

# EvalNet: A Practical Toolchain for Generation and Analysis of Extreme-Scale Interconnects

Maciej Besta<sup>1\*</sup>, Patrick Iff<sup>1</sup>, Marcel Schneider<sup>1</sup>, Nils Blach<sup>1</sup>, Alessandro Maissen<sup>1</sup>,  
Salvatore Di Girolamo<sup>1</sup>, Jens Domke<sup>2</sup>, Jascha Krattenmacher<sup>1</sup>, Ankit Singla<sup>1</sup>,  
Kartik Lakhotia<sup>3</sup>, Laura Monroe<sup>4</sup>, Fabrizio Petrini<sup>3</sup>, Robert Gerstenberger<sup>1</sup>, Torsten Hoefler<sup>1</sup>

<sup>1</sup>Department of Computer Science, ETH Zurich; <sup>2</sup>RIKEN Center for Computational Science (R-CCS); <sup>3</sup>Intel Labs; <sup>4</sup>Los Alamos National Laboratory, \*Corresponding author

## ABSTRACT

The diversity of communication paths in a network—especially non-minimal paths—is a key enabler of performance at extreme scales. We present EvalNet, a toolchain for scalable generation and analysis over 25 important network topologies, such as Slim Fly, PolarFly, and Orthogonal Fat Trees, with a strong focus on path diversity metrics. EvalNet provides an extensive and fine-grained analysis of shortest and non-shortest paths, including their multiplicities, lengths, and interference. It supports exact measurement and visualization of bandwidth and throughput between every router pair, enabling unprecedented insight into routing potential. EvalNet also includes detailed models for construction cost and power consumption, and interfaces seamlessly with established simulators, which we tune to support large-scale evaluations on low-cost hardware. Using EvalNet, we deliver the widest and most comprehensive path diversity study to date, demonstrating how path diversity underpins throughput and scalability, and facilitating progress towards new frontiers in extreme-scale network design.

## CCS CONCEPTS

• **Networks** → **Network performance evaluation; Network performance analysis; Network experimentation; Data center networks; Packet-switching networks.**

**Code:** <https://github.com/spcl/EvalNet>

## 1 INTRODUCTION

The interconnection network plays an important role in today’s large-scale computing systems [13, 15, 28, 31, 32, 42, 45, 78, 100]. Large networks with tens of thousands of nodes are deployed in warehouse-sized HPC and data centers [37, 72]. Future supercomputers as well as mega data centers will require even larger scales with hundreds of thousands of servers.

A modern network topology must satisfy many stringent requirements related to different aspects of the interconnect design, such as high performance or low construction cost. However, the evaluation of all these aspects is challenging. First, there are many networks that must be considered for comparison when designing a new topology, for example Fat Tree [69], Hypercube [29], Torus [29], Dragonfly [64], Slim Fly [15], Xpander [2], or Jellyfish [95] – to name only a few. On top of that, many of these baseline networks, for example Xpander or Orthogonal Fat Trees, do not have straightforward construction algorithms, making it challenging and time-consuming to compare to these baselines.

Similarly, there are many performance measures that should be considered for overall evaluation, and these measures often require different tools. A core challenge in evaluating interconnects is the analysis of path diversity, which has emerged as a critical enabler of performance and scalability in modern networks [17]. As networks grow to extreme sizes, there is a growing need for fine-grained and scalable tools that can assess not only the existence of diverse paths but also their structure, interference, and utility for multipath routing. In parallel, several complementary performance metrics must also be considered, such as throughput and latency. Throughput is ideally evaluated using packet-level simulators, which offer high accuracy but often do not scale to large topologies. Flow-based simulations scale more easily, but they tend to produce less accurate predictions [23]. Furthermore, additional aspects like construction cost and power consumption must be analyzed. Unfortunately, the tools supporting these various metrics often rely on different formats for specifying networks, which significantly complicates integrated and efficient evaluation.

To address all these issues, we propose EvalNet: a publicly available toolchain that facilitates the integrated and comprehensive evaluation of networks. EvalNet consists of several modules, some of which were developed from scratch, while others are interfaces to existing tools which facilitate seamless operation. First, EvalNet enables generation of a very large number of network topologies. We cover the state-of-the-art class of low-diameter topologies developed by optimizing the network structure towards the Moore Bound (Slim Fly [15], PolarFly [65], BundleFly [68], Xpander [100]), Dragonfly topologies (fully-connected Dragonfly [64], Cascade Dragonfly [34]), random networks (Jellyfish [95]), torus related topologies (Mesh, Torus, Hypercube [29], Express Mesh [54], HyperX [2], Flattened Butterfly [63], Tofu [3, 4]), indirect topologies (Fat Tree [69], Multi-Layer Full-Mesh [60], Orthogonal Fat Tree [60], eXtended Generalized Fat Tree [80], k-ary n-tree [85]), and Kautz networks (Undirected Kautz [70], Arrangement Network [30]).

Second, we observe that the high performance of network workloads is becoming increasingly dependent on effectively harnessing various forms of path diversity, particularly through non-minimal multipathing [17]. This is especially critical in state-of-the-art low-diameter topologies, such as Slim Fly and PolarFly, which typically offer limited shortest-path diversity but compensate with a rich structure of non-minimal paths. In such networks, the ability to balance load and avoid congestion hinges on the capacity to exploit this non-shortest-path structure. To address this need, EvalNet provides a comprehensive suite of fine-grained tools for analyzing path diversity. It supports the computation of shortest and non-shortest path multiplicities, detailed distributions of path

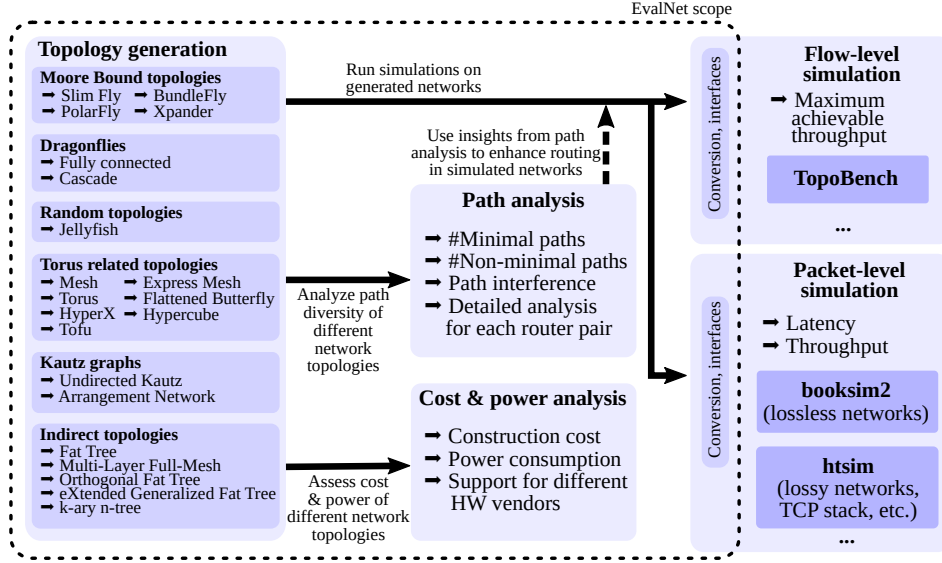


Figure 1: An overview of EvalNet.

lengths, and interference-aware metrics that capture how routes interact with each other. EvalNet also includes visualization and measurement tools that expose the exact bandwidth and multi-pathing potential between every pair of routers, enabling both coarse-grained insights and fine-grained optimization of routing and scheduling schemes [66]. All path diversity analyses are designed to scale to very large topologies, supporting networks with more than 100,000 servers, thus making them practical for evaluating future-generation systems.

Moreover, EvalNet seamlessly interfaces with different tools for network simulation. This includes the flow-level evaluation tool TopoBench [57] that enables approximating throughput for arbitrary traffic patterns and comes with a specially designed worst-case scenario that puts particularly large stress on the interconnect. EvalNet also harnesses two complementary packet-level simulators (htsim [92] and Booksim2 [56]), which target, respectively, lossy and lossless architectures<sup>1</sup>. We tune these simulators to support large-scale evaluations on low-cost hardware, achieving simulations of more than 1 million endpoint networks on a commodity PC. Finally, EvalNet also comes with a construction cost and power consumption assessment module. The user can parametrize this module with arbitrary cost and power models for both routers and cables, considering arbitrary vendors and architectures such as Ethernet or InfiniBand.

Finally, we use EvalNet to conduct the widest-ever comparative analysis of interconnect topologies, focusing on the path diversity measures, but also combining them with classical metrics that include latency, global bandwidth, construction cost, and power consumption. We investigate path diversity in-depth because other measures (latency, bandwidth under flow/packet simulations, fault-tolerance) have already been extensively researched. However, path diversity has only been analyzed in a limited scope. We provide a total of 55 rich plots for path diversity metrics, combined with

observations, analyses, and insights that could be used to develop better routing strategies.

## 2 EVALNET TOOLCHAIN

We illustrate the overall structure of EvalNet in Figure 1. First, the user selects a topology to be generated; it can be any of the networks described in Section 3; the user can also provide a script generating their own interconnect. The user can then use a module for analyzing the path diversity of a network, which can subsequently be used to enhance routing schemes in simulations (an overview is in Section 4.1, details and evaluation are in Section 5). The generated networks can also be evaluated using either flow or packet-based simulations (overview is in Section 4.2, details and evaluation are in Section 6 and 7, respectively). These simulations are conducted by external tools (TopoBench [57] for flow-based and htsim [92] as well as Booksim2 [56] for packet-based simulations of lossy and lossless networks, respectively); we provide input/output conversion routines to integrate these tools seamlessly. Finally, the user can use our cost and power models to assess the construction cost or static power consumption (Section 4.3).

We now proceed to outline these factors (Sections 3–4) and then detail each of them and provide extensive evaluations (Sections 5–7). We will finalize with a comprehensive summary comparing all considered interconnects and measures (Section 8).

## 3 RELEVANT NETWORK TOPOLOGIES

We first describe network topologies included in EvalNet – see also Figure 2. The most important symbols are listed in Table 1.

### 3.1 Network Model

We model an interconnection network as an undirected graph  $G = (V, E)$ ;  $V$  and  $E$  are sets of vertices and edges. A vertex models a router<sup>2</sup> ( $|V| = N_r$ ). An edge models a full-duplex inter-router

<sup>1</sup>EvalNet does not compete with existing simulators as it does not introduce a new simulator. Instead, it tunes two of these simulators and harnesses them within its full comprehensive pipeline.

<sup>2</sup>We abstract away HW details and use “router” for both L2 switches and L3 routers.

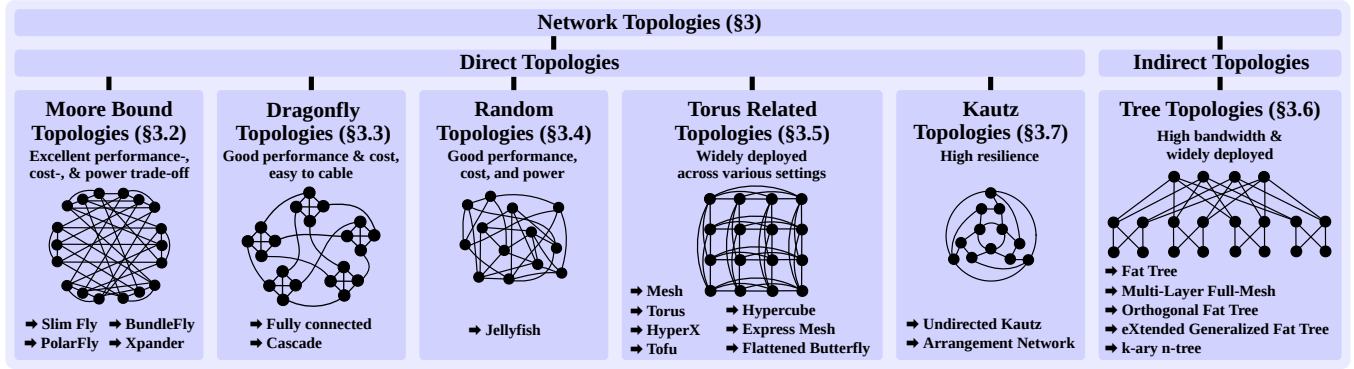


Figure 2: An overview of network topologies offered by default in EvalNet. New topologies can be straightforwardly added to the pipeline.

$V, E$	Sets of vertices/edges (routers/links, $V = \{0, \dots, N_r - 1\}$ ).
$N, N_r$	#endpoints and #routers in the network ( $N_r =  V $ ).
$p, k'$	#endpoints attached to a router, #channels to other routers.
$k$	Router radix ( $k = k' + p$ ).
$D, d$	Network diameter and the average path length.
$c_l(A, B)$	Count of (at most $l$ -hop) disjoint paths between router sets $A, B$ .
$c_{\min}(s, t), l_{\min}(s, t)$	Diversity and lengths of minimal paths between routers $s$ and $t$ .
$I_{ac,bd}$	Path interference between pairs of routers $a, b$ and $c, d$ .
$\lambda, v$	Flow arrival rate [flows/s] and the flow volume [bytes].

Table 1: The most important symbols used in this work.

physical link. Servers (endpoints) are modeled implicitly. There are  $N$  servers in total, and  $p$  endpoints are attached to each router (*concentration*). There are  $k'$  channels between any two routers (*network radix*).  $k = p + k'$  is the total router *radix*. The diameter is  $D$  while the average path length is  $d$ .  $\lambda$  and  $v$  are the flow arrival rate [flows/s] and the flow volume [bytes] of a used workload.

