FatPaths: Routing in Supercomputers, Data Centers, and Clouds with Low-Diameter Networks when Shortest Paths Fall Short

Maciej Besta[1], Marcel Schneider[1], Karolina Cynk[2], Marek Konieczny[2], Erik Henriksson[1], Salvatore Di Girolamo[1], Ankit Singla[1], Torsten Hoefler[1]

[1]Department of Computer Science; ETH Zurich
[2]Department of Computer Science, Electronics and Telecommunications; AGH-UST

Abstract

We introduce FatPaths: a simple, generic, and robust routing architecture for Ethernet stacks. FatPaths enables state-of-the-art low-diameter topologies such as Slim Fly to achieve unprecedented performance, targeting both HPC supercomputers as well as data centers and clusters used by cloud computing. FatPaths exposes and exploits the rich ("fat") diversity of both minimal and non-minimal paths for high-performance multi-pathing. Moreover, FatPaths features a redesigned "purified" transport layer, based on recent advances in data center networking, that removes virtually all TCP performance issues (e.g., the slow start). FatPaths also uses 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.

CCS Concepts

• **Computer systems organization → Cloud computing**; **Interconnection architectures**; • **Networks → Network architectures**; **Network protocols**; *Link-layer protocols*; *Routing protocols*; **Network performance evaluation**; **Network structure**; **Topology analysis and generation**; **Physical topologies**; Transport protocols; *Network topology types*; **Data center networks**;

Keywords

Network Architectures, High-Performance Networks, Data Center Networks, Cloud Computing Infrastructure, Routing, Multipath Routing, Layered Routing, Non-Minimal Routing, Adaptive Routing, Topology Structure, Low-Diameter Topologies, Path Diversity, Load Balancing, ECMP, Ethernet, TCP/IP

FatPaths website:
https://spcl.inf.ethz.ch/Research/Scalable_Networking/FatPaths

1 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 (the November 2018 issue) machines [40] are based on Ethernet, see Figure 1 (the left plot). We observe similar numbers for the Green500 list. 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 [60, 142, 147]. Another example is Mellanox, with its Ethernet sales for the 3rd quarter of 2017 being higher than those for Infiniband [111].

Yet, Ethernet systems are scarce in the highest 100 positions of Top500. For example, in November 2018, only *four* 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. As Ethernet is prevalent in cloud systems [17, 156], it would similarly accelerate cloud infrastructure.

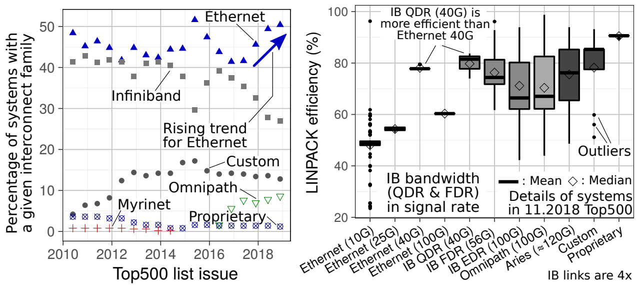

Figure 1: The percentage of Ethernet systems in the Top500 list (on the left) and the LINPACK efficiency of Top500 systems (on the right).

Clos is the most commonly deployed topology in data centers and supercomputers today, and it dominates the landscape of Ethernet clusters [60, 105, 142]. Yet, many low-diameter topologies have recently been proposed which claim to improve the cost-performance tradeoff compared to Clos networks. For instance, Slim Fly is ≈2× more cost- and power-efficient than fat trees and Clos [7] while offering ≈25% lower latency. Similar numbers have been reported for Jellyfish [131] and Xpander [142]. *These topologies could significantly enhance the compute capabilities of Ethernet clusters.*

However, the above comparisons (low-diameter topologies vs. Clos) assume hard-to-deploy routing, for example in the case of Jellyfish [131]. Moreover, *the bar for comparison with Clos interconnects has been raised substantially.* Clos was traditionally deployed using ECMP, which tries to approximate an equal split of a fluid flow across shortest paths. Bleeding-edge Clos proposals based on per-packet load balancing[1] and novel transport mechanisms *achieve* 3-4× *smaller tail flow completion time (FCT) than ECMP* [51, 60].

[1]These schemes account for packet-reordering.

The above two research threads raise two questions we have not seen addressed so far. **First, what is a high-performance routing architecture for low-diameter networks, assuming an Ethernet stack?** The key issue here is that traditional routing schemes such as ECMP cannot be directly used in networks such as Slim Fly, because (as we will show) **shortest paths fall short** in these topologies: there is almost always *only one* shortest path between endpoint pairs. Restricting traffic to these paths does not utilize such topologies' path diversity, and it remains unclear how to split traffic across non-shortest paths of *unequal* lengths. **Second, can low-diameter networks continue to claim an improvement in the cost-performance tradeoff against the new, superior Clos baselines?** The key issue here is that the recent progress on Clos and fat trees also does *not* directly translate to topologies like Slim Fly, because the optimality of splitting traffic equally for Clos does *not* extend to recent low-diameter topologies.

In this work, we answer both questions affirmatively. We first analyze in detail path diversity in five low-diameter topologies and we discover that, even though **low-diameter topologies fall short of shortest paths, they have enough path diversity when using "almost" shortest paths**. We then present ***FatPaths*, a high-performance, simple, and robust *routing architecture* for Ethernet low-diameter networks**, aiming to accelerate both HPC systems and cloud infrastructure. FatPaths encodes the rich diversity of non-minimal paths in low-diameter networks in commodity hardware using *layered routing*. It also uses a redesigned ("purified") transport layer (based on recent data center designs for fat trees [60]) with lossless metadata exchange (packet headers always reach their destinations), almost no dropped packet payload, fast start (senders start transmitting at line rate), shallow buffers, and priority queues for retransmitted packets to avoid head-of-line congestion [60], ultimately ensuring low latency and high bandwidth. Finally, FatPaths uses flowlet switching [75], a scheme proposed for Clos to prevent packet reordering, to enable very simple but powerful load balancing in non-Clos low-diameter networks.

We extensively compare FatPaths to other routing schemes in Table 1. 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 for highest performance, (5) offers adaptive load balancing, and (6) is generic, being applicable across topologies. Table 1 focuses on various aspects of path diversity, because as topologies lower their diameter and reduce link count, path diversity, which is key to high performance of routing, *becomes a scarce resource demanding careful examination and use.*

Even if FatPaths primarily targets Ethernet networks, most its schemes are generic. We briefly discuss the feasibility of **implementing Remote Direct Memory Access (RDMA)** [49] technologies such as RDMA over Converged Ethernet (RoCE) [15] and Internet Wide Area RDMA Protocol (iWARP) [53] on top of FatPaths. For wide applicability in data centers and cloud systems, we integrate FatPaths with TCP protocols such as Data Center TCP (DCTCP) [11] and MPTCP [117]. We also summarize advantages of FatPaths over flow control schemes such as Priority Flow Control (PFC) [1, 2]. Finally, we discuss how FatPaths **could enhance Infiniband**, possibly starting a line of future work on more powerful lossless routing on low-diameter topologies.

Routing Scheme (Name, Abbreviation, Reference)	**Stack Layer**	Features of routing schemes						
		SP	**NP**	**SM**	**MP**	**DP**	**ALB**	**AT**
(1) SIMPLE ROUTING PROTOCOLS (often used as building blocks):								
Valiant load balancing (VLB) [144]	L2–L3	✗	✓	✗	✗	✗	✗	✓
Simple Spanning Tree (ST) [108]	L2	✗^{S}	✗^{S}	✗	✗	✗	✗	✓
Simple routing, e.g., OSPF [97, 102, 106, 122]	L2, L3	✓	✗	✗	✗	✗	✗	✓
ECMP [64], OSPF-OMP [151]	L3	✓	✗	✗	✓	✗	✗	✓
UGAL [86]	L2–L3	✓	✓	✗	✗	✗	✓	✓
Simple Packet Spraying (PR) [39, 129]	L2–L3	✓	✗	✗	✓	✗	✗	✓
(2) ROUTING ARCHITECTURES:								
DCell [58]	L2–L3	✗	✓	✗	✗	✗	✗	✗
Monsoon [55]	L2, L3	◐	◐	✗	◐	✗	✗	✗
PortLand [105]	L2	✓	✗	✗	✓	✗	✗	✗
DRILL [50]	L2	✓	✗	✗	✓	✗	✓	✗
LocalFlow [129], DRB [29]	L2	✓	✗	✗	✓	✗	✓	✗
VL2 [54]	L3	✓	✗	✗	✓	✗	◐	✗
Architecture by Al-Fares et al. [7]	L2–L3	✓	✗	✗	✓	✓	✓	✗
BCube [56]	L2–L3	✓	✗	✗	✓	✓	✗	✗
SEATTLE [84], others* [48, 95, 109, 124]	L2	✓	✗	✗	✗	✗	✗	✓
VIRO [70]	L2–L3	✗^{S}	✗^{S}	✗	✗	✗	✗	✓
Ethernet on Air [125]	L2	✗^{S}	✗^{S}	✗	◐^{R}	✗	✗	✓
PAST [134]	L2	◐^{S}	◐^{S}	✗	✗	✓	✗	✓
MLAG, MC-LAG, others [135]	L2	◐	◐	✗	◐^{R}	✗	✗	✓
MOOSE [126]	L2	✓	✗	✗	◐	✗	✗	✓
MPA [104]	L3	✓	✓	✗	✓	✗	✗	✓
AMP [52]	L3	✓	✗	✗	✓	✗	✓	✓
MSTP [37], GOE [69], Viking [130]	L2	✗^{S}	✗^{S}	✗	✓	✗	✗	✓
SPB [13], TRILL [138], Shadow MACs [3]	L2	✓	✗^{R}	✗	✓	✗	✗	✓
SPAIN [103]	L2	◐^{S}	◐^{S}	◐^{S}	✓	✓	✗	✓
(3) Schemes for exposing/encoding paths (can be combined with FatPaths):								
XPath [65]	L3	✓	◐	◐	✓	✓	◐	✓
Source routing for flexible DC fabric [72]	L3	✓	◐^{R}	◐^{R}	✗	✗	✗	◐†
(3) FatPaths [This work]	L2–L3	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of the considered routing and transport schemes, focusing on how well they utilize path diversity. "Stack Layer" indicates the location of each routing scheme in the TCP/IP stack. **SP**, **NP**, **SM**, **MP**, **DP**, **ALB**, and **AT** are various aspects of path diversity. Specifically: **SP**: A given scheme enables using arbitrary **shortest** paths. **NP**: A given scheme enables using arbitrary **non-minimal** paths. **SM**: A given scheme **simultaneously** enables minimal and non-minimal paths. **MP**: A given scheme enables **multi-pathing** (between two hosts). **DP**: A given scheme considers **disjoint** paths. **ALB**: A given scheme offers **adaptive load balancing**. **AT**: A given scheme works with an **arbitrary topology**. ✓: A given scheme does offer a given feature. ◐: A given scheme offers a given feature in a limited way (e.g., Monsoon [55] uses ECMP for multi-pathing only between border and access routers). ✗: A given scheme does not offer a given feature. ^{R}A given feature (e.g., multi-pathing) is offered *only* for resilience, *not* for more performance. ^{S}Minimal or non-minimal paths are offered *only* within spanning trees.

We conduct extensive, large-scale packet-level simulations, and a comprehensive theoretical analysis. ***We simulate topologies with up to ≈1 million endpoints*** (to the best of our knowledge, these are the largest shared-memory simulations so far). We motivate FatPaths in Figure 2. Slim Fly and Xpander equipped with FatPaths ensure ≈15% higher throughput and ≈2 lower latency than similar-cost fat trees, for various flow sizes[2] and for heavily-skewed traffic.

[2]We use the term "flow", which is equivalent to a "message".

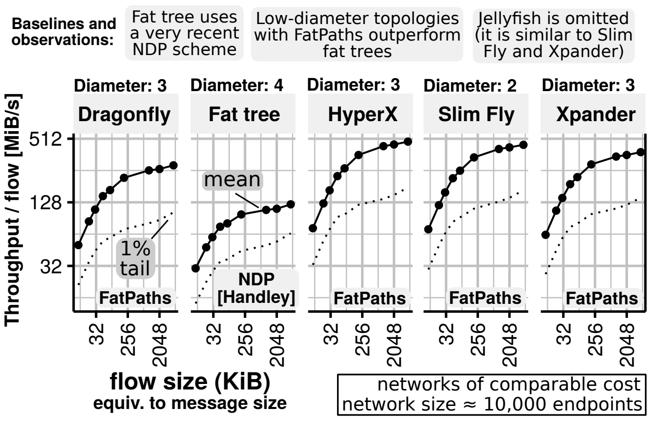

Figure 2: Example **performance advantages of low-diameter topologies that use FatPaths over fat trees** equipped with NDP (very recent routing architecture by Handley et al. [60]). Evaluation methodology is discussed in detail in § 7.

Towards the above goals, we contribute:

- A **simple and resilient routing architecture, FatPaths**, that successfully combines existing techniques from the HPC and datacenter communities, requires simple network hardware, and supports both shortest and non-minimal paths (§ 3, § 5).
- The identification of diversity of non-minimal paths as **a key resource** and the first **detailed analysis of the potential for multipath routing** in five low-diameter network topologies, considering several metrics for their path diversity (§ 4).
- A **novel path diversity metric** (Path Interference) that captures bandwidth loss between specific pairs of routers (§ 4) and enhaces the path diversity analysis.
- A comprehensive analysis of existing routing schemes in terms of their support for path diversity (Table 1).
- A **theoretical analysis** illustrating the advantages coming from FatPaths (§ 6).
- Extensive, large-scale packet-level simulations (reaching around **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 cost (§ 7).

2 Notation, Background, Concepts

We first introduce the notation and basic concepts. The most important used symbols are summarized in Table 2.

2.1 Network Model

We model an interconnection network as an undirected graph $G = (V, E)$; V and E are sets of routers[3] ($|V| = N_r$) and full-duplex inter-router physical links. Endpoints are *not* modeled explicitly. There are N endpoints in total, 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.

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

Network structure	V, E	Sets of vertices/edges (routers/links, $V = \{0, \ldots, N_r - 1\}$).
	N, N_r	#endpoints and #routers in the network ($N_r = |V|$).
	p	#endpoints attached to a router (*concentration*).
	k'	#channels to other routers (*network radix*).
	k	*Router radix* ($k = k' + p$).
	D, d	Network diameter and the average path length.
Diversity of paths (§ 4)	$x \in V$	Different routers used in § 4 ($x \in \{s, t, a, b, c, d\}$).
	$X \subset V$	Different router sets used in § 4 ($X \in \{A, B\}$).
	$c_l(A, B)$	*Count of (at most l-hop) disjoint paths* between router sets A, B.
	$c_{\min}(s, t)$	*Diversity of minimal paths* between routers s and t.
	$l_{\min}(s, t)$	*Lengths of minimal paths* between routers s and t.
	$I_{ac,bd}$	*Path interference* between pairs of routers a, b and c, d.
Layers (§ 5)	n	The total number of layers in FatPaths routing.
	σ_i	A layer, defined by its forwarding function, $i \in \{1, \ldots, n\}$.
	ρ	Fraction of edges used in routing.

Table 2: The **most important symbols** used in this work.

2.2 Network Topologies

We summarize the considered topologies in Table 3. We consider Slim Fly (SF) [23] (a variant with $D = 2$), Dragonfly (DF) [86] (the "balanced" variant with $D = 3$), Jellyfish (JF) [131] (with $D = 3$), Xpander (XP) [142] (with $D \le 3$), HyperX (Hamming graph) (HX) [5] that generalizes Flattened Butterflies (FBF) [85] with $D = 3$. We also use established three-stage fat trees (FT3) [90] that are a variant of the Clos network [34].

2.2.1 Topology Types Some selected networks are *flexible* (parameters determining their structure can have arbitrary values) while most are *fixed* (parameters must follow well-defined closed-form expressions). Next, networks can be *group hierarchical* (routers form *groups* connected with the same pattern of intra-group *local* cables and then groups are connected with *global* inter-group links), *semi-hierarchical* (there is some structure but no such groups), or *flat* (no distinctive hierarchical structure at all). Finally, topologies can be *random* (based on randomized constructions) or *deterministic*.

