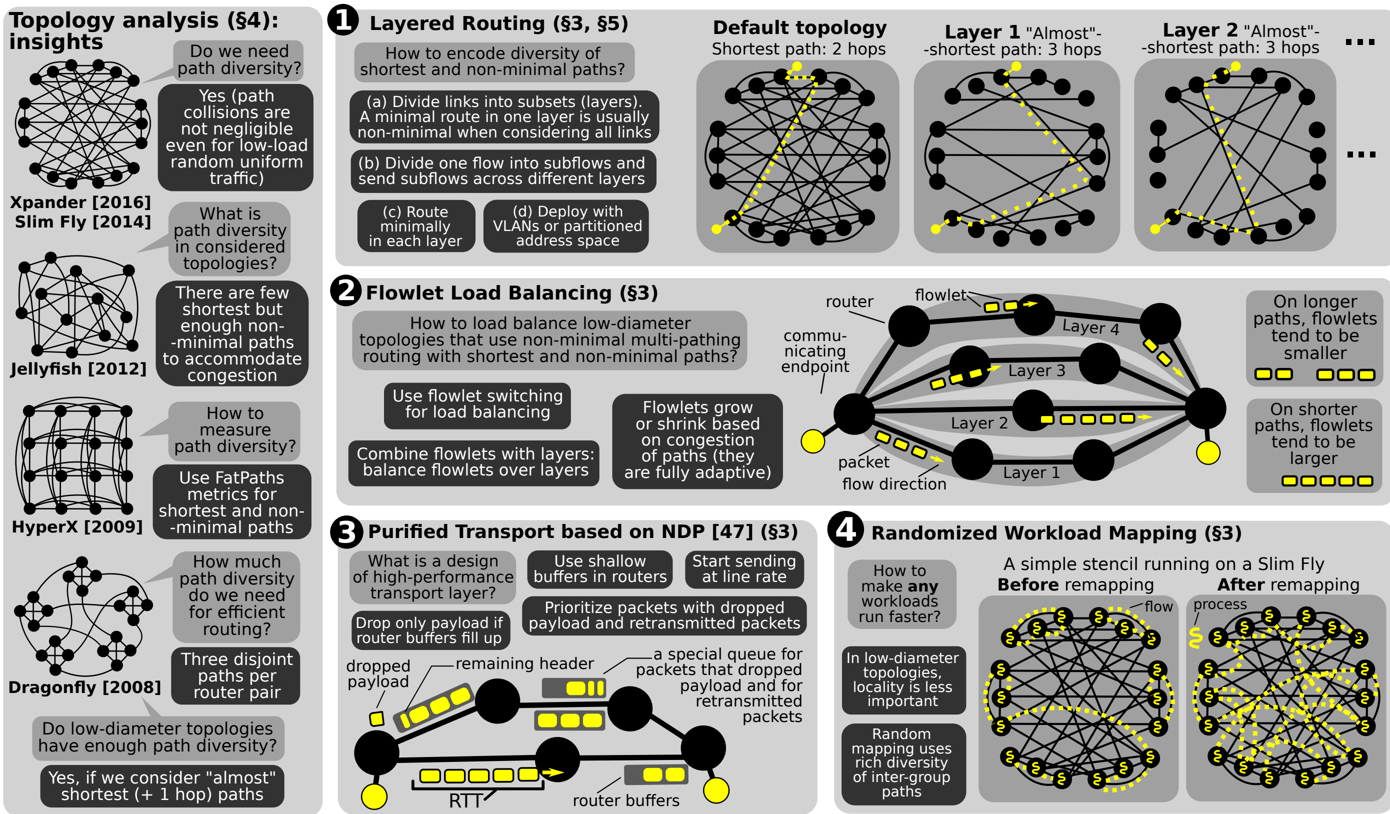

Fig. 3: **Overview of FatPaths**. Numbers (❶ – ❹) refer to parts of Section III. We present selected considered research questions with the obtained answers. "Topology analysis" summarizes our work on the path diversity of low-diameter topologies. "Path diversity" intuitively means the number of edge-disjoint paths between router pairs (details in § IV).

to other routers (*network radix*). The total router *radix* is $k = p + k'$. The diameter is D while the average path length is d.

B. Topologies and Fair Topology Setup

We consider all recent low-diameter networks: Slim Fly (SF) [50] (a variant with $D = 2$), Dragonfly (DF) [19] (the "balanced" variant with $D = 3$), Jellyfish (JF) [10] (with $D = 3$), Xpander (XP) [3] (with $D \leq 3$), HyperX (Hamming graph) (HX) [51] that generalizes Flattened Butterflies (FBF) [52] with $D = 3$. We also use established three-stage fat trees (FT3) [53] that are a variant of Clos [54]. Note that we do not detail the considered topologies. This is because our design does *not* rely on any specifics of these networks (i.e., FatPaths can be used in *any* topology, but from performance perspective, it is most beneficial for low-diameter networks).

We use four classes of sizes N: small ($N \approx 1,000$), medium ($N \approx 10,000$), large ($N \approx 100,000$), and huge ($N \approx 1,000,000$). We set $p = \frac{k'}{d}$ (in the technical report, we show that, assuming random uniform traffic, $p = \frac{k'}{d}$ maximizes throughput while minimizing congestion and network cost). Third, we select network radix k' and router count N_r so that, for a fixed N, the compared topologies use similar amounts of networking hardware and thus have similar construction costs.

Jellyfish – unlike other topologies – is "fully flexible": There is a JF instance for each combination of N_r and k'. Thus, to fully test JF, for each other network X, we use an equivalent JF (denoted as X-JF) with identical N_r, k'.

C. Traffic Patterns

We analyze recent works [50], [55]–[66] to select traffic patterns that represent important HPC and datacenter workloads. Denote a set of endpoint IDs $\{1, ..., N\}$ as V_e. Formally, a traffic pattern is a mapping from source endpoint IDs $s \in V_e$ to destination endpoints $t(s) \in V_e$. First, we select **random uniform** ($t(s) \in V_e$ u.a.r.,) and **random permutation** ($t(s) = \pi_N(s)$, where π_N is a permutation selected u.a.r.,) that represent **irregular workloads** such as graph computations, sparse linear algebra solvers, and adaptive mesh refinement methods [67]. Second, we pick **off-diagonals** ($t(s) = (s + c) \mod N$, for fixed c) and **shuffle** ($t(s) = \mathrm{rotl}_i(s) \mod N$, where the bitwise left rotation on i bits is denoted as rotl_i and $2^i < N < 2^{i+1}$). They represent **collective operations** such as MPI-all-to-all or MPI-all-gather [50], [67]. We also use **stencils, realistic traffic patterns common in HPC**. We model 2D stencils as four off-diagonals at fixed offsets $c \in \{\pm 1, \pm 1, \pm 42, \pm 42\}$. For large simulations ($N > 10,000$) we also use offsets $c \in \{\pm 1, \pm 1, \pm 1337, \pm 1337\}$ to reduce counts of communicating endpoint pairs that sit on the same switches. Finally, we use **adversarial** and **worst-case** traffic patterns. In the former, we use a skewed off-diagonal with large offsets (we make sure that it has many colliding paths). For the latter, we use a pattern (detailed in § VI) that maximizes stress on the interconnect *individually for each topology*.

III. FatPaths Architecture: Overview

We outline the FatPaths architecture in Figure 3. The key part, layered routing, is summarized here and detailed in Section V. For higher performance in TCP settings, FatPaths uses simple and robust flowlet load balancing, "purified" high-performance transport, and randomized workload mapping.

A. Layered Routing ❶

To encode minimal *and non-minimal* paths with commodity HW, FatPaths divides all links into (not necessarily disjoint) subsets called *layers*[3]. Routing within each layer uses shortest paths; these paths are usually *not* shortest when considering all network links. Different layers encode different paths between each endpoint pair. The number of layers is minimized to reduce hardware resources needed to deploy layers. Layers can easily be implemented with commodity schemes, e.g., VLANs or a simple partitioning of the address space.

B. Simple and Effective Load Balancing ❷

For simple but robust load balancing, we use flowlet switching [12], [69], originally used to alleviate packet reordering in TCP. A flowlet is a sequence of packets within one flow, separated from other flowlets by sufficient time gaps, which prevents packet reordering at the receiver. Flowlet switching can provide a *very* simple load balancing: a router simply picks a random path for each flowlet, without *any* probing for congestion [5]. Such load balancing is powerful because flowlets are *elastic*: their size changes *automatically* based on network conditions. On high-latency and low-bandwidth paths, flowlets are usually shorter as time gaps large enough to separate two flowlets are more frequent. Then, low-latency and high-bandwidth paths host longer flowlets as such time gaps appear less often. Now, we propose to use flowlets in *low-diameter* networks, to load balance FatPaths. We combine flowlets with layered routing: flowlets are sent using *different layers*. The key observation is that elasticity of flowlets *automatically* ensures that such load balancing considers both static and dynamic network properties (e.g., longer vs. shorter paths and more vs. less congestion). Consider a pair of communicating routers. As we show in § IV, virtually all router pairs in a low-diameter network are connected with exactly one shortest part but multiple non-minimal paths, possibly of different lengths. A shortest path often has smallest congestion while longer paths are more likely to be congested. Here, flowlet elasticity ensures that larger flowlets are sent over shorter and less congested paths. Shorter flowlets are then transmitted over longer and usually more congested paths.

C. Purified Transport ❸

FatPaths' transport layer incorporates the NDP [4] Clos schemes to remove TCP/Ethernet performance issues in the context of low-diameter non-Clos networks. First, if router queues fill up, *only packet payload is dropped*; headers with all the metadata are preserved and the receiver has full information on the congestion in the network and can pull the data from the sender at a rate dictated by the evolving network conditions. Specifically, the receiver can request to change a layer, if packets transmitted over this layer arrive without payload, indicating congestion. Second, routers can prioritize headers of packets that lost payload, and retransmitted packets. Thus, congested flows finish quickly and head-of-line-blocking is reduced. Third, senders transmit the first RTT at line rate (no probing for available bandwidth). Finally, router queues are shallow. The resulting transport layer has low latency and high throughput, it meets demands of various traffic patterns, and it can be implemented with existing network technology.

[3]In FatPaths, a "layer" is formally a *subset* of links. We use the term "layer" as our concept is similar to "virtual layers" known from works on deadlock-freedom [68]

D. Randomized Workload Mapping ❹

We optionally assign communicating endpoints to routers *randomly*. This is often done in HPC; details are in the report. We stress that this scheme is *even more beneficial in FatPaths* due to the low diameter of targeted networks.

IV. Path Diversity in Low-Diameter Topologies

To develop FatPaths, we first need to understand the "nature" of path diversity that FatPaths benefits from. For this, we first show that low-diameter topologies exhibit congestion due to conflicting flows even in mild traffic scenarios, and we derive the minimum number of disjoint paths that eliminate flow conflicts (§ IV-A). We then formalize the "path diversity" notion (§ IV-B) to show that all low-diameter topologies have few shortest but enough non-minimal paths to accommodate flow collisions, an important type of flow conflicts (§ IV-C). To the best of our knowledge, we provide the most extensive analysis of path diversity in low-diameter networks so far (considering the number of path diversity metrics and topologies), cf. Related Work. We summarize key insights; full data is in the report (the link is on page 1).