While EvalNet supports arbitrary configurations of  $p$ ,  $k$ , and  $k'$ , in the experiments, we usually set  $p = \frac{k'}{d}$ , as it maximizes throughput while minimizing congestion and network cost (assuming a random uniform traffic pattern) [17]. We also usually 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. We assume router-based topologies hereafter, i.e., no direct channels between any two endpoints.

### 3.2 Moore Bound Topologies

A recent class of low-diameter topologies explicitly optimizes the network structure towards the Moore Bound [15]. This maximizes the number of routers and endpoints for a fixed diameter and radix, minimizing cost and power consumption. These topologies are important as they have achieved new frontiers in performance, cost, and power consumption.

**Slim Fly** (SF) [15], **PolarFly** (PF) [65], and **PolarStar** [67] belong to this class, they are based on the underlying McKay-Miller-Širáň and Erdős-Rényi polar graphs, respectively. **Spectralfly** (SpF) [106] is based on the Ramanujan graph construction of Lubotzky, Phillips, and Sarnak (LPS). Moreover, a  $\tau$ -regular **Xpander** (XP) [100] constructions starts with a complete graph of  $\tau + 1$  vertices, and is then repeatedly “lifted” ( $\frac{N_r}{\tau+1}$  times) to obtain the final graph with good expansion properties. Finally, **BundleFly** (BF) [68] is using SF

together with a Paley graph. The routers in both graphs are linked via a multi-star product based on a bijection.

### 3.3 Dragonfly Topologies

The **Fully-Connected Dragonfly** (DF) [64] is comprised of  $g$  groups of nodes internally interconnected in an all-to-all fashion. The topology among groups is an all-to-all, too. A **Cascade Dragonfly** (C-DF) [34] alters the original Dragonfly by assuming a  $16 \times 6$  2D HyperX for intra-group and an all-to-all graph for inter-group connectivity. **MegaFly** (MF) [36] is another Dragonfly variant. These networks are relevant as they come with good performance and cost, albeit achieving worse values for these metrics than low-diameter networks, but have been found to be easier to cable. They also offer relatively easy scheduling: their modular structure facilitates achieving data locality for workloads [99].

### 3.4 Random Topologies

Fully random networks are infeasible due to the unbounded radix. The **Jellyfish** [95] is a random graph with a pre-selected router radix  $k$ . We support these networks as they come with advantageous performance, cost, and power consumption [17, 95].

### 3.5 Torus Related Topologies

We consider traditional **Mesh** (MH), **Torus** (TR), and **Hypercube** (HC) topologies. We also incorporate their more recent enhancements: **HyperX** (HX) [2] (each router in  $n$ -D grid connects to all other routers within the same dimension), **Express Meshes** (EM) [54] (cut-down versions of the HyperX), **Flattened Butterfly** (FBF) [63] (a 2D variation of HyperX), and **Tofu** (TF) [3, 4] (a 6D torus variant with inner 3D mesh/tori networks embedded as virtual routers). These networks have been widely deployed across different settings and thus are considered in EvalNet.

### 3.6 Indirect Topologies

In indirect networks, some routers attach endpoints while other routers only forward packets. A **Fat Tree** (FT) [69] is a binary tree, with either higher bandwidth links towards the root or multiple parallel links towards the root to avoid communication bottlenecks. It is one of the most widely researched and used network topologies [5–9, 11, 21, 22, 42–44, 48, 49, 61, 62, 71, 73–76, 81, 82, 84, 91,

Evaluation metrics (§4)	Path diversity measures (§4.1)	Count of minimal paths	Shortest path lengths	Facilitates reasoning about the expected latency of a topology with shortest-path minimum routing, for different router pairs.
			Shortest path multiplicity	Enables taking informed decisions on whether to use multipathing over shortest paths (e.g., with ECMP), or whether one has to use non-minimal multipath routing.
		Count of non-minimal paths	Non-shortest path lengths	It facilitates reasoning about the expected latency of a topology with non-shortest path routing, for different router pairs, and offers hints on routing design.
			Non-shortest path multiplicity	Enables taking informed decisions on how to use non-minimal multipath routing.
		Path interference		Facilitates reasoning about the performance of both minimal and non-minimal routing, and of scheduling potential processes to different servers.
	Cost and power (§4.3)	Total network load (TNL)		Acts as an upper bound on the number of flows that a network can maintain without congestion.
		Construction cost		Enables comparisons between different baselines and insights into construction cost, for different cost models & equipment.
	Throughput and latency (§4.2)	Power consumption		Enables rudimentary comparisons between different baselines and preliminary insights into power consumption.
		Throughput (saturation for a given workload)		Measures how much load a network can accommodate, before its latency grows uncontrollably.
		Maximum achievable throughput (MAT)		Measures how much load a network can accommodate for a very adversarial traffic pattern, before its latency grows uncontrollably.
		Network latency		Measures the end-to-end communication delay without congestion effects in the network.
		Average path length		Provides insights into the impact of a routing design decision on the number of links that must be traversed.

Figure 3: An overview of evaluation measures of network topologies offered by default in EvalNet. New measures can be straightforwardly added to the pipeline.

101, 102, 104, 109, 111]. The **k-ary n-tree** (AT) [85] avoids infeasible high-radix roots of FT by using the same number of routers per tree-level and connecting them with increasing strides towards the top. An **eXtended Generalized Fat Tree** (xGFT) [80] is a k-ary n-tree variant which allows reducing deployment cost. **Orthogonal Fat-Trees** (OFT) [60] are based on k-ary 2-trees; the wiring pattern between the two levels is modified to increase scalability. Finally, a **Multi-Layer Full-Mesh** (MLFM) [60] consists of multiple 2D meshes (incl. diagonal connections) stacked together, but instead of direct links it uses global routers to link meshes. EvalNet also supports indirect networks as they offer very high bandwidth; FTs are also widely deployed.

### 3.7 Kautz Topologies

Both **Undirected Kautz** (KA) [70] and **Arrangement Network** [30] have strings assigned to each router. Then, links between routers are determined by similarities of the corresponding router strings. Kautz networks offer high resilience properties and offer an interesting alternative to other classes of interconnects.

### 3.8 Other Topologies

EvalNet can be easily extended to model and evaluate interconnects from other domains, such as on-chip networks between different cores [1, 14], chiplets [50–52], or FPGA interconnects [20], due to its modular and flexible architecture. The toolchain’s seamless integration with external simulators for packet-level and flow-level analysis ensures that new topologies, such as those designed for heterogeneous [24] or spatial computing environments [10, 40], can be evaluated with minimal effort, making EvalNet a versatile tool for broader interconnect research.

## 4 EVALUATING NETWORK TOPOLOGIES

We next describe the considered measures used to evaluate a network topology. We analyzed a large body of related work to provide a comprehensive overview [13, 15, 25, 29, 58–60, 63, 64, 86–90, 94, 107, 108]. We distinguish five classes of measures, used to analyze five aspects of a network: path diversity, bandwidth, latency, construction cost, and power consumption. All these metrics are also listed in Figure 3.

### 4.1 Path Diversity Measures

Recent studies in high-performance interconnect design indicate that harnessing different forms of path diversity, especially non-minimal multipathing, is a key performance enabler, especially for state-of-the-art low-diameter networks such as SF or PF. This is because these networks – while ensuring low latency, high throughput, as well as low construction cost and power consumption – have only *one* shortest path between almost all router pairs [13, 17]. Hence, one cannot use traditional load-balancing mechanisms such as equal-cost multi-path routing (ECMP) [47] for more performance, and must resort to multipath routing over non-minimal paths [17].

To address this, EvalNet supports analyzing all recently proposed measures for path diversity. First, it enables computing **counts** of both **minimal** and **non-minimal** paths between any two routers. These two measures illustrate the potential for multipathing over both traditional and recent networks. Moreover, we use the **path interference** (PI) measure, which illustrates how multipathing changes when considering interactions between different pairs of routers (i.e., some parts of a given network may be more or less advantageous for multipathing than other parts of the same network). This enables deriving detailed insights into path diversity in the local network structure, facilitating the fine-grained design of routing protocols. Finally, EvalNet enables computing the **total network load** (TNL), which is an upper bound on the number of flows that a network can maintain without congestion; TNL constitutes the maximum supply of path diversity offered by a given topology. We define all these measures in more detail in Section 5 while analyzing their values for different topologies.

Many of the above measures can be analyzed at **different levels of detail**. They can be summarized with histograms, or they can be computed and plotted separately for each individual router pair. EvalNet supports both options.

### 4.2 Throughput & Latency Measures

In EvalNet, we offer seamless interfaces to established tools, facilitating assessing **throughput** using packet-based simulations. We define throughput to be the normalized rate of injecting input packets for which the network saturates, i.e., the average latency of packets becomes higher than a pre-specified threshold. Moreover, we also consider **Maximum Achievable Throughput** (MAT), i.e.,



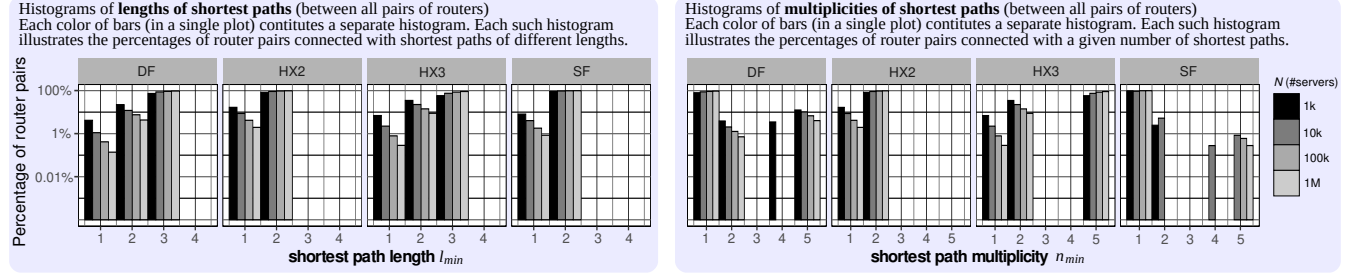


Figure 4: (left) Histograms of shortest path lengths  $l_{min}$ . (right) Histograms of shortest path multiplicities  $n_{min}$ . Considered networks: DF, SF, and two HX variants with  $D \in \{2, 3\}$ .

an approximation of throughput obtained using flow-based simulations conducted with TopoBench. MAT is defined as the ratio of the flow that can be forwarded between each pair of endpoints, to the demanded amount of flow for this pair.

**Network latency** is also obtained using packet-based simulations. The network latency is defined as the latency achieved for a very low injection rate that is close to zero. Thus, latency measures the end-to-end communication delay without congestion effects in the network. Moreover, the interfaced simulators enable delivering the **average path length**, i.e., the average number of router-router links traversed between any two communicated endpoints. This measure is useful because it is related to the network topological structure, i.e., one can see more clearly (than with latency) the impact of a routing design decision on the number of links that must be traversed.

### 4.3 Construction Cost & Power Consumption

EvalNet enables assessing the network **construction cost**. The user can plug in different cable and router cost models, coming from arbitrary vendors such as Ethernet or InfiniBand, and incorporating different rack placement. EvalNet also offers a model assessing the base power consumption required to power the network. This enables rudimentary comparisons between different baselines and preliminary insights into power consumption; detailed power models are notoriously hard to devise and vendors resort to their in-house infrastructures.

## 5 PATH DIVERSITY ANALYSIS

We present path diversity measures and conduct analyses in EvalNet, obtaining interesting insights about various interconnects. These measures are defined on pairs of routers [17]. Extended results are provided in the Appendix. We start with basic definitions shared by all measures.

**DEFINITION 5.1.** The neighborhood  $h(A)$  of a set of vertices  $A \subseteq V$  contains all the vertices adjacent to vertices in  $A$ , i.e.,  $h(A) = \{t \in V : \exists s \in A, (s, t) \in E\}$ .

Intuitively,  $h(A)$  are all the routers connected to routers in set  $A$ .

**DEFINITION 5.2.** The  $l$ -step neighborhood  $h^l(A)$  (or a transitive hull) of  $A \subseteq V$  contains all the vertices reachable by a path of length  $l$  from some vertex in  $A$ . Hence,  $h^l(A) = \underbrace{h(\dots h(A) \dots)}_{l \text{ times}}$ .

Intuitively,  $h^l(A)$  are all the routers reachable with  $l$  hops from routers in set  $A$ .

**DEFINITION 5.3.** For  $s, t \in V$ , the number of length- $l$  paths  $n_l(s, t)$  from  $s$  to  $t$  is recursively defined as  $n_l(s, t) = \sum_{u \in V} n_{l-1}(s, u) \cdot n_1(u, t)$ ,  $n_1(s, t) = 1$  iff  $(s, t) \in E$ , else 0.

### 5.1 Diversity of Shortest Paths

First, we consider the lengths and multiplicities of shortest paths. Understanding the former facilitates reasoning about the expected latency of a topology with shortest-path minimum routing, for different router pairs. Knowing the latter, on the other hand, enables taking informed decisions on whether to use multipathing over shortest paths (e.g., with ECMP), or whether one has to use non-minimal multipath routing [13, 17].

**DEFINITION 5.4.** For  $s, t \in V$ , the length of the shortest path  $l_{min}(s, t)$  between them is

$$l_{min}(s, t) = \min \{i \in \mathbb{N} : t \in h^i(\{s\})\}.$$

**DEFINITION 5.5.** The shortest path multiplicity between  $s, t \in V$  counts the number of shortest paths between  $s$  and  $t$  and is defined as

$$n_{min}(s, t) = n_l(s, t) \text{ with } l = l_{min}(s, t).$$

We provide example results in Figure 4, for both lengths of shortest paths (the left panel) and for their multiplicities (the right panel), for different sizes of four selected networks (indicated with different colors). The left panel illustrates the rich information about lengths of shortest paths. For example, most pairs are connected with a shortest path having two hops. Specifically, more router pairs in SF are connected with a 2-hop shortest path, than in HX2, for all topology sizes. This motivates more focus on making best-possible routing decisions over 2-hop paths in SF; 1-hop paths are negligible (especially for large networks). The right plot shows that, for most router pairs, the multiplicity of DF, HX2, and SF is low ( $n_{min} \leq 2$ ) compared to HX3. We also observe that nearly all router pairs in SF have only 1 shortest path between them, making non-minimal multipathing crucial. Still, note that DF and SF host some outlier router pairs connected with large counts of shortest paths. Finally, most router pairs in HX have many shortest paths between them, and this ratio is growing with  $N$ , making ECMP a good option.