Topology	Structure remarks	D	Variant	Deployed?
Slim Fly (SF) [23]	Consists of groups	2	MMS	unknown
HyperX (HX2) [5]	Consists of groups	2	Flat. Butterfly [85]	unknown
Dragonfly (DF) [86]	Consists of groups	3	"balanced"	PERCS [14], Cascade [41]
HyperX (HX3) [5]	Consists of groups	3	"regular" (cube)	unknown
Xpander (XP) [142]	Consists of metanodes	≤3	randomized	unknown
Jellyfish (JF) [131]	Random network	≤3	"homogeneous"	unknown
Fat tree (FT) [90]	Endpoints form pods	4	3 router layers	Many systems

Table 3: Used topologies.

2.2.2 Fair Selection of Topology Parameters 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 concentration to $p = \frac{k'}{d}$; it maximizes throughput while minimizing congestion and network cost (we analyze this later in § 7). 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.

2.2.3 Special Case: Jellyfish The considered topologies cannot use arbitrary values of N_r and k'. An exception is Jellyfish, which is "fully flexible": There is a JF instance for each combination of N_r and k'. Thus, to fully evaluate JF, *for every other network* X, *we consider an equivalent* JF (denoted as X-JF) with identical N_r, k'.

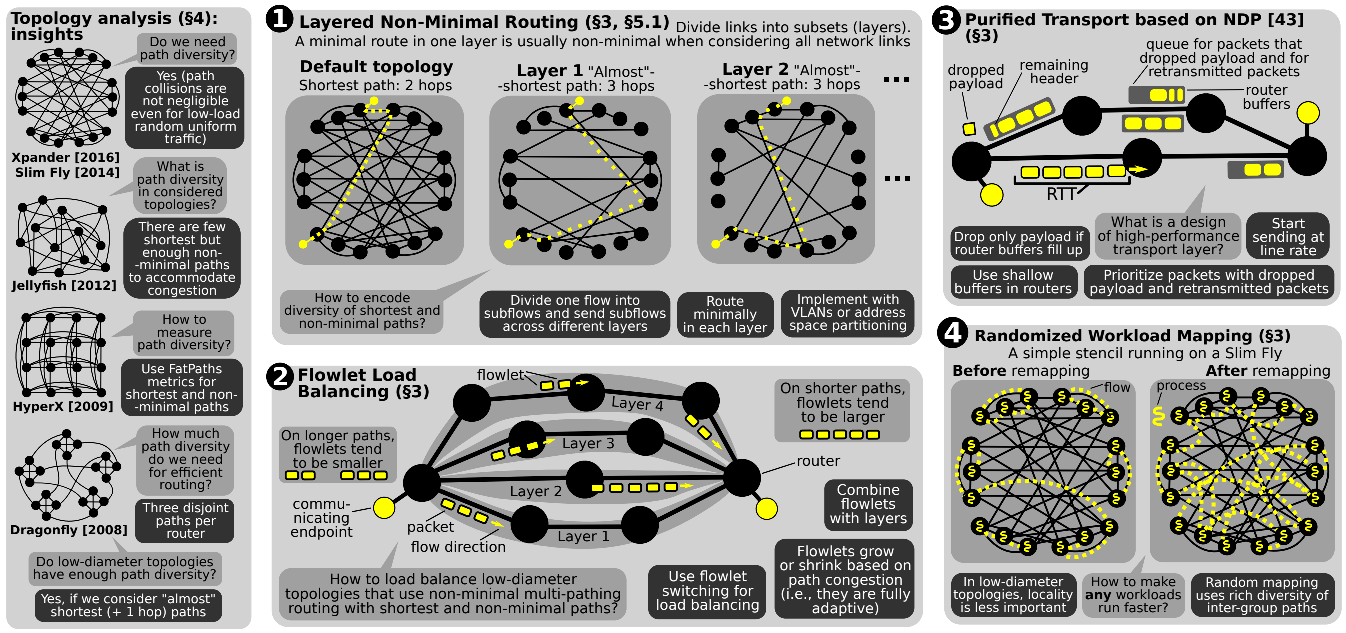
Topology analysis (§4): insights
Do we need path diversity?
Yes (path collisions are not negligible even for low-load random uniform traffic)
Xpander [2016]
Slim Fly [2014]
What is path diversity in considered topologies?
There are few shortest but enough non-minimal paths to accommodate congestion
Jellyfish [2012]
How to measure path diversity?
Use FatPaths metrics for shortest and non-minimal paths
HyperX [2009]
How much path diversity do we need for efficient routing?
Three disjoint paths per router
Dragonfly [2008]
Do low-diameter topologies have enough path diversity?
Yes, if we consider "almost" shortest (+ 1 hop) paths
1 Layered Non-Minimal Routing (§3, §5.1) Divide links into subsets (layers).
A minimal route in one layer is usually non-minimal when considering all network links
Default topology
Shortest path: 2 hops
Layer 1 "Almost"-shortest path: 3 hops
Layer 2 "Almost"-shortest path: 3 hops
...
How to encode diversity of shortest and non-minimal paths?
Divide one flow into subflows and send subflows across different layers
Route minimally in each layer
Implement with VLANs or address space partitioning
2 Flowlet Load Balancing (§3)
flowlet
Layer 4
Layer 3
Layer 2
Layer 1
On longer paths, flowlets tend to be smaller
On shorter paths, flowlets tend to be larger
router
communicating endpoint
packet
flow direction
Combine flowlets with layers
How to load balance low-diameter topologies that use non-minimal multi-pathing routing with shortest and non-minimal paths?
Use flowlet switching for load balancing
Flowlets grow or shrink based on path congestion (i.e., they are fully adaptive)
3 Purified Transport based on NDP [43] (§3)
dropped payload
remaining header
queue for packets that dropped payload and for retransmitted packets
router buffers
RTT
What is a design of high-performance transport layer?
Drop only payload if router buffers fill up
Start sending at line rate
Use shallow buffers in routers
Prioritize packets with dropped payload and retransmitted packets
4 Randomized Workload Mapping (§3)
A simple stencil running on a Slim Fly
Before remapping
After remapping
flow
process
In low-diameter topologies, locality is less important
How to make any workloads run faster?
Random mapping uses rich diversity of inter-group paths

Figure 3: Overview of FatPaths. Numbers (❶ – ❹) refer to elements of the routing architecture while "Topology analysis" summarizes our work on the structure of low-diameter topologies in terms of their path diversity. We present some of the considered research questions as well as the obtained insights and answers. In the leftmost panel, the term "path diversity" intuitively means the number of edge-disjoint paths between each pair of routers (details in § 4).

2.3 Flow Model

We use a Poisson-distributed flow arrival rate and a matrix defined on endpoint pairs to model flow sizes and traffic.

2.4 Considered Traffic Patterns

We analyze recent works in high-performance and datacenter networking [23, 31, 76–78, 112–116, 127, 152, 153] to select traffic patterns that represent important HPC workloads and cloud or datacenter traffic. 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 $t(s) \in V_e$.

2.4.1 Random Patterns First, we select **random uniform**

$$t(s) \in V_e \text{ u.a.r.},$$

and **random permutation**

$$t(s) = \pi_N(s) \in V_e, \quad \pi_N \text{ is a permutation selected u.a.r..}$$

These patterns represent **irregular workloads** such as graph computations, sparse linear algebra solvers, and adaptive mesh refinement methods [154]. They are used in both HPC studies [23] and data center and cloud infrastructure analyses [76, 89, 127].

2.4.2 Off-Diagonal Patterns We also use **off-diagonals**:

$$t(s) = (s + c) \mod N, \quad \text{for fixed } c.$$

These patterns are often used in workloads such as nearest neighbor data exchanges [154], used in HPC and data centers [73].

2.4.3 Bit Permutation Patterns Next, we pick **shuffle**, a traffic pattern that represents bit permutation pattern:

$$t(s) = \text{rotl}_i(s) \mod N, \quad 2^i < N < 2^{i+1}$$

where the bitwise left rotation on i bits is denoted as rotl_i.

They represent **collective operations** such as MPI-all-to-all or MPI-all-gather [23, 154], used in HPC. They are also used for the evaluation of data center networks [77, 89, 127].

2.4.4 Stencils We also use **stencils, realistic traffic patterns often used 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.

2.4.5 All-To-One In this pattern, traffic from all endpoints is directed towards a single random endpoint in the network.

2.4.6 Adversarial Pattern We use a skewed off-diagonal with large offsets (we make sure it has a very high amount of colliding paths).

2.4.7 Worst-Case Pattern Finally, we use **worst-case** traffic patterns. We focus on a recently proposed pattern, developed specifically to maximize stress on the interconnect while hampering effective routing [73]. 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 the generation process is individual for each network, our worst-case pattern stresses the interconnect in any setting, including HPC systems, data centers, or any other cloud infrastructure.

3 FatPaths Architecture: Overview

We first outline the FatPaths architecture. **A design summary is in Figure 3.** FatPaths stands on four key design ideas that, combined, effectively use the "fat" diversity of minimal and non-minimal paths. These ideas are layered non-minimal routing, flowlet load balancing, "purified" transport, and randomized workload mapping.

3.1 Layered Routing

To encode the diversity of minimal *and non-minimal* paths with commodity hardware, FatPaths divides all the 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. This enables taking advantage of the diversity of non-minimal paths in low-diameter topologies. 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.

We provide two schemes for the construction of layers: a simple randomized approach and an augmentation that minimizes the number of overlapping paths between communicating endpoints. Moreover, we encode existing routing schemes that enable multi-pathing, such as SPAIN [103], PAST [134], and k-shortest paths [131], using FatPaths layers. We analyze which scheme is most advantageous for which topology.

3.2 Load Balancing

To achieve very simple but powerful load balancing, we use flowlet switching [75, 132], a technique used in the past to alleviate packet reordering in TCP. A flowlet is a sequence (also referred to as a burst) of packets within one flow, separated from other flowlets by sufficient time gaps. Now, flowlet switching can also be used for a *very* simple load balancing: a router simply picks a random path for each flowlet, without *any* probing for congestion. This scheme was used for Clos networks [147]. The power of such load balancing lies in the fact that flowlets are *elastic*: their size changes *automatically* based on conditions in the network. On paths that have higher latency or lower bandwidth, flowlets are usually smaller in size because time gaps large enough to separate two flowlets are more frequent. Contrarily, paths with lower latency and more bandwidth feature longer flowlets because such time gaps appear less often.

We propose to use flowlets in low-diameter non-Clos networks, as a load balancing part of FatPaths. Here, we combine flowlets with layered routing: *each flow is divided into flowlets that are sent using different layers*. The key observation is that elasticity of flowlets *automatically* ensures that such load balancing takes into account both static network properties (e.g., longer vs. shorter paths) and dynamic network properties (e.g., more vs. less congestion). Consider a pair of communicating routers. As we will show later (§ 4), virtually all router pairs are connected with exactly one shortest part but multiple non-minimal paths, possibly of different lengths. In many workload scenarios, a shortest path experiences smallest congestion. Contrarily, longer paths are more likely to be congested. Here, the elasticity of flowlet load balancing ensures that larger flowlets are sent over shorter and less congested paths. Shorter flowlets are then transmitted over longer and usually more congested paths.

3.3 Purified Transport with NDP [60]

Transport layer in FatPaths is inspired by recent Clos transport designs, namely NDP [60], in that it removes virtually all TCP and Ethernet issues that hamper latency and throughput. First, if router queues fill up, *only packet payload is dropped*. As packet headers with all the metadata are preserved, 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 i, when packets within flowlets transmitted over paths belonging to i arrive without payload, indicating congestion.

Second, routers enable prioritization of (1) headers of packets that lost their payload, and (2) retransmitted packets. This ensures that congested flows finish quickly and it reduces head-of-line-blocking. Third, senders transmit the first RTT at line rate, without probing for available bandwidth. Finally, router queues are shallow. All these elements result in a low-latency and high-throughput transport layer that meets demands of various traffic patterns and can be implemented with existing network technology.

3.4 Randomized Workload Mapping

We optionally use random assignments, where communicating endpoints are located at routers chosen u.a.r (uniformly at random). First, one often cannot rely on locality due to schedulers or virtualization. For example, Cray machines often host processes from one job in different machine parts to increase utilization. Second, many workloads, such as distributed graph processing, have little or no locality [96]. Finally, perhaps most importantly, the low diameter of used topologies, especially the ones with $D = 2$, mostly eliminates the need for locality-aware software. We predict that this will be a future trend as *reducing cost and power consumption with simultaneous increase in scale is inherently associated with reducing diameter* [23]. However, to cover applications tuned for locality, **we also evaluate non-randomized workloads** and show that FatPaths ensures the highest performance in such cases as well.

4 Path Diversity in Modern Topologies

FatPaths *enables* using the diversity of paths in low-diameter topologies for high-performance routing. To develop FatPaths, we first need to understand the "nature" of this path diversity. We also justify and motivate using multi-pathing in low-diameter networks. Namely, we 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 would eliminate flow conflicts (§ 4.1). We then formalize the notion of "path diversity" (§ 4.2) and we use our formal measures 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* (§ 4.3). In evaluation (§ 7), we show that another type of flow conflicts, flow overlaps, is also alleviated by FatPaths. *To the best of our knowledge, compared to recent works in low-diameter networks [6, 16, 22, 42, 61, 73, 77, 78, 81–83, 91, 131, 139, 140, 142]*, **we provide the most extensive analysis of path diversity in low-diameter networks so far (with respect to the number of path diversity metrics and topologies).**

4.1 How Much Path Diversity Do We Need?

FatPaths uses path diversity to avoid congestion due to *conflicting flows*. Consider two communicating pairs of endpoints. Generated flows *conflict* when their paths **collide** (i.e., flows use an identical path) or **overlap** (i.e., flows share some links), see Figure 5. Collisions only depend on how communicating endpoints are attached to routers (i.e., on p, N_r, and thus also indirectly D). Intuitively, collisions measure *workload demand for path diversity*

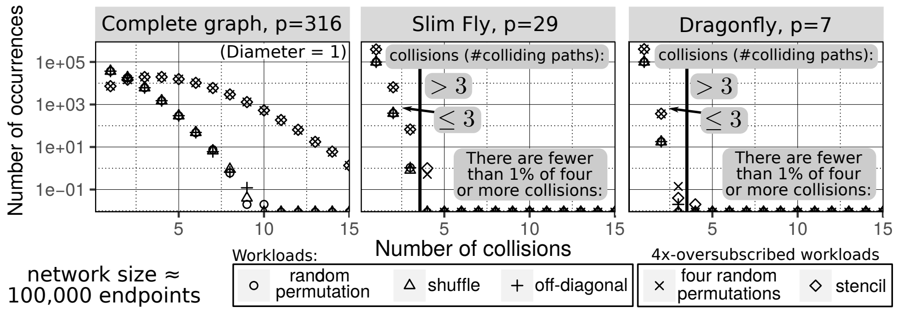

Figure 4: Histogram of colliding paths (per router pair) with $p = \frac{k'}{D}$. We plot data for Slim Fly, Dragonfly, and a complete graph, using $N \approx 100,000$. The results for other values of N and other topologies have identical shares of colliding paths.

(multi-pathing). Contrarily, overlaps depend on the topology details (i.e., how routers are connected to other routers). Intuitively, overlaps capture *how well a topology can sustain a workload.*

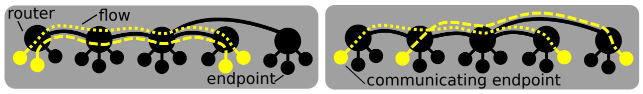

Figure 5: Example collision (left) and overlap (right) of paths and flows.

To understand how much path diversity is needed to alleviate flow conflicts, we analyze the impact of topology properties (diameter D, concentration p, size 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.

We consider five traffic patterns: a random permutation, a randomly-mapped off-diagonal, a randomly mapped shuffle, four random permutations in parallel, and a randomly mapped 4-point stencil composed of four off-diagonals. The last two patterns are 4× oversubscribed and thus *expected to generate even more collisions.*
✓ Takeaway We need at least three disjoint paths per router pair to handle colliding paths in any considered workloads, assuming random mapping. As the number of colliding paths lower bounds the number of overlapping paths, the same holds for overlaps.

4.2 How Should We Measure Path Diversity?

To analyze whether 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 (§ 4.2.1). This measures address path **collisions**. Moreover, to analyze path **overlaps**, we define two further measures: *path interference* (PI, § 4.2.2) and *total network load* (TNL, § 4.2.3). 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.