A. How Much Path Diversity Do We Need?

FatPaths uses path diversity to avoid congestion due to *conflicting flows*. Consider two communicating pairs of endpoints. The generated flows *conflict* when their paths **collide** (flows use an identical path) or **overlap** (flows share some links), see Figure 5. Collisions are caused by *workload mapping*, when communicating endpoint pairs occupy the same router pairs. Thus, collisions *only* depend on concentration p and #routers N_r. Contrarily, overlaps depend on *topology details* (i.e., connections *between routers*). Thus, overlaps capture how well a topology can sustain a given workload.

To understand how much path diversity is needed to alleviate flow conflicts, we analyze the impact of topology properties (D, p, N) and a traffic pattern on the number of colliding paths, see Figure 4. For $D > 1$, the number of collisions is at most *three* in most cases, especially when lowering D (while increasing p). Importantly, this holds for the adversarial $4\times$ oversubscribed patterns that stress the interconnect. For $D = 1$, at least nine collisions occur for more than 1% of router pairs, even in mild traffic patterns. While we do not consider $D = 1$ in practical applications, we indicate that global DF links form a complete graph, demanding high path diversity at least with respect to the global links.

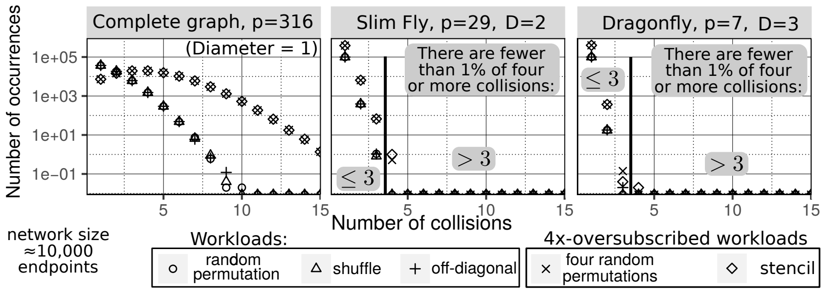

Fig. 4: Histogram of colliding paths (per router pair) with $p = \frac{k'}{D}$. We plot data for SF, DF, and a complete graph, for $N \approx 100,000$. Other sizes N and other topologies result in identical shares of colliding paths.

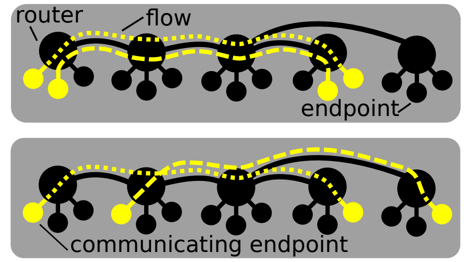

Fig. 5: Example collision (top) and overlap (bottom) of paths and flows.

Takeaway We need at least three disjoint paths per router pair to handle colliding paths in considered workloads, assuming random mapping. Now, we observe that there are at least as many overlapping paths as colliding paths (as seen from a simple counting argument: for each pair of colliding flows x and y, any other flow in the network may potentially overlap with x and y). Thus, the same holds for overlaps.

B. How Should We Measure Path Diversity?

To analyze if low-diameter topologies provide at least three disjoint paths per router pair, we need to first formalize the notion of "disjoint paths" and "path diversity" in general. For example, we must be able to distinguish between partially or fully disjoint paths that may have different lengths. Thus, we first define *the count of disjoint paths* (CDP), minimal and non-minimal, between routers (§ IV-B1). These measures address path collisions. Moreover, to analyze path overlaps, we define two further measures: *path interference* (PI, § IV-B2) and *total network load* (TNL, § IV-B3). We summarize each measure and we provide all formal details for reproducibility; these details can be omitted by readers only interested in intuition. We use several measures because any single measure that we tested cannot fully capture the rich concept of path diversity.

1) Count of Disjoint Paths (CDP)

We define the count of *disjoint* paths (CDP) between router sets $A, B \subseteq V$ at length l as the smallest number $c_l(A, B)$ of edges that must be removed so that no path of length at most l exists from any router in A to any router in B. To compute $c_l(A, B)$, first define the l-step neighborhood $h^l(A)$ of a router set A as "a set of routers at l hops away from A":

$$h(A) = \{t \in V : \exists_{s \in A}\ \{s, t\} \in E\} \quad \text{("routers attached to } A\text{")}$$

$$h^l(A) = \underbrace{h(\cdots h(}_{l \text{ times}} A) \cdots) \quad \text{("}l\text{-step neighborhood of } A\text{").}$$

Now, the condition that no path of length at most l exists between any router in A to any router in B is $h^l(A) \cap B = \emptyset$. To derive the values of $c_l(A, B)$, we use a variant of the Ford-Fulkerson algorithm [70] (with various pruning heuristics) that removes edges in paths between designated routers in A and B (at various distances l) and verifies whether $h^l(A) \cap B = \emptyset$. We are most often interested in pairs of designated routers s and t, and we use $A = \{s\}, B = \{t\}$.

Minimal paths are vital in congestion reduction as they use fewest resources for each flow. We derive the distribution of minimal path *lengths* $l_{\min}$ and *counts* $c_{\min}$. Intuitively, $l_{\min}$ describes (statistically) *distances* between any router pairs while $c_{\min}$ provides their respective *diversities*. We have:

$$l_{\min}(s, t) = \arg\min_{i \in \mathbb{N}} \{t \in h^i(\{s\})\} \quad \text{("minimal path lengths")}$$

$$c_{\min}(s, t) = c_l(\{s\}, \{t\}) \text{ with } l = l_{\min}(s, t) \quad \text{("minimal path counts")}$$

Note that the diameter D equals $\max_{s,t}\ l_{\min}(s, t)$.

To analyze **non-minimal paths**, we reuse the count of disjoint paths CDP $c_l(A, B)$ of random router pairs $s \in A, t \in B$, but with path lengths l larger than $l_{\min}(s, t)$ ($l > l_{\min}(s, t)$). Here, we are interested in distributions of counts of non-minimal paths for fixed non-minimal distances l.

2) Path Interference (PI)

With Path Interference (PI), we want to quantify path overlaps. This is challenging because overlaps depend on the details of the structure of each topology as well as on workload mappings. Thus, a PI definition must be *local* in that it should consider *all* router pairs that may possibly communicate. Consider two router pairs a, b and c, d where a communicates with b and c communicates with d. Now, paths between these two pairs *interfere* if their total count of disjoint paths (at a given distance l), $c_l(\{a, c\}, \{b, d\})$, is lower than the sum of individual counts of disjoint paths (at l): $c_l(\{a\}, \{b\}) + c_l(\{c\}, \{d\})$. We denote path interference with $I^l_{ac,bd}$ and define it as

$$I^l_{ac,bd} = c_l(\{a, c\}, \{b\}) + c_l(\{a, c\}, \{d\}) - c_l(\{a, c\}, \{b, d\})$$

Path interference captures and quantifies the fact that, if a and b communicate *and* c and d communicate *and* the flows between these two pairs use paths that are not fully disjoint (due to, e.g., not ideal routing), then the available bandwidth between any of these two pairs of routers is reduced.

3) Total Network Load (TNL)

TNL is a simple upper bound on the number of flows that a network can maintain without congestion (i.e., without flow conflicts). There are $k'N_r$ links in a topology. Now, a flow occupying a path of length l "consumes" l links. Thus, with the average path length of d, TNL is defined as $\frac{k'N_r}{d}$, because $\#flows \leq \frac{k'N_r}{d}$. Thus, TNL constitutes the maximum *supply of path diversity* offered by a specific topology.

Takeaway We suggest to use several measures to analyze the rich nature of path diversity, e.g., the count of minimal and non-minimal paths (for collisions), and path interference as well as the total network load (for overlaps).

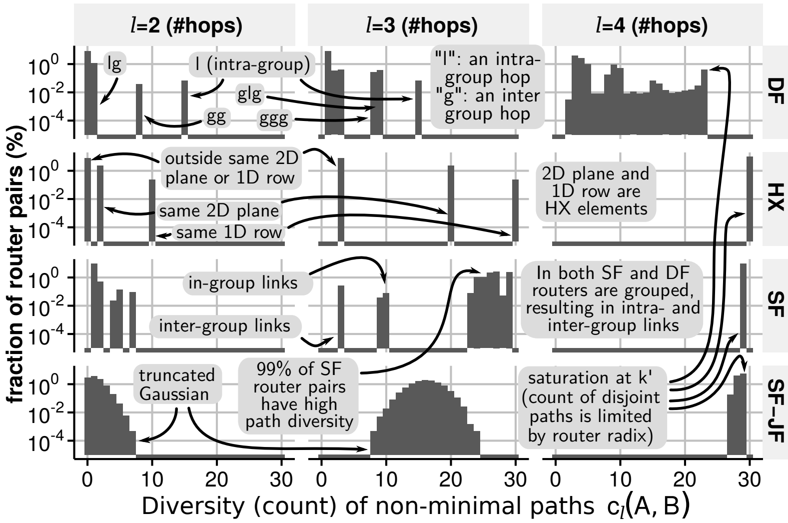

Fig. 7: Distribution of the number of non-minimal edge-disjoint paths with up to l hops ($c_l(A, B)$) between random router pairs ($N \approx 10,000$).

	d'	D	k'	N_r	N
Topology parameters					
clique	2	1	100	101	10100
SF	3	2	29	722	10108
XP	3	3	32	1056	16896
HX	3	3	30	1331	13310
DF	4	3	23	2064	16512
FT3	4	4	18	1620	11664
Default topology variant	d'	**CDP** (mean)	**CDP** (1% tail)	**PI** (mean)	**PI** (99.9% tail)
clique	2	100%	100%	2%	2%
SF	3	89%	10%	26%	79%
XP	3	49%	34%	20%	41%
HX	3	25%	10%	9%	67%
DF	4	25%	13%	8%	74%
FT3	4	100%	100%	0	0
Equivalent Jellyfish	d'	**CDP** (mean)	**CDP** (1% tail)	**PI** (mean)	**PI** (99.9% tail)
SF-JF	3	56%	38%	23%	45%
XP-JF	3	51%	34%	21%	41%
HX-JF	3	50%	23%	17%	37%
DF-JF	4	87%	78%	13%	26%
FT3-JF	4	96%	90%	5%	14%

TABLE III: Counts of disjoint non-minimal paths CDP ($c_{d'}(A, B)$) and path interference PI $\left(I^{d'}_{ac,bd}\right)$ at distance d'; d' is chosen such that the tail $c_{d'}(A, B) \geq 3$.