### 5.2 Diversity of Non-Shortest Paths

Next, we are interested in the number of edge disjoint paths between two vertices  $s, t \in V$ , that are not necessarily shortest. It

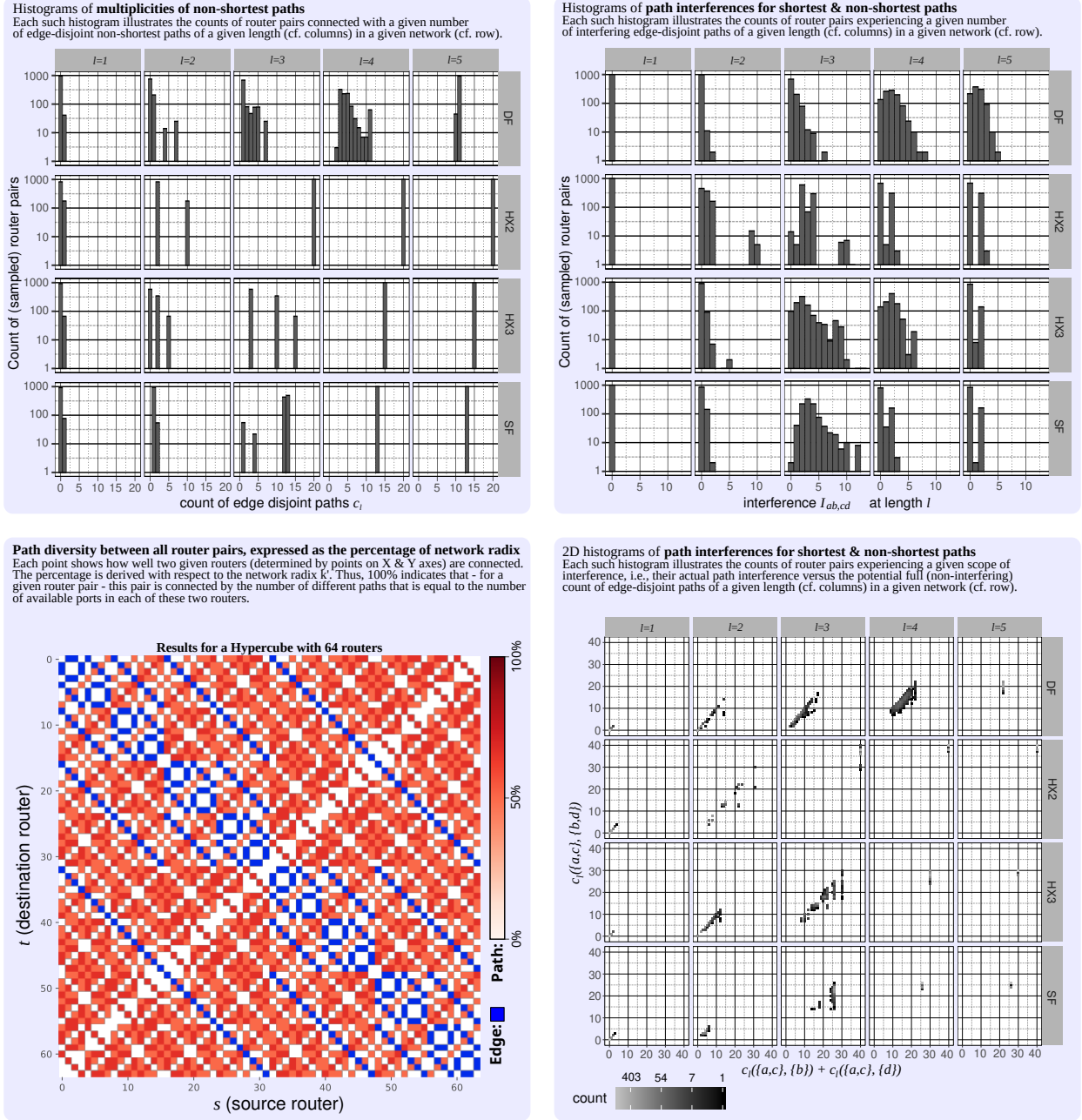


Figure 5: The analysis of non-shortest path diversity and path interference.

facilitates reasoning about the expected latency of a topology with non-shortest path routing, for different router pairs, and offers hints on routing design.

For convenience, we define this measure for two *groups* of vertices  $A, B$ :

**DEFINITION 5.6.** *The count of edge disjoint paths  $c_l(A, B)$  at length  $l$  between sets of routers  $A \subset V, B \subset V$  is the smallest number of edges that can be removed such that  $A$  and  $B$  are not connected by any path of length at most  $l$ .*

To obtain this count for two specific routers, we set  $A = \{s\}, B = \{t\}$ . The above definition lets us easily derive the number of shortest *edge disjoint* paths between two vertices:

**DEFINITION 5.7.** *The count of shortest edge disjoint paths  $c_{\min}(s, t)$  between  $s, t \in V$  is defined as*

$$c_{\min}(s, t) = c_l(\{s\}, \{t\}) \text{ with } l = l_{\min}(s, t)$$

$c_{\min}(s, t)$ , in contrast to  $l_{\min}(s, t)$ , does not count shortest paths that share any edges. Hence,  $c_{\min}(s, t) \leq l_{\min}(s, t)$ .

We show example results in Figure 5. The top-left panel contains the histograms of  $c_l$  at various lengths for DF, SF, and two HX instances. The values of  $c_l$  of DF are in general lower than in the low dimensional HX instances (HX2 and HX3) and SF. Accordingly,  $c_l$  of the HX and SF instances attains the network radix at lower lengths. HX2 has very high path diversity that is also “stable” (20 for all router pairs) for all higher path lengths ( $l = 3, 4, 5$ ). This facilitates the design for complex adaptive routing that routes around hotspots, uses complex paths, et cetera. Next, DF has relatively large diversity of paths for  $l = 5$ , suggesting similar insights as for HX2, but the count is 11, so less than 20 by nearly 50%. Then, for  $l = 4$  or  $l = 3$ , there is quite some diversity but the counts for different pairs may differ significantly, requiring attention when developing routing.

We further investigate  $c_l$  with a detailed view of the count of edge disjoint paths for *every router pair* in a small SF, showing pairs with  $\frac{c_l}{k} \leq 75\%$  (the bottom-left panel). This plot reveals all pairs of routers that do *not* come with a large path diversity between them. This enables a fine-grained performance debugging of routing and job scheduling schemes. The results show that the majority of router pairs are connected by a number of different edge-disjoint paths that is larger than 1/3 of all their available network ports (indicated with dark shade). Such pairs do not require a lot of attention when selecting a path to route. However, certain pairs do not come with large path diversity (cf. light shade). These specific pairs are not good candidates for scheduling communication-intense jobs.

### 5.3 Path Interference

So far, we considered measures that only take into account paths between two vertices. To capture the interactions of paths (of different lengths) between various pairs of vertices, we now define path interference. Here, paths between two router pairs  $a, b$  and  $c, d$  *interfere* if their total count  $c_l(\{a, c\}, \{b, d\})$  of edge disjoint paths at length  $l$  is lower than the sum of the single destination count of edge disjoint paths  $c_l(\{a, c\}, \{b\}) + c_l(\{a, c\}, \{d\})$ .

**DEFINITION 5.8.** *Path interference  $I_{ac,bd}^l$  between two router pairs  $a, b$  and  $c, d$  is defined as*

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

Understanding path interference facilitates reasoning about the performance of both minimal and non-minimal routing, and the scheduling of potential processes to different servers.

Note that  $I_{ac,bd}^l \geq 0$ , where equality is achieved when the disjoint paths from  $a, c$  to  $b$  do not share any edges with the disjoint paths from  $a, c$  to  $d$ . Thus, if  $I_{ac,bd}^l = 0$ , then connections can use fully the available link bandwidth. On the other hand, the higher the value of  $I_{ac,bd}^l$ , the more edges overlap so that the connections have to share this bandwidth.

Figure 5 (the top-right panel) shows histograms of interference  $I_{ac,bd}^l$  up to length  $l = 5$  for SF, DF, and two instances of HX. At length  $l = 4$  and  $l = 5$  DF shows higher interference than HX2, HX3 and SF in contrast to length  $l = 3$  where interference of DF is lower. This insight could be used when deciding on non-minimal adaptive routing decisions: taking paths of length 3 is more beneficial than 4 or 5. Also, DF, HX3, and SF have almost no interference at two hops ( $l = 2$ ). Thus, one does not have to pay much attention to

how to schedule pairs of communicating jobs, assuming they are executed on servers attached to routers separated by the distance  $l = 2$  (or  $l = 1$ ). Moreover, DF has also little interference between router pairs separated by 3 hops. Finally, at  $l = 3$  (i.e., for 3 inter-router hops), HX2, HX3, and SF have significant amounts of path interference, calling for care if devising routing/scheduling with such path lengths.

We further show a more detailed view of interference up to length  $l = 5$  on the bottom-right panel. Each single histogram illustrates the counts of router pairs experiencing a given scope of interference, i.e., their actual path interference versus the potential full (non-interfering) count of edge-disjoint paths, for a given length (as determined by the column) and for a given network (as determined by the row). Here, our pipeline also enables delivering various performance related insights. In general, we observe that the further a data point is from the “diagonal” of each histogram, the more path interference this data point indicates. Such far data points denote a large difference between the ideal amount of path diversity (on the X axis) versus the actual amount of diversity (on the Y axis). No topology is much imbalanced in this respect, as all data points far from the diagonal are dark (indicating few router pairs characterized a given difference between ideal and actual path diversity). Such pairs still require care when scheduling communication-intense jobs.

### 5.4 Computing Path Diversity Efficiently

In EvalNet, we primarily use algebraic algorithms to compute path diversity metrics, as they offer better scalability and parallelizability compared to purely combinatorial approaches, particularly for large, low-diameter networks [16, 96–98]. For example, for counting the number of paths of fixed length between all node pairs, we use adjacency matrix exponentiation, where the  $\ell$ -th power of the adjacency matrix  $A$  gives the number of length- $\ell$  paths via  $(A^\ell)_{i,j}$ . This computation is efficient for small diameters and can be adapted for next-hop tracking using semiring-based variants. For length-restricted connectivity, we extend the algebraic path propagation algorithm of Cheung et al. [26]. For other metrics, we also explore max-flow-based approaches such as Push-Relabel and Ford-Fulkerson with BFS path constraints (used for sampled length-limited disjoint path analysis), as well as classic Gomory-Hu trees for all-pairs edge connectivity. While these combinatorial algorithms serve as correctness baselines or targeted tools for small-scale samples, EvalNet’s core design centers on the algebraic methods due to their efficiency, extensibility, and compatibility with parallel computation.

## 6 FLOW-LEVEL SIMULATIONS

Flow-level simulators enable evaluation of large-scale systems, at a cost of a coarse design that may hinder realistic insights into performance. Flow-level simulations assume a continuous, real-valued flow model. Each flow has a rate  $\mu \in \mathbb{R}^+$ . The sum of all flows on each link must not exceed the link capacity. Without any additional constraints, this leads to constrained max-flow solutions in the topology graph. Other constraints (e.g., on the routes that can be taken by specific flows) lead to more realistic upper-bounds for performance of various routing schemes on specific topologies.

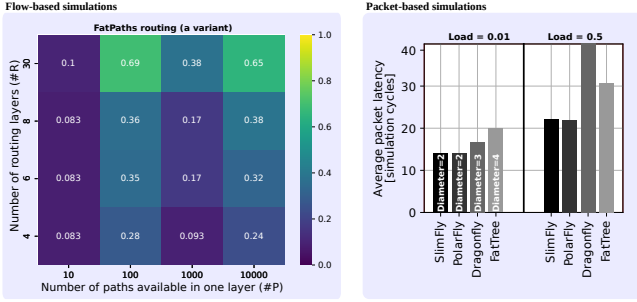


Figure 6: (left) Heatmap showing the Maximum Achievable Throughput (MAT), obtained with TopoBench, for a SF with 10k endpoints and a variant of FatPaths routing with a worst-case traffic pattern. (right) Packet-based simulation runtimes [cycles] with Booksim2, for networks of 50k endpoints, for two injection rates, 0.01 and 0.5. As expected, networks with lower diameter usually provide lower latency (load=0.01). For load=0.5, the network global bandwidth starts to play an important role; here, DF has higher latency than FT3 because it has much lower global bandwidth.

However, since these models do not represent time accurately, they are unable to estimate latency or provide realistic insights into the impact of routing choices on flow control. As such, they are a useful tool for initial insights when designing a network.

For flow simulations, we provide an interface to TopoBench, an evaluation tool [57] that uses linear programming (LP) to derive MAT. Our TopoBench version supports state-of-the-art layered routing schemes such as FatPaths [17] or extended layered routing [18], built specifically for enabling non-minimal multi-path routing. Example results are in the left part of Figure 6, where we analyze a variant of FatPaths that offers non-minimal multipath routing. That part contains a heatmap showing MAT for a SF with 10k endpoints and a variant of FatPaths routing with a worst-case traffic pattern. “#R” (Y axis) as well as “#P” (X axis) are parameters of the FatPaths routing scheme; they can be tuned for a specific topology. Usually, the higher #R is, the better the throughput becomes (because more layers entail more potential for multipathing [17]). Simultaneously, higher #R indicates more HW costs because one needs larger forwarding tables. Then, higher #P does not always enhance MAT (as confirmed by the results), because certain value ranges may cause larger path interference across routing layers, ultimately diminishing performance [17]. Overall, for certain configurations, FatPaths on SF achieves very high MAT.