4.2.1 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 [46] (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 routing and congestion reduction as they use fewest resources for each flow. We derive the *distribution* of minimal path *distances* $l_{\min}$ and *diversities* $c_{\min}$. Intuitively, $l_{\min}$ describes (statistically) distances between any router pairs while $c_{\min}$ provides their respective counts. We have:

$$l_{\min}(s, t) = \underset{i \in \mathbb{N}}{\arg\min}\{t \in h^i(\{s\})\} \quad \text{("minimal path distances")}$$
$$c_{\min}(s, t) = c_l(\{s\}, \{t\}) \text{ with } l = l_{\min}(s, t) \quad \text{("counts of minimal paths")}$$

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

For **non-minimal paths**, we consider the CDP $c_l(A, B)$ of random router pairs $s \in A, t \in B$, with path lengths $l > l_{\min}(s, t)$.

4.2.2 Path Interference (PI)

We define Path Interference (PI) which is – to the best of our knowledge – the first metric that measures path overlap while considering the local topology structure. Here, paths between two router pairs a, b and c, d (a communicates with b; c communicates with d) *interfere* if their total count of disjoint paths at length 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

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

***Path interference captures the fact that, if** a **and** b **communicate, the available bandwidth between** c **and** d **is reduced**.*

4.2.3 ***Total Network Load (TNL)*** TNL is a simple *upper bound on the number of flows that a network can maintain without congestion.* Intuitively, it constitutes the maximum *supply of path diversity* offered by a topology. It uses the notion that a flow occupying a path of length l "consumes" l links. TNL is defined as $\frac{k' N_r}{d}$.

✓ **Takeaway** Due to the rich nature of path diversity, we suggest to use several measures, for example count of minimal as well as non-minimal disjoint paths (measuring collisions) and path interference as well as total network load (measuring overlaps).

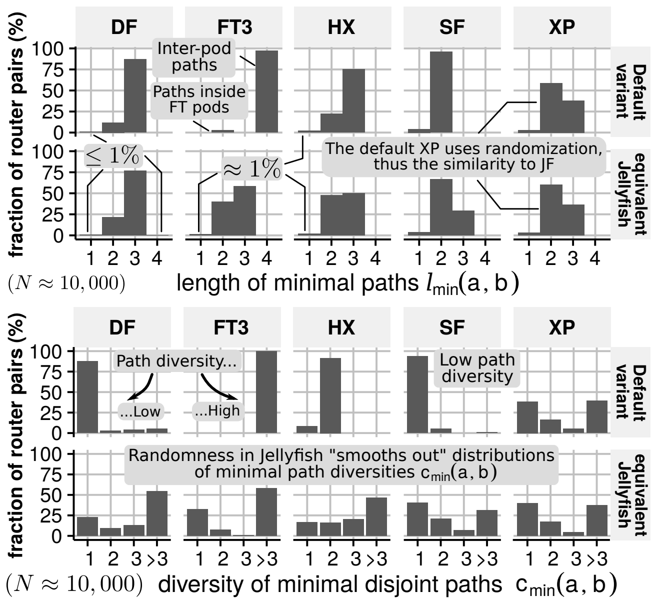

Figure 6: Distributions of lengths and counts of shortest paths.

4.3 Do We Have Enough Path Diversity?

We now use our measures to analyze path diversity in low-diameter networks. First, selected results on **minimum 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 only. In the corresponding JF networks, 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.

For **non-minimal paths**, we first summarize the results in Table 4. 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. ***Considered topologies provide three 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 large, most samples fall into a common case, where PI is small (c.f. small fractions). We thus focus on the extreme tail of the distribution (we show both mean and tail), see Table 4. We use radix-invariant PI values (as for CDP) at a distance d' selected to ensure that the 99.9% tail of collisions $c_{d'}(A, B) \geq 3$. Thus, we analyze PI in cases where demand from a workload is larger than 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.*

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

Table 4: 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 $c_{d'}(A, B) \geq 3$.

4.4 Final Takeaway on Path Diversity

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

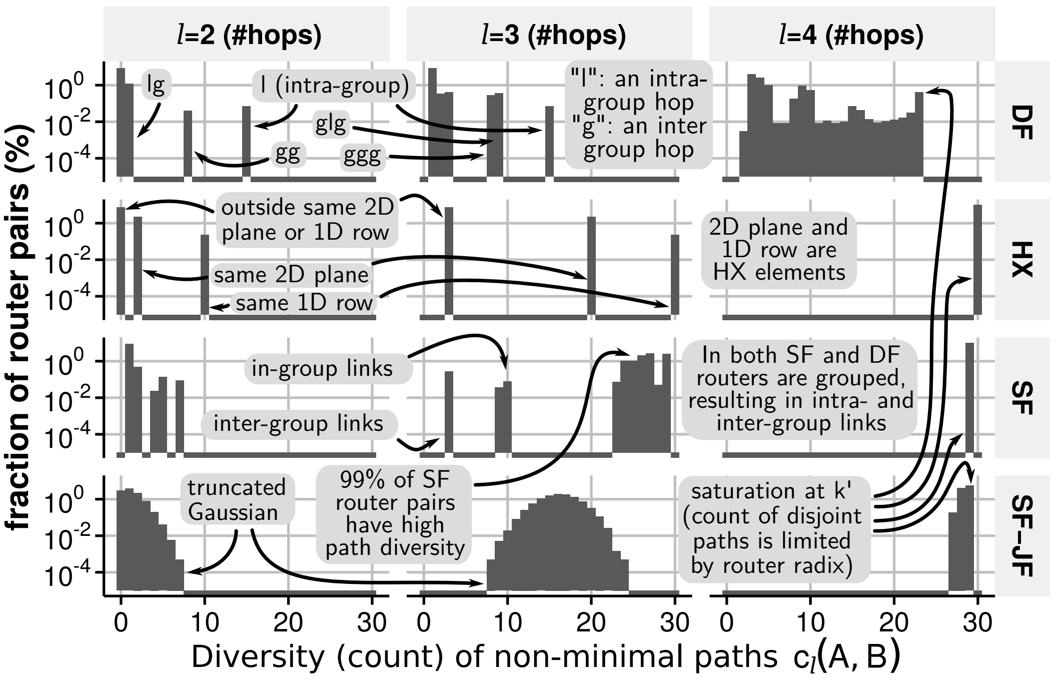

Figure 7: Distribution of the number of non-minimal edge-disjoint paths with up to l hops ($c_l(A, B)$) between random router pairs ($N \approx 10,000$).

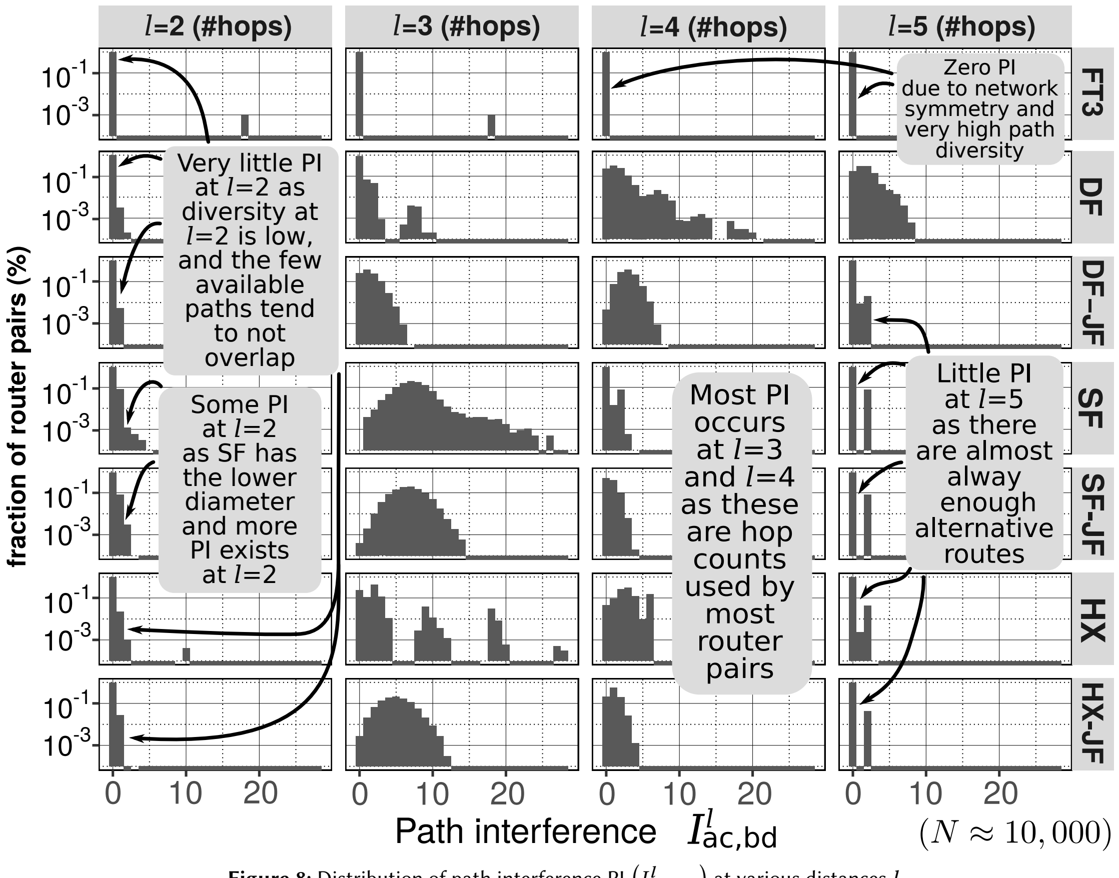

Figure 8: Distribution of path interference PI $\left(I^l_{ac,bd}\right)$ at various distances l.

5 FatPaths: Design and Implementation

FatPaths is a high-performance, simple, and robust ***routing architecture*** *that uses rich path diversity in low-diameter topologies* (analyzed in § 4) *to enhance Ethernet stacks in clusters, data centers, and supercomputers.* FatPaths aims to accelerate both cloud computing and HPC workloads. We now summarize key design ideas behind FatPaths. First, we develop the layered routing scheme that (1) is capable of encoding the rich diversity of both minimal and non-minimal paths, and (2) can be implemented with commodity Ethernet hardware. Second, we combine layered routing with flowlet switching and a "purified" transport layer based on very recent Clos designs [60]. The former enables very simple but powerful load balancing. The latter ensures low-latency and high-throughput transport. The design of both load balancing and transport layer is straightforward and presented in § 3 and Figure 3. *Here, we focus on the* ***layered routing****, the key element of FatPaths that enables using the rich "fat" path diversity analyzed in § 4.*

5.1 Routing Model

We assume simple destination-based routing, compatible with any relevant technology, including source-based systems like NDP. To compute the output port $j \in \{1, \ldots, k'\}$ 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 σ with fixed t we eventually reach $s' = t$ and finish. The forwarding function σ must be defined such that a path from any s to any t is *loop-free*.

5.2 Layered Routing

We use n routing functions $\sigma_1, \ldots, \sigma_n$ for n layers, where each router uses function σ_i for a packet with a *layer tag i* attached. The layer tags are chosen on the endpoint by the adaptivity algorithm. We use n layers associated with n routing functions. Each router uses the i-th routing function, denoted as σ_i, for a packet with a *layer tag i* attached. All layers but one accommodate a fraction of links, maintaining non-minimal paths. One layer (associated with σ_1) uses all links, maintaining minimal paths. A single layer constitutes a Directed Acyclic Graph (DAG). The fraction of links in one layer is controlled by a parameter $\rho \in [0; 1]$. 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 ($\rho = 1$ can be used if there is high minimal-path diversity in the topology.) To facilitate implementation of FatPaths, we provide configurations of layers (ρ, n) that ensure high-performance routing for each used topology. Files with full specifications are in a dedicated repository (see link on page 1) while *performance analysis of different ρ and n is in § 6 and § 7.*

5.3 Construction of Layers

We develop two schemes for constructing layers in FatPaths; we also adapt selected existing protocols.

5.3.1 *Random Permutations*

An overview of layer construction is in Listing 1. We start with one layer with all links for 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.. The network may become disconnected with E', but for the used values of ρ, this is unlikely and a small number of attempts delivers a connected network.

```
L = {E} //Init a set of layers L; we start with E that corresponds to σ1.
P = {π1(V), ..., πn−1(V)} //Generate n − 1 random permutations of vertices.
foreach π ∈ P do: //Iterate over each of the generated permutations...
  //One iteration of the main loop derives one layer associated with some σi.
  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:
      //ρ ∈ [0;1] (see § 5.2) is a parameter that controls layer's sparsity.
      //If we end up here, it means the edge (u, v) was sampled.
      E′=E′ ∪ (u, v) //Add a sampled edge to the layer.
  L = L ∪ {E′}
```

Listing 1: Overview of the FatPaths algorithm for constructing routing layers **(a simple variant based on random permutations)**.

5.3.2 *Minimizing Path Overlap*

We also use a variant 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 other layers. Importantly, while computing paths, *we prefer paths that are one hop longer than minimal ones, using the insights from the path diversity analysis (§ 4).*

5.3.3 *Adapting Existing Schemes*

In addition to our two schemes for generating layers, we also adapt existing approaches that provide multi-pathing. These are SPAIN [103], PAST [134], and k-shortest paths [131], three recent schemes that support (1) multi-pathing and (2) disjoint paths (as identified in Table 1).

5.4 Populating Forwarding Entries

The σ_i functions are deployed with forwarding tables. To derive these tables, we compute minimum paths between every two routers s, t within layer σ_i. Then, 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. For any hypothetical network size, constructing layers is not a computational bottleneck, given the $O(|V|^2 \log |V|)$ complexity of Dijkstra's shortest path algorithm for $|V|$ vertices [38].

5.5 Implementation Details

We briefly discuss the most important implementation details.

5.5.1 *Implementation of Layers*

We propose two schemes to deploy 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 [103] or PAST [134], one can use **VLANs** [47] 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.

5.5.2 *Implementation of Forwarding Functions*

Forwarding functions can be implemented with well-known static schemes such as simple lookup tables, either flat **Ethernet exact matching** or hierarchical **TCAM longest prefix matching** tables. In the former, one entry maps a single input tuple to a single next hop. The latter are usually much smaller but more powerful: one entry can provide the next hop information for many input tuples.

As not all the considered topologies are hierarchical, we cannot use all the properties of longest match tables. Still, we observe that all endpoints on one router share the routes towards that router. We can thus use prefix-match tables to *reduce the required number of entries from $O(N)$ to $O(N_r)$*. This only requires exact matching on a fixed address part. As we mainly target low-diameter topologies, space savings due to moving from $O(N)$ to $O(N_r)$ *can be significant*. For example, an SF with $N = 10,830$ has $N_r = 722$. Such semi-hierarchical forwarding was proposed in, for example, PortLand [105] and shadow MACs [3]. Since we use a simple, static forwarding function, it can also be implemented on the endpoints themselves, using source routing [72].

5.5.3 *Addressing*

To integrate FatPaths with **L2/Ethernet**, one can use 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 [25]. Alternatively, one could also use a simple **L3/IP** scheme. First, every endpoint has an IP address of the form $10.i.s.h$ for each layer (s, h, and i identify a router, an endpoint within the router, and the layer ID). Second, for the inter-router links, addresses from a disjoint range are used, e.g,. $192.168.*.*$, with one /30 subnet per link. Finally, each router s has one forwarding rule for each other router, of the form $10.i.t.*/24$ *via* $192.168.x.y$, where the inter-router link address is chosen from the router's ports according to the forwarding function $\sigma_i(s,t)$.

5.5.4 ***Fault-Tolerance***

Fault-tolerance in FatPaths is based on preprovisioning multiple paths within different layers. For major (infrequent) topology updates, we recompute layers [103]. 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 [60, 65, 70, 103, 147] for the exact mechanisms of failure detection. Traffic redirection is based on flowlets [147]. Failures are treated similarly to congestion: the elasticity of flowlets automatically ensures that no data is sent over an unavailable path.

Besides flowlet elasticity, the layered FatPaths design enables other fault-tolerance schemes. Assuming L2/Ethernet forwarding and addressing, we propose to adapt a scheme from SPAIN [103] or PAST [134], both of which use the concept of layered routing similar to that in FatPaths. We first identify the layer with a failed element and then reroute the affected flows to a new randomly selected layer. This is done only for endpoints directly affected by the failure; thus, the affected layer continues to operate for endpoints where no failure was detected. The utilization of the affected layer is reestablished upon a receipt of any packet from this layer. Moreover, FatPaths could limit each layer to be a spanning tree and use mechanisms such as Cisco's proprietary Per-VLAN Spanning Tree (PVST) or IEEE 802.1s MST to fall back to the secondary backup ports offered by these schemes.

Finally, assuming L3/IP forwarding and addressing, one could rely on resilience schemes such as VIRO's [70].