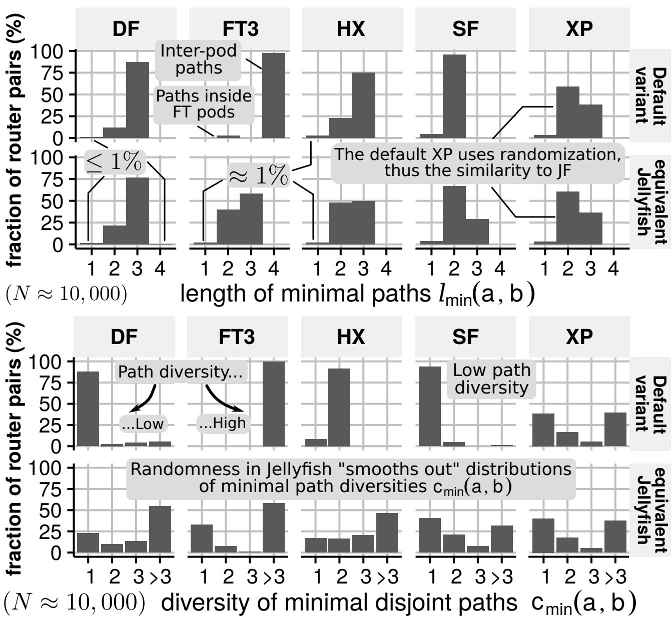

Fig. 6: Distributions of lengths and counts of shortest paths.

C. *Do We Have Enough Path Diversity?*

We now use our measures for path diversity analysis.

1) *Analysis of Minimal Paths*

Selected results for minimal paths are in Figure 6. In DF and SF, most routers are connected with one minimal path. In XP, more than 30% of routers are connected with *one* minimal path. In JF, the results are more leveled out, but pairs of routers with one shortest part in-between still form large fractions. FT3 and HX show the highest diversity, with very few unique minimal paths, while the matching JFs have lower diversities. The results match the structure of each topology (e.g., one can distinguish intra- and inter-pod paths in FT3).

Takeaway In all the considered low-diameter topologies, shortest paths fall short: at least a large fraction of router pairs are connected with *only one* shortest path.

2) *Analysis of Non-Minimal Paths*

For non-minimal paths, we first summarize the results in Table III. We report counts of disjoint paths as *fractions of router radix* k' to make these counts radix-invariant. For example, the mean CDP of 89% in SF means that 89% of router links host disjoint paths. In general, all deterministic topologies provide higher disjoint path diversity than their corresponding JFs, but there are specific router pairs with lower diversity that lead to undesired tail behavior. JFs have more predictable tail behavior due to the Gaussian distribution of $c_l(A, B)$. A closer analysis of this distribution (Figure 7) reveals details about each topology. For example, for HX, router pairs can clearly be separated into classes sharing zero, one, or two coordinate values, corresponding to the HX array structure. Another example is SF, where lower $c_l(A, B)$ are related to pairs connected with an edge while higher $c_l(A, B)$ in DF are related to pairs in the same group or pairs connected with specific sequences of local and global links. We describe all remaining data in the extended report.

Takeaway Overall, considered topologies have 3 disjoint "almost"-minimal (one hop longer) paths per router pair.

3) *Analysis of Path Interference*

Next, we sample router pairs u.a.r. and derive full **path interference** distributions; they all follow the Gaussian distribution. Selected results are in Figure 8 (we omit XP and XP-JF; both are nearly identical to SF-JF) As the combination space is very large, most samples fall into a common case, where PI is small. We thus provide full data in the report and focus on the extreme tail of the distribution (we show both mean and tail), see Table III. We use radix-invariant PI values (as for CDP) at a distance d' selected to ensure that the 99.9% tail of collisions $c_{d'}(A, B)$ is at least 3. Thus, we analyze PI in cases where demand from a workload outgrows the "supply of path diversity" from a network (three disjoint paths per router pair). All topologies except for DF achieve negligible PI for $d' = 4$, but the diameter-2 topologies do experience PI at $d' = 3$. SF shows the lowest PI in general, but has (few) high-interference outliers. In general, random JFs have higher average PI but less PI in tails, while deterministic topologies tend to perform better on average with worse tails.

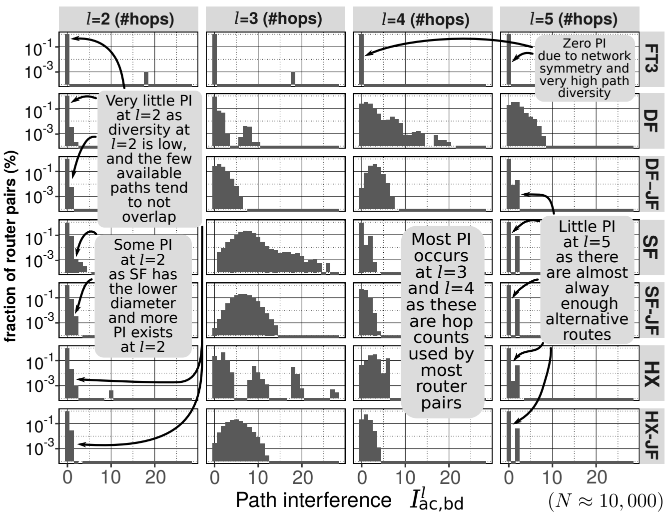

Fig. 8: Distribution of path interference PI $\left(I^l_{ac,bd}\right)$ at various distances l.

D. Final Takeaways on Path Diversity

We show a fundamental tradeoff between path length and diversity. High-diameter topologies (e.g., FT) have high path diversity, even on minimal paths. Yet, due to longer paths, they need more links for an equivalent N and performance. Low-diameter topologies fall short of shortest paths, but do provide enough diversity of non-minimal paths, requiring non-minimal routing. Yet, this may reduce the cost advantage of low-diameter networks *with adversarial workloads* that use many non-minimal paths, consuming additional links. *Workload randomization in FatPaths suffices to avoid this effect*. **Overall, low-diameter topologies host enough path diversity for alleviating flow conflicts. We now show how to effectively use this diversity in FatPaths.**

V. FatPaths: Design and Implementation

FatPaths is a high-performance, simple, and robust *routing architecture* that uses rich path diversity in low-diameter topologies to enhance Ethernet stacks in data centers and supercomputers. FatPaths aims to accelerate both datacenter and HPC workloads. We outlined FatPaths in § III. Here, we detail the layered routing scheme that is capable of encoding the rich diversity of both minimal and non-minimal paths, and can be implemented with commodity Ethernet hardware.

A. Routing Model

We assume simple destination-based routing, compatible with any relevant technology, including source-based systems like NDP. To compute the output port $j \in \{1, \dots, k'\}$ in a router $s \in V$ for a packet addressed to a router $t \in V$, and simultaneously the ID of the next-hop router $s' \in V$, a routing function $(j, s') = \sigma(s, t)$ is evaluated. By iteratively applying σ with fixed t we eventually reach $s' = t$ and finish. The function σ must ensure no loops on any path.

B. Layered Routing in FatPaths

We use n routing functions $\sigma_1, \dots, \sigma_n$ for n layers. Each router uses σ_i for a packet with a *layer tag* i attached. The layer tags are chosen on the endpoint by the adaptivity algorithm. All layers but one accommodate a fraction of links, maintaining non-minimal paths. One layer (associated with σ_1) uses all links to host minimal paths. The fraction of links in one layer is controlled by $\rho \in [0; 1]$. Now, the interplay between ρ and n is important. More layers (higher n) that are sparse (lower ρ) give more paths that are long, giving more path diversity, but also more wasted bandwidth (as paths are long). More layers that are dense reduce wasted bandwidth but also give fewer disjoint paths. Still, this may be enough as we need three paths per router pair. *One ideally needs more dense layers or fewer sparse layers*. Thus, an important part of deploying FatPaths is selecting the best ρ and n for a given network. To facilitate implementation of FatPaths, the project repository contains layer configurations (ρ, n) that ensure high-performance routing for used topologies. *We analyze performance of different ρ and n in § VI* and § VII.

An overview of layer construction is in Listing 1. We start with one layer with all links, maintaining shortest paths. We use $n-1$ random permutations of vertices to generate $n-1$ random layers. Each such layer is a subset $E' \subset E$ with $\lfloor \rho \cdot |E| \rfloor$ edges sampled u.a.r.. E' may disconnect the network, but for the used values of ρ, this is unlikely and a small number of attempts delivers a connected network. Note that the algorithm for constructing layers is general and can be used with any topology; cf. § II-B and Section VIII.

```
L = {E} //Init a set of layers L; we start with E that corresponds to σ_1
P = {π_1(V), ..., π_{n−1}(V)} //Generate n − 1 random permutations of vertices
foreach π ∈ P do: //One iteration derives one layer associated with some σ_i
  E' = {}; foreach (u,v) ∈ E do:
    //Below, a condition "π(u) <π(v)" ensures layer's acyclicity
    //Below, a call to rnd(0,1) returns a random number ∈ [0;1)
    if(π(u) < π(v) and rnd(0,1) < ρ) then:
      E'=E' ∪ (u,v) //Add a sampled edge to the layer
  L = L ∪ {E'}
```

Listing 1: Overview of the algorithm for constructing routing layers.

We also use a variant of the above scheme in which, instead of randomized edge picking while creating paths within layers, we use a simple heuristic that minimizes path interference. For each router pair, we pick a set of paths with minimized overlap with paths already placed in any of the layers. Most importantly, while computing paths, *we prefer paths that are one hop longer than minimal ones, using the insights from the path diversity analysis (§ IV)*.

The σ_i functions are deployed using forwarding tables with minimum paths between every two routers s, t within layer σ_i. For each router s, we populate the entry for s, t in σ_i with a port that corresponds to the router s_i that is the first step on a path from s to t. We compute all such paths and choose a random first step port, if there are multiple options.

We propose two schemes to implement layers. First, a simple way to achieve separation is *partitioning of the address space*. This requires no hardware support, except for sufficiently long addresses. One inserts the layer tag anywhere in the address, the resulting forwarding tables are then simply concatenated. The software stack must support multiple addresses per interface (deployed in Linux since v2.6.12, 2005). Next, similarly to schemes like SPAIN [48] or PAST [37], one can use *VLANs* [71] that are a part of the L2 forwarding tuple and provide full separation. Still, the number of available VLANs is hardware limited, and FatPaths does not require separated queues per layer. Finally, *L2/Ethernet* addressing can be done with exact match tables; they should only support masking out a fixed field in the address before lookup, which could be achieved with, for example, P4 [72].

C. Fault-Tolerance

Fault-tolerance in FatPaths is based on preprovisioning multiple paths within different layers. For major (infrequent) topology updates, we recompute layers [48]. Contrarily, when a failure in some layer is detected, FatPaths redirects the affected flows to a different layer. We rely on established fault tolerance schemes [4], [5], [35], [48], [73] for the exact mechanisms of failure detection. Traffic redirection relies on flowlets [5], as with congestion: the elasticity of flowlets automatically prevents data from using an unavailable path.