## 7 PACKET-LEVEL SIMULATIONS

Packet-level simulators offer a more detailed packet-based model of the network and routing. All actions in the simulation are modeled as *events*, which are scheduled to happen at a given point in time. For each event, some code is executed, which can update the state of the simulation, and produce new events that are scheduled to happen in the future. The main simulation loop keeps the scheduled events in a priority queue and advances the simulation by dequeuing and executing successive events.

### 7.1 Which Simulator to Use?

A large number of simulators exist, and it is challenging to select the one best suited for one’s purposes. We analyze a broad selection of available simulators, and show that none scale to the desired

Simulator	Scalability [#servers]	Design
LogGOPSim [46]	1024 [46]	SM
htsim [92]	128 [92]	SM
NS2 [53]	<1,000	SM
booksim [56]	10,000 [15]	SM
FOGSim [39]	16,512 [12]	SM
OMNet++ [103]	100,000 [38]	SM
NS3 [79]	5,000 [83]	DM
NS4 [35]	N.A.	DM
BigSim [110]	46,656 [55]	DM
SST [93]	110,592 [41]	DM
xSim [19]	2,097,152 [33]	DM
ROSS/CODES [27]	1,000,000 [105] (Slim Fly), 50,000,000 [77] (Dragonfly)	DM
htsim [this work]	1,000,000	SM
Booksim2 [this work]	50,000	SM

Table 2: The comparison of available packet-level network simulators. “Scalability” is the largest size of a topology (simulated with a specific simulator) that we were able to find in the literature. “Design” includes the details of the compute platform used for simulations in the “Scalability” column: “SM” (shared-memory design), “DM” (distributed-memory design).

sizes of hundreds of thousands of servers. The overview is in Table 2. We analyzed their codebases, investigating which (1) facilitate modifications that could enhance scalability, (2) are easy to use and extend without requiring distributed-memory infrastructure support, and (3) cover a large number of protocols and settings. Based on that analysis, we selected two simulators: htsim [92] for lossy TCP/IP stacks and Booksim2 [56] for lossless HPC architectures. Other simulators do not meet these criteria. For example, SST and ROSS natively target HPC clusters and we concluded that they are relatively complex to set up, use, and extend on low-cost PCs.

**htsim** The *htsim* simulator [92] provides a lightweight infrastructure for lossy networks. It only models the transport layer: there is no model for links or routers. Instead, the route for each packet is pre-computed as a sequence of queues that the packet will pass through. Such a route specification is attached to the packet. Each queue has a finite service rate, which models the link capacity. Due to this design, htsim is highly flexible, and new topologies can be added straightforwardly: the topology only affects the route computation, which is explicitly called for every flow during its setup. The disadvantage is that many adaptive routing schemes cannot be modeled, since there is no per-router state that could affect the routing. Furthermore, all flows are initialized before the first event is processed, and all routes are kept in memory. EvalNet offers a seamless interface to htsim. We select htsim as it targets TCP stacks; it supports by default modern TCP-based schemes such as NDP [42], and it can be easily extended with new routing schemes.

**Booksim2** Booksim2 targets lossless networks. It explicitly models input-queued routers, thus one can easily extend the simulator with new routing protocols. Moreover, there is support for flow control unit (flit) packets, and thus for schemes such as worm-hole routing or virtual cut-through flow control. Virtual channels are also supported. One can specify the total buffering/port, router delay for credit processing, delays for channel latency, router and VC allocations, and processing in a crossbar. As with htsim, EvalNet offers modules that enable seamless evaluation of any generated topologies within the Booksim2 setting.



Element		Range (lower–upper bounds)			10,000 servers			100,000 servers			1,000,000 servers		
		Count (relative)	Count (total)	Memory (per item)	Count (relative)	Count (total)	Memory (total)	Count (relative)	Count (total)	Memory (total)	Count (relative)	Count (total)	Memory (total)
Network	router	1	$10^2$ – $10^4$	<sup>(1)</sup>	1	242	<sup>(1)</sup>	1	1,058	<sup>(1)</sup>	1	5,618	<sup>(1)</sup>
	cable (per router)	10–1000	$10^4$ – $10^6$	<sup>(1)</sup>	<sup>(4)</sup> 8.5+40	11,737	<sup>(1)</sup>	<sup>(4)</sup> 17.5+90	113,735	<sup>(1)</sup>	<sup>(4)</sup> 39.5+200	1,345,511	<sup>(1)</sup>
	server (per router)	10–1000	$10^4$ – $10^6$	5kB	40	9,680	47MB	90	95,220	460MB	200	1,123,600	5.4GB
	routing entries	$O(N_r^2)$	$10^4$ – $10^8$	<sup>(2)</sup> 100B	$O(N_r^2)$	58,564	6MB	$O(N_r^2)$	1,119,364	110MB	$O(N_r^2)$	31,561,924	3GB
Workload	task (per server)	1–10	$10^2$ – $10^5$	<sup>(1)</sup>	1	968,0	<sup>(1)</sup>	1	95,220	<sup>(1)</sup>	1	1,123,600	<sup>(1)</sup>
	flow (per task)	1–100	$10^2$ – $10^7$	2kB	100	968,000	1.8GB	10	952,200	1.8GB	1	1,123,600	2.1GB
	path (per flow)	1–10	$10^2$ – $10^8$	600B	5	4,840,000	2.7GB	5	4,761,000	2.7MB	5	5,618,000	3.2GB
	packet (per flow)	10–1000	$10^3$ – $10^{10}$	<sup>(3)</sup>	<sup>(5)</sup> 110	106,480,000	<sup>(3)</sup>	<sup>(5)</sup> 110	104,742,000	<sup>(3)</sup>	<sup>(5)</sup> 110	123,596,000	<sup>(3)</sup>

Table 3: Estimation of the number of elements of a data center Slim Fly simulation. The numbers and thus required resources vary widely depending on a chosen configuration. “Range” lists parameter bounds considered in this work, in the following columns we show numbers for example configurations. Memory estimates are based on htSim. The chosen workload parameters keep the simulations feasible on a laptop (configuration details are in 7.3). <sup>1</sup> Negligible for relevant configurations. <sup>2</sup> Data depends on the routing scheme. <sup>3</sup> Objects are transient and accounted for in the flow counts. <sup>4</sup> Inter-router cables + server links. <sup>5</sup> The average over the flow size distribution, excluding re-transmissions.

## 7.2 Can We Simulate Large Networks on Low-Cost HW?

Next, we conduct a feasibility analysis in which we argue that million-server packet-level simulators should in theory be achievable on a low-cost PC. For this, we analyze the approximate memory and time cost of targeted large-scale simulations.

**Number of Simulation Elements** We estimate the number of elements of a simulated topology (e.g., servers) and of a simulated workload (e.g., flows which require flow control state). The results of the analysis are in Table 3. The number of network elements does not pose the most serious scalability problems. Instead, *the number and corresponding size of elements related to workloads dominates the memory usage and the running time.*

**Memory and Time Requirements** The total memory usage and simulation time depend on the simulation software and on *granularity*, i.e., what elements of the simulated workload are explicitly stored and simulated. We estimate these numbers in Table 3, which also shows what may *not* be feasible in the largest-scale simulations. Specifically, we cannot simulate large workloads on large networks, and must not store any per-packet state.

*The lowest required amount of memory is determined by the number of simulated flows*, because we want to simulate end-to-end flow control for each flow. Thus, the flow control algorithm state for each flow needs to be stored. We also account for the packets in flight on a per-flow basis, since the memory occupied by packets is tightly coupled to the number of concurrent flows. Memory consumed is about 2kB/flow plus 600B/path, which could probably be further reduced. To make large simulations feasible, it is crucial to keep the per-flow overhead low. Special attention needs to be paid to monitoring and debugging code, which can easily dominate the per-flow state. For very large networks, the quadratically increasing size of the routing tables starts to dominate, but as Table 3 shows, this was not a problem for our simulations.

*The lowest amount of time is determined by the amount of packet forwarding.* Each packet forwarding has to be processed as at least one event, and these events constitute the majority of simulation time. In the modern low-diameter networks, each packet has to be forwarded  $\approx 2$ –4 times to reach its destination, and we observed

around 60 events processed for each transmitted packet (this includes entering and leaving queues on the path through the network, for the data packet as well as the ACK packet and potential retransmissions). We observe event rates of around  $10^6$  per second on one CPU core. Since each event likely causes at least one cache miss, we cannot expect much higher rates without distributed simulation or middle- to high-cost PC stations, which are outside the scope of this work.

## 7.3 How to Simulate Really Large Networks on Low-Cost HW?

Finally, we show how to use htSim [92] and Booksim2 [56] to enable more scalability. In Booksim2, the key limiting factor that we identified was processing related to single packets. This processing is usually related to gathering statistics, which is not relevant for the actual simulation outcomes. In Booksim2, eliminating this processing was the main enabler for being able to simulate networks with 10,000 endpoints and more. In htSim, on the contrary, the main bottleneck was the storage related to flows. Specifically, we had to add a more efficient, routing table based scheme for arbitrary topologies. We do not use the provided logging solution, which tends to produce too much output at the simulation scales that we consider. Instead, the statistics of each flow are printed in text format to standard output when the flow finishes. Another scalability obstacle was the `net_paths` structure present in the sample programs, a preallocated  $N^2$  size matrix of routes, which would only be sparsely populated for permutation workloads and would dominate memory use for large networks.

## 7.4 Representative Simulation Results

We simulate networks with  $\approx 10k$ ,  $\approx 100k$ , and  $\approx 1M$  servers using a laptop with 16GB RAM and an Intel Core i7-8550U CPU; each run takes 1–3 hours. Networks are  $5\times$  oversubscribed, as in configurations studied in [59], and include topologies larger than those used in prior large-scale SF simulations. Due to the cost of simulating large networks, we capture only a few milliseconds of activity—sufficient for reliable relative performance comparisons. We focus on flow completion time (FCT) distributions as a function of flow size, using the pFabric web search distribution (average  $\approx 1MB$ ) and injection rates of 40–60 flows/server/s. Results are shown as

mean, 10th, and 99th percentiles. To mitigate artifacts from the fixed injection window, we carefully interpret short and long flows.

As an example, we analyze 10k, 100k, and 1M server simulations, see Figure 7. The left plot shows FCT as a function of size. For long flows ( $v > 200\text{kB}$ ), FCT is limited by bandwidth, while for shorter flows, latency dominates. The middle plot illustrates the distribution of FCTs for middle-size flows: here, the better tail behavior of the randomized Jellyfish topology is clearly visible. In the right plot, we show the impact of the flow arrival rate  $\lambda$ . Even a small increase in load causes a large degradation in FCT in the considered networks, due to the used oversubscription. The 1M server simulation is less affected due to the limited warmup time.

Our analysis illustrates that million-server packet-level simulations on a simple low-cost laptop are feasible. However, they require some compromises on the quality of simulation, most importantly on the number of allowed flows and on the methodology (i.e., simulating a fixed amount of flows instead of a fixed time window after reaching a steady-state).

## 8 COMPREHENSIVE ANALYSIS & TAKEAWAYS

EvalNet enables multifaceted analysis of numerous network topologies. We list the considered networks in Table 4, and provide an assessment of their important properties (performance, cost & power consumption, path diversity), obtained using EvalNet. The table compiles the results described in the previous sections of this paper, as well as further results illustrated in the Appendix in Figures 8–56.

### 8.1 General Observations & Patterns

A central observation is the critical role of path diversity (both shortest-path (MS) and non-shortest-path (MnS)) in determining a network’s performance and scalability. Topologies such as Xpander and OFT exhibit high MS, meaning they provide many equal-cost paths between node pairs. This allows routing schemes like ECMP to distribute traffic efficiently, reducing congestion and delivering both low latency and high global bandwidth under minimal routing. These networks are thus well-suited for environments that rely on deterministic routing, where fault tolerance and load balancing are achieved through structural redundancy alone.

In contrast, Moore-bound diameter-2 networks like SF and PF attain excellent performance through topological minimalism (i.e., by minimizing average path length and diameter). However, their MS is low: most router pairs have only a single shortest path, rendering traditional ECMP ineffective. These networks compensate by offering very high MnS, which enables advanced non-minimal multipathing strategies such as Valiant load balancing and FatPaths, allowing traffic to be dispersed over longer, disjoint paths to avoid congestion. This tradeoff is critical: although SF, PF and other such networks lack diversity in minimal paths, their rich non-minimal structure, if properly exploited, enables them to outperform MS-rich networks under high loads—provided the routing stack supports sufficient adaptivity and complexity (e.g., congestion-aware routing, larger tables, multipath tracking).