5.6 ECMP for Non-Minimal and Minimal Paths

We separately discuss how one could use ECMP for multipathing on *both minimal and non-minimal paths*. Our design facilitates deploying FatPaths on commodity hardware used in data centers, clusters, and other cloud infrastructure.

5.6.1 *ECMP Polarization*

Polarization is an ECMP-related issue [27, 123]. Namely, for a given TCP flow, successive switches select identical output ports for each packet, which may leave some links unused. This effect is called *polarization*.

5.6.2 *Using Polarization for Non-Minimal ECMP*

As another option, layered routing can even be implemented using ECMP hardware. In ECMP, a switch selects the i-th forwarding rule out of n equal-cost options based on a pseudo-random hash value computed from the forwarding parameters, including the pseudo-random TCP source port. In normal ECMP deployments, these hash functions are randomized, to make sure the switches select different combinations of i's for varying source ports. Instead, we can deliberately introduce *polarization* by selecting *the same hash function for each switch*. Now, each forwarding tuple will get the same i assigned over the entire network, and we can use i as a layer tag, populating each slot in the routing table with with a different forwarding function σ_i. The paths in each slot might not be equal-cost, and the resulting forwarding table not correct according to ECMP rules, but thanks to polarization, the routing will exactly represent our layered routing and therefore also be loop-free. A minor disadvantage of this approach is that we can't systematically choose which layer i a flow will use, since i is still determined by a hash function; however this is not a problem with the proposed flowlet routing, which randomly chooses layers and will not be affected by another randomization step.

6 Theoretical Analysis

We now conduct a theoretical analysis.

6.1 Traffic Patterns

We focus on a recently proposed worst-case traffic pattern, developed specifically to maximize stress on the interconnect while hampering effective routing [73]. 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.

6.2 Considered Schemes

We use both variants of layered routing proposed in this work. We also consider SPAIN, PAST, and k-shortest paths, adopted to the layered setting. Originally, SPAIN uses a set of spanning trees, using greedy coloring to minimize their number and maximize path disjointness; one tree is one layer. PAST uses one spanning tree per host, aiming at distributing the trees uniformly over available physical links. k-shortest paths [131] spreads traffic over multiple shortest paths (if available) between endpoints.

6.3 Number of Layers

SPAIN and PAST use trees as layers while FatPaths allows for arbitrary DAGs. This brings drawbacks, as each SPAIN layer can use at most $N_r - 1$ links, while the topology contains $\frac{N_r k'}{2}$ links.

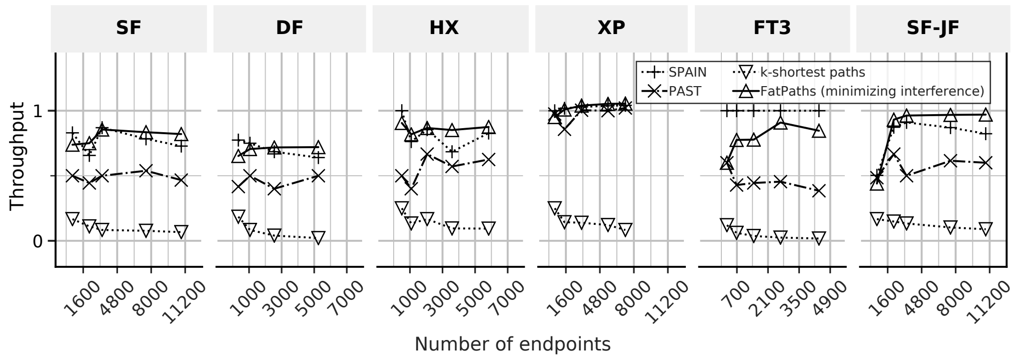

Figure 9: Theoretical analysis of FatPaths performance: Maximum achievable throughput in various layered routing schemes implemented in FatPaths (traffic intensity: 0.55).

Thus, at least $O(k')$ layers are required to cover all minimal paths, and SPAIN may require even $O(N_r)$. Moreover, PAST 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.*

6.4 Throughput

We also analyze maximum achievable throughput (MAT) in various layered 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. [73]).

We test all layered routing schemes implemented in FatPaths (including SPAIN, PAST, and k-shortest paths) on *all considered topologies, topology sizes, traffic patterns, and traffic intensity (fraction of communicating endpoint pairs).* We use TopoBench, a throughput evaluation tool [73] that uses linear programming (LP) to derive $\mathcal{T}$. We extended TopoBench's LP formulation of MCF so that it includes layered routing. Most importantly, instead of one network for accommodating MCF, we use n networks (that represent layers) for allocating flows. We also introduce constraints that prevent one flow from being allocated over multiple layers.

Selected results are in Figure 9. As expected, SPAIN – a scheme developed specifically for Clos – delivers more performance on fat trees. However, it uses up to $O(N_r)$ layers. The layered routing scheme that minimizes path interference generally outperforms the SPAIN variant (that we tuned to perform well on low-diameter topologies) on other networks. 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).

7 Simulations

We now analyze the performance of the FatPaths architecture, including layered routing but also adaptive load balancing, the transport protocol based on NDP [60], and randomized mapping. We consider the combined performance advantages but we also investigate how each single element impacts the final performance. Specifically, we will illustrate how low-diameter topologies equipped with FatPaths outperform novel superior fat tree designs. We use two different simulation tools that reflect two considered environments: HPC systems and cloud infrastructure.

7.1 Methodology, Parameters, Baselines

We first discuss used parameters, methodology, and baselines.

7.1.1 Topologies We consider all the discussed topologies as specified in § 2: SF, XP, JF, HX, DF, and FT. We use their most advantageous variants (e.g., the "balanced" Dragonfly [86]) while fixing the network size N (N varies by up to ≈10% as there are limited numbers of configurations of each network). Slim Fly represents a recent family of diameter-2 topologies such as Multi-Layer Full-Mesh [78] and Two-Level Orthogonal Fat-Trees [143, 144]. To achieve fixed cost and N we use 2× oversubscribed fat trees.

7.1.2 Topology Parameters We now extend the discussion on the selection of key topology parameters. We select N_r and k' so that considered topologies use similar amounts of hardware. To analyze these amounts, we analyze the *edge density*: a ratio between the number of all the cables $\frac{1}{2}N_r k' + N_r p$ and endpoints $N = N_r p$. It turns out to be (asymptotically) constant for all topologies (the left plot in Figure 10) and *related to D*. Next, higher-diameter networks such as DF require more cables. As explained before [23]: Packets traverse more cables on the way to their destination. We also illustrate k as a function of N (the right plot in Figure 10). An interesting outlier is FT. It scales with k similarly to networks with $D = 2$, but with a much lower constant factor, at the cost of a higher D and thus more routers and cables. This implies that FT is most attractive for small networks using routers with constrained k. We can also observe the unique SF properties: For a fixed (low) number of cables, the required k is lower by a constant factor (than in, e.g., HX), resulting in better N scaling.

7.1.3 Routing and Transport Schemes We use flow-based non-adaptive ECMP as the baseline (routing performance lower bound). Low-diameter topologies use FatPaths while fat trees use NDP with all optimizations [60], additionally enhanced with LetFlow [147], 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. [60]. 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

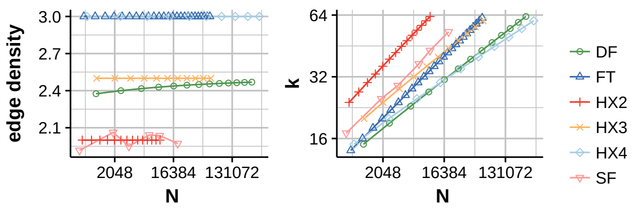

Figure 10: Edge density (number of edges)/(number of endpoints) = $(\frac{1}{2}N_r k' + N_r p)/N$ and radix k as a function of network size N. All cables, including endpoint links, are considered. This allows for a fair comparison with fat trees. We use radix values $k \leq 64$.

ECN [11, 44, 119]. We use these schemes to illustrate that FatPaths can accelerate not only bare Ethernet systems but also cloud computing environments that usually use full TCP stacks [17, 68].

7.1.4 Flows and Messages We vary flow sizes (and thus message sizes as a flow is equivalent to a message) from 32 KiB to 2 MiB.

7.1.5 Metrics We use **flow completion time** (FCT), which also represents **throughput per flow** TPF = $\frac{\text{flow size}}{\text{FCT}}$. We also consider **total time to complete** a tested workload.

7.1.6 Performance Validation Our evaluation is influenced by a plethora of parameters and effects, many of which are not necessarily related to the core paper domain. Some of them may be related to the incorporated protocols (e.g., TCP), others to the used traffic patterns. Thus, we also establish baseline comparison targets and we fix various parameters to ensure fair comparisons. To characterize **TCP effects**, **one baseline is a star (ST) topology** that contains a single crossbar switch and attached endpoints. It should not exhibit any behavior that depends on the topology structure as it does not contain any inter-switch links. We use the same flow distribution and traffic pattern as in the large-scale simulation, as well as the same transport protocols. This serves as an upper bound on performance. Compared to measurements, we observe the lowest realistic latency and the maximum achievable link throughput, as well as flow control effects that we did not explicitly model, such as TCP slow start. There is no additional congestion compared to measured data since we use randomized workloads. Second, as a lower bound on **routing performance**, we show results for flow-based **ECMP** as an example of a non-adaptive routing scheme, and **LetFlow** as an example of an adaptive routing scheme. We also include results of unmodified **NDP** (with oblivious load balancing) on FTs.

7.1.7 Simulation Infrastructure and Methodology We use the OMNeT++ [148, 149] parallel discrete event simulator with the INET model package [150] and the *htsim* packet-level simulator with the NDP reference implementation [60]. We use OMNeT++ to enable detailed simulations of full networking stack based on Ethernet and TCP together 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.

We extend the INET TCP stack with ECN (RFC 3168 [120]), MPTCP (RFC 6824 [45], RFC 6356 [118]), and DCTCP. We extend the default router model with ECMP (Fowler-Noll-Vo hash [87]) and LetFlow. 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. In htsim, we use similar parameters; they match those used by Handley et al.. We extend htsim to support arbitrary topologies, FatPaths routing and adaptivity, and our workload model. Routers use tail-dropping with a maximum queue size of 100 packets per port. ECN marks packets once a queue reaches more than 33 packets. Fast retransmissions use the default threshold of three segments. We also model a latency in the software stack (corresponding to interrupt throttling) to 100kHz rate. For FatPaths, we use 9KB jumbo frames, an 8-packet congestion window, and a queue length of 8 full-size packets.

7.1.8 Scale of Simulations We fix various scalability issues in INET and OMNeT++ to allow parallel simulation of large systems, *with up to ≈1 million endpoints.* To the best of our knowledge, *we conduct the largest shared-memory simulations (endpoint count) so far in the networking community* for the used precision and simulation setup.

7.1.9 Gathering Results 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, and we only record per-flow quantities, this suffices for statistical significance. We simulate a fixed number of flows starting in a fixed window of time, and drop the results from the first half of that window for warmup. We summarize the resulting distributions with arithmetic means or percentiles of distributions.

7.1.10 Traffic Patterns We use the traffic patterns discussed in § 2, in both **randomized** and **skewed non-randomized** variants. We also vary the fraction of communicating endpoints.

7.1.11 Shown Data When some variants or parameters are omitted (e.g., we only show SF-JF to cover Jellyfish), this indicates that *the shown data is representative.*

7.2 Performance Analysis: HPC Systems

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

Low-Diameter Topologies with FatPaths Beat Fat Trees We analyze both Figure 2 from page 2 (randomized workload) and Figure 11 (skewed non-randomized workload). In each case, low-diameter topologies outperform 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 lies in 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 § 1, we conclude that *FatPaths enables low-diameter topologies to outperform state-of-the-art fat trees.*

FatPaths Uses "Fat" Non-Minimal Path Diversity Well We now focus on the performance of FatPaths with heavily skewed

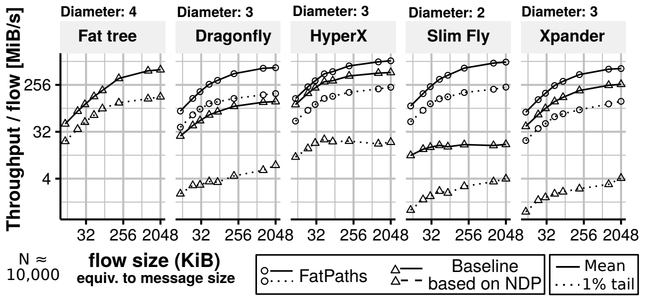

Figure 11: Performance analysis of a skewed adversarial traffic.

non-randomized workloads, see Figure 11. Non-minimal routing with FatPaths, 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 diversity of minimal paths (cf. Figure 6). Thus, *FatPaths effectively leverages the diversity of non-minimal paths.*

What Layer Setup Fares Best? We also investigate the impact of the number (n) and the sparsity (ρ) of layers in FatPaths on performance and resolution of collisions; see Figure 12 (layers are computed with simple random edge sampling). Nine layers (one complete and eight sparsified) suffice to produce three disjoint paths per router pair, resolving most collisions for both SF and DF (other networks follow similar patterns). To understand what n resolves collisions on global channels in DF, we also consider a clique. Here, more layers are required, since higher-multiplicity path collisions appear (visible in the 99% tail). We also observe that, when more layers *can* be used, it is better to use a higher ρ (cf. FCT for $n = 64$ and different ρ). This reduces the maximum achievable path diversity, but it also keeps more links available for alternative routes *within each layer*, increasing chances of choosing disjoint paths. It also increases the number of minimal paths in use across all entries, reducing total network load.

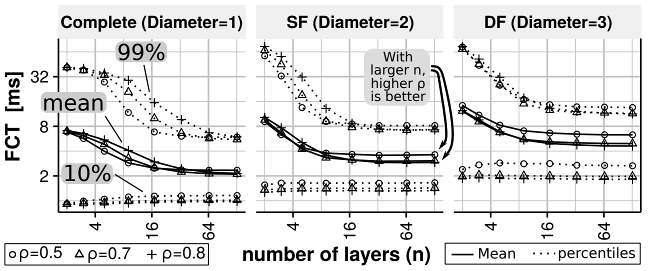

Figure 12: Effects of the number of layers n and the amount of remaining edges ρ on FatPaths, on long flows (size 1MiB); $N \approx 10,000$ (htsim).

FatPaths Scales to Large Networks We also simulate large-scale SF, DF, and JF (we could not simulate several similar-size networks such as FT with high path diversity that leads to very excessive memory use in the simulator). We start with SF, SF-JF, and DF ($N \approx$ 80,000), see Figure 13. 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 needs to handle high multiplicities of overlapping flows. We also conduct runs with $N \approx 1,000,000$ endpoints. Here, we illustrate the distribution of the FCT of flows for SF and SF-JF. Our analysis indicates that flows on SF tend to finish later that on SF-JF.

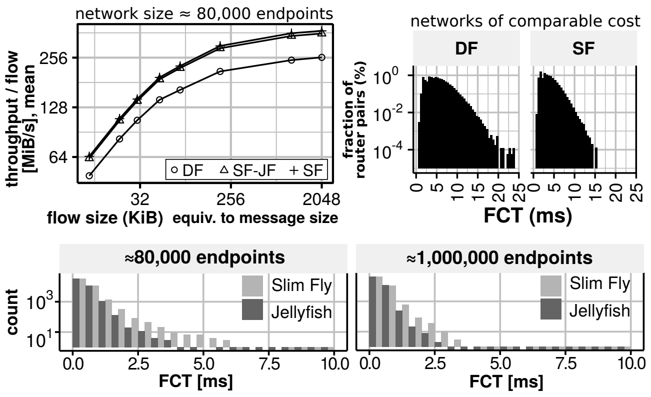

Figure 13: FatPaths on large networks; FCT histograms for flow size 1MiB (htsim).

7.3 Performance Analysis: Cloud Systems

We also analyze FatPaths on networks with Ethernet *and full TCP stack.* This setting represents TCP data centers and clusters often used as cloud infrastructure [68]. 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 § 7.2, we only summarize TCP-related insights.

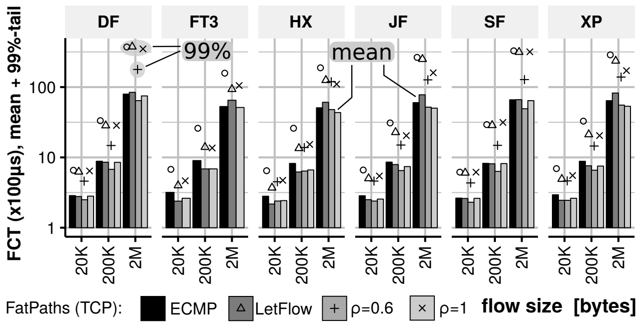

Figure 14: FatPaths on TCP compared to ECMP and LetFlow.