VI. Theoretical Analysis

We first conduct a theoretical analysis. The main goal is to illustrate that layered routing in FatPaths enables higher throughput than SPAIN [48], PAST [37], [73], and k-shortest paths [10], three recent schemes that support (1) multi-pathing and (2) disjoint paths (as identified in Table I). SPAIN uses a set of spanning trees, using greedy coloring to minimize their number; one tree is one layer. Then, paths between endpoints are mapped to the trees, maximizing path disjointness. PAST uses one spanning tree per host, aiming at distributing the trees uniformly over available physical links. k-shortest paths [10] spreads traffic over multiple shortest paths (if available) between endpoints.

A. Analysis of Number of Layers

Both SPAIN and PAST use trees as layers. This brings many drawbacks, as each SPAIN layer can use at most $N_r - 1$ links, while the topology contains $\frac{N_r k'}{2}$ links. Thus, at least $\mathcal{O}(k')$ layers are required to cover all minimal paths, and SPAIN requires even $\mathcal{O}(N_r)$ on many topologies. Moreover, PAST always needs $O(N)$ trees by its design. By using layers that are arbitrary DAGs and contain a large, constant fraction of links, *FatPaths provides sufficient path diversity with a low, $O(1)$ number of layers.*

B. Analysis of Throughput

We also analyze maximum achievable throughput (MAT) in routing schemes. MAT is defined as the maximum value $\mathcal{T}$ for which there exists a feasible multicommodity flow (MCF) that routes a flow $T(s,t) \cdot \mathcal{T}$ between all router pairs s and t, satisfying link capacity and flow conservation constraints. $T(s,t)$ specifies traffic demand; it is an amount of requested flow from s to t (more details are provided by Jyothi et al. [74]). We test all considered topologies, topology sizes, traffic patterns and intensities (fraction of communicating endpoint pairs). We consider two FatPaths variants from § V-B. We use TopoBench, a throughput evaluation tool [74] that uses linear programming (LP) to derive $\mathcal{T}$. We extended TopoBench's LP formulation of MCF to include layered routing. Most importantly, instead of one network for accommodating MCF, we use n networks (that represent layers) to allocate flows. We also introduce constraints that prevent one flow from being allocated over multiple layers.

Selected results are in Figure 9. We focus on a recently proposed worst-case traffic pattern which maximizes stress on the interconnect while hampering effective routing [74]. This pattern is generated *individually* for each topology; it uses maximum weighted matching algorithms to find a pairing of endpoints that maximizes average flow path length, using both elephant and small flows. As expected, SPAIN – a scheme developed specifically for Clos – delivers more performance on fat trees. Yet, it uses up to $O(N_r)$ layers. The layered routing that minimizes path interference generally outperforms SPAIN on other networks (we tuned SPAIN to perform as well as possible on low-diameter topologies). Finally, also as expected, our heuristic that minimizes path overlap delivers more speedup than simple random edge picking (we only plot the former for more clarity).

Tested schemes use equally many layers (n) to fix the amount of HW resources. Increasing n accelerates all comparison targets but also increases counts of forwarding entries in routing tables. Here, SPAIN and PAST become faster on fat trees and approach FatPaths, but they use up to $O(N_r)$ layers. FatPaths maintains its advantages for different traffic intensities. As expected, our heuristic that minimizes path overlap outperforms a simple random edge picking.

Takeaway FatPaths layered routing outperforms competitive schemes in the used count of layers (and thus the amount of needed hardware resources) and achieved throughput.

VII. Simulations

We now illustrate how low-diameter topologies equipped with FatPaths outperform novel high-performance fat tree designs.

A. Methodology, Parameters, and Baselines

We first discuss parameters, methodology, and baselines.

1) Topologies and Traffic Patterns

We use all topologies specified in § II-B: SF, XP, JF, HX, DF, and FT, in their most beneficial variants (e.g., the "balanced" Dragonfly [19]). We fix the network size N (N varies by up to $\approx$10% as there are limited numbers of configurations of each network). SF represents a recent family of diameter-2 topologies such as Multi-Layer Full-Mesh [60] and Two-Level Orthogonal Fat-Trees [13], [75]. To achieve similar costs and N we use $2\times$ oversubscribed fat trees.

We use the traffic patterns discussed in § II, in both randomized and skewed non-randomized variants.

2) Cost Model for Using Topologies of Comparable Cost

We select specific topologies such that they have comparable construction cost. For this, we use the established cost models from past works [19], [50], [52]. Overall, for a selected "network size category" ($N \in \{\approx 1\text{k}, \approx 10\text{k}, \approx 100\text{k}, \approx 1\text{M}\}$,

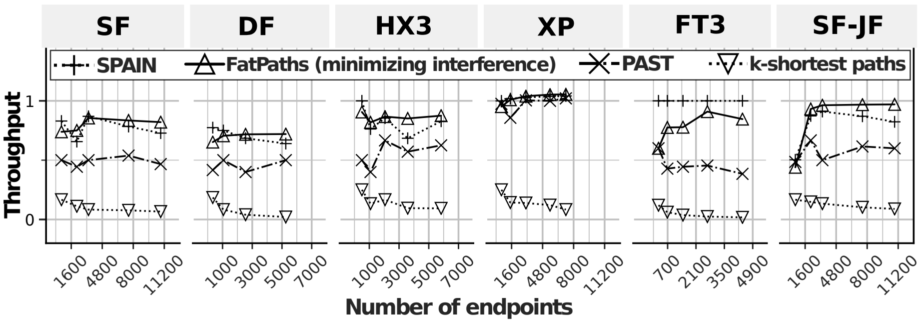

Fig. 9: **Theoretical analysis of FatPaths performance:** Maximum achievable throughput in FatPaths and other layered routing mechanisms (traffic intensity: 0.55) for the adversarial traffic pattern that maximizes stress on the interconnect.

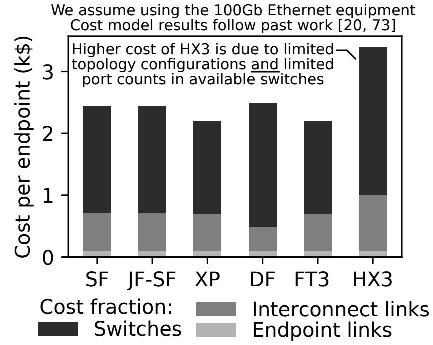

Fig. 10: An example cost model for the class of topologies with $N \approx 10,000$, assuming the 100GbE equipment.

cf. § II-B), we search for specific topology configurations with minimal differences in their sizes N. The total cost of such configurations is derived based on existing router and cable cost models [19], [50], [52]. These models use linear cost functions for both cables and routers parametrized with prices of modern equipment (e.g., Mellanox switches and cables listed on ColfaxDirect `http://www.colfaxdirect.com`). In the following analysis, we focus on $N \approx 10,000$. The model distinguishes between fiber and copper cables, with the former being used for longer router-router links (e.g., links between groups in DF or SF) and the latter forming short router-router connections (e.g., intra-group links in DF or SF) and endpoint connections. As used specific topology configurations vary in their sizes (because there is always a limited number of configurations of each used topology), to maximize fairness, the final prices are normalized per single endpoint. An example cost model is in Figure 10. One can distinguish effects caused by details of each specific topology, for example lower cable costs in DF because of relatively few expensive global inter-group connections. Variations in final costs are caused by a limited number of topology configurations combined with limited counts of ports in available switches.

3) Routing and Transport Schemes

We use flow-based non-adaptive ECMP as the routing performance lower bound. Low-diameter topologies use FatPaths while fat trees use NDP with all optimizations [4], additionally enhanced with LetFlow [5], a recent scheme that uses flowlet switching for load balancing in fat trees. We also compare to a fat tree system using NDP with per-packet congestion-oblivious load balancing as introduced by Handley et al. [4]. For FatPaths, we vary ρ and n to account for *different layer configurations*, including $\rho = 1$ (minimal paths only). Finally, we consider simple TCP, MPTCP, and DCTCP with ECN [76]–[78], showing that FatPaths can accelerate not only bare Ethernet systems but also cloud environments that usually use full TCP stacks [8], [79].

4) Flows, Messages, Metrics

We vary flow sizes (and thus message sizes as a flow is equivalent to a message) from 32 KiB to 2 MiB. We use a Poisson distributed flow arrival rate.

We use (1) flow completion time (FCT), which also represents (2) throughput per flow $\text{TPF} = \frac{\text{flow size}}{\text{FCT}}$. We also consider (3) total time to complete a tested workload [4].

5) Simulation Infrastructure and Methodology

We use the OMNeT++ [80], [81] parallel discrete event simulator with the INET model package [82] and the *htsim* packet-level simulator with the NDP reference implementation [4]. OMNeT++ enables detailed simulations of full Ethernet/TCP networking stack, with all overheads coming from protocols such as ARP. We use htsim as its simplified structure enables simulations of networks of much larger scales. We extend both simulators with any required schemes, such as flowlets, ECMP, layered routing, workload randomization. In LetFlow, we use precise timestamps to detect flowlets, with a low gap time of 50μs to reflect the low-latency network. As INET does not model hardware or software latency, we add a 1μs fixed delay to each link. All our code is available online.

6) Gathering Results and Shown Data

We evaluate each combination of topology and routing method. As each such simulation contains thousands of flows with randomized source, destination, size, and start time, we only record per-flow quantities; this suffices for statistical significance. We simulate a fixed number of flows starting in a fixed time window, and drop the results from the first window half for warmup. We summarize the resulting distributions with arithmetic means of the underlying time measurements, or percentiles of distributions.

When some variants or parameters are omitted (e.g., we only show SF-JF to cover Jellyfish), this means that the shown data is representative; the rest is in the full report.

B. Performance Analysis: HPC Systems

First, we analyze FatPaths with Ethernet *but without the TCP transport*. This setting represents HPC systems that use Ethernet for its low cost, but avoid TCP due to its performance issues. We use *htsim* that can deliver such a setting.

[4] When reporting some runtimes (cf. Figures 14-17), we use a relative speedup over the plain ECMP baseline for clarity of presentation (as each plot contains runtimes for flows of different sizes, some absolute runtime data becomes hard to read).

1) Low-Diameter Networks + FatPaths Beat Fat Trees

We analyze Figure 2 (page 2, randomized workload) and Figure 11 (skewed non-randomized workload). In each case, low-diameter topologies outperform similar-cost fat trees, with up to 2× and 4× *improvement* in throughput for non-randomized and randomized workload, respectively. Both fat tree and low-diameter networks use similar load balancing based on flowlet switching and purified transport. Thus, the advantage of low-diameter networks is their *low diameter* combined with the ability of FatPaths to *effectively use the diversity of "almost" minimal paths*. Answering one of two main questions from § I, we conclude that *FatPaths enables low-diameter topologies to outperform state-of-the-art fat trees.*

2) FatPaths Uses "Fat" Non-Minimal Path Diversity Well

We now focus on skewed non-randomized workloads, see Figure 11. Non-minimal balanced routing over FatPaths layers, in each low-diameter topology, *leads to an up to* 30× *FCT improvement* over minimal routing (i.e., "circles on topology X outperform triangles on X"). The exception is HyperX, due to its higher minimal path diversity (cf. Figure 6). Thus, *FatPaths effectively leverages the non-minimal path diversity*.