This divergence between MS and MnS has a broader design tradeoff. Networks with high MS (e.g., Xpander) typically have slightly larger diameters but benefit from simplicity and robustness of minimal routing. Networks with low MS but high MnS (e.g., SF,

Topology	Lat.	GLB.	Ext.	C&P	MS	MnS
Slim Fly [15]						
PolarFly [65]						
Xpander [100]						
BundleFly [68]						
Dragonfly [64]						
Cascade Dragonfly [34]						
Jellyfish [95]						
Mesh						
Torus 2D						
Torus 3D						
Torus 4D						
Torus 5D						
Torus 6D						
Hypercube						
HyperX (2-dimensional) [2]						
HyperX (3-dimensional) [2]						
Express Mesh [54]						
Flattened Butterfly [63]						
Fat Tree (2-level) [69]						
Fat Tree (3-level) [69]						
k-ary n-tree [85]						
eXtended Generalized Fat Tree [80]						
Orthogonal Fat-Trees [60]						
Multi-Layer Full-Mesh [60]						
Undirected Kautz [70]						
Arrangement Network [30]						

**Table 4: A general comparison of different network topologies.** Lat.: latency. GLB.: global bandwidth (i.e., the saturation point for the random uniform traffic pattern). Ext.: extensibility (i.e., how far can we extend a given concrete network with new servers?). Note that fixed diameter networks have lowest extensibility, because they have strict upper bounds on how many new servers can be attached; the lower the diameter is, the lower this bound is. Contrarily, networks such as torus can attach arbitrarily many new servers, because their diameter grows to infinity. C&P: construction cost and static power consumption. They are assessed as proportional to the total number of ports in a network, for a fixed network size  $N$  (i.e., for fixed  $N$ , networks with higher radix tend to have higher cost and power consumption). MS: diversity of shortest paths. It is assessed as a weighted average of shortest path diversities at different lengths. MnS: diversity of non-shortest paths. It is assessed as a weighted average of non-shortest path diversities at different lengths. The battery symbols serve as a rudimentary measure of relative comparison between networks. : worst, : bad, : medium, : good, : excellent. They always have a positive meaning, e.g., “” used for cost means that the cost of a given network is *very low* compared to other networks.

PF) require intelligent routing to achieve high performance but can then perform remarkably well. On the other hand, low-dimensional regular topologies such as torus and mesh suffer from low values of both MS and MnS. While these networks are easily extensible in terms of size, their limited path diversity leads to high latency. Without structural augmentation (e.g., express links or randomized shortcuts), their performance does not scale with node count.

Tree-based networks, including Fat Trees, k-ary n-trees, and eXtended Generalized Fat Trees, offer a moderate level of both MS and MnS due to the presence of multiple upper-layer routes. These designs are non-blocking when fully provisioned and can deliver high global bandwidth by supporting ECMP and basic multipathing. However, they suffer from moderate-to-high latency due to their hierarchical depth, and may lack resilience of the more modern high-MS or high-MnS topologies that have good expansion properties.

These insights tie into performance, cost, and scalability tradeoffs. While SF and PF achieve top-tier latency and bandwidth, their fixed diameter fundamentally limits extensibility. Namely, they

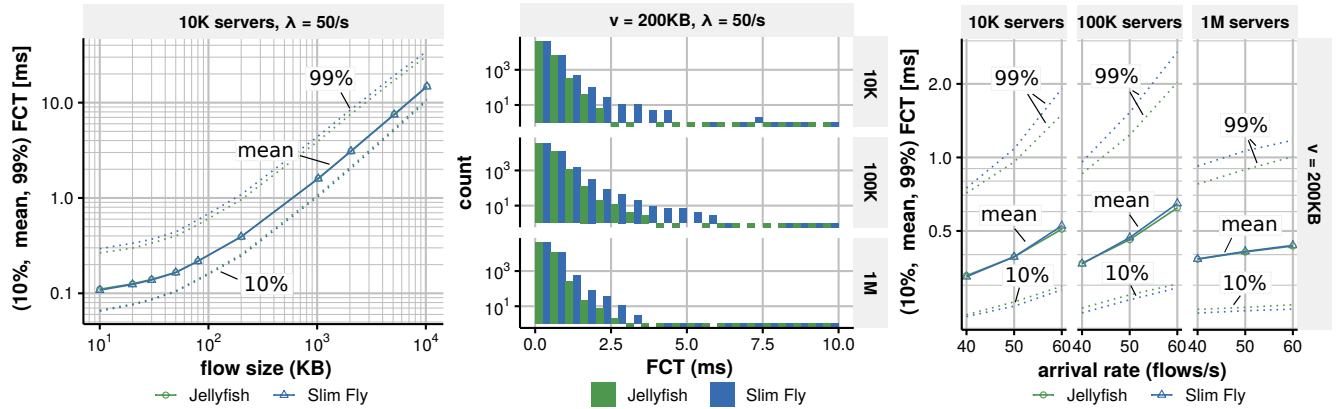


Figure 7: Results of large-scale simulations.

cannot grow beyond a certain size without losing their optimal structural properties. In contrast, torus and mesh networks offer near-unlimited extensibility due to their non-constant diameter, but this comes at the cost of poor latency and reduced throughput. Topologies with very high radix, like HyperX and Multi-Layer Full Mesh, provide rich connectivity and sometimes high MS/MnS, but incur substantially higher construction and power costs, as they require more expensive switching hardware due to very high radix. Dragonfly and Cascade Dragonfly offer a middle ground, balancing latency, cost, and path diversity, which explains their popularity in practical deployments.

Ultimately, while our findings highlight that each topology occupies a different point in the design tradeoff space, low-diameter networks such as Slim Fly and PolarFly emerge as particularly desirable candidates. Their minimal path lengths enable very low latency and bandwidth, which are critical for modern high-performance workloads. Although they offer limited shortest-path diversity, their rich structure of non-minimal paths allows them to sustain high throughput under load—provided that routing schemes can exploit this flexibility. EvalNet enables these tradeoffs to be quantified and visualized precisely, supporting informed design decisions based on latency, throughput, scalability, cost-efficiency, and path diversity.

## 8.2 Example Detailed Insights

We also provide a few selected more detailed insights that could be used, e.g., as a basis for enhancements of routing strategies. Full results and observations are in the Appendix.

**Topologies based on algebraic expanders minimize contention for multipathing.** For example, Spectralfly’s path interference is nearly zero, even matching random Jellyfish variants, implying that one can exploit non-minimal detours almost for free, boosting available parallel paths without contention.

**Express Meshes have bimodal latency.** Express Meshes have a “gap” in their shortest-path multiplicity: most node-pairs have very few ( $\leq 2$ ) or a large number ( $\geq 5$ ) of disjoint shortest paths, with almost no pairs in between. Hence, under uniform ECMP, about half the traffic has rich bandwidth, while the other half is starved, creating a bimodal latency distribution (unless one employs adaptivity).

**Megaflly has hop-bound tail-latency.** Megaflly variants have little path interference for up to 3 hops, yet develop a small set ( $\approx 10$ ) of high-interference outliers at more than 3 hops. Hence, one can exploit non-minimal multipathing up to 3 hops with near-ideal bandwidth, but more hops risk large tail-latency spikes.

**PolarStar variants have performance trade-offs.** PolarStars with Paley-subgraphs have marginally higher shortest-path multiplicity at 5 hops and only moderate path interference, whereas PolarStars with BDF-subgraphs yield lower overall interference (smoother latency) but slightly fewer total paths. Thus, one can tune PolarStar for either peak throughput (Paley) or tighter latency tails (BDF) based on application’s burstiness.

## 9 CONCLUSION

In this work, we introduce EvalNet, a comprehensive toolchain for the generation and in-depth analysis of extreme-scale interconnection networks, with a strong focus on path diversity as a core enabler of performance and scalability. EvalNet integrates topology generation, fine-grained path diversity analysis, performance modeling, and cost evaluation into a unified and extensible framework. Using EvalNet, we conduct the widest comparative study to date across 25 network topologies, revealing how shortest- and non-shortest-path diversity relate to key performance and deployment metrics such as latency, throughput, cost, and power. Our analysis shows that the ability to exploit non-minimal multipathing is essential for maintaining high throughput in low-diameter, scalable networks. EvalNet interfaces seamlessly with established simulators and serves as a practical and versatile tool for guiding future innovations in large-scale network design in data centers and supercomputers.

## ACKNOWLEDGMENTS

We thank Haoran Zhao for his code contributions. We thank Colin McMurtrie, Mark Klein, Angelo Mangili, and the whole CSCS team granting access to the Ault and Daint machines. We thank Timo Schneider for help with infrastructure at SPCL. This project received funding from the European Research Council (Project PSAP, No. 101002047), and the European High-Performance Computing

Joint Undertaking (JU) under grant agreement No. 955513 (MAEL-STROM). This project received funding from the European Union's HE research and innovation programme under the grant agreement No. 101070141 (Project GLACIATION). This project was supported by JSPS KAKENHI Grant Number JP19H04119.

## REFERENCES

- [1] Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoun Choi. 2015. A Scalable Processing-In-Memory Accelerator for Parallel Graph Processing. *SIGARCH Comput. Archit. News* 43, 3S (June 2015), 105–117. <https://doi.org/10.1145/2872887.2750386>
- [2] Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S. Schreiber. 2009. HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Portland, OR, USA) (SC '09). Association for Computing Machinery, New York, NY, USA, Article 41, 11 pages. <https://doi.org/10.1145/1654059.1654101>
- [3] Yuuichirou Ajima, Tomohiro Inoue, Shinya Hiramoto, and Toshiyuki Shimizu. 2012. Tofu: Interconnect for the K Computer. *Fujitsu Sci. Tech. J* 48, 3 (July 2012), 280–285. <https://www.fujitsu.com/global/documents/about/resources/publications/fstj/archives/vol48-3/paper05.pdf>
- [4] Yuichiro Ajima, Takahiro Kawashima, Takayuki Okamoto, Naoyuki Shida, Kouichi Hirai, Toshiyuki Shimizu, Shinya Hiramoto, Yoshiro Ikeda, Takahide Yoshikawa, Kenji Uchida, and Tomohiro Inoue. 2018. The Tofu Interconnect D. In *Proceedings of the IEEE International Conference on Cluster Computing* (Belfast, Northern Ireland) (CLUSTER '18). IEEE Press, Washington, DC, USA, 646–654. <https://doi.org/10.1109/CLUSTER.2018.00090>
- [5] Mohammed Alasmar, George Parisi, and Jon Crowcroft. 2018. Polyraport: Embracing Path and Data Redundancy in Data Centres for Efficient Data Transport. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos* (Budapest, Hungary) (SIGCOMM '18). Association for Computing Machinery, New York, NY, USA, 69–71. <https://doi.org/10.1145/3234200.3234222>
- [6] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. 2014. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters. *SIGCOMM Comput. Commun. Rev.* 44, 4 (Aug. 2014), 503–514. <https://doi.org/10.1145/2740070.2626316>
- [7] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data Center TCP (DCTCP). *SIGCOMM Comput. Commun. Rev.* 40, 4 (Aug. 2010), 63–74. <https://doi.org/10.1145/1851275.1851192>
- [8] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. pFabric: Minimal Near-Optimal Datacenter Transport. *SIGCOMM Comput. Commun. Rev.* 43, 4 (Aug. 2013), 435–446. <https://doi.org/10.1145/2534169.2486031>
- [9] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Weicheng Sun. 2014. PIAS: Practical Information-Agnostic Flow Scheduling for Data Center Networks. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks* (Los Angeles, CA, USA) (HotNets '14). Association for Computing Machinery, New York, NY, USA, 1–7. <https://doi.org/10.1145/2670518.2673871>
- [10] Yves Baumann, Tal Ben-Nun, Maciej Besta, Lukas Gianinazzi, Torsten Hoefer, and Piotr Luczynski. 2024. Low-Depth Spatial Tree Algorithms. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium* (San Francisco, CA, USA) (IPDPS '24). IEEE Press, Washington, DC, USA, 180–192. <https://doi.org/10.1109/IPDPS57955.2024.00024>
- [11] Cristian Hernandez Benet, Andreas J. Kessler, Theophilus Benson, and Gergely Pongracz. 2018. MP-HULA: Multipath Transport Aware Load Balancing Using Programmable Data Planes. In *Proceedings of the Morning Workshop on In-Network Computing* (Budapest, Hungary) (NetCompute '18). Association for Computing Machinery, New York, NY, USA, 7–13. <https://doi.org/10.1145/3229591.3229596>
- [12] Mariano Benito, Enrique Vallejo, and Ramon Beivide. 2015. On the Use of Commodity Ethernet Technology in Exascale HPC Systems. In *Proceedings of the IEEE 22nd International Conference on High Performance Computing* (Washington, DC, USA) (HPC '15). IEEE Press, Washington, DC, USA, 254–263. <https://doi.org/10.1109/HPC.2015.32>
- [13] Maciej Besta, Jens Domke, Marcel Schneider, Marek Konieczny, Salvatore Di Girolamo, Timo Schneider, Ankit Singla, and Torsten Hoefer. 2021. High-Performance Routing with Multipathing and Path Diversity in Ethernet and HPC Networks. *IEEE Transactions on Parallel and Distributed Systems* 32, 4 (April 2021), 943–959. <https://doi.org/10.1109/TPDS.2020.3035761>
- [14] Maciej Besta, Syed Minhaj Hassan, Sudhakar Yalamanchili, Rachata Ausavarungrun, Onur Mutlu, and Torsten Hoefer. 2018. Slim NoC: A Low-Diameter On-Chip Network Topology for High Energy Efficiency and Scalability. *SIGPLAN Not.* 53, 2 (March 2018), 43–55. <https://doi.org/10.1145/3296957.3177158>
- [15] Maciej Besta and Torsten Hoefer. 2014. Slim Fly: A Cost Effective Low-Diameter Network Topology. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (New Orleans, LA, USA) (SC '14). IEEE Press, New York, NY, USA, 348–359. <https://doi.org/10.1109/SC.2014.34>
- [16] Maciej Besta, Florian Marending, Edgar Solomonik, and Torsten Hoefer. 2017. SlimSell: A Vectorizable Graph Representation for Breadth-First Search. In *Proceedings of the 31st IEEE International Parallel and Distributed Processing Symposium* (Orlando, FL, USA) (IPDPS '17). IEEE Press, Washington, DC, USA, 32–41. <https://doi.org/10.1109/IPDPS.2017.93>
- [17] Maciej Besta, Marcel Schneider, Marek Konieczny, Karolina Cynk, Erik Henriksen, Salvatore Di Girolamo, Ankit Singla, and Torsten Hoefer. 2020. FatPaths: Routing in Supercomputers and Data Centers When Shortest Paths Fall Short. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Atlanta, GA, USA) (SC '20). IEEE Press, Washington, DC, USA, Article 27, 18 pages. <https://doi.org/10.1109/SC41405.2020.00031>
- [18] Nils Blach, Maciej Besta, Daniele De Sensi, Jens Domke, Hussein Harake, Shigang Li, Patrick Iff, Marek Konieczny, Kartik Lakhotia, Ales Kubicek, Marcel Ferrari, Fabrizio Petrini, and Torsten Hoefer. 2024. A High-Performance Design, Implementation, Deployment, and Evaluation of the Slim Fly Network. In *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation* (Santa Clara, CA, USA) (NSDI '24). USENIX Association, Berkeley, CA, USA, Article 57, 20 pages. <https://www.usenix.org/conference/nsdi24/presentation/blach>
- [19] Swen Böhm and Christian Engelmann. 2011. xSim: The Extreme-Scale Simulator. In *Proceedings of the International Conference on High Performance Computing & Simulation* (Istanbul, Turkey) (HPCS '11). IEEE Press, Washington, DC, USA, 280–286. <https://doi.org/10.1109/HPCSim.2011.5999835>
- [20] Yu Cai, Ken Mai, and Onur Mutlu. 2015. Comparative Evaluation of FPGA and ASIC Implementations of Bufferless and Buffered Routing Algorithms for On-Chip Networks. In *Proceedings of the Sixteenth International Symposium on Quality Electronic Design* (Santa Clara, CA, USA) (ISQED '15). IEEE Press, Washington, DC, USA, 475–484. <https://doi.org/10.1109/ISQED.2015.7085472>
- [21] Christian Callegari, Stefano Giordano, Michele Pagano, and Teresa Pepe. 2014. A Survey of Congestion Control Mechanisms in Linux TCP. In *Proceedings of the 17th International Conference on Distributed Computer and Communication Networks* (DCCN '13) (Moscow, Russia) (Communications in Computer and Information Science (CCIS), Vol. 279). Vladimir Vishnevsky, Dmitry Kozyrev, and Andrey Larionov (Eds.). Springer, Berlin/Heidelberg, 28–42. [https://doi.org/10.1007/978-3-319-05209-0\\_3](https://doi.org/10.1007/978-3-319-05209-0_3)
- [22] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: Congestion-Based Congestion Control. *Commun. ACM* 60, 2 (Jan. 2017), 58–66. <https://doi.org/10.1145/3009824>
- [23] Henri Casanova, Arnaud Legrand, and Martin Quinson. 2008. Simgrid: A Generic Framework for Large-Scale Distributed Experiments. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation* (Liverpool, England) (UKSIM '08). IEEE Press, Washington, DC, USA, 126–131. <https://doi.org/10.1109/UKSIM.2008.28>
- [24] Kevin Kai-Wei Chang, Rachata Ausavarungrun, Chris Fallin, and Onur Mutlu. 2012. HAT: Heterogeneous Adaptive Throttling for On-Chip Networks. In *Proceedings of the IEEE 24th International Symposium on Computer Architecture and High Performance Computing* (New York, NY, USA) (SBAC-PAD '12). IEEE Press, Washington, DC, USA, 9–18. <https://doi.org/10.1109/SBAC-PAD.2012.44>
- [25] Dong Chen, Philip Heidelberger, Craig Stunkel, Yutaka Sugawara, Cyriel Minkenberg, Bogdan Prisacari, and German Rodriguez. 2016. An Evaluation of Network Architectures for Next Generation Supercomputers. In *Proceedings of the 7th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems* (Salt Lake City, UT, USA) (PMBS '16). IEEE Press, Washington, DC, USA, 11–21. <https://doi.org/10.1109/PMBS.2016.007>
- [26] Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. 2011. Graph Connectivities, Network Coding, and Expander Graphs. In *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science* (Palm Springs, CA, USA) (FOCS '11). IEEE Press, Washington, DC, USA, 190–199. <https://doi.org/10.1109/FOCS.2011.55>
- [27] Jason Cope, Ning Liu, Sam Lang, Phil Carns, Chris Carothers, and Robert Ross. 2011. CODES: Enabling Co-Design of Multilayer Exascale Storage Architectures. In *Proceedings of the Workshop on Emerging Supercomputing Technologies* (Glasgow, Scotland) (WEST '11). Argonne National Laboratory, Argonne, IL, USA, 303–312. <http://www.mcs.anl.gov/uploads/cels/papers/P1884.pdf>
- [28] Marcin Copik, Grzegorz Kwaśniewski, Maciej Besta, Michał Podstawski, and Torsten Hoefer. 2021. SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing. In *Proceedings of the 22nd International Middleware Conference* (Québec, Canada) (Middleware '21). Association for Computing Machinery, New York, NY, USA, 64–78. <https://doi.org/10.1145/3464298.3476133>
- [29] William Dally and Brian Towles. 2003. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