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 do not provide minimal-path diversity. *Non-minimal routing in FatPaths and* $\rho = 0.6$ *fixes it*, even with only $n = 4$ layers. On the 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 can fully resolve this problem.* Short-flow FCTs are dominated by TCP flow control effects, which are not affected much by routing changes.

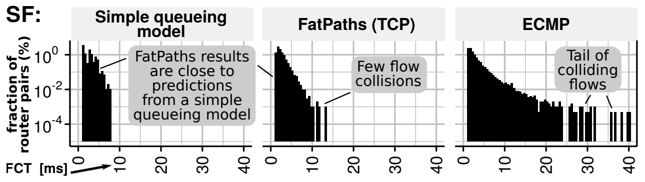

Figure 15: FCT distribution of long flows (size 1MiB**) on SF**. FatPaths on TCP with non-minimal routing approaches predictions from a simple queuing model (model details omitted due to space constrains); ECMP shows a long tail of colliding flows.

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 § 7.2 for SF and DF.

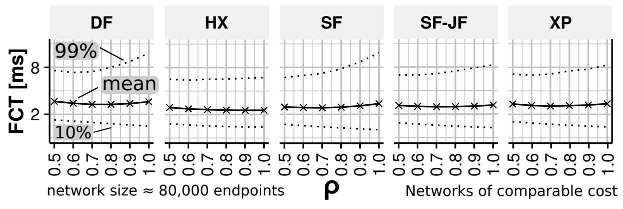

Figure 16: Impact of ρ on long-flow (1MiB) FCT with FatPaths on TCP, $n = 4$. The largest impact of non-minimal routing is for DF and SF, with a 2× improvement in tail FCT; small improvements on tail FCT are seen in all topologies, but there are no throughput improvements on networks with higher minimal-path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

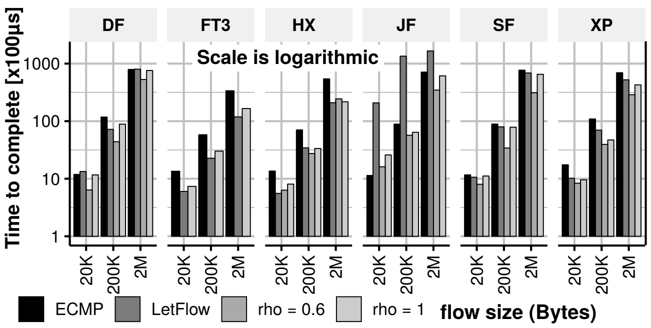

Figure 17: FatPaths on TCP vs. ECMP and LetFlow (stencil).

7.4 Performance Analysis: Vertical Effects

To facilitate analysis of the large amounts of included performance data, we now summarize analyzes of different FatPaths design choices ("vertical" analysis). First (1), different layer configurations (ρ, n) for various D are investigated in Figure 12 and in § 7.2 (bare Ethernet systems) as well as in Figure 16 and in § 7.3 (TCP systems). Differences (in FCT) across layer configurations are up to 4×; increasing both n and ρ maximizes performance. Second (2), the comparison of adaptive load balancing ("LetFlow") based on flowlet switching vs. static load balancing ("ECMP") is in Figure 14 and in sec:clouds; adaptivity improves tail and short flow FCTs at the cost of long flow throughput. Third (3), the comparison of FatPaths with and without Purified Transport is omitted due to space constraints; performance with no Purified Transport is always significantly worse. (4) We also analyze performance with and without layered routing (Figure 14, "ECMP" and "LetFlow" use no layers at all); not using layers is detrimental for performance on topologies that do not provide minimal-path diversity (e.g., SF or DF). Moreover (5), we also study the impact of using *only* the shortest paths in FatPaths (Figure 14, baseline "$\rho = 1$"); it is almost always disadvantageous. Finally (6), the effect from workload randomization is illustrated in Figures 2 (randomization) and 11 (no randomization); randomization increases throughput by ≈2×.

Figure 11 shows that fat trees with NDP outperform low-diameter networks that do *not* use non-minimal paths (the "NDP" baseline). *FatPaths, by accommodating non-minimal paths, enables low-diameter topologies to outperform fat trees, even for up to 2× for the* ***adversarial*** *traffic pattern.*

7.5 Final Takeaway on Performance

We are now able to answer the main question from § 1. Most importantly, 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 is a routing architecture that enables this.** Moreover, it combines random workload mapping, purified transport, flowlet load balancing, and layered routing, achieving high performance on both bare Ethernet systems and full TCP stacks. Thus, *it enables speedups on HPC systems such as supercomputers or tightly coupled clusters, or cloud infrastructure such as data centers.*

8 Discussion

Relations Between Metrics 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 [77]) in Figure 18. The figure shows how various measures vary when increasing the network load expressed by count of communicating router pairs x. The values of measures are expressed with numbers of disjoint paths P. In this expression, bandwidth measures are numbers of disjoint paths between two router sets; these numbers must match corresponding counts in the original definitions of bandwidth measures. For example, path count associated with BB must equal the BB cut size.
Integration with TCP For applicability in data centers and cloud services, we integrate FatPaths with simple TCP, DCTCP [11], MPTCP [117], and ECN [119] for congestion control. These are less interesting designs and we exclude their description. *Most importantly, all these designs require minor changes to the TCP stack.*
Integration with RDMA As FatPaths fully preserves the semantics of TCP, one could seamlessly use iWARP [53] on top of FatPaths. FatPaths could also be used together with RoCE [15]. RoCE

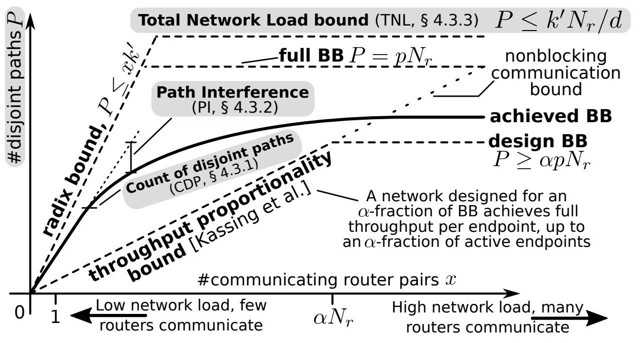
Total Network Load bound (TNL, § 4.3.3) $P \leq k'N_r/d$
full BB $P = pN_r$
nonblocking communication bound
Path Interference (PI, § 4.3.2)
achieved BB
design BB $P \geq \alpha p N_r$
Count of disjoint paths (CDP, § 4.3.1)
radix bound, $P \leq xk'$
throughput proportionality bound [Kassing et al.]
A network designed for an α-fraction of BB achieves full throughput per endpoint, up to an α-fraction of active endpoints
#disjoint paths P
#communicating router pairs x
0
1
Low network load, few routers communicate
αN_r
High network load, many routers communicate

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

has traditionally relied on Ethernet with Priority Flow Control [2] (PFC) for lossless data transfer. However, numerous works illustrate that PFC introduces inherent issues such as head-of-line blocking [57, 88, 100, 158]. Now, the design of FatPaths reduces counts of dropped packets to almost zero ($\leq$ 0.01%) due to flowlet load balancing. With its packet-oriented design and a thin protocol layer over simple Ethernet, *FatPaths could become the basis for RoCE.* Moreover, many modern RDMA schemes (e.g., work by Lu et al. [94]) are similar to NDP in that they, e.g., also use packet spraying. Thus, many of our results may be representative for such RDMA environments. For example, using RDMA on top of FatPaths could provide similar advantages on low-diameter topologies as presented in Figure 2 and 11. We leave this for future work.

Enhancing Infiniband Although we focus on Ethernet, most of the schemes in FatPaths *do not assume anything Ethernet-specific* and they could be *straightforwardly used to enhance IB routing architecture.* For example, all the insights from path diversity analysis, layered routing for multi-pathing, or flowlet load balancing, *could also be used with IB.* We leave these directions for future work.

FatPaths Limitations To facilitate applicability of our work in real-world installations, we discuss FatPaths' limitations. First, as FatPaths addresses low-diameter topologies, it is less advantageous on high-diameter older interconnects such as torus. This is mostly because such networks provide multiple (almost disjoint) shortest paths between most router pairs. Second, FatPaths inherits some of NDP's limitations, namely interrupt throttling. However, similarly to NDP, we alleviate this by assuming that a single CPU core is dedicated to polling for incoming packets. Finally, even if FatPaths delivers decent performance for non-randomized workloads (as illustrated in § 7.2 and in Figure 11), it ensures much higher performance with workload randomization. Yet, as discussed in § 3, this is (1) a standard technique in HPC systems and (2) it is not detrimental for application performance on low-diameter networks that – by design – have very low latencies for all router pairs.

9 Related Work

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

Topologies FatPaths high-performance adaptive routing targets low-diameter networks: Slim Fly [23], Jellyfish [131], Xpander [142], HyperX [5], and Dragonfly [86]. *FatPaths enables these topologies to achieve low latency and high throughput under various traffic patterns (uniform, skewed), and outperform similar-cost fat trees.*

Routing We survey routing schemes in detail in Table 1 and in § 6. *FatPaths is the first one to offer generic and adaptive multi-pathing using both shortest and non-shortest disjoint paths.*

Load Balancing Adaptive load balancing can be implemented using flows [8, 21, 36, 64, 74, 121, 129, 141, 157], flowcells (fixed-sized packet series) [62], and packets [29, 39, 50, 60, 110, 117, 155, 155]. We choose an intermediate level, flowlets (variable-size packet series) [10, 75, 79, 80, 147]. *FatPaths is the first architecture to use load balancing based on flowlets for low-diameter networks.*

Congestion and Flow Control We do not compete with congestion or flow control schemes but instead use them for more performance. *FatPaths can use any such scheme in its design* [9, 11, 12, 18, 19, 30, 60, 63, 67, 71, 93, 99, 101, 117, 145, 159].

Multi-pathing Many works on multi-pathing exist [4, 20, 20, 26, 28, 28, 29, 54, 66, 77, 92, 103, 110, 129, 133, 136, 136, 137, 146, 157]. Our work differs from them all: *it focuses on path diversity in low-diameter topologies, it considers both minimal and non-minimal paths, and it shows a routing scheme using the explored path diversity.*

Network Analyses Some works analyze various properties of low-diameter topologies, for example path length, throughput, and bandwidth [6, 16, 22, 42, 61, 73, 77, 78, 81–83, 91, 131, 139, 140, 142]. *FatPaths offers the most extensive analysis on path diversity so far.*

Encoding Path Diversity Some schemes complement FatPaths in their focus. For example, XPath [65] and source routing [72] could be used together with FatPaths by providing effective means to encode the rich path diversity exposed by FatPaths.

10 Conclusion

We introduce ***FatPaths: a simple, high-performance, and robust routing architecture***. FatPaths enables modern low-diameter topologies to achieve unprecedented performance on Ethernet networks 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 enough non-minimal disjoint paths to avoid congestion. Our path diversity metrics and methodology can be used to analyze other properties of networks.

FatPaths routing stands on three core elements: purified transport, flowlet load balancing, and layered routing. Our theoretical analysis and simulations illustrate that all these elements contribute to the low-latency and high-bandwidth FatPaths design, outperforming very recent fat tree architectures. Even though we focus on Ethernet in this work, most of these schemes – for example adaptive flowlet load balancing and layers – are generic and they could enhance technologies such as RDMA and Infiniband.

Simulations with up to *one million* endpoints show that low-diameter topologies equipped with FatPaths outperform novel superior fat trees [60]. 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.

References

[1] 802.1qbb - priority-based flow control. http://www.ieee802.org/1/pages/802.1bb.html. Retrieved 2015-02-06.

[2] Priority flow control: Build reliable layer 2 infrastructure. http://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white_paper_c11-542809.pdf.

[3] Agarwal, K., Dixon, C., Rozner, E., and Carter, J. B. Shadow macs: scalable label-switching for commodity ethernet. In *Proceedings of the third workshop on Hot topics in software defined networking, HotSDN '14, Chicago, Illinois, USA, August 22, 2014* (2014), pp. 157–162.

[4] Aggarwal, S., and Mittal, P. Performance evaluation of single path and multipath regarding bandwidth and delay. *International Journal of Computer Applications 145*, 9 (2016).

[5] Ahn, J. H., Binkert, N., Davis, A., McLaren, M., and Schreiber, R. S. HyperX: topology, routing, and packaging of efficient large-scale networks. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (2009), ACM, p. 41.

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

[7] Al-Fares, M., Loukissas, A., and Vahdat, A. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Seattle, WA, USA, August 17-22, 2008* (2008), pp. 63–74.

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

[9] Alasmar, M., Parisis, G., and Crowcroft, J. 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* (2018), ACM, pp. 69–71.

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

[11] Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and Sridharan, M. Data center TCP (DCTCP). *ACM SIGCOMM computer communication review 41*, 4 (2011), 63–74.

[12] Alizadeh, M., Yang, S., Sharif, M., Katti, S., McKeown, N., Prabhakar, B., and Shenker, S. pFabric: Minimal near-optimal datacenter transport. *ACM SIGCOMM Computer Communication Review 43*, 4 (2013), 435–446.

[13] Allan, D., Ashwood-Smith, P., Bragg, N., Farkas, J., Fedyk, D., Ouellete, M., Seaman, M., and Unbehagen, P. Shortest path bridging: Efficient control of larger ethernet networks. *IEEE Communications Magazine 48*, 10 (2010).

[14] Arimilli, L. B., Arimilli, R., Chung, V., Clark, S., Denzel, W. E., Drerup, B. C., Hoefler, T., Joyner, J. B., Lewis, J., Li, J., Ni, N., and Rajamony, R. The PERCS high-performance interconnect. In *IEEE 18th Annual Symposium on High Performance Interconnects, HOTI 2010, Google Campus, Mountain View, California, USA, August 18-20,2010* (2010), pp. 75–82.

[15] Association, I. T., et al. Rocev2, 2014.

[16] Azizi, S., Hashemi, N., and Khonsari, A. Hhs: an efficient network topology for large-scale data centers. *The Journal of Supercomputing 72*, 3 (2016), 874–899.

[17] Azodolmolky, S., Wieder, P., and Yahyapour, R. Cloud computing networking: Challenges and opportunities for innovations. *IEEE Communications Magazine 51*, 7 (2013), 54–62.

[18] Bai, W., Chen, L., Chen, K., Han, D., Tian, C., and Sun, W. PIAS: practical information-agnostic flow scheduling for data center networks. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks, HotNets-XIII, Los Angeles, CA, USA, October 27-28, 2014* (2014), pp. 25:1–25:7.

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

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

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

[22] Besta, M., Hassan, S. M., Yalamanchili, S., Ausavarungnirun, R., Mutlu, O., and Hoefler, T. Slim noc: A low-diameter on-chip network topology for high energy efficiency and scalability. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems* (2018), ACM, pp. 43–55.

[23] Besta, M., and Hoefler, T. Slim Fly: A Cost Effective Low-Diameter Network Topology. Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC14).

[24] Bondy, J. A., and Murty, U. S. R. *Graph theory with applications*, vol. 290. Macmillan London, 1976.

[25] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., and Walker, D. P4: programming protocol-independent packet processors. *Computer Communication Review 44*, 3 (2014), 87–95.

[26] Bredel, M., Bozakov, Z., Barczyk, A., and Newman, H. Flow-based load balancing in multipathed layer-2 networks using openflow and multipath-tcp. In *Proceedings of the third workshop on Hot topics in software defined networking* (2014), ACM, pp. 213–214.

[27] Broadcom. Avoiding network polarization and increasing visibility in cloud networks using broadcom smart hash technology. Tech. rep., 2012.

[28] Caesar, M., Casado, M., Koponen, T., Rexford, J., and Shenker, S. Dynamic route recomputation considered harmful. *ACM SIGCOMM Computer Communication Review 40*, 2 (2010), 66–71.

[29] Cao, J., Xia, R., Yang, P., Guo, C., Lu, G., Yuan, L., Zheng, Y., Wu, H., Xiong, Y., and Maltz, D. Per-packet load-balanced, low-latency routing for clos-based data center networks. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies* (2013), ACM, pp. 49–60.

[30] Cardwell, N., Cheng, Y., Gunn, C. S., Yeganeh, S. H., and Jacobson, V. BBR: congestion-based congestion control. *ACM Queue 14*, 5 (2016), 20–53.