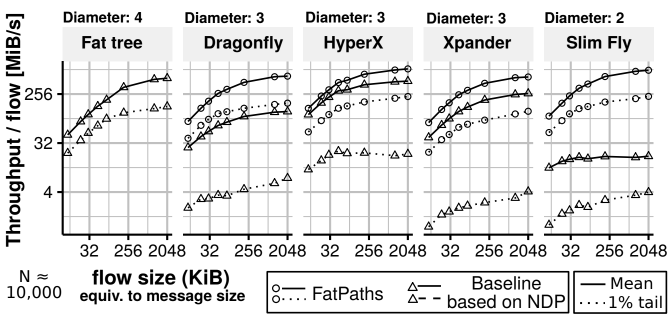

Fig. 11: Performance analysis of a skewed adversarial traffic for similar-cost networks.

3) What Layer Setup Is Best?

We also study the impact of the number n and the sparsity ρ of layers in FatPaths on performance and collision resolution; see Figure 12 (layers are computed with random edge sampling, cf. Listing 1). Nine layers (one complete and eight sparsified) suffice for three disjoint paths per router pair, resolving most collisions for SF and DF (other networks behave similarly). To understand which n resolves collisions on global channels in DF, we use a complete graph. Here, more layers are needed, since higher-multiplicity path collisions appear (cf. the 99% tail). Moreover, when more layers *can* be used, a higher ρ *is better* (cf. FCT for $n = 64$). This reduces the maximum achievable path diversity, but also keeps more links available for alternative routes *within each layer*, increasing chances of choosing disjoint paths. It also increases the count of minimal paths in use across all entries, reducing total network load.

4) FatPaths Scales to Large Networks

We also simulate large-scale SF, DF, and JF for $N = 80,000$ and $N = 1,000,000$ (other topologies lead to excessive memory use in the simulator). Figure 13 shows example results. A slight mean throughput decrease compared to the smaller

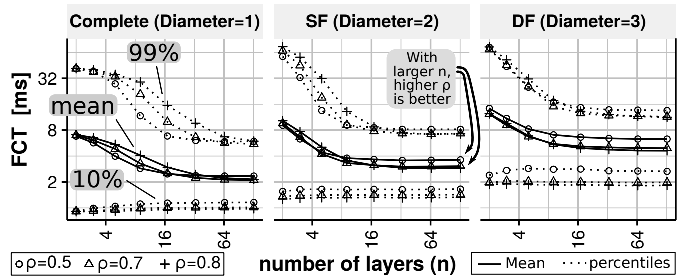

Fig. 12: Effects of the number of layers n and the amount of remaining edges ρ on FatPaths, on long flows (size 1MiB); $N \approx 10,000$ (htsim).

instances is noticeable, but latency and tail FCTs remain tightly bounded. The comparatively bad tail performance of DF is due to path overlap on the global links, where the adaptivity mechanism must handle many overlapping flows. Our analysis also indicates that flows on SF tend to finish later that on SF-JF.

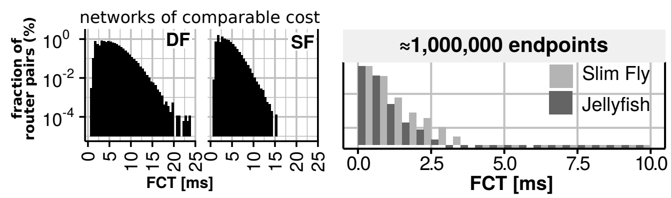

Fig. 13: FatPaths on large networks; FCT histograms for flow size 1MiB (htsim).

C. Performance Analysis: Cloud Systems

We also analyze FatPaths on networks with Ethernet *and full TCP stack*. This represents TCP data centers often used as cloud infrastructure [79]. Here, we use OMNeT++/INET.

We compare FatPaths to ECMP (traditional static load balancing) and LetFlow (recent adaptive load balancing), see Figure 14. The number of layers was limited to $n = 4$ to keep routing tables small; as they are precomputed for all routers and loaded into the simulation in a configuration file (this turned out to be a major performance and memory concern). Most observations follow those from § VII-B, we only summarize TCP-related insights.

LetFlow improves tail and short flow FCTs at the cost of long flow throughput, compared to ECMP. Both are ineffective on SF and DF which have little minimal-path diversity. *Non-minimal routing in FatPaths and* $\rho = 0.6$ *fixes it*, even with only $n = 4$ layers. On other topologies, even with minimal paths ($\rho = 1$), *FatPaths adaptivity outperforms ECMP and LetFlow.* A detailed analysis into the FCT distributions in Figure 15 shows that with minimal routing and low minimal-path diversity, there are many flows with low performance due to path collisions and overlap, although they do not vastly affect the mean throughput. *FatPaths fully resolves this problem*. Short-flow FCTs are dominated by TCP flow control effects, which are not affected much by routing changes.

We also analyze in detail performance effects in flows of different sizes vs. different layer configurations. The findings match those in the "bare Ethernet" simulations in § VII-B. For example, for large flows (1MiB), with $n = 4$, the higher ρ is, the faster flows finish. The largest impact of non-minimal

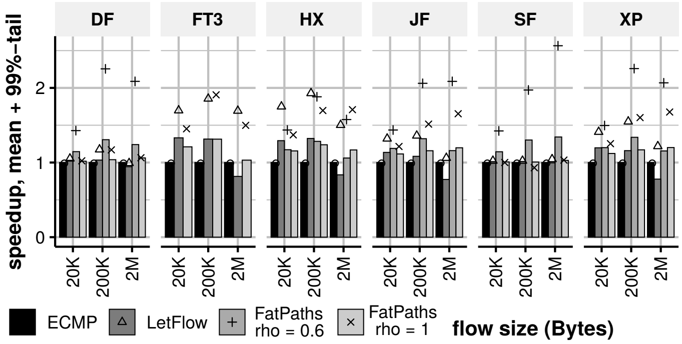

Fig. 14: FatPaths+TCP compared to ECMP and LetFlow (mean and 99% tail values). Some flows on SF finish more than 2.5× faster with FatPaths than ECMP or LetFlow.

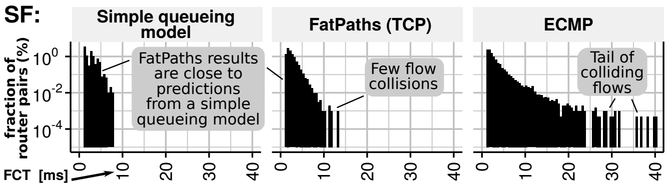

Fig. 15: **FCT distribution of long flows** (1MiB) **on SF**. FatPaths on TCP with non-minimal routing approaches predictions from a simple queuing model (model details omitted due to space constrains); ECMP has a long tail of colliding flows.

routing is for DF and SF, with a 2× improvement in tail FCT; small improvements on tail FCT are seen in all topologies.

We also observe a cost in long flow throughput due to the higher total network load with non-minimal paths. To understand this effect better, Figure 16 shows the impact of the fraction of remaining edges ρ in each layer, and therefore the amount of non-minimal paths, on FCT for long flows. The optimum choice of ρ matches the findings from the Ethernet simulations in § VII-B for SF and DF.

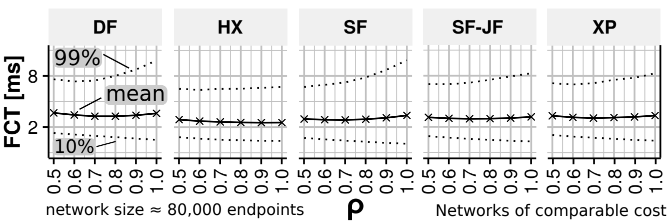

Fig. 16: **Impact of ρ on long-flow (1MiB) FCT with FatPaths on TCP**, $n = 4$. The largest impact of non-minimal routing is for DF and SF, with a 2× improvement in tail FCT; small improvements on tail FCT are seen in all topologies, but there are no throughput improvements on networks with higher minimal-path diversity.

Besides FCT means/tails, we also consider a full completion time of a stencil workload that is representative of an HPC application, in which processes conduct local computation, communicate, and synchronize with a barrier; see Figure 17. Results follow the same performance patterns as others. An interesting outcome is JF: high values for LetFlow are caused by packet loss and do *not* affect the mean/99% tail (cf. Figure 14), only the total completion runtime. Overall, FatPaths ensures high speedups of completion times, e.g., more than 2.5× and nearly 2× faster completion times on SF and XP, for flows of the sizes of 200K and 2M bytes, respectively.

FatPaths also enables influencing communication latency: Specifically, whenever lowest latency is prioritized, one can solely use a layer that provides all shortest paths. This ensures low latencies matching those achieved with shortest-path routing in respective networks [50]. For more throughput, one can use any layer configuration offering diversity of almost-minimal paths. Here, any (marginal) latency overheads from the additional router-router hop are caused by the properties of the underlying topology, *not* the routing protocol.

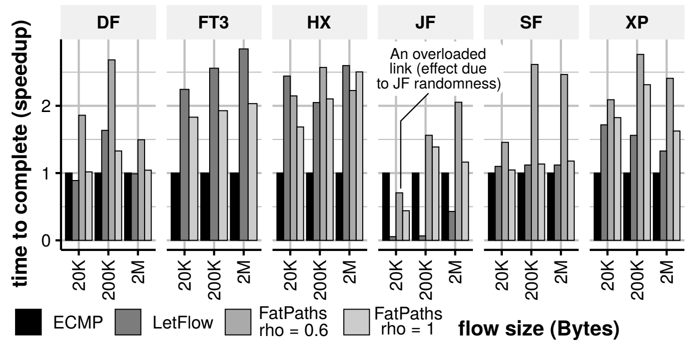

Fig. 17: FatPaths on TCP compared to ECMP and LetFlow (stencil + barrier).

D. Performance Analysis: Routing vs. Topology

How much performance gains in FatPaths come from its routing vs. from simply the benefits of low diameter [50]? Here, we extensively analyzed various design choices in FatPaths; full description is in the extended report. The takeaway is that simple past routing schemes make low-diameter topologies worse (≈2× and more in FCT) than recent fat tree designs. This is because low diameter *must* be enhanced with effective tacking of flow conflicts and other detrimental effects, which is addressed by multipathing in FatPaths.

E. Performance Analysis: Impact from Partial Design Choices

We also analyze speedups from *specific parts of FatPaths*, e.g., only the purified transport, flowlet load balancing, layered routing, or non-minimal paths. While many of these elements can solely accelerate workloads in low-diameter networks, *it is the combination of effective non-minimal multipath routing, load balancing, and transport that gives superior performance*. For example, Figure 11 shows that fat trees with NDP outperform low-diameter networks that do *not* use multipathing based on non-minimal paths (the "NDP" baseline).