- [30] Khaled Day and Anand Tripathi. 1992. Arrangement Graphs: A Class of Generalized Star Graphs. *Inform. Process. Lett.* 42, 5 (July 1992), 235–241. [https://doi.org/10.1016/0020-0190\(92\)90030-Y](https://doi.org/10.1016/0020-0190(92)90030-Y)
- [31] Salvatore Di Girolamo, Daniele De Sensi, Konstantin Taranov, Milos Malesevic, Maciej Besta, Timo Schneider, Severin Kistler, and Torsten Hoefler. 2022. Building Blocks for Network-Accelerated Distributed File Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Dallas, TX, USA) (SC '22). IEEE Press, Washington, DC, USA, Article 10, 14 pages. <https://doi.org/10.1109/SC41404.2022.00015>
- [32] Salvatore Di Girolamo, Konstantin Taranov, Andreas Kurth, Michael Schaffner, Timo Schneider, Jakub Beránek, Maciej Besta, Luca Benini, Duncan Roweth, and Torsten Hoefler. 2019. Network-Accelerated Non-Contiguous Memory Transfers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, CO, USA) (SC '19). Association for Computing Machinery, New York, NY, USA, Article 56, 14 pages. <https://doi.org/10.1145/3295500.3356189>
- [33] Christian Engelmann. 2014. Scaling to a Million Cores and Beyond: Using Light-Weight Simulation to Understand the Challenges Ahead on the Road to Exascale. *Future Generation Computer Systems* 30 (2014), 59–65. <https://doi.org/10.1016/j.future.2013.04.014>
- [34] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. 2012. Cray Cascade: A Scalable HPC System Based on a Dragonfly Network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Salt Lake City, UT, USA) (SC '12). IEEE Press, Washington, DC, USA, Article 103, 9 pages. <https://doi.org/10.1109/SC.2012.39>
- [35] Chengze Fan, Jun Bi, Yu Zhou, Cheng Zhang, and Haisu Yu. 2017. NS4: A P4-Driven Network Simulator. In *Proceedings of the SIGCOMM Posters and Demos* (Los Angeles, CA, USA) (SIGCOMM Posters and Demos '17). Association for Computing Machinery, New York, NY, USA, 105–107. <https://doi.org/10.1145/3123878.3132002>
- [36] Mario Flajslik, Eric Borch, and Mike A Parker. 2018. Megaflj: A Topology for Exascale Systems. In *Proceedings of the 33rd International Conference on High Performance Computing (ISC '18)* (Frankfurt, Germany) (Lecture Notes in Computer Science (LNCS), Vol. 10876). Rio Yokota, Michèle Weiland, David Keyes, and Carsten Trinitis (Eds.), Springer, Berlin/Heidelberg, Germany, 289–310. [https://doi.org/10.1007/978-3-319-92040-5\\_15](https://doi.org/10.1007/978-3-319-92040-5_15)
- [37] Haohuan Fu, Junfeng Liao, Jinzhe Yang, Lanning Wang, Zhenya Song, Xiaomeng Huang, Chao Yang, Wei Xue, Fangfang Liu, Fangli Qiao, Wei Zhao, Xunqiang Yin, Chaofeng Hou, Chenglong Zhang, Wei Ge, Jian Zhang, Yangang Wang, Chunbo Zhou, and Guangwen Yang. 2016. The Sunway TaihuLight Supercomputer: System and Applications. *Science China Information Sciences* 59, 7 (June 2016), 072001. <https://doi.org/10.1007/s11432-016-5588-7>
- [38] Thomas Gamer and Christoph P. Mayer. 2009. Large-Scale Evaluation of Distributed Attack Detection. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques* (Rome, Italy) (Simutools '09). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, Belgium, Article 68, 8 pages. <https://doi.org/10.4108/ICST.SIMUTOOLS2009.5552>
- [39] M. Garcia, P. Fuentes, M. Odriozola, E. Vallejo, and R. Beivide. 2014. FOGSim Interconnection Network Simulator. <https://github.com/fuentesp/fgsim>
- [40] Lukas Gianinazzi, Tal Ben-Nun, Maciej Besta, Saleh Ashkboos, Yves Baumann, Piotr Luczynski, and Torsten Hoefler. 2023. The Spatial Computer: A Model for Energy-Efficient Parallel Computation. <https://doi.org/10.48550/arXiv.2205.04934>
- [41] Taylor Groves, Ryan E. Grant, Scott Hemmer, Simon Hammond, Michael Levenhagen, and Dorian C. Arnold. 2016. (SAI) Stalled, Active and Idle: Characterizing Power and Performance of Large-Scale Dragonfly Networks. In *Proceedings of the IEEE International Conference on Cluster Computing* (Taipei, Taiwan) (CLUSTER '16). IEEE Press, Washington, DC, USA, 50–59. <https://doi.org/10.1109/CLUSTER.2016.52>
- [42] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W. Moore, Gianni Antichi, and Marcin Wójcik. 2017. 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* (Los Angeles, CA, USA) (SIGCOMM '17). Association for Computing Machinery, New York, NY, USA, 29–42. <https://doi.org/10.1145/3098822.3098825>
- [43] Keqiang He, Eric Rozner, Kanak Agarwal, Wes Felter, John Carter, and Aditya Akella. 2015. Presto: Edge-based Load Balancing for Fast Datacenter Networks. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 465–478. <https://doi.org/10.1145/2829988.2787507>
- [44] Keqiang He, Eric Rozner, Kanak Agarwal, Yu (Jason) Gu, Wes Felter, John Carter, and Aditya Akella. 2016. AC/DC TCP: Virtual Congestion Control Enforcement for Datacenter Networks. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) (SIGCOMM '16). Association for Computing Machinery, New York, NY, USA, 244–257. <https://doi.org/10.1145/2934872.2934903>
- [45] Torsten Hoefler, Amnon Barak, Zvi Drezner, Amnon Shilo, Marc Snir, William Gropp, Maciej Besta, Salvatore Di Girolamo, Konstantin Taranov, Grzegorz Kwaśniewski, et al. 2020. High-Performance Distributed Memory Systems – From Supercomputers to Data Centers. In *34th International Symposium on Distributed Computing* (Virtual Event) (DISC '20). ETH Zurich, Scalable Parallel Computing Laboratory, Zurich, Switzerland. <https://spcl.inf.ethz.ch/Publications/index.php?pub=386> Keynote Talk.
- [46] Torsten Hoefler, Salvatore Di Girolamo, Konstantin Taranov, Ryan E. Grant, and Ron Brightwell. 2017. sPIN: High-Performance streaming Processing In the Network. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, CO, USA) (SC '17). Association for Computing Machinery, New York, NY, USA, Article 59, 16 pages. <https://doi.org/10.1145/3126908.3126970>
- [47] C. Hopps. 2000. RFC2992: Analysis of an Equal-Cost Multi-Path Algorithm.
- [48] Jiawei Huang, Weihe Li, Qi Li, Tao Zhang, Pingping Dong, and Jianxin Wang. 2020. Tuning High Flow Concurrency for MPTCP in Data Center Networks. *Journal of Cloud Computing* 9, Article 13 (Feb. 2020), 15 pages. <https://doi.org/10.1186/s13677-020-00160-3>
- [49] Jaehyun Hwang, Joon Yoo, and Nakjung Choi. 2014. Deadline and Incast Aware TCP for Cloud Data Center Networks. *Computer Networks* 68 (Aug. 2014), 20–34. <https://doi.org/10.1016/j.comnet.2013.12.002>
- [50] Patrick Iff, Maciej Besta, Matheus Cavalcante, Tim Fischer, Luca Benini, and Torsten Hoefler. 2023. HexaMesh: Scaling to Hundreds of Chiplets with an Optimized Chiplet Arrangement. In *Proceedings of the 2023 60th ACM/IEEE Design Automation Conference* (San Francisco, CA, USA) (DAC '23). IEEE Press, Washington, DC, USA, 1–6. <https://doi.org/10.1109/DAC56929.2023.10248006>
- [51] Patrick Iff, Maciej Besta, Matheus Cavalcante, Tim Fischer, Luca Benini, and Torsten Hoefler. 2023. Sparse Hamming Graph: A Customizable Network-on-Chip Topology. In *Proceedings of the 2023 60th ACM/IEEE Design Automation Conference* (San Francisco, CA, USA) (DAC '23). IEEE Press, Washington, DC, USA, 1–6. <https://doi.org/10.1109/DAC56929.2023.10247754>
- [52] Patrick Iff, Benigna Bruggmann, Blaise Morel, Maciej Besta, Luca Benini, and Torsten Hoefler. 2025. RapidChiplet: A Toolchain for Rapid Design Space Exploration of Chiplet Architectures. In *Proceedings of the 22nd ACM International Conference on Computing Frontiers* (Cagliari, Italy) (CF '25). Association for Computing Machinery, New York, NY, USA, 7 pages. <https://doi.org/10.48550/arXiv.2311.06081>
- [53] Teerawat Issariyakul and Ekram Hossain. 2009. Introduction to Network Simulator 2 (NS2). In *Introduction to Network Simulator NS2*. Springer US, Boston, MA, USA, 1–18. [https://doi.org/10.1007/978-0-387-71760-9\\_2](https://doi.org/10.1007/978-0-387-71760-9_2)
- [54] Nikhil Jain, Abhinav Bhatel, Xiang Ni, Todd Gamblin, and Laxmikant V. Kale. 2017. Partitioning Low-Diameter Networks to Eliminate Inter-Job Interference. In *Proceedings of the 31st IEEE International Parallel and Distributed Processing Symposium* (Orlando, FL, USA) (IPDPS '17). IEEE Press, Washington, DC, USA, 439–448. <https://doi.org/10.1109/IPDPS.2017.91>
- [55] Nikhil Jain, Abhinav Bhatel, Sam White, Todd Gamblin, and Laxmikant V. Kale. 2016. Evaluating HPC Networks via Simulation of Parallel Workloads. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Salt Lake City, UT, USA) (SC '16). IEEE Press, Washington, DC, USA, Article 14, 12 pages. <https://doi.org/10.1109/SC.2016.13>
- [56] Nan Jiang, Daniel U. Becker, George Michelogiannakis, James Balfour, Brian Towles, D. E. Shaw, John Kim, and William J. Dally. 2013. A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software* (Austin, TX, USA) (ISPASS '13). IEEE Press, Washington, DC, USA, 86–96. <https://doi.org/10.1109/ISPASS.2013.6557149>
- [57] Sangeetha Abdu Jyothi, Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla. 2016. Measuring and Understanding Throughput of Network Topologies. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Salt Lake City, UT, USA) (SC '16). IEEE Press, Washington, DC, USA, Article 65, 12 pages. <https://doi.org/10.1109/SC.2016.64>
- [58] Bengi Karacali, John M. Tracey, Paul G. Crumley, and Claude Basso. 2018. Assessing Cloud Network Performance. In *Proceedings of the IEEE International Conference on Communications* (Kansas City, MO, USA) (ICC '18). IEEE Press, Washington, DC, USA, 1–7. <https://doi.org/10.1109/ICC.2018.8422770>
- [59] Simon Kassing, Asaf Valadarsky, Gal Shahaf, Michael Schapira, and Ankit Singla. 2017. Beyond Fat-Trees without Antennae, Mirrors, and Disco-Balls. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) (SIGCOMM '17). Association for Computing Machinery, New York, NY, USA, 281–294. <https://doi.org/10.1145/3098822.3098836>
- [60] Georgios Kathareios, Cyriel Minkenbergh, Bogdan Prisacari, German Rodriguez, and Torsten Hoefler. 2015. Cost-Effective Diameter-Two Topologies: Analysis and Evaluation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Austin, TX, USA) (SC '15). Association for Computing Machinery, New York, NY, USA, Article 36, 11 pages. <https://doi.org/10.1145/2807591.2807652>
- [61] Naga Katta, Mukesh Hira, Aditi Ghag, Changhoon Kim, Isaac Keslassy, and Jennifer Rexford. 2016. 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* (Atlanta, GA, USA) (HotNets '16). Association for Computing