[31] Chen, D., Heidelberger, P., Stunkel, C., Sugawara, Y., Minkenberg, C., Prisacari, B., and Rodriguez, G. 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)* (2016), IEEE, pp. 11–21.

[32] Cherkassky, B. V., and Goldberg, A. V. On implementing the push—relabel method for the maximum flow problem. *Algorithmica 19*, 4 (1997), 390–410.

[33] Cheung, H. Y., Lau, L. C., and Leung, K. M. Graph connectivities, network coding, and expander graphs. *SIAM J. Comput. 42*, 3 (2013), 733–751.

[34] Clos, C. A study of non-blocking switching networks. *Bell Labs Technical Journal 32*, 2 (1953), 406–424.

[35] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms.* MIT press, 2009.

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

[37] De Sousa, A. F. Improving load balance and resilience of ethernet carrier networks with ieee 802.1 s multiple spanning tree protocol. In *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on* (2006), IEEE, pp. 95–95.

[38] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271.

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

[40] Dongarra, J. J., Meuer, H. W., Strohmaier, E., et al. Top500 supercomputer sites. *Supercomputer 13* (1997), 89–111.

[41] Faanes, G., Bataineh, A., Roweth, D., Froese, E., Alverson, B., Johnson, T., Kopnick, J., Higgins, M., Reinhard, J., et al. Cray cascade: a scalable HPC system based on a Dragonfly network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (2012), IEEE Computer Society Press, p. 103.

[42] Flajslik, M., Borch, E., and Parker, M. A. Megafly: A topology for exascale systems. In *International Conference on High Performance Computing* (2018), Springer, pp. 289–310.

[43] Floyd, R. W. Algorithm 97: shortest path. *Communications of the ACM 5*, 6 (1962), 345.

[44] Floyd, S. Tcp and explicit congestion notification. *ACM SIGCOMM Computer Communication Review 24*, 5 (1994), 8–23.

[45] Ford, A., Raiciu, C., Handley, M., and Bonaventure, O. RFC 6824, TCP extensions for multipath operation with multiple addresses.

[46] Ford, L. R., and Fulkerson, D. R. Maximal flow through a network. *Canadian journal of Mathematics 8*, 3 (1956), 399–404.

[47] Frantz, P. J., and Thompson, G. O. Vlan frame format, Sept. 28 1999. US Patent 5,959,990.

[48] García, R., Duato, J., and Silla, F. Lsom: A link state protocol over mac addresses for metropolitan backbones using optical ethernet switches. In *Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium on* (2003), IEEE, pp. 315–321.

[49] Gerstenberger, R., Besta, M., and Hoefler, T. Enabling Highly-scalable Remote Memory Access Programming with MPI-3 One Sided. In *Proc. of ACM/IEEE Supercomputing* (2013), SC '13, pp. 53:1–53:12.

[50] Ghorbani, S., Yang, Z., Godfrey, P., Ganjali, Y., and Firoozshahian, A. Drill: Micro load balancing for low-latency data center networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (2017), ACM, pp. 225–238.

[51] Ghorbani, S., Yang, Z., Godfrey, P. B., Ganjali, Y., and Firoozshahian, A. 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).

[52] Gojmerac, I., Ziegler, T., and Reichl, P. Adaptive multipath routing based on local distribution of link load information. In *International Workshop on Quality of Future Internet Services* (2003), Springer, pp. 122–131.

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

[54] Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. VL2: a scalable and flexible data center network. *ACM SIGCOMM computer communication review 39*, 4 (2009), 51–62.

[55] Greenberg, A., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. Towards a next generation data center architecture: scalability and commoditization. In *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow* (2008), ACM, pp. 57–62.

[56] Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., and Lu, S. Bcube: a high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM Computer Communication Review 39*, 4 (2009), 63–74.

[57] Guo, C., Wu, H., Deng, Z., Soni, G., Ye, J., Padhye, J., and Lipshteyn, M. Rdma over commodity ethernet at scale. In *Proceedings of the 2016 ACM SIGCOMM Conference* (2016), ACM, pp. 202–215.

[58] Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., and Lu, S. Dcell: a scalable and fault-tolerant network structure for data centers. In *ACM SIGCOMM Computer Communication Review* (2008), vol. 38, ACM, pp. 75–86.

[59] Gusfield, D. Very simple methods for all pairs network flow analysis. *SIAM J. Comput. 19*, 1 (1990), 143–155.

[60] Handley, M., Raiciu, C., Agache, A., Voinescu, A., Moore, A. W., Antichi, G., and Wojcik, M. Re-architecting datacenter networks and stacks for low latency and high performance. 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), pp. 29–42.

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

[62] He, K., Rozner, E., Agarwal, K., Felter, W., Carter, J. B., and Akella, A. Presto: Edge-based load balancing for fast datacenter networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015* (2015), pp. 465–478.

[63] He, K., Rozner, E., Agarwal, K., Gu, Y. J., Felter, W., Carter, J. B., and Akella, A. AC/DC TCP: virtual congestion control enforcement for datacenter networks. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016* (2016), pp. 244–257.

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

[65] Hu, S., Chen, K., Wu, H., Bai, W., Lan, C., Wang, H., Zhao, H., and Guo, C. Explicit path control in commodity data centers: Design and applications. *IEEE/ACM Transactions on Networking 24*, 5 (2016), 2768–2781.

[66] Huang, X., and Fang, Y. Performance study of node-disjoint multipath routing in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology 58*, 4 (2009), 1942–1950.

[67] Hwang, J., Yoo, J., and Choi, N. Deadline and incast aware tcp for cloud data center networks. *Computer Networks 68* (2014), 20–34.

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

[69] Iwata, A., Hidaka, Y., Umayabashi, M., Enomoto, N., and Arutaki, A. Global open ethernet (goe) system and its performance evaluation. *IEEE Journal on Selected Areas in Communications 22*, 8 (2004), 1432–1442.

[70] Jain, S., Chen, Y., and Zhang, Z.-L. Viro: A scalable, robust and namespace independent virtual id routing for future networks. In *INFOCOM, 2011 Proceedings IEEE* (2011), Citeseer, pp. 2381–2389.

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

[72] Jyothi, S. A., Dong, M., and Godfrey, P. Towards a flexible data center fabric with source routing. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research* (2015), ACM, p. 10.

[73] Jyothi, S. A., Singla, A., Godfrey, P. B., and Kolla, A. Measuring and understanding throughput of network topologies. In *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for* (2016), IEEE, pp. 761–772.

[74] Kabbani, A., Vamanan, B., Hasan, J., and Duchene, F. 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* (2014), ACM, pp. 149–160.

[75] Kandula, S., Katabi, D., Sinha, S., and Berger, A. Dynamic load balancing without packet reordering. *ACM SIGCOMM Computer Communication Review 37*, 2 (2007), 51–62.

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

[77] Kassing, S., Valadarsky, A., Shahaf, G., Schapira, M., and Singla, A. Beyond fat-trees without antennae, mirrors, and disco-balls. 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), pp. 281–294.

[78] Kathareios, G., Minkenberg, C., Prisacari, B., Rodriguez, G., and Hoefler, T. Cost-effective diameter-two topologies: Analysis and evaluation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2015), ACM, p. 36.

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

[80] Katta, N. P., Hira, M., Ghag, A., Kim, C., Keslassy, I., and Rexford, J. 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, Atlanta, GA, USA, November 9-10, 2016* (2016), pp. 155–161.

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

[82] Kawano, R., Nakahara, H., Fujiwara, I., Matsutani, H., Koibuchi, M., and Amano, H. A layout-oriented routing method for low-latency hpc networks. *IEICE TRANSACTIONS on Information and Systems 100*, 12 (2017), 2796–2807.

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

[84] Kim, C., Caesar, M., and Rexford, J. Floodless in seattle: a scalable ethernet architecture for large enterprises. In *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Seattle, WA, USA, August 17-22, 2008* (2008), pp. 3–14.

[85] Kim, J., Dally, W. J., and Abts, D. Flattened butterfly: a cost-efficient topology for high-radix networks. In *ACM SIGARCH Computer Architecture News* (2007), vol. 35, ACM, pp. 126–137.

[86] Kim, J., Dally, W. J., Scott, S., and Abts, D. Technology-driven, highly-scalable dragonfly topology. In *35th International Symposium on Computer Architecture (ISCA 2008), June 21-25, 2008, Beijing, China* (2008), pp. 77–88.

[87] Kornblum, J. Identifying almost identical files using context triggered piecewise hashing. *Digital investigation 3* (2006), 91–97.

[88] Le, Y., Stephens, B., Singhvi, A., Akella, A., and Swift, M. M. Rogue: Rdma over generic unconverged ethernet. In *Proceedings of the ACM Symposium on Cloud Computing* (2018), ACM, pp. 225–236.

[89] Lebiednik, B., Mangal, A., and Tiwari, N. A survey and evaluation of data center network topologies. *arXiv preprint arXiv:1605.01701* (2016).

[90] Leiserson, C. E., Abuhamdeh, Z. S., Douglas, D. C., Feynman, C. R., Ganmukhi, M. N., Hill, J. V., Hillis, W. D., Kuszmaul, B. C., Pierre, M. A. S., Wells, D. S., Wong-Chan, M. C., Yang, S., and Zak, R. The network architecture of the connection machine CM-5. *J. Parallel Distrib. Comput. 33*, 2 (1996), 145–158.

[91] Li, S., Huang, P.-C., and Jacob, B. Exascale interconnect topology characterization and parameter exploration. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)* (2018), IEEE, pp. 810–819.

[92] Li, Y., and Pan, D. 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.

[93] Lu, Y. Sed: An sdn-based explicit-deadline-aware tcp for cloud data center networks. *Tsinghua Science and Technology 21*, 5 (2016), 491–499.

[94] Lu, Y., Chen, G., Li, B., Tan, K., Xiong, Y., Cheng, P., Zhang, J., Chen, E., and Moscibroda, T. Multi-path transport for RDMA in datacenters. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)* (2018), pp. 357–371.

[95] Lui, K.-S., Lee, W. C., and Nahrstedt, K. Star: a transparent spanning tree bridge protocol with alternate routing. *ACM SIGCOMM Computer Communication Review 32*, 3 (2002), 33–46.

[96] Lumsdaine, A., Gregor, D., Hendrickson, B., and Berry, J. Challenges in parallel graph processing. *Parallel Processing Letters 17*, 01 (2007), 5–20.

[97] Malkin, G. Rip version 2-carrying additional information. Tech. rep., 1994.

[98] McKay, B. D., Miller, M., and Širáň, J. A note on large graphs of diameter two and given maximum degree. *J. Comb. Theory Ser. B 74*, 1 (Sept. 1998), 110–118.

[99] Mittal, R., Lam, V. T., Dukkipati, N., Blem, E. R., Wassel, H. M. G., Ghobadi, M., Vahdat, A., Wang, Y., Wetherall, D., and Zats, D. TIMELY: rtt-based congestion control for the datacenter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015* (2015), pp. 537–550.

[100] Mittal, R., Shpiner, A., Panda, A., Zahavi, E., Krishnamurthy, A., Ratnasamy, S., and Shenker, S. Revisiting network support for rdma. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (2018), ACM, pp. 313–326.

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

[102] Moy, J. Ospf version 2. Tech. rep., 1997.

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

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

[105] Niranjan Mysore, R., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., and Vahdat, A. Portland: a scalable fault-tolerant layer 2 data center network fabric. *ACM SIGCOMM Computer Communication Review 39*, 4 (2009), 39–50.

[106] Oran, D. Osi is-is intra-domain routing protocol. Tech. rep., 1990.

[107] Panigrahi, D. Gomory–hu trees. In *Encyclopedia of algorithms*. Springer, 2008, pp. 1–99.

[108] Perlman, R. An algorithm for distributed computation of a spanningtree in an extended lan. In *ACM SIGCOMM Computer Communication Review* (1985), vol. 15, ACM, pp. 44–53.

[109] Perlman, R. Rbridges: transparent routing. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* (2004), vol. 2, IEEE, pp. 1211–1218.

[110] Perry, J., Ousterhout, A., Balakrishnan, H., Shah, D., and Fugal, H. Fastpass: A centralized zero-queue datacenter network. *ACM SIGCOMM Computer Communication Review 44*, 4 (2015), 307–318.

[111] Platform, T. N. THE TUG OF WAR BETWEEN INFINIBAND AND ETHERNET. https://www.nextplatform.com/2017/10/30/tug-war-infiniband-ethernet/.

[112] Prisacari, B., Rodriguez, G., Heidelberger, P., Chen, D., Minkenberg, C., and Hoefler, T. 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* (2014), ACM, pp. 129–140.

[113] Prisacari, B., Rodriguez, G., Jokanovic, A., and Minkenberg, C. Randomizing task placement and route selection do not randomize traffic (enough). *Design Automation for Embedded Systems 18*, 3-4 (2014), 171–182.

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

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

[116] Prisacari, B., Rodriguez, G., Minkenberg, C., and Hoefler, T. Fast pattern-specific routing for fat tree networks. *ACM Transactions on Architecture and Code Optimization (TACO) 10*, 4 (2013), 36.

[117] Raiciu, C., Barré, S., Pluntke, C., Greenhalgh, A., Wischik, D., and Handley, M. 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, Toronto, ON, Canada, August 15-19, 2011* (2011), pp. 266–277.

[118] Raiciu, C., Handley, M., and Wischik, D. RFC 6356, Coupled Congestion Control for Multipath Transport Protocols.

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

[120] Ramakrishnan, K., Floyd, S., and Black, D. RFC 3168, The addition of Explicit Congestion Notification (ECN) to IP.

[121] Rasley, J., Stephens, B., Dixon, C., Rozner, E., Felter, W., Agarwal, K., Carter, J., and Fonseca, R. Planck: Millisecond-scale monitoring and control for commodity networks. In *ACM SIGCOMM Computer Communication Review* (2014), vol. 44, ACM, pp. 407–418.

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

[123] Rezaaifar, E., and Bassin, Y. Equivalent multiple path traffic distribution in communications networks, Mar. 18 2008. US Patent 7,346,706.

[124] Rodeheffer, T. L., Thekkath, C. A., and Anderson, D. C. Smartbridge: A scalable bridge architecture. *ACM SIGCOMM Computer Communication Review 30*, 4 (2000), 205–216.

[125] Sampath, D., Agarwal, S., and Garcia-Luna-Aceves, J. " ethernet on air': Scalable routing in very large ethernet-based networks. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on* (2010), IEEE, pp. 1–9.

[126] Scott, M., Moore, A., and Crowcroft, J. Addressing the scalability of ethernet with moose. In *Proc. DC CAVES Workshop* (2009).

[127] Sehery, W., and Clancy, C. Flow optimization in data centers with clos networks in support of cloud applications. *IEEE Transactions on Network and Service Management 14*, 4 (2017), 847–859.

[128] Seidel, R. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of computer and system sciences 51*, 3 (1995), 400–403.

[129] Sen, S., Shue, D., Ihm, S., and Freedman, M. J. Scalable, optimal flow routing in datacenters via local link balancing. In *Conference on emerging Networking Experiments and Technologies, CoNEXT '13, Santa Barbara, CA, USA, December 9-12, 2013* (2013), pp. 151–162.

[130] Sharma, S., Gopalan, K., Nanda, S., and Chiueh, T.-c. Viking: A multi-spanning-tree ethernet architecture for metropolitan area and cluster networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* (2004), vol. 4, IEEE, pp. 2283–2294.

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

[132] Sinha, S., Kandula, S., and Katabi, D. Harnessing tcp'sburstiness with flowlet switching. *San Diego, November 83* (2004).

[133] Sohn, S., Mark, B. L., and Brassil, J. T. Congestion-triggered multipath routing based on shortest path information. In *Computer Communications and Networks, 2006. ICCCN 2006. Proceedings. 15th International Conference on* (2006), IEEE, pp. 191–196.

[134] Stephens, B., Cox, A., Felter, W., Dixon, C., and Carter, J. PAST: Scalable Ethernet for data centers. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies* (2012), ACM, pp. 49–60.

[135] Subramanian, K. Multi-chassis link aggregation on network devices, June 24 2014. US Patent 8,761,005.

[136] Suchara, M., Xu, D., Doverspike, R., Johnson, D., and Rexford, J. Network architecture for joint failure recovery and traffic engineering. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems* (2011), ACM, pp. 97–108.

[137] Suurballe, J. W., and Tarjan, R. E. A quick method for finding shortest pairs of disjoint paths. *Networks 14*, 2 (1984), 325–336.

[138] Touch, J., and Perlman, R. Transparent interconnection of lots of links (trill): Problem and applicability statement. Tech. rep., 2009.