F. Final Takeaway on Performance

A high-performance routing architecture for low-diameter networks should expose and use diversity of *almost minimal* paths (because they are numerous, as opposed to minimal paths). *FatPaths enables this, achieving speedups on both HPC systems or cloud infrastructure*.

VIII. Discussion

A. Integration with Other Protocols for Wide Applicability

We also integrate FatPaths with Data Center TCP (DCTCP) [76] and we discuss integration with RDMA [7] (iWARP [83], RoCE [6]), Infiniband [84], MPTCP [85], and we discuss non-minimal ECMP on FatPaths. Details are in the full report.

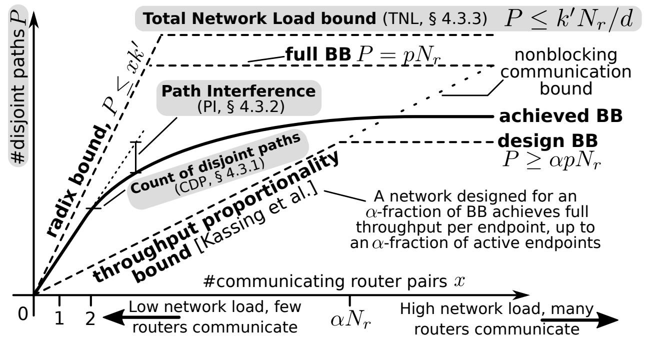

Fig. 18: Relations between connectivity- and BB-related measures. Shade indicates metrics formalized and/or proposed as a part of FatPaths.

B. Integration with Performance Measures and Bounds

For deeper understanding, we *intuitively* connect our path diversity measures to established network performance measures and bounds (e.g., bisection bandwidth (BB) or throughput proportionality [66]). Figure 18 shows how various measures vary when increasing the network load expressed by count of communicating router pairs x. Values of measures are expressed with numbers of disjoint paths P. In this expression, bandwidth measures are counts of disjoint paths between two router sets; these numbers must match corresponding counts in the original measure definitions (e.g., path count associated with BB must equal the BB cut size).

C. FatPaths Limitations

To facilitate applicability of our work in real-world installations, we discuss its limitations. First, FatPaths addresses low-diameter topologies, being less beneficial on high-diameter older interconnects such as torus, because such networks provide multiple (almost or completely disjoint) shortest paths between most router pairs. FatPaths also inherits some NDP's limitations, namely interrupt throttling. As in NDP, this is fixed by dedicating one CPU core to polling for incoming packets.

IX. Related Work

FatPaths touches on various areas. We now briefly discuss related works, excluding the ones covered in past sections.

Our work targets modern **low-diameter topologies** such as Slim Fly [3], [10], [19], [50], [51]. *FatPaths enables these networks to achieve low latency and high throughput with various workloads, outperforming similar-cost fat trees.*

We survey **routing** schemes in Table I and in § VI. *FatPaths is the first to offer generic and adaptive multi-pathing using both shortest and non-shortest disjoint paths.*

Adaptive **load balancing** can be implemented using flows [11], [25], [86]–[92], flowcells (fixed-sized packet series) [93], and packets [4], [21], [24], [26], [85], [94], [94], [95]. We choose an intermediate level, flowlets (variable-size packet series) [5], [12], [96]–[98]. *FatPaths is the first architecture to use load balancing based on flowlets for low-diameter networks*.

We do not compete with **congestion or flow control** schemes; we use them for more performance. *FatPaths can use such schemes in its design* [4], [76], [85], [99]–[111].

Many works on **multi-pathing** exist [25]–[27], [48], [66], [90], [95], [112], [112]–[121]. Our work differs from them all: *it focuses on path diversity in low-diameter topologies and it uses both minimal and non-minimal paths*.

Some works **analyze various properties of low-diameter topologies**, for example path length, throughput, and bandwidth [3], [10], [60], [66], [74], [122]–[132]. *FatPaths offers the most extensive analysis on path diversity so far*.

Some schemes complement FatPaths. For example, XPath [73] and source routing [133] deliver means to **encode different paths**. They could be used *together with FatPaths* by encoding the rich path diversity *exposed by FatPaths*.

Finally, FatPaths could be used to accelerate communication-efficient workloads that benefit from low-diameter properties of Slim Fly and other modern topologies, including deep learning [134]–[139], linear algebra computations [140]–[143], graph processing [122], [144]–[153], and other distributed workloads [7], [154]–[158] and algorithms [159]–[162]. One could possibly use some elements of the FatPaths routing for the associated problems in the on-chip networking [122], [163].

X. Conclusion

We introduce ***FatPaths: a simple, high-performance, and robust routing architecture for a modern family of low-diameter topologies***. FatPaths enables such networks to achieve unprecedented performance by exposing the rich ("fat") diversity of minimal *and non-minimal* paths. We formalize and extensively analyze this path diversity and show that, even though the considered topologies *fall short of shortest paths*, they can accommodate three "almost" minimal disjoint paths, which is enough to avoid congestion in many traffic scenarios. Our path diversity metrics and methodology can be used to analyze other properties of networks.

The key part of FatPaths, layered routing, enables harnessing diversity of both shortest and non-minimal paths. Supported with simple yet effective flowlet load balancing, and high-performance transport in TCP settings, FatPaths achieves low-latency and high-bandwidth, outperforming very recent fat tree architectures [4] by 15% in net throughput at 2$\times$ in latency, for comparable cost. Even though we focus on Ethernet, most of these schemes – for example adaptive flowlet load balancing and layers – are generic and they could enhance technologies such as RDMA (RoCE, iWARP) and Infiniband.

We deliver simulations with up to *one million* endpoints. Our code is online and can be used to foster novel research on next-generation large-scale compute centers.

FatPaths uses Ethernet for maximum versatility. We argue that it can accelerate both HPC clusters or supercomputers as well as data centers and other types of cloud infrastructure. FatPaths will help to bring the areas of HPC networks and cloud computing closer, fostering technology transfer and facilitating exchange of ideas.

Acknowledgements: We thank Hussein Harake, Colin McMurtrie, Mark Klein, Angelo Mangili, and the CSCS team granting access to Ault/Daint, and for their excellent technical support. We thank Timo Schneider for help with computing infrastructure at SPCL, and Nils Blach, Alessandro Maissen, and Adam Latos for useful comments.

REFERENCES

[1] J. J. Dongarra, H. W. Meuer, E. Strohmaier *et al.*, "Top500 supercomputer sites," *Supercomputer*, vol. 13, pp. 89–111, 1997.

[2] W.-c. Feng and K. Cameron, "The green500 list: Encouraging sustainable supercomputing," *Computer*, vol. 40, no. 12, pp. 50–55, 2007.

[3] A. Valadarsky, M. Dinitz, and M. Schapira, "Xpander: Unveiling the secrets of high-performance datacenters," in *ACM HotNets*, 2015.

[4] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wojcik, "Re-architecting datacenter networks and stacks for low latency and high performance," in *ACM SIGCOMM*, 2017.

[5] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, and T. Edsall, "Let it flow: Resilient asymmetric load balancing with flowlet switching," in *NSDI*, 2017, pp. 407–420.

[6] Infiniband Trade Association and others, "Rocev2," 2014.

[7] R. Gerstenberger, M. Besta, and T. Hoefler, "Enabling Highly-scalable Remote Memory Access Programming with MPI-3 One Sided," in *ACM/IEEE Supercomputing*, 2013.

[8] S. Azodolmolky, P. Wieder, and R. Yahyapour, "Cloud computing networking: Challenges and opportunities for innovations." *IEEE Communications Magazine*, vol. 51, no. 7, pp. 54–62, 2013.

[9] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM CCR*, vol. 39, no. 4, pp. 39–50, 2009.

[10] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.

[11] C. Hopps, "RFC 2992: Analysis of an Equal-Cost Multi-Path Algorithm," 2000.

[12] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, pp. 51–62, 2007.

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

[14] R. Perlman, "An algorithm for distributed computation of a spanningtree in an extended lan," in *ACM SIGCOMM CCR*, vol. 15, no. 4. ACM, 1985, pp. 44–53.

[15] J. Moy, "Ospf version 2," Tech. Rep., 1997.

[16] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)," Tech. Rep., 2005.

[17] G. Malkin, "Rip version 2-carrying additional information," Tech. Rep., 1994.

[18] D. Oran, "Osi is-is intra-domain routing protocol," Tech. Rep., 1990.

[19] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *35th International Symposium on Computer Architecture (ISCA)*, 2008, pp. 77–88.

[20] C. Villamizar, "OSPF optimized multipath (OSPF-OMP)," 1999.

[21] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2130–2138.

[22] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 75–86.

[23] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Towards a next generation data center architecture: scalability and commoditization," in *ACM PRESTO*, 2008.

[24] S. Ghorbani, Z. Yang, P. Godfrey, Y. Ganjali, and A. Firoozshahian, "Drill: Micro load balancing for low-latency data center networks," in *ACM SIGCOMM*, 2017.

[25] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," in *CoNEXT*, 2013.

[26] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, and D. Maltz, "Per-packet load-balanced, low-latency routing for clos-based data center networks," in *ACM CoNEXT*, 2013, pp. 49–60.

[27] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," *ACM SIGCOMM computer communication review*, vol. 39, no. 4, pp. 51–62, 2009.

[28] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM*, 2008, pp. 63–74.

[29] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM CCR*, vol. 39, no. 4, pp. 63–74, 2009.

[30] C. Kim, M. Caesar, and J. Rexford, "Floodless in seattle: a scalable ethernet architecture for large enterprises," in *ACM SIGCOMM*, 2008, pp. 3–14.

[31] K.-S. Lui, W. C. Lee, and K. Nahrstedt, "Star: a transparent spanning tree bridge protocol with alternate routing," *ACM SIGCOMM CCR*, vol. 32, no. 3, pp. 33–46, 2002.

[32] T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson, "Smartbridge: A scalable bridge architecture," *ACM SIGCOMM CCR*, vol. 30, no. 4, pp. 205–216, 2000.

[33] R. Perlman, "Rbridges: transparent routing," in *IEEE INFOCOM*, 2004.

[34] R. Garcia *et al.*, "Lsom: A link state protocol over mac addresses for metropolitan backbones using optical ethernet switches," in *IEEE NCA*, 2003.

[35] S. Jain *et al.*, "Viro: A scalable, robust and namespace independent virtual id routing for future networks," in *IEEE INFOCOM*, 2011.

[36] D. Sampath, S. Agarwal, and J. Garcia-Luna-Aceves, "'ethernet on air': Scalable routing in very large ethernet-based networks," in *IEEE ICDCS*, 2010.

[37] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter, "PAST: Scalable Ethernet for data centers," in *ACM CoNEXT*, 2012.