- Machinery, New York, NY, USA, 155–161. <https://doi.org/10.1145/3005745.3005751>
- [62] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. 2016. HULA: Scalable Load Balancing Using Programmable Data Planes. In *Proceedings of the Symposium on SDN Research* (Santa Clara, CA, USA) (SOSR '16). Association for Computing Machinery, New York, NY, USA, Article 10, 12 pages. <https://doi.org/10.1145/2890955.2890968>
- [63] John Kim, William J. Dally, and Dennis Abts. 2007. Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks. *SIGARCH Comput. Archit. News* 35, 2 (June 2007), 126–137. <https://doi.org/10.1145/1273440.1250679>
- [64] John Kim, William J. Dally, Steve Scott, and Dennis Abts. 2008. Technology-Driven, Highly-Scalable Dragonfly Topology. *SIGARCH Comput. Archit. News* 36, 3 (June 2008), 77–88. <https://doi.org/10.1145/1394608.1382129>
- [65] Kartik Lakhota, Maciej Besta, Laura Monroe, Kelly Isham, Patrick Iff, Torsten Hoefler, and Fabrizio Petrini. 2022. PolarFly: A Cost-Effective and Flexible Low-Diameter Topology. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Dallas, TX, USA) (SC '22). IEEE Press, Washington, DC, USA, Article 12, 15 pages. <https://doi.org/10.1109/SC41404.2022.00017>
- [66] Kartik Lakhota, Kelly Isham, Laura Monroe, Maciej Besta, Torsten Hoefler, and Fabrizio Petrini. 2023. In-Network Allreduce with Multiple Spanning Trees on PolarFly. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures* (Orlando, FL, USA) (SPAA '23). Association for Computing Machinery, New York, NY, USA, 165–176. <https://doi.org/10.1145/3558481.3591073>
- [67] Kartik Lakhota, Laura Monroe, Kelly Isham, Maciej Besta, Nils Blach, Torsten Hoefler, and Fabrizio Petrini. 2024. PolarStar: Expanding the Horizon of Diameter-3 Networks. In *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures* (Nantes, France) (SPAA '24). Association for Computing Machinery, New York, NY, USA, 345–357. <https://doi.org/10.1145/3626183.3659975>
- [68] Fei Lei, Dezun Dong, Xiangke Liao, and José Duato. 2020. Bundlefly: A Low-Diameter Topology for Multicore Fiber. In *Proceedings of the 34th ACM International Conference on Supercomputing* (Barcelona, Spain) (ICS '20). Association for Computing Machinery, New York, NY, USA, Article 20, 11 pages. <https://doi.org/10.1145/3392717.3392747>
- [69] Charles E. Leiserson, Zahi S. Abuhamdeh, David C. Douglas, Carl R. Feynman, Mahesh N. Ganmukhi, Jeffrey V. Hill, W. Daniel Hillis, Bradley C. Kuszmaul, Margaret A. St. Pierre, David S. Wells, Monica C. Wong-Chan, Shaw-Wen Yang, and Robert Zak. 1996. The Network Architecture of the Connection Machine CM-5. *J. Parallel and Distrib. Comput.* 33, 2 (March 1996), 145–158. <https://doi.org/10.1006/jpdc.1996.0033>
- [70] Dongsheng Li, Xicheng Lu, and Jinshu Su. 2004. Graph-Theoretic Analysis of Kautz Topology and DHT Schemes. In *Proceedings of the IFIP International Conference on Network and Parallel Computing* (Wuhan, China) (NPC '04). Hai Jin, Guang R. Gao, Zhiwei Xu, and Hao Chen (Eds.). Springer, Berlin/Heidelberg, Germany, 308–315. [https://doi.org/10.1007/978-3-540-30141-7\\_45](https://doi.org/10.1007/978-3-540-30141-7_45)
- [71] Yu Li and Deng Pan. 2013. OpenFlow Based Load Balancing for Fat-Tree Networks with Multipath Support. In *Proceedings of the 12th IEEE International Conference on Communications* (Budapest, Hungary) (ICC '13). IEEE Press, Washington, DC, USA, 1–5. <https://users.cs.fiu.edu/~pand/publications/13icc-yu.pdf>
- [72] Xiangke Liao, Liquan Xiao, Canqun Yang, and Yutong Lu. 2014. MilkyWay-2 Supercomputer: System and Application. *Frontiers of Computer Science* 8, 3 (May 2014), 345–356. <https://doi.org/10.1007/s11704-014-3501-3>
- [73] Yifei Lu. 2016. SED: An SDN-Based Explicit-Deadline-Aware TCP for Cloud Data Center Networks. *Tsinghua Science and Technology* 21, 5 (Oct. 2016), 491–499. <https://doi.org/10.1109/TST.2016.7590318>
- [74] Yuanwei Lu, Guo Chen, Bojie Li, Kun Tan, Yongqiang Xiong, Peng Cheng, Jiansong Zhang, Enhong Chen, and Thomas Moscibroda. 2018. Multi-Path Transport for RDMA in Datacenters. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation* (Renton, WA, USA) (NSDI '18). USENIX Association, Berkeley, CA, USA, 357–371. <https://www.usenix.org/conference/nsdi18/presentation/lu>
- [75] Radhika Mittal, Vinh The Lam, Nandita Dukkkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based Congestion Control for the Datacenter. *SIGCOMM Comput. Commun. Rev.* 45, 4 (Aug. 2015), 537–550. <https://doi.org/10.1145/2829988.2787510>
- [76] Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John Ousterhout. 2018. Homa: A Receiver-Driven Low-Latency Transport Protocol Using Network Priorities. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) (SIGCOMM '18). Association for Computing Machinery, New York, NY, USA, 221–235. <https://doi.org/10.1145/3230543.3230564>
- [77] Misbah Mubarak, Christopher D. Carothers, Robert B. Ross, and Philip H. Carns. 2012. Modeling a Million-Node Dragonfly Network Using Massively Parallel Discrete-Event Simulation. In *Companion of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Salt Lake City, UT, USA) (SC Companion '12). IEEE Press, Washington, DC, USA, 366–376. <https://doi.org/10.1109/SC.Companion.2012.56>
- [78] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. 2009. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. *SIGCOMM Comput. Commun. Rev.* 39, 4 (Aug. 2009), 39–50. <https://doi.org/10.1145/1594977.1592575>
- [79] University of Washington NS-3 Consortium. 2024. Network Simulator 3. <https://www.nsnam.org/> accessed 2025-04-07.
- [80] Sabine R. Ohring, Maximilian Ibel, Sajal K. Das, and Mohan J. Kumar. 1995. On Generalized Fat Trees. In *Proceedings of the 9th International Parallel Processing Symposium* (Santa Barbara, CA, USA) (IPPS '95). IEEE Press, Washington, DC, USA, 37–44. <https://doi.org/10.1109/IPPS.1995.395911>
- [81] Vladimir Olteanu and Costin Raiciu. 2016. Datacenter Scale Load Balancing for Multipath Transport. In *Proceedings of the Workshop on Hot Topics in Middleboxes and Network Function Virtualization* (Florianopolis, Brazil) (HotMiddlebox '16). Association for Computing Machinery, New York, NY, USA, 20–25. <https://doi.org/10.1145/2940147.2940154>
- [82] Minkyung Park, Sungmin Sohn, Kwangwook Kwon, and Ted Taekyoung Kwon. 2019. DataPass: Credit-Based Multipath Transmission for Load Balancing in Data Centers. *Journal of Communications and Networks* 21, 6 (Dec. 2019), 558–568. <https://doi.org/10.1109/JCN.2019.000047>
- [83] Joshua Pelkey and George Riley. 2011. Distributed Simulation with MPI in NS-3. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques* (Barcelona, Spain) (SIMUTools '11). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, Belgium, 410–414. <https://dl.acm.org/doi/10.5555/2151054.2151128>
- [84] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. 2014. Fastpass: A Centralized "Zero-Queue" Datacenter Network. *SIGCOMM Comput. Commun. Rev.* 44, 4 (Aug. 2014), 307–318. <https://doi.org/10.1145/2740070.2626309>
- [85] Fabrizio Petrini and Marco Vanneschi. 1997. k-ary n-Trees: High Performance Networks for Massively Parallel Architectures. In *Proceedings of the 11th International Parallel Processing Symposium* (Geneva, Switzerland) (IPPS '97). IEEE Press, Washington, DC, USA, 87–93. <https://doi.org/10.1109/IPPS.1997.580853>
- [86] Bogdan Prisacari, German Rodriguez, Philip Heidelberger, Dong Chen, Cyriel Minkenberg, and Torsten Hoefler. 2014. 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* (Vancouver, BC, Canada) (HPDC '14). Association for Computing Machinery, New York, NY, USA, 129–140. <https://doi.org/10.1145/2600212.2600225>
- [87] Bogdan Prisacari, German Rodriguez, Ana Jokanovic, and Cyriel Minkenberg. 2014. Randomizing Task Placement and Route Selection Do Not Randomize Traffic (Enough). *Design Automation for Embedded Systems* 18, 3 (2014), 171–182. <https://doi.org/10.1007/s10617-014-9133-x>
- [88] Bogdan Prisacari, German Rodriguez, Cyriel Minkenberg, Marina Garcia, Enrique Vallejo, and Ramon Beivide. 2015. Performance Optimization of Load Imbalanced Workloads in Large Scale Dragonfly Systems. In *Proceedings of the IEEE 16th International Conference on High Performance Switching and Routing* (Budapest, Hungary) (HPSR '15). IEEE Press, Washington, DC, USA, 1–6. <https://doi.org/10.1109/HPSR.2015.7483107>
- [89] Bogdan Prisacari, German Rodriguez, Cyriel Minkenberg, and Torsten Hoefler. 2013. Bandwidth-Optimal All-to-All Exchanges in Fat Tree Networks. In *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing* (Eugene, OR, USA) (ICS '13). Association for Computing Machinery, New York, NY, USA, 139–148. <https://doi.org/10.1145/2464996.2465434>
- [90] Bogdan Prisacari, German Rodriguez, Cyriel Minkenberg, and Torsten Hoefler. 2013. Fast Pattern-Specific Routing for Fat Tree Networks. *ACM Trans. Archit. Code Optim.* 10, 4, Article 36 (Dec. 2013), 25 pages. <https://doi.org/10.1145/2541228.2555293>
- [91] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. 2011. Improving Datacenter Performance and Robustness with Multipath TCP. *SIGCOMM Comput. Commun. Rev.* 41, 4 (Aug. 2011), 266–277. <https://doi.org/10.1145/2043164.2018467>
- [92] Costin Raiciu, Christopher Pluntke, Sebastien Barre, Adam Greenhalgh, Damon Wischik, and Mark Handley. 2010. Data Center Networking with Multipath TCP. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (Monterey, CA, USA) (Hotnets-IX). Association for Computing Machinery, New York, NY, USA, Article 10, 6 pages. <https://doi.org/10.1145/1868447.1868457>
- [93] Arun F. Rodrigues, Gwendolyn Renae Voskuilen, Simon David Hammond, and Karl Scott Hemmert. 2016. *Structural Simulation Toolkit (SST)*. Technical Report. Sandia National Lab. <https://www.osti.gov/biblio/1365223>
- [94] Wile Seherly and Charles Clancy. 2017. Flow Optimization in Data Centers with Clos Networks in Support of Cloud Applications. *IEEE Transactions on Network and Service Management* 14, 4 (Dec. 2017), 847–859. <https://doi.org/10.1109/TNSM.2017.2761321>

