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

Maciej Besta¹, Marcel Schneider¹, Marek Konieczny², Karolina Cynk², Erik Henrikkson¹, Salvatore Di Girolamo¹, Ankit Singla¹, Torsten Hoefler¹
¹Department of Computer Science, ETH Zurich
²Faculty 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.

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...
HyperX

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 ≈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× lower latency for comparable performance.

II. NOTATION, BACKGROUND, CONCEPTS

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

A. Network Model

We model an interconnection network as an undirected graph \( G = (V, E) \); \( V \) and \( E \) are sets of routers 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.
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 (FFB) [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 \( p = \frac{k'}{2} \) (in the technical report, we show that, assuming random uniform traffic, \( p = \frac{k'}{2} \) 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 \( \{1, \ldots, N\} \) 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 \( t(s) \in V_e \) (randomly and randomly selected u.a.r.) and random permutation \( t(s) = \pi_N(s) \), where \( \pi_N \) 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 \( t(s) = \text{rot}_i(s) \mod N \), where the bitwise left rotation on \( i \) bits is denoted as \( \text{rot}_i \), 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 sten-

cils, realistic traffic patterns common in HPC. We model 2D stencils as four off-diagonals at fixed offsets \( c \in \{\pm1, \pm1, \pm42, \pm42\} \). For large simulations (\( N > 10,000 \)) we also use offsets \( c \in \{\pm1, \pm1, \pm1337, \pm1337\} \) 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
To encode minimal and non-minimal paths with commodity HW, FatPaths divides all links into (not necessarily disjoint) subsets called layers. 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
For simple but robust load balancing, we use flowlet switching [12]. 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. 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.

D. Randomized Workload Mapping
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 share a link), 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× 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.

3In 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].
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 test 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 $A$’;

$$h(A) = \{ t \in V : \exists s \in A \ { s, t } \in E \} \ (“routers \ attached \ to \ A”)$$

$$h^l(A) = h(h( \ldots h(A) \ldots )) \ (“l$-step neighborhood of $A”).$$

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:

$$l_{\min}(s,t) = \arg \min_{i \in N} \{ t \in h^l(\{s\}) \} \ (“minimal \ path \ lengths”)$$

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

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 $I_{a,b,c,d}^l$ and define it as

$$I_{a,b,c,d}^l = c_l(\{a,c\}, \{b\}) + c_l(\{a,c\}, \{d\}) - c_l(\{a,c\}, \{b,d\})$$

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 $k^d N_c$ 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 $k^d N_c / d$, because $\# \text{flows} \leq k^d N_c / d$. 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).
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 path 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 \( c_d(A, B) \). 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 \( c_d(A, B) \) are related to pairs connected with an edge while higher \( c_d(A, B) \) 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

### Table III: Counts of disjoint non-minimal paths CDP

<table>
<thead>
<tr>
<th>Topology</th>
<th>( d' )</th>
<th>( N )</th>
<th>( k' )</th>
<th>( N_r )</th>
<th>( N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF-JF</td>
<td>2</td>
<td>10000</td>
<td>89%</td>
<td>10%</td>
<td>39%</td>
</tr>
<tr>
<td>XP-JF</td>
<td>3</td>
<td>10000</td>
<td>50%</td>
<td>86%</td>
<td>26%</td>
</tr>
<tr>
<td>HX-JF</td>
<td>4</td>
<td>10000</td>
<td>26%</td>
<td>80%</td>
<td>23%</td>
</tr>
<tr>
<td>DF-JF</td>
<td>4</td>
<td>10000</td>
<td>56%</td>
<td>38%</td>
<td>22%</td>
</tr>
<tr>
<td>FT3-JF</td>
<td>4</td>
<td>10000</td>
<td>96%</td>
<td>90%</td>
<td>14%</td>
</tr>
</tbody>
</table>

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 \( c_d(A, B) \). 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 \( c_d(A, B) \) are related to pairs connected with an edge while higher \( c_d(A, B) \) 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.

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.

**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 some destination-based routing, compatible with any relevant technology, including source-based systems like NDP. To compute the output port \( j \in \{1, \ldots, k'\} \) 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 \( \sigma \) with fixed \( t \) we eventually reach \( s' = t \) and finish. The function \( \sigma \) must ensure no loops on any path.

**B. Layered Routing in FatPaths**

We use \( n \) routing functions \( \sigma_1, \ldots, \sigma_n \) for \( n \) layers. Each router uses \( \sigma_i \) 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 \( \sigma_1 \)) 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 \( \rho \) and \( n \) is important. More layers (higher \( n \)) that are sparse (lower \( \rho \)) 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 \( \rho \) and \( n \) for a given network. To facilitate implementation of FatPaths, the project repository contains layer configurations (\( \rho, n \)) that ensure high-performance routing for used topologies. We analyze performance of different \( \rho \) 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 \( E' \subseteq E \) with \( \rho \cdot |E'| \) edges sampled u.a.r.. \( E' \) may disconnect the network, but for the used values of \( \rho \), 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.

---

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 \( \sigma_i \) functions are deployed using forwarding tables with minimum paths between every two routers \( s, t \) within layer \( \sigma_i \). For each router \( s \), we populate the entry for \( s, t \) in \( \sigma_i \) 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 \( N_r \) links. Thus, at least \( O(k') \) layers are required to cover all minimal paths, and SPAIN requires even \( O(N_r) \) on many topologies. Moreover, PAST always needs \( O(N_r) \) 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 \( T \) for which there exists a feasible multicommodity flow (MCF) that routes a flow \( T(s, t) \cdot T \) 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 \( T \). 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, HF, 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” (\( N \in \{ \approx 1k, \approx 10k, \approx 100k, \approx 1M \} \),...
We use (1) flow completion time (FCT), which also represents (2) throughput per flow $TPF = \frac{flow\ size}{FCT}$. 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 $2 \times$ and $4 \times$ 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 $30 \times$ 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.

3) What Layer Setup Is Best?
We also study the impact of the number $n$ and the sparsity $\rho$ 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 $\rho$ 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 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.

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 $\rho$ is, the faster flows finish. 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.

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 $\rho$ in each layer, and therefore the amount of non-minimal paths, on FCT for long flows. The optimum choice of $\rho$ matches the findings from the Ethernet simulations in § VII-B for SF and DF.

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.

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 ($\approx 2×$ 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.
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× 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 McMurtie, Mark Klein, Angelo Mangili, and the CSCS team granting access to Anri/Daint, and for their excellent technical support. We thank Timo Schneider for help with computing infrastructure at SPCL, and Niils Blach, Alessandro Maissen, and Adam Latos for useful comments.