[38] K. Subramanian, "Multi-chassis link aggregation on network devices," Jun. 24 2014, uS Patent 8,761,005.

[39] M. Scott, A. Moore, and J. Crowcroft, "Addressing the scalability of ethernet with moose," in *Proc. DC CAVES Workshop*, 2009.

[40] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng, "Efficient algorithms for multi-path link-state routing," 1999.

[41] I. Gojmerac, T. Ziegler, and P. Reichl, "Adaptive multipath routing based on local distribution of link load information," in *Springer QofIS*, 2003.

[42] A. F. De Sousa, "Improving load balance and resilience of ethernet carrier networks with ieee 802.1 s multiple spanning tree protocol," in *IEEE ICN/ICONS/MCL*, 2006.

[43] A. Iwata, Y. Hidaka, M. Umayabashi, N. Enomoto, and A. Arutaki, "Global open ethernet (goe) system and its performance evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 8, pp. 1432–1442, 2004.

[44] S. Sharma, K. Gopalan, S. Nanda, and T.-c. Chiueh, "Viking: A multi-spanning-tree ethernet architecture for metropolitan area and cluster networks," in *IEEE INFOCOM*, 2004.

[45] D. Allan, P. Ashwood-Smith, N. Bragg, J. Farkas, D. Fedyk, M. Ouellete, M. Seaman, and P. Unbehagen, "Shortest path bridging: Efficient control of larger ethernet networks," *IEEE Communications Magazine*, vol. 48, no. 10, 2010.

[46] J. Touch and R. Perlman, "Transparent interconnection of lots of links (TRILL): Problem and applicability statement," Tech. Rep., 2009.

[47] K. Agarwal, C. Dixon, E. Rozner, and J. B. Carter, "Shadow macs: scalable label-switching for commodity ethernet," in *HotSDN'14*, 2014, pp. 157–162.

[48] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "SPAIN: COTS Data-Center Ethernet for Multipathing over Arbitrary Topologies." in *NSDI*, 2010, pp. 265–280.

[49] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Drill: Micro load balancing for low-latency data center neworks," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*, 2017.

[50] M. Besta and T. Hoefler, "Slim Fly: A Cost Effective Low-Diameter Network Topology," Nov. 2014, aCM/IEEE Supercomputing.

[51] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "HyperX: topology, routing, and packaging of efficient large-scale networks," in *ACM/IEEE Supercomputing*, 2009, p. 41.

[52] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *ACM SIGARCH Comp. Arch. News*, 2007.

[53] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong-Chan, S. Yang, and R. Zak, "The network architecture of the connection machine CM-5," *J. Parallel Distrib. Comput.*, vol. 33, no. 2, pp. 145–158, 1996.

[54] C. Clos, "A study of non-blocking switching networks," *Bell Labs Technical Journal*, vol. 32, no. 2, pp. 406–424, 1953.

[55] B. Prisacari, G. Rodriguez, C. Minkenberg, and T. Hoefler, "Fast pattern-specific routing for fat tree networks," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 10, no. 4, p. 36, 2013.

[56] X. Yuan, S. Mahapatra, M. Lang, and S. Pakin, "Lfti: A new performance metric for assessing interconnect designs for extreme-scale hpc systems," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, 2014, pp. 273–282.

[57] X. Yuan, S. Mahapatra, W. Nienaber, S. Pakin, and M. Lang, "A new routing scheme for Jellyfish and its performance with HPC workloads," in *ACM/IEEE Supercomputing*, 2013, p. 36.

[58] B. Prisacari, G. Rodriguez, P. Heidelberger, D. Chen, C. Minkenberg, and T. Hoefler, "Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks," in *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*. ACM, 2014, pp. 129–140.

[59] B. Prisacari, G. Rodriguez, A. Jokanovic, and C. Minkenberg, "Randomizing task placement and route selection do not randomize traffic (enough)," *Design Automation for Embedded Systems*, vol. 18, no. 3-4, pp. 171–182, 2014.

[60] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. Hoefler, "Cost-effective diameter-two topologies: Analysis and evaluation," in *ACM/IEEE Supercomputing*. ACM, 2015, p. 36.

[61] B. Prisacari, G. Rodriguez, C. Minkenberg, M. Garcia, E. Vallejo, and R. Beivide, "Performance optimization of load imbalanced workloads in large scale dragonfly systems," in *2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 2015, pp. 1–6.

[62] D. Chen, P. Heidelberger, C. Stunkel, Y. Sugawara, C. Minkenberg, B. Prisacari, and G. Rodriguez, "An evaluation of network architectures for next generation supercomputers," in *2016 7th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, 2016, pp. 11–21.

[63] B. Prisacari, G. Rodriguez, C. Minkenberg, and T. Hoefler, "Bandwidth-optimal all-to-all exchanges in fat tree networks," in *Proceedings of the 27th international ACM conference on International conference on supercomputing*. ACM, 2013, pp. 139–148.

[64] B. Karacali, J. M. Tracey, P. G. Crumley, and C. Basso, "Assessing cloud network performance," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[65] W. Sehery and C. Clancy, "Flow optimization in data centers with clos networks in support of cloud applications," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 847–859, 2017.

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

[67] X. Yuan, S. Mahapatra, W. Nienaber, S. Pakin, and M. Lang, "A New Routing Scheme for Jellyfish and Its Performance with HPC Workloads," in *Proceedings of 2013 ACM/IEEE Supercomputing*, ser. SC '13, 2013, pp. 36:1–36:11.

[68] T. Skeie, O. Lysne, and I. Theiss, "Layered shortest path (lash) routing in irregular system area networks," in *ipdps*. Citeseer, 2002, p. 0162.

[69] S. Sinha, S. Kandula, and D. Katabi, "Harnessing tcp'sburstiness with flowlet switching," *San Diego, November*, vol. 83, 2004.

[70] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[71] P. J. Frantz and G. O. Thompson, "Vlan frame format," Sep. 28 1999, uS Patent 5,959,990.

[72] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[73] S. Hu, K. Chen, H. Wu, W. Bai, C. Lan, H. Wang, H. Zhao, and C. Guo, "Explicit path control in commodity data centers: Design and applications," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2768–2781, 2016.

[74] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla, "Measuring and understanding throughput of network topologies," in *ACM/IEEE Supercomputing*, 2016.

[75] M. Valerio, L. E. Moser, and P. Melliar-Smith, "Recursively scalable fat-trees as interconnection networks," in *Phoenix Conference on Computers and Communications*, vol. 13. Citeseer, 1994, pp. 40–40.

[76] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," *ACM SIGCOMM computer communication review*, vol. 41, no. 4, pp. 63–74, 2011.

[77] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ecn) to ip," Tech. Rep., 2001.

[78] S. Floyd, "Tcp and explicit congestion notification," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 8–23, 1994.

[79] T. Isobe, N. Tanida, Y. Oishi, and K. Yoshida, "Tcp acceleration technology for cloud computing: Algorithm, performance evaluation in real network," in *2014 International Conference on Advanced Technologies for Communications (ATC 2014)*. IEEE, 2014, pp. 714–719.

[80] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European simulation multiconference (ESM'2001)*, vol. 9, no. S 185. sn, 2001, p. 65.

[81] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008, p. 60.

[82] ——, "INET Framework for OMNeT++," Tech. Rep., 2012.

[83] R. Grant, M. Rashti, A. Afsahi, and P. Balaji, "RDMA Capable iWARP over Datagrams," in *Par. Dist. Proc. Symp. (IPDPS), 2011 IEEE Intl.*, 2011, pp. 628–639.

[84] G. F. Pfister, "An introduction to the infiniband architecture," *High Performance Mass Storage and Parallel I/O*, vol. 42, pp. 617–632, 2001.

[85] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2011, pp. 266–277.

[86] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1629–1637.

[87] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, and R. Fonseca, "Planck: Millisecond-scale monitoring and control for commodity networks," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 407–418.

[88] F. P. Tso, G. Hamilton, R. Weber, C. Perkins, and D. P. Pezaros, "Longer is better: Exploiting path diversity in data center networks," in *IEEE 33rd International Conference on Distributed Computing Systems, ICDCS*, 2013, pp. 430–439.

[89] T. Benson, A. Anand, A. Akella, and M. Zhang, "Microte: Fine grained traffic engineering for data centers," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM, 2011, p. 8.

[90] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "WCMP: weighted cost multipathing for improved fairness in data centers," in *ACM EuroSys*, 2014.

[91] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks." in *NSDI*, vol. 10, 2010, pp. 19–19.

[92] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "FlowBender: Flow-level Adaptive Routing for Improved Latency and Throughput in Datacenter Networks," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. ACM, 2014, pp. 149–160.

[93] K. He, E. Rozner, K. Agarwal, W. Felter, J. B. Carter, and A. Akella, "Presto: Edge-based load balancing for fast datacenter networks," in *ACM SIGCOMM*, 2015.

[94] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. H. Katz, "Detail: reducing the flow completion time tail in datacenter networks," in *ACM SIGCOMM*, 2012, pp. 139–150.

[95] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized zero-queue datacenter network," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 307–318, 2015.

[96] N. P. Katta, M. Hira, A. Ghag, C. Kim, I. Keslassy, and J. Rexford, "CLOVE: how I learned to stop worrying about the core and love the edge," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks, HotNets*, 2016, pp. 155–161.

[97] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, G. Varghese *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 2014, pp. 503–514.

[98] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "Hula: Scalable load balancing using programmable data planes," in *Proceedings of the Symposium on SDN Research*. ACM, 2016, p. 10.

[99] R. Mittal, V. T. Lam, N. Dukkipati, E. R. Blem, H. M. G. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY: rtt-based congestion control for the datacenter," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM*, 2015, pp. 537–550.

[100] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: congestion-based congestion control," *ACM Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[101] K. He, E. Rozner, K. Agarwal, Y. J. Gu, W. Felter, J. B. Carter, and A. Akella, "AC/DC TCP: virtual congestion control enforcement for datacenter networks," in *Proceedings of the 2016 conference on ACM SIGCOMM*, 2016, pp. 244–257.

[102] D. Zhuo, Q. Zhang, V. Liu, A. Krishnamurthy, and T. Anderson, "Rack-level congestion control," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. ACM, 2016, pp. 148–154.

[103] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and W. Sun, "PIAS: practical information-agnostic flow scheduling for data center networks," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks, HotNets-XIII*, 2014, pp. 25:1–25:7.

[104] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 435–446, 2013.

[105] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 115–126, 2012.

[106] Y. Lu, "Sed: An sdn-based explicit-deadline-aware tcp for cloud data center networks," *Tsinghua Science and Technology*, vol. 21, no. 5, pp. 491–499, 2016.

[107] J. Hwang, J. Yoo, and N. Choi, "Deadline and incast aware tcp for cloud data center networks," *Computer Networks*, vol. 68, pp. 20–34, 2014.

[108] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," *arXiv preprint arXiv:1803.09615*, 2018.