[139] Truong, N. T., Fujiwara, I., Koibuchi, M., and Nguyen, K.-V. 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 28*, 4 (2016), 989–1001.

[140] Truong, T.-N., Nguyen, K.-V., Fujiwara, I., and Koibuchi, M. Layout-conscious expandable topology for low-degree interconnection networks. *IEICE TRANSACTIONS on Information and Systems 99*, 5 (2016), 1275–1284.

[141] Tso, F. P., Hamilton, G., Weber, R., Perkins, C., and Pezaros, D. P. Longer is better: Exploiting path diversity in data center networks. In *IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, 8-11 July, 2013, Philadelphia, Pennsylvania, USA* (2013), pp. 430–439.

[142] Valadarsky, A., Dinitz, M., and Schapira, M. Xpander: Unveiling the secrets of high-performance datacenters. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks, Philadelphia, PA, USA, November 16 - 17, 2015* (2015), pp. 16:1–16:7.

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

[144] Valiant, L. G. A scheme for fast parallel communication. *SIAM journal on computing 11*, 2 (1982), 350–361.

[145] Vamanan, B., Hasan, J., and Vijaykumar, T. Deadline-aware datacenter tcp (d2tcp). *ACM SIGCOMM Computer Communication Review 42*, 4 (2012), 115–126.

[146] Van der Linden, S., Detal, G., and Bonaventure, O. Revisiting next-hop selection in multipath networks. In *ACM SIGCOMM Computer Communication Review* (2011), vol. 41, ACM, pp. 420–421.

[147] Vanini, E., Pan, R., Alizadeh, M., Taheri, P., and Edsall, T. Let it flow: Resilient asymmetric load balancing with flowlet switching. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017* (2017), pp. 407–420.

[148] Varga, A., et al. The OMNeT++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESM'2001)* (2001), vol. 9, sn, p. 65.

[149] Varga, A., and Hornig, R. 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), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p. 60.

[150] Varga, A., and Hornig, R. INET Framework for OMNeT++. Tech. rep., 2012.

[151] Villamizar, C. Ospf optimized multipath (ospf-omp).

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

[153] Yuan, X., Mahapatra, S., Nienaber, W., Pakin, S., and Lang, M. A new routing scheme for Jellyfish and its performance with HPC workloads. In *Proceedings of the International Conference on High Performance Computing, Networking,*

Storage and Analysis (2013), ACM, p. 36.

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

[155] Zats, D., Das, T., Mohan, P., Borthakur, D., and Katz, R. H. Detail: reducing the flow completion time tail in datacenter networks. In *ACM SIGCOMM 2012 Conference, SIGCOMM '12, Helsinki, Finland - August 13 - 17, 2012* (2012), pp. 139–150.

[156] Zhang, Q., Cheng, L., and Boutaba, R. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications 1*, 1 (2010), 7–18.

[157] Zhou, J., Tewari, M., Zhu, M., Kabbani, A., Poutievski, L., Singh, A., and Vahdat, A. WCMP: weighted cost multipathing for improved fairness in data centers. In *Proceedings of the Ninth European Conference on Computer Systems* (2014), ACM, p. 5.

[158] Zhu, Y., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., Liron, Y., Padhye, J., Raindel, S., Yahia, M. H., and Zhang, M. Congestion control for large-scale rdma deployments. *ACM SIGCOMM Computer Communication Review 45*, 4 (2015), 523–536.

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

APPENDIX

We now provide full discussions, analyses, and results omitted in the main paper body to maintain its clarity.

A Formal Description of Topologies

We first extend the discussion of the considered topologies. Table 5 provides details. Now, each topology uses certain *input parameters* that define the structure of this topology. These parameters are as follows: q (SF), a, h (DF), ℓ (XP), and L, S, K (HX).

A.1 Slim Fly

Slim Fly [23] is a state-of-the-art cost-effective topology for large computing centers that uses mathematical optimization to minimize diameter D for a given radix k while maximizing size N. SF's low diameter ($D = 2$) ensures the lowest latency for many traffic patterns and it reduces the number of required network resources (packets traverse fewer routers and cables), lowering cost, static, and dynamic power consumption. SF is based on graphs approaching the Moore Bound (MB): The upper bound on the number of vertices in a graph with a given D and k'. This ensures full global bandwidth and high resilience to link failures due to good expansion properties. Next, SF is group hierarchical. A group is not necessarily complete but all the groups are connected to one another (with the same number of global links) and form a complete network of groups. We select SF because it is a state-of-the-art design based on optimization that outperforms virtually all other targets in most metrics and represents topologies with $D = 2$.

Associated Parameters N_r and k' depend on a parameter q that is a prime power with certain properties (detailed in the original work [23]). Some flexibility is ensured by allowing changes to p and with a large number of suitable values of the parameter q. We use the suggested value of $p = \lceil k'/2 \rceil$.

A.2 Dragonfly

Dragonfly [86] is a group hierarchical network with $D = 3$ and a layout that reduces the number of global wires. Routers form complete groups; groups are connected to one another to form a complete network of groups with one link between any two groups. DF comes with an intuitive design and represents deployed networks with $D = 3$.

Associated Parameters Input is: the group size a, the number of channels from one router to routers in other groups h, and concentration p. We use the *maximum capacity* DF (with the number of groups $g = ah + 1$) that is *balanced*, i.e., the load on global links is balanced to avoid bottlenecks ($a = 2p = 2h$). In such a DF, a single parameter p determines all others.

A.3 Jellyfish

Jellyfish [131] networks are random regular graphs constructed by a simple greedy algorithm that adds randomly selected edges until no additions as possible. The resulting construction has good expansion properties [24]. Yet, all guarantees are probabilistic and rare degenerate cases, although unlikely, do exist. Even if D can be arbitrarily high in degenerate cases, usually $D < 4$ with much lower d. We select JF as it represents flexible topologies that use randomization and offer very good performance properties.

Associated Parameters JF is flexible. N_r and k' can be arbitrary; we use parameters matching less flexible topologies. To compensate for the different amounts of hardware used in different topologies, we include a Jellyfish network constructed from the same routers for each topology; the performance differences observed between those networks are due to the different hardware and need to be factored in when comparing the deterministic topologies.

A.4 Xpander

Xpander [142] networks resemble JF but have a deterministic variant. They are constructed by applying one or more so called *ℓ-lifts* to a k'-clique G. The ℓ-lift of G consists of ℓ copies of G, where for each edge e in G, the copies of e that connect vertices $s_1, \ldots, s_\ell$ to $t_1, \ldots, t_\ell$, are replaced with a *random matching* (can be derandomized): s_i is connected to $t_{\pi(i)}$ for a random ℓ-permutation π. This construction yields a k'-regular graph with $N = \ell k'$ and good expansion properties. The randomized ℓ-lifts ensure good properties in the expectation. We select XP as it offers the advantages of JF in a deterministically constructed topology.

Associated Parameters We create XP with a single lift of arbitrary ℓ. Such XP is flexible although there are more constraints than in JF. Thus, we cannot create matching instances for each topology. We select $k' \in \{16, 32\}$ and $\ell = k'$, which is comparable to diameter-2 topologies. We also consider $\ell = 2$ with multiple lifts as this ensures good properties [142], but we do not notice any additional speedup. We use $p = \frac{k'}{2}$, matching the diameter-2 topologies.

A.5 HyperX

HyperX [5] is formed by arranging vertices in an L-dimensional array and forming a clique along each 1-dimensional row. Several topologies are special cases of HX, including complete graphs, hypercubes (HCs) [24], and Flattened Butterflies (FBF) [85]. HX is a generic design that represents a wide range of networks.

Associated Parameters An HX is defined by a 4-tuple (L, S, K, p). L is the number of dimensions and $D = L$, S and K are L-dimensional vectors (they respectively denote the array size in each dimension and the relative capacity of links along each dimension). Networks with uniform K and S (for all dimensions) are called *regular*. We only use regular $(L, S, 1, \cdot)$ networks with $L \in \{2, 3\}$. HX with $L = 2$ is about a factor of two away from the MB ($k' \approx 2\sqrt{N_r}$) resulting

	Topology	Hierarchy	Flexibility	Input	N_r	N	k'	p	D	Remarks	Deployed?
deterministic	Slim Fly [23]	group hierarchical	fixed	q	$2q^2$	pN_r		$\left\lceil \frac{k'}{2} \right\rceil$	2	"MMS" variant [23, 98]	unknown
	Dragonfly [86]	group hierarchical	fixed	p	$4p^3 + 2p$	pN_r	$3p - 1$	p	3	"balanced" variant [86] (§3.1)	PERCS [14], Cascade [41]
	HyperX [5]	semi-hierarchical	fixed	S	S^2	pN_r	$2(S-1)$	$\left\lceil \frac{k'}{2} \right\rceil$	2	"regular" variant, 2x-oversubscribed, forms a Flattened Butterfly [85]	unknown
	HyperX [5]	semi-hierarchical	fixed	S	S^3	pN_r	$3(S-1)$	$\left\lceil \frac{k'}{3} \right\rceil$	3	"regular" variant, 2x-oversubscribed, forms a cube	unknown
	Fat tree [90]	semi-hierarchical	fixed	k	$5\left\lfloor \frac{k^2}{4} \right\rfloor$	$p\left\lfloor \frac{k^2}{2} \right\rfloor$	$\frac{k}{2}$	$\left\lceil \frac{k}{2} \right\rceil$	4	2-stage variant (3 router layers)	Many installations
	Complete (clique)	flat	fixed	k'	$k' + 1$	pN_r	k'	k'	1	$D = 1$ HyperX, 2x-oversubscribed	crossbar routers
rand.	Jellyfish [131]	flat	flexible	k', N_r, p	N_r	pN_r	k'	p	n/a	"homogeneous" variant [131]	unknown
	Xpander [142]	flat	semi-flexible	ℓ	$\ell(k'+1)$	pN_r	ℓ	$\left\lceil \frac{k'}{2} \right\rceil$	n/a	Restricted to $\ell = k'$, $D \approx 2$, $p = \left\lceil \frac{k'}{2} \right\rceil$	unknown

Table 5: The considered topologies. "Input" is a column with input parameters used to derive other network parameters.

in more edges than other topologies. Thus, we include higher-diameter variants with k' similar to that of other networks. Now, for full bisection bandwidth (BB), one should set $p = \frac{k'}{2D}$. Yet, since HX already has the highest k' and N_r (for a fixed N) among the considered topologies, we use a higher $p = \frac{k'}{D}$ as with the other topologies to reduce the amount of used hardware. As we do not consider worst-case bisections, we still expect HX to perform well.

A.6 Fat Tree

Fat tree [90] is based on the Clos network [34] with disjoint inputs and outputs and unidirectional links. By "folding" inputs with outputs, a multistage fat tree that connects any two ports with bidirectional links is constructed. We use three-stage FTs with $D = 4$; fewer stages reduce scalability while more stages lead to high D. FT represents designs that are in widespread use and feature excellent performance properties such as full BB and non-blocking routing.
Associated Parameters A three-stage FT with full BB can be constructed from routers with uniform radix k: It connects $k^3/4$ endpoints using five groups of $k^2/4$ routers. Two of these groups, $k^2/2$ routers, form an *edge group* with $k/2$ endpoints. Another two groups form an *aggregation layer*: each of the edge groups forms a complete bipartite graph with one of the aggregation groups using the remaining $k/2$ ports, which are called *upstream*. Finally, the remaining group is called the *core*: each of the two aggregation groups forms a fully connected bipartite graph with the core, again using the remaining $k/2$ upstream ports. This also uses all k ports of the core routers. Now, for FT, it is not always possible to construct a matching JF as N/N_r can be fractional. In this case, we select p and k' such that $k = p + k'$ and $k'/p \approx 4$, which potentially changes N. Note also that for FT, p is the number of endpoints per edge router, while in the other topologies, all routers are edge routers.

A.7 Fully-Connected Graphs

We also consider fully-connected graphs. They represent interesting corner-cases, offer lower bounds on various metrics such as minimal path length, and can be used for validation purposes.
Associated Parameters A clique is defined by a single parameter k', leading to $N_r = k' + 1$. We use $p = k'$ with the same rationale as for the HyperX topologies.

B Efficient Path Counting

Some measures for path diversity are computationally hard to derive for large graphs. Algorithms for all-pairs shortest paths analysis based on adjacency matrices are well known, and we reintroduce one such method here for the purpose of reproducibility. For the disjoint-paths analysis however, all-pairs algorithms exist, but are not commonly known. We introduce a method by Cheung et. al [33] and *we adapt for length-limited edge connectivity computation.*

B.1 Matrix Multiplication for Path Counting

It is well known that for a graph represented as an adjacency matrix, matrix multiplication (MM) can be used to obtain information about paths in that graph. Variations of this include the Floyd-Warshall algorithm [43] for transitive closure and all-pairs shortest paths [128], which use different semirings to aggregate the respective quantities. To recapitulate how these algorithms work, consider standard MM using $\cdot$ and $+$ operators on non-negative integers, which computes the number of paths $n_l(s,t)$ between each pair of vertices.

Theorem 1. *If A is the adjacency matrix of a directed graph $G = (V, E)$, $A_{i,j} = 1$ iff $(i,j) \in E$ and $A_{i,j} = 0$ iff $(i,j) \notin E$, then each cell $i \in V, j \in V$ of $Q = A^l = \underbrace{A \cdot \ldots \cdot A}_{l\ times}$ contains the number of paths from i to j with exactly l steps in G.*

Proof. By induction on the path length l: For $l = 1$, $A^l = A$ and the adjacency matrix contains a 1 in cell i,j iff $(i,j) \in E$, else 0. Since length-1 paths consist of exactly one edge, this satisfies the theorem. Now consider matrices A^p, A^q for $p + q = l$ for which the theorem holds since $p, q < l$. We now prove the theorem also holds for $A^l = A^p \cdot A^q$. Matrix multiplication is defined as

$$(A^p \cdot A^q)_{i,j} = \sum_k A^p_{i,k} \cdot A^q_{k,j}\,. \tag{1}$$

According to the theorem, $A^p_{i,k}$ is the number of length-p paths from i to some vertex k, and $A^q_{k,j}$ is the number of length-q paths from said vertex k to j. To reach j from i via k, we can choose any path from i to k and any from k to j, giving $A^p_{i,k} \cdot A^q_{k,j}$ options. As we regard *all* paths from i to j, we consider *all* intermediate vertices k and count the total number (sum) of paths. This is exactly the count of length-l paths demanded by the theorem, as each length-l path can be uniquely split into a length-p and a length-q segment. □

In the proof we ignored a few details caused by the adjacency matrix representation: first, the adjacency matrix models a directed graph. We can also use the representation for undirected graphs by making sure A is symmetrical (then also A^l is symmetrical). Adjacency matrices contain the entry $A_{i,j} = 0$ to indicate $(i,j) \notin E$ and $A_{i,j} = 1$ for $(i,j) \in E$. By generalizing $A_{i,j}$ to be the number of length-1 paths (= number of edges) from i to j as in the theorem, we can also represent multi-edges; the proof still holds.

Finally, the diagonal entries $A_{i,i}$ represent self-loops in the graph, which need to be explicitly modeled. Note that also $i = j$ is allowed above and the intermediate vertex k can be equal to i and/or j. Usually self-loops should be avoided by setting $A_{i,i} = 0$. Then $A^l_{i,i}$ will be the number of cycles of length l passing through i, and the paths counted in $A_{i,j}$ will include paths containing cycles. These cannot easily be avoided in this scheme[4]. For most measures, e.g., shortest paths or disjoint paths, this is not a problem, since paths containing cycles will naturally never affect these metrics.

On general graphs, the algorithms outlined here are not attractive since it might take up to the maximum shortest path length D iterations to reach a fixed point, however since we are interested in low-diameter graphs, they are practical and easier to reason about than the Floyd-Warshall algorithms.

B.1.1 Matrix Multiplication for Routing Tables As another example, we will later use a variation of this algorithm to compute next-hop tables that encode for each source s and each destination t which out-edge of s should be used to reach t. In this algorithm, the matrix entries are sets of possible next hops. The initial adjacency matrix will contain for each edge in G a set with the out edge index of this edge, otherwise empty sets. Instead of summing up path counts, we union the next-hop sets, and instead of multiplying with zero or one for each additional step, depending if there is an edge, we retain the set only if there is an edge for the next step. Since this procedure is not associative, it cannot be used to form longer paths from shorter segments, but it works as long as we always use the original adjacency matrix on the right side of the multiplication. The correctness proof is analogous to the path counting procedure.