[109] J. Jiang, R. Jain, and C. So-In, "An explicit rate control framework for lossless ethernet operation," in *Communications, 2008. ICC'08. IEEE International Conference on*. IEEE, 2008, pp. 5914–5918.

[110] B. G. Banavalikar, C. M. DeCusatis, M. Gusat, K. G. Kamble, and R. J. Recio, "Credit-based flow control in lossless Ethernet networks," Jan. 12 2016, uS Patent 9,237,111.

[111] M. Alasmar, G. Parisis, and J. Crowcroft, "Polyraptor: embracing path and data redundancy in data centres for efficient data transport," in *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*. ACM, 2018, pp. 69–71.

[112] C. H. Benet, A. J. Kassler, T. Benson, and G. Pongracz, "Mp-hula: Multipath transport aware load balancing using programmable data planes," in *Proceedings of the 2018 Morning Workshop on In-Network Computing*. ACM, 2018, pp. 7–13.

[113] M. Caesar, M. Casado, T. Koponen, J. Rexford, and S. Shenker, "Dynamic route recomputation considered harmful," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 66–71, 2010.

[114] S. Aggarwal and P. Mittal, "Performance evaluation of single path and multipath regarding bandwidth and delay," *Intl. J. Comp. App.*, vol. 145, no. 9, 2016.

[115] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

[116] X. Huang and Y. Fang, "Performance study of node-disjoint multipath routing in vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, 2009.

[117] S. Sohn, B. L. Mark, and J. T. Brassil, "Congestion-triggered multipath routing based on shortest path information," in *IEEE ICCCN*, 2006.

[118] Y. Li and D. Pan, "Openflow based load balancing for fat-tree networks with multipath support," in *Proc. 12th IEEE International Conference on Communications (ICC'13), Budapest, Hungary*, 2013, pp. 1–5.

[119] M. Bredel, Z. Bozakov, A. Barczyk, and H. Newman, "Flow-based load balancing in multipathed layer-2 networks using openflow and multipath-tcp," in *Hot topics in software defined networking*. ACM, 2014, pp. 213–214.

[120] S. Van der Linden, G. Detal, and O. Bonaventure, "Revisiting next-hop selection in multipath networks," in *ACM SIGCOMM CCR*, vol. 41, no. 4, 2011.

[121] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," in *ACM SIGMETRICS*, 2011.

[122] M. Besta, S. M. Hassan, S. Yalamanchili, R. Ausavarungnirun, O. Mutlu, and T. Hoefler, "Slim noc: A low-diameter on-chip network topology for high energy efficiency and scalability," in *ACM SIGPLAN Notices*, 2018.

[123] S. Li, P.-C. Huang, and B. Jacob, "Exascale interconnect topology characterization and parameter exploration," in *HPCC/SmartCity/DSS*. IEEE, 2018, pp. 810–819.

[124] R. Kawano, R. Yasudo, H. Matsutani, and H. Amano, "k-optimized path routing for high-throughput data center networks," in *2018 Sixth International Symposium on Computing and Networking (CANDAR)*. IEEE, 2018, pp. 99–105.

[125] V. Harsh, S. A. Jyothi, I. Singh, and P. Godfrey, "Expander datacenters: From theory to practice," *arXiv preprint arXiv:1811.00212*, 2018.

[126] R. Kawano, H. Nakahara, I. Fujiwara, H. Matsutani, M. Koibuchi, and H. Amano, "Loren: A scalable routing method for layout-conscious random topologies," in *2016 Fourth International Symposium on Computing and Networking (CANDAR)*. IEEE, 2016, pp. 9–18.

[127] T.-N. Truong, K.-V. Nguyen, I. Fujiwara, and M. Koibuchi, "Layout-conscious expandable topology for low-degree interconnection networks," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 5, pp. 1275–1284, 2016.

[128] M. Flajslik, E. Borch, and M. A. Parker, "Megafly: A topology for exascale systems," in *International Conference on High Performance Computing*, 2018.

[129] R. Kawano, H. Nakahara, I. Fujiwara, H. Matsutani, M. Koibuchi, and H. Amano, "A layout-oriented routing method for low-latency hpc networks," *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 12, pp. 2796–2807, 2017.

[130] S. Azizi, N. Hashemi, and A. Khonsari, "Hhs: an efficient network topology for large-scale data centers," *The Journal of Supercomputing*, vol. 72, no. 3, pp. 874–899, 2016.

[131] N. T. Truong, I. Fujiwara, M. Koibuchi, and K.-V. Nguyen, "Distributed shortcut networks: Low-latency low-degree non-random topologies targeting the diameter and cable length trade-off," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 989–1001, 2016.

[132] F. Al Faisal, M. H. Rahman, and Y. Inoguchi, "A new power efficient high performance interconnection network for many-core processors," *Journal of Parallel and Distributed Computing*, vol. 101, pp. 92–102, 2017.

[133] S. A. Jyothi, M. Dong, and P. Godfrey, "Towards a flexible data center fabric with source routing," in *ACM SOSR*, 2015.

[134] T. Ben-Nun, M. Besta, S. Huber, A. N. Ziogas, D. Peter, and T. Hoefler, "A modular benchmarking infrastructure for high-performance and reproducible deep learning," *arXiv preprint arXiv:1901.10183*, 2019.

[135] P. Grönquist, T. Ben-Nun, N. Dryden, P. Dueben, L. Lavarini, S. Li, and T. Hoefler, "Predicting weather uncertainty with deep convnets," *arXiv preprint arXiv:1911.00630*, 2019.

[136] S. Li, T. Ben-Nun, S. D. Girolamo, D. Alistarh, and T. Hoefler, "Taming unbalanced training workloads in deep learning with partial collective operations," in *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2020, pp. 45–61.

[137] Y. Oyama, T. Ben-Nun, T. Hoefler, and S. Matsuoka, "Accelerating deep learning frameworks with micro-batches," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 402–412.

[138] T. Ben-Nun, A. S. Jakobovits, and T. Hoefler, "Neural code comprehension: A learnable representation of code semantics," in *Advances in Neural Information Processing Systems*, 2018, pp. 3585–3597.

[139] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–43, 2019.

[140] M. Besta, R. Kanakagiri, H. Mustafa, M. Karasikov, G. Rätsch, T. Hoefler, and E. Solomonik, "Communication-efficient jaccard similarity

for high-performance distributed genome comparisons," *IEEE IPDPS*, 2020.
[141] M. Besta, F. Marending, E. Solomonik, and T. Hoefler, "Slimsell: A vectorizable graph representation for breadth-first search," in *IEEE IPDPS*, 2017, pp. 32–41.
[142] E. Solomonik, M. Besta, F. Vella, and T. Hoefler, "Scaling betweenness centrality using communication-efficient sparse matrix multiplication," in *ACM/IEEE Supercomputing*, 2017, p. 47.
[143] G. Kwasniewski, M. Kabić, M. Besta, J. VandeVondele, R. Solcà, and T. Hoefler, "Red-blue pebbling revisited: near optimal parallel matrix-matrix multiplication," in *ACM/IEEE Supercomputing*. ACM, 2019, p. 24.
[144] M. Besta and T. Hoefler, "Accelerating irregular computations with hardware transactional memory and active messages," in *ACM HPDC*, 2015.
[145] M. Besta, M. Fischer, T. Ben-Nun, J. De Fine Licht, and T. Hoefler, "Substream-centric maximum matchings on fpga," in *ACM/SIGDA FPGA*, 2019, pp. 152–161.
[146] M. Besta *et al.*, "Slim graph: Practical lossy graph compression for approximate graph processing, storage, and analytics," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019.
[147] M. Besta, D. Stanojevic, J. D. F. Licht, T. Ben-Nun, and T. Hoefler, "Graph processing on fpgas: Taxonomy, survey, challenges," *arXiv preprint arXiv:1903.06697*, 2019.
[148] M. Besta, M. Podstawski, L. Groner, E. Solomonik, and T. Hoefler, "To push or to pull: On reducing communication and synchronization in graph computations," in *ACM HPDC*, 2017.
[149] M. Besta, D. Stanojevic, T. Zivic, J. Singh, M. Hoerold, and T. Hoefler, "Log (graph) a near-optimal high-performance graph representation," in *ACM PACT*, 2018, pp. 1–13.
[150] M. Besta and T. Hoefler, "Survey and taxonomy of lossless graph compression and space-efficient graph representations," *arXiv preprint arXiv:1806.01799*, 2018.
[151] L. Gianinazzi, P. Kalvoda, A. De Palma, M. Besta, and T. Hoefler, "Communication-avoiding parallel minimum cuts and connected components," in *ACM SIGPLAN Notices*, vol. 53, no. 1. ACM, 2018, pp. 219–232.
[152] M. Besta, M. Fischer, V. Kalavri, M. Kapralov, and T. Hoefler, "Practice of streaming and dynamic graphs: Concepts, models, systems, and parallelism," *arXiv preprint arXiv:1912.12740*, 2019.
[153] M. Besta, E. Peter, R. Gerstenberger, M. Fischer, M. Podstawski, C. Barthels, G. Alonso, and T. Hoefler, "Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries," *arXiv preprint arXiv:1910.09017*, 2019.
[154] M. Besta and T. Hoefler, "Active access: A mechanism for high-performance distributed data-centric computations," in *ACM ICS*, 2015.
[155] ——, "Fault tolerance for remote memory access programming models," in *ACM HPDC*, 2014, pp. 37–48.
[156] A. N. Ziogas, T. Ben-Nun, G. I. Fernández, T. Schneider, M. Luisier, and T. Hoefler, "A data-centric approach to extreme-scale ab initio dissipative quantum transport simulations," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–13.
[157] T. Ben-Nun, J. de Fine Licht, A. N. Ziogas, T. Schneider, and T. Hoefler, "Stateful dataflow multigraphs: A data-centric model for performance portability on heterogeneous architectures," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–14.
[158] S. Di Girolamo, K. Taranov, A. Kurth, M. Schaffner, T. Schneider, J. Beránek, M. Besta, L. Benini, D. Roweth, and T. Hoefler, "Network-accelerated non-contiguous memory transfers," *ACM/IEEE Supercomputing*, 2019.
[159] H. Schweizer, M. Besta, and T. Hoefler, "Evaluating the cost of atomic operations on modern architectures," in *IEEE PACT*, 2015, pp. 445–456.
[160] P. Schmid, M. Besta, and T. Hoefler, "High-performance distributed RMA locks," in *ACM HPDC*, 2016, pp. 19–30.
[161] M. Sutton, T. Ben-Nun, and A. Barak, "Optimizing parallel graph connectivity computation via subgraph sampling," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2018, pp. 12–21.
[162] A. Tate *et al.*, "Programming abstractions for data locality." PADAL Workshop 2014, 2014.
[163] J. de Fine Licht *et al.*, "Transformations of high-level synthesis codes for high-performance computing," *arXiv:1805.08288*, 2018.