B.2 Counting Disjoint Paths

The problem of counting all-pairs disjoint paths per pair is equivalent to the all-pairs edge connectivity problem which is a special case of the all-pairs max flow problem for uniform edge capacities. It can be solved using a spanning tree (*Gomory-Hu tree* [107]) with minimum $s-t$-cut values for the respective partitions on the edges. The minimum $s-t$ cut for each pair is then the minimum edge weight on the path in this tree, which can be computed cheaply for all pairs. The construction of the tree requires $O(N_r)$ $s-t$-cuts, which cost $O(N_r^3)$ each (e.g., using the Push-Relabel scheme [32]).

Since we are more interested in the max flow values, rather than the min-cut partitions, a simplified approach can be used: while the Gomory-Hu tree has max flow values and min cut partitions equivalent to the original graph, a *equivalent flow tree* [59] only preserves the max flow values. While constructing it needs the same number of max-flow computations, these can be performed on the original input graph rather than the contracted graphs of Gomory-Hu, which makes the implementation much easier.

For length-restricted connectivity, common max-flow algorithms have to be adapted to respect the path length constraint. The Gomory-Hu approach does not work, since it is based on the principle that the distances in the original graph do not need to be respected. We implemented an algorithm based on the Ford-Fulkerson method [46], using BFS [35], which is not suitable for an all-pairs analysis, but can provide results for small sets of samples.

The spanning-tree based approaches only work for undirected graphs, and solve the more general max-flow problem. There are also algorithms that only solve the edge-connectivity problem, using completely different approaches. Cheung et. al [33] propose an algorithm based on linear algebra which can compute all-pairs connectivity in $O(|E|^\omega + |V|^2 k'^\omega)$; $\omega \leq 3$ is the exponent for matrix-matrix multiplication. For our case of $k' \approx \sqrt{N_r}$ and naive matrix inversion, this is $O(N_r^{4.5})$ with massive space use, but there are many options to use sparse representations and iterative solvers, which might enable $O(N_r^{3.5})$. Due to their construction, those algorithms also allow a limitation of maximum path length (with a corresponding complexity reduction) and the heavy computations are built on well-known primitives with low constant overhead and good parallel scaling, compared to classical graph schemes.

B.3 Deriving Edge Connectivity

This scheme is based on the ideas of Cheung et. al. [33]. First we adapt the algorithm for vertex connectivity, which allows lower space- and time complexity than the original algorithm and might also be easier to understand. The original edge-connectivity algorithm is obtained by applying it to a transformed graph.[5] We then introduce the path-length constraint by replacing the exact solution obtained by matrix inversion with an approximated one based on iterations, which correspond to incrementally adding steps. The algorithm is randomized in the same way as the original is; we will ignore the probability analysis for now, as the randomization is only required to avoid degenerate matrices in the process and allow the use of a finite domain. The domain $\mathbb{F}$ is defined to be a finite field of sufficient size to make the analysis work and allow a real-world implementation; we can assume $\mathbb{F} = \mathbb{R}^+$ for the algorithm itself.

First, we consider a *connection matrix*, which is just the adjacency matrix with random coefficients for the edges:

$$K_{i,j} = \begin{cases} x \in \mathbb{F} \text{ u.a.r.} & \text{iff } (i,j) \in E \\ 0 & \text{else .} \end{cases} \tag{2}$$

In the edge-connectivity algorithm we use a much larger adjacency matrix of a transformed graph here (empty rows and columns could be dropped, leaving an $|E| \times |E|$ matrix, but our implementation does not do this since the empty rows and columns are free in a sparse matrix representation):

$$K'_{(i,k),(k,j)} = \begin{cases} x \in \mathbb{F} \text{ u.a.r.} & \text{iff } (i,k) \in E \wedge (k,j) \in E \\ 0 & \text{else .} \end{cases} \tag{3}$$

Now, we assign a vector $F_i \in \mathbb{F}^k$, where k is the maximum vertex degree, to each vertex i and consider the system of equations

[4] Setting $A^l_{i,i} = 0$ before/after each step does not prevent cycles, since a path from i to k might pass j, causing a cycle, and we cannot tell this is the case without actually recording the path.

[5] Vertex-connectivity, defined as the minimum size of a cut set $c_{st} \subset V \setminus \{s, t\}$ of vertices that have to be removed to make s and t disconnected, is not well defined for neighbors in the graph. The edge-connectivity algorithm avoids this problem, but this cannot be generalized for vertex-connectivity.

defined by the graph: the value of each vertex shall be the linear combination of its neighbors weighted by the edge coefficients in K. To force a non-trivial solution, we designate a source vertex s and add pairwise orthogonal vectors to each of its neighbors. For simplicity we use unit vectors in the respective columns of a $k \times |V|$ matrix P_s (same shape as F). So, we get the condition

$$F = FK + P_s \,. \tag{4}$$

This can be solved as

$$F = -P_s(\mathbb{I} - K)^{-1} \,. \tag{5}$$

The work-intensive part is inverting $(\mathbb{I} - K)$, which can be done explicitly and independently from s, to get a computationally inexpensive all-pairs solution, or implicitly only for the vectors in P_s for a computationally inexpensive single-source solution. To compute connectivity, we use the following theorem. The scheme outlined in the following proof counts vertex-disjoint paths of any length.

Theorem 2. *The size of the vertex cut set c_{st} from s to t equals $\mathrm{rank}(FQ_t)$, where $F = -P_s(\mathbb{I} - K)^{-1}$ and Q_t is a $|V| \times k$ permutation matrix selecting the incoming neighbors of t.*

Proof. First, $c_{st} \le \mathrm{rank}(FQ_t)$, because all non-zero vectors were injected around s and all vectors propagated through the cut set of c_{st} vertices to t, so there cannot be more than c_{st} linearly independent vectors near t. Second, $c_{st} \ge \mathrm{rank}(FQ_t)$, because there are c_{st} vertex-disjoint paths from s to t. Each passes through one of the c_{st} outgoing neighbors of s, which has one of the linearly independent vectors of P_s assigned (combined with potentially other components). As there is a path from s to t trough this vertex, on each edge of this path the component of P_s will be propagated to the next vertex, multiplied by the respective coefficient in K. So, at t each of the paths will contribute one orthogonal component. □

To count length-limited paths instead, we simply use an iterative approximation of the fixed point instead of the explicit solution. Since we are only interested in the ranks of sub-matrices, it is also not necessary to actually find a precise solution; rather, following the argument of the proof above, we want to follow the propagation of linearly independent components through the network. The first approach is simply iterating Equation 4 from some initial guess. For this guess we use zero vectors, due to P_s in there we still get nontrivial solutions but we can be certain to not introduce additional linearly dependent vectors:

$$\begin{aligned} F_0 &= \left(0\right) \quad (k \times |V|) \\ F_l &= F_{l-1}K + P_s \,. \end{aligned} \tag{6}$$

This iteration still depends on a specific source vertex s. For an all-pairs solution, we can iterate for all source vertices in parallel by using more dimensions in the vectors; we set $k = |V|$. Now we can assign every vertex a pairwise orthogonal start vector, e.g., by factoring out P_s and selecting rows by multiplying with P_s in the end. The intermediate products are now $|V| \times |V|$ matrices, and we add the identity matrix after each step. Putting all together gives

$$c_{st} = \mathrm{rank}(P_s \underbrace{(((K + \mathbb{I}) \cdot K + \mathbb{I}) \cdot \ldots)}_{l \text{ times, precomputed}} Q_t) \,. \tag{7}$$

The total complexity includes the $O\left(|V|^3 l\right)$ operations to precompute the matrix for a maximum path length of l and $O\left(|V|^2 k^3\right)$ operations for the rank operations for all vertex pairs in the end, which will be the leading term for the $k = O\left((\sqrt{|V|}\right)$ (diameter 2) undirected graphs considered here, for a total of $O\left(|V|^{3.5}\right)$.

For the edge connectivity version, we use the edge incidence connection matrix K', and select rows and columns based on edge incidence, instead of vertex adjacency. Apart from that, the algorithm stays identical, but the measured cut set will now be a cut set of edges, yielding edge connectivity values. However, the algorithm is more expensive in terms of space use and running time: $O\left(|E|^3 l\right)$ to precompute the propagation matrix.

C Details of Evaluation Setup

We now provide details of the evaluation setup.

C.1 Behavior of Flows in OMNet++

First, we use Figure 19 to illustrate the behavior of long flows (2MB) in the pFabric web search distribution in response to the flow arrival rate λ (flows per endpoint per second) on a 60-endpoint crossbar (tail limited to 90% due to the low sample size). The left plot shows how the per-flow throughput decreases beyond $\lambda = 250$, which is a sign of network saturation; the right figure shows three samples of completion time distributions for low, moderate, and high loads.

C.2 Behavior of Flows in htsim

In htsim simulations, on the star network as well as the baseline $2x$-oversubscribed fat tree, we observe better performance compared to the OMNet++ TCP simulations. This leads to a lower network load, which would be misleading in topology comparisons. Therefore, we use $\lambda = 300$, where the system starts to show congestion at the network level. At lower λ, we only observe endpoint congestion (crossbar (Star) results equal fat tree results), while at higher λ, the FCTs increase beyond the 2× expected from the oversubscription: the lower service rate leads to more concurrent flows, decreasing the throughput per flow (see Figure 20).

C.3 Selection of TCP Parameters

TCP retransmissions on packet loss are controlled by two separate timeouts: one for the initial SYN and SYNACK handshake packets, and one that is used during the flow. Both are guarded by an upper and lower limit. For the handshake timeout, the lower limit is used in the beginning, increasing it up to the upper limit on retries. For the normal limit, it is adapted in response to the measured RTT but limited by the bounds, initially starting from a high value.

Since we usually do not see lost SYN packets, we did not optimize the handshake timeouts. Most of the time, they simply have no effect at all. We did optimize the normal retransmission timeouts, and observed that limiting the lower bound can decrease performance at high load, while the upper bound does not have much impact (again, this is because it is unlikely that a packet is lost before an RTT estimate is produced, so this parameter is not used at all). The value of $200\mu s$ for the RTO lower bound is fairly high and can lead to performance degradation, but it also models a limited timer granularity on the endpoints, which makes low timeouts unrealistic. In the usual workload model considered in this work, packet loss rates

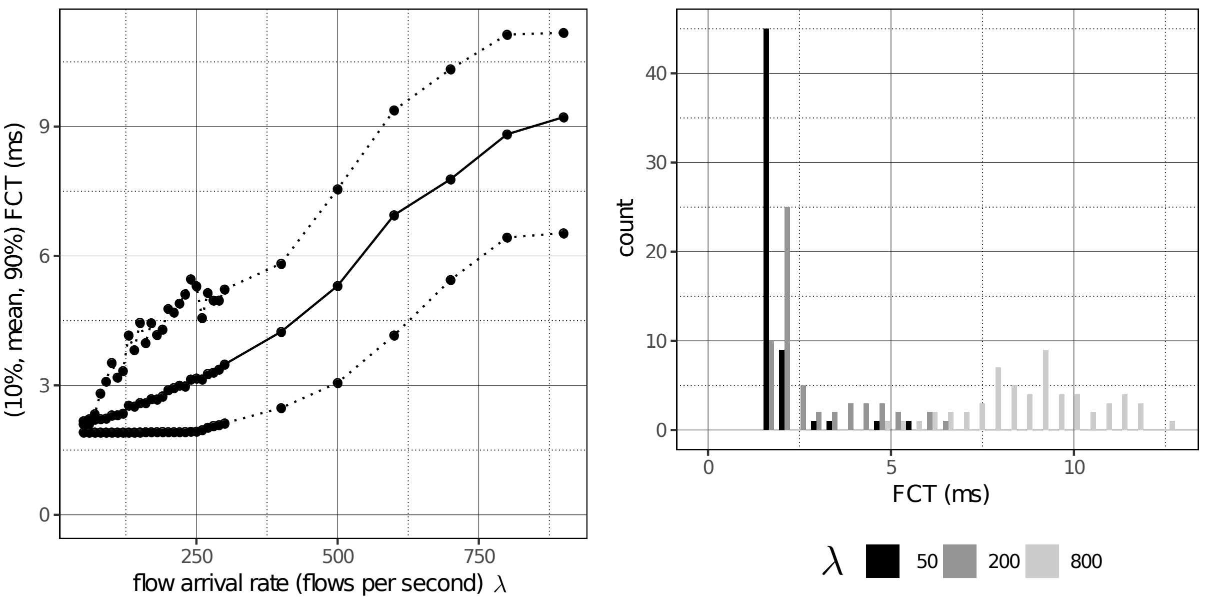

Figure 19: Behavior of long flows (2MB) in the pFabric web search distribution in response to the flow arrival rate λ (flows per endpoint per second) on a 60-endpoint crossbar (tail limited to 90% due to the low sample size).

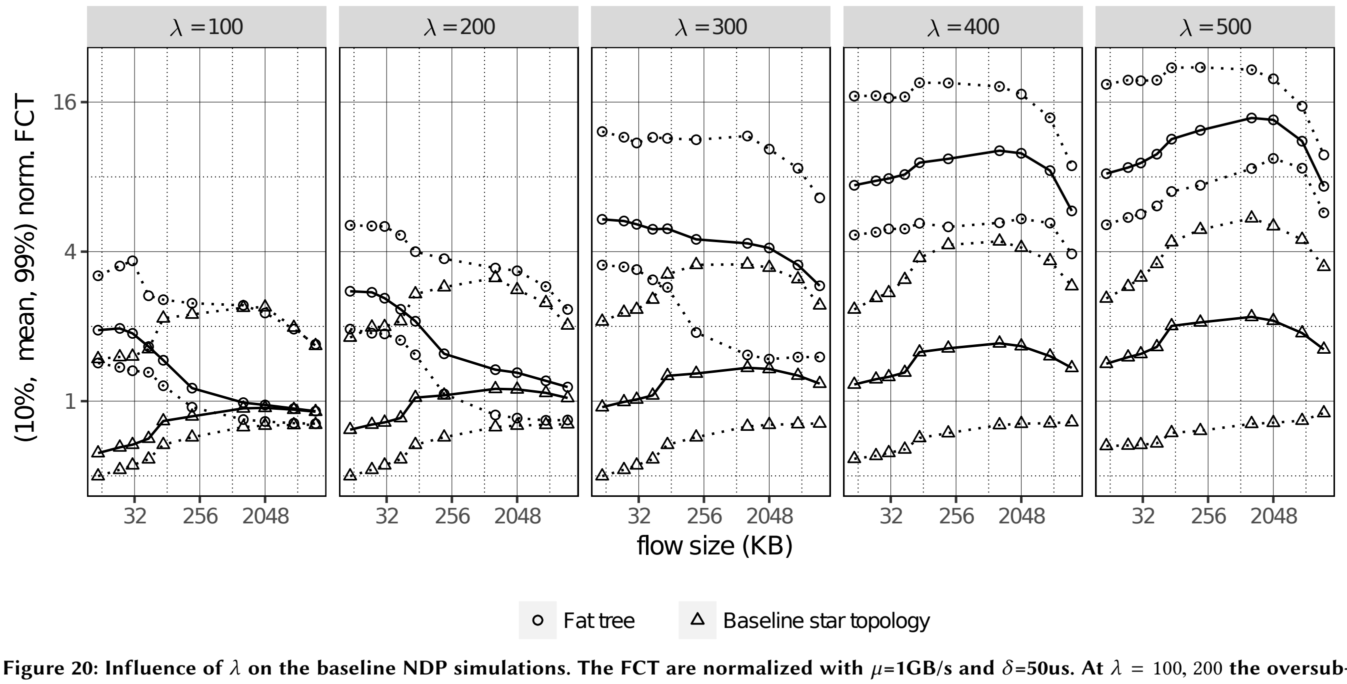

Figure 20: Influence of λ on the baseline NDP simulations. The FCT are normalized with μ=1GB/s and δ=50us. At λ = 100, 200 the oversubscription is not noticeable (similar long-flow behavior for crossbar and fat tree), which indicates that the total network load is still low. The difference in short-flow FCT is due to the drastically different network diameters.

are low enough that the RTO does not have any measurable impact, as long as the timeouts are not very high (with the INET default value of 1s, a single flow experiencing a RTO can influence the mean significantly). The TCP retransmission parameters become more relevant if very sparse, and therefore incomplete, layers are used, where packets are lost not only due to congestion but also due to being non-routable. However, in this case we use feedback via ICMP to trigger an immediate retransmission on a different layer, therefore the RTO limits also have no significant impact in this scenario.