- [95] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P Brighten Godfrey. 2012. Jellyfish: Networking Data Centers Randomly. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation* (San Jose, CA, USA) (NSDI '12). USENIX Association, Berkeley, CA, USA, 225–238. <https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/singla>
- [96] Edgar Solomonik, Maciej Besta, Flavio Vella, and Torsten Hoefler. 2017. Scaling Betweenness Centrality Using Communication-Efficient Sparse Matrix Multiplication. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, CO, USA) (SC '17). Association for Computing Machinery, New York, NY, USA, Article 47, 14 pages. <https://doi.org/10.1145/3126908.3126971>
- [97] Edgar Solomonik, Erin Carson, Nicholas Knight, and James Demmel. 2014. Tradeoffs Between Synchronization, Communication, and Computation in Parallel Linear Algebra Computations. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures* (Prague, Czech Republic) (SPAA '14). Association for Computing Machinery, New York, NY, USA, 307–318. <https://doi.org/10.1145/2612669.2612671>
- [98] Edgar Solomonik and James Demmel. 2011. Communication-Optimal Parallel 2.5D Matrix Multiplication and LU Factorization Algorithms. In *Proceedings of the European Conference on Parallel Processing (Euro-Par '11)*, Emmanuel Jeannot, Raymond Namyst, and Jean Roman (Eds.). Lecture Notes in Computer Science, Vol. 6853. Springer, Berlin/Heidelberg, Germany, 90–109. [https://doi.org/10.1007/978-3-642-23397-5\\_10](https://doi.org/10.1007/978-3-642-23397-5_10)
- [99] Adrian Tate, Amir Kamil, Anshu Dubey, Armin Größlinger, Brad Chamberlain, Brice Goglin, Carter Edwards, Chris J. Newburn, David Padua, Didem Unat, Emmanuel Jeannot, Frank Hannig, Tobias Gysi, Hatem Ltaief, James Sexton, Jesús Labarta, John Shalf, Karl Furlinger, Kathryn O'Brien, Leonidas Linardakis, Maciej Besta, Marie-Christine Sawley, Mark Abraham, Mauro Bianco, Miquel Pericás, Naoya Maruyama, Paul Kelly, Peter Messmer, Robert B. Ross, Romain Cledat, Satoshi Matsuoka, Thomas Schulthess, Torsten Hoefler, and Vitus Leung. 2014. Programming Abstractions for Data Locality. In *Proceedings of the Workshop on Programming Abstractions for Data Locality* (Lugano, Switzerland) (PADAL '14). Swiss National Supercomputing Center, Lugano, Switzerland, 54 pages.
- [100] Asaf Valadarsky, Michael Dinitz, and Michael Schapira. 2015. Xpander: Unveiling the Secrets of High-Performance Datacenters. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks* (Philadelphia, PA, USA) (HotNets '15). Association for Computing Machinery, New York, NY, USA, Article 16, 7 pages. <https://doi.org/10.1145/2834050.2834059>
- [101] Balajee Vamanan, Jahangir Hasan, and T.N. Vijaykumar. 2012. Deadline-Aware Datacenter TCP (D2TCP). *SIGCOMM Comput. Commun. Rev.* 42, 4 (Aug. 2012), 115–126. <https://doi.org/10.1145/2377677.2377709>
- [102] Erico Vanini, Rong Pan, Mohammad Alizadeh, Parvin Taheri, and Tom Edsall. 2017. Let It Flow: Resilient Asymmetric Load Balancing with Flowlet Switching. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation* (Boston, MA, USA) (NSDI '17). USENIX Association, Berkeley, CA, USA, 407–420. <https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/vanini>
- [103] András Varga and Rudolf Hornig. 2008. 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* (Marseille, France) (Simutools '08). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, Belgium, Article 60, 10 pages. <https://dl.acm.org/doi/10.5555/1416222.1416290>
- [104] Jörg Widmer, Robert Denda, and Martin Mauve. 2001. A Survey on TCP-Friendly Congestion Control. *IEEE Network* 15, 3 (May 2001), 28–37. <https://doi.org/10.1109/65.923938>
- [105] Noah Wolfe, Christopher D. Carothers, Misbah Mubarak, Robert Ross, and Philip Carns. 2016. Modeling a Million-Node Slim Fly Network Using Parallel Discrete-Event Simulation. In *Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation* (Banff, AB, Canada) (SIGSIM-PADS '16). Association for Computing Machinery, New York, NY, USA, 189–199. <https://doi.org/10.1145/2901378.2901389>
- [106] Stephen Young, Sinan Aksoy, Jesun Firoz, Roberto Gioiosa, Tobias Hagge, Mark Kempton, Juan Escobedo, and Mark Raugas. 2022. Spectralfly: Ramanujan Graphs as Flexible and Efficient Interconnection Networks. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium* (Lyon, France) (IPDPS '22). IEEE Press, Washington, DC, USA, 1040–1050. <https://doi.org/10.1109/IPDPS53621.2022.00105>
- [107] Xin Yuan, Santosh Mahapatra, Michael Lang, and Scott Pakin. 2014. LFTI: A New Performance Metric for Assessing Interconnect Designs for Extreme-Scale HPC Systems. In *Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium* (Phoenix, AZ, USA) (IPDPS '14). IEEE Press, Washington, DC, USA, 273–282. <https://doi.org/10.1109/IPDPS.2014.38>
- [108] Xin Yuan, Santosh Mahapatra, Wickus Nienaber, Scott Pakin, and Michael Lang. 2013. 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* (Denver, CO, USA) (SC '13). Association for Computing Machinery, New York, NY, USA, Article 36, 11 pages. <https://doi.org/10.1145/2503210.2503229>
- [109] Hong Zhang, Junxue Zhang, Wei Bai, Kai Chen, and Mosharaf Chowdhury. 2017. Resilient Datacenter Load Balancing in the Wild. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) (SIGCOMM '17). Association for Computing Machinery, New York, NY, USA, 253–266. <https://doi.org/10.1145/3098822.3098841>
- [110] Gengbin Zheng, Gunavardhan Kakulapati, and Laxmikant V. Kalé. 2004. BigSim: A Parallel Simulator for Performance Prediction of Extremely Large Parallel Machines. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium* (Santa Fe, NM, USA) (IPDPS '04). IEEE Press, Washington, DC, USA, 78–87. <https://doi.org/10.1109/IPDPS.2004.1303013>
- [111] Danyang Zhuo, Qiao Zhang, Vincent Liu, Arvind Krishnamurthy, and Thomas Anderson. 2016. Rack-Level Congestion Control. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks* (Atlanta, GA, USA) (HotNets '16). Association for Computing Machinery, New York, NY, USA, 148–154. <https://doi.org/10.1145/3005745.3005772>

## A APPENDIX

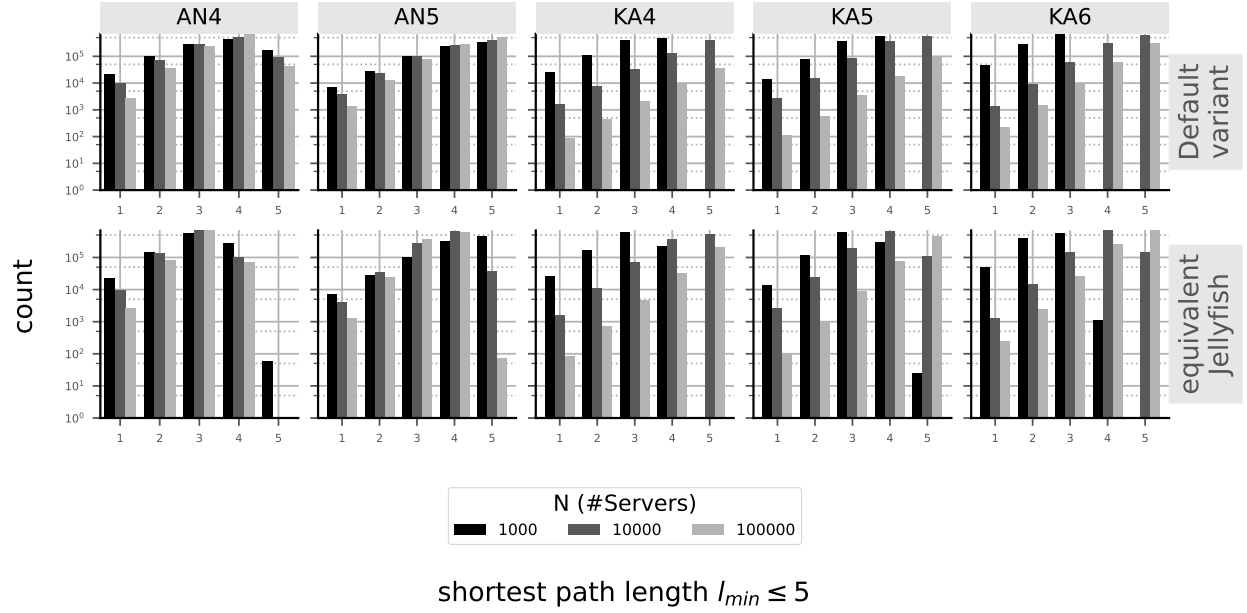


Figure 8: Shortest path plot for Kautz and Arrangement graphs. Kautz graphs have lower shortest paths compared to their Jellyfish equivalent for small networks. They also have lower shortest paths when compared to a small Arrangement network. A small Kautz graph has only a diameter of 3, while the Arrangement graph has already a diameter  $\geq 5$ . The analysis of the Arrangement graphs compared to their Jellyfish equilibrium, which is a Jellyfish topology built with the same hardware, looks fairly similar. The Jellyfish equilibrium has less shortest paths of higher lengths  $l = 5$ . The amount of shortest paths of low length (between 1 and 3) gets smaller, the bigger the network gets for all topologies. For Kautz graphs, these drop-offs are much more pronounced. There are about 100 times fewer shortest paths of lengths 1 when we compare a small Kautz graph with a large Kautz graph. In comparison, Arrangement networks have a drop-off between 5 and 10 times.

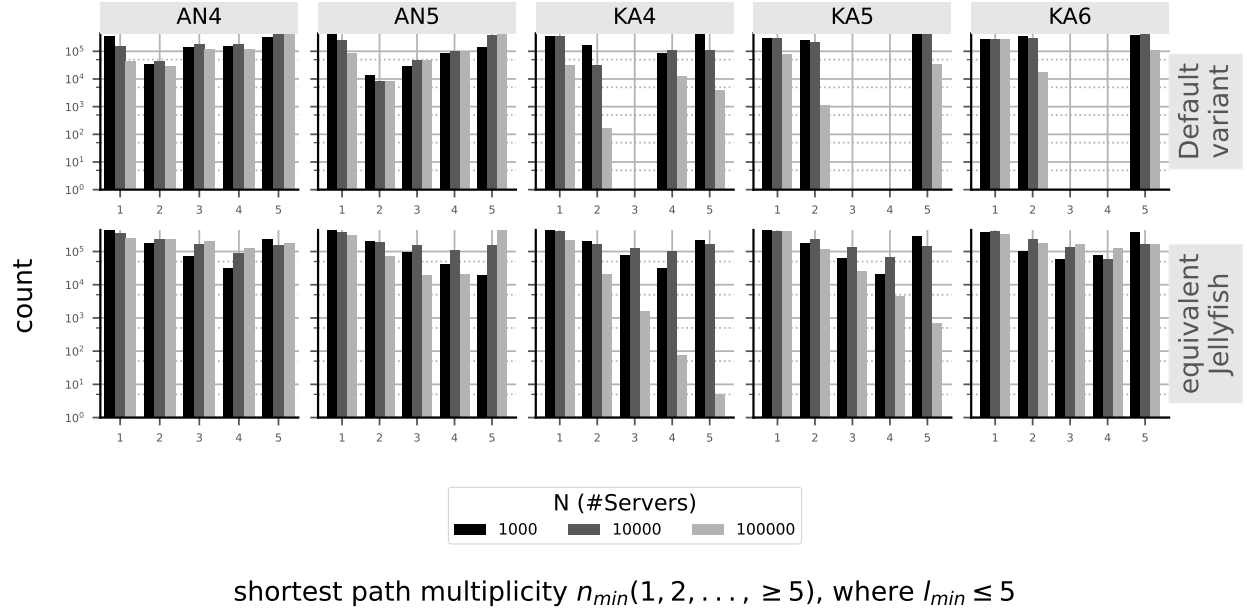


Figure 9: Multiplicity plot for Kautz and Arrangement graphs. For Arrangement networks, the shortest path density first decreases, then increases. In contrast, Kautz graphs have about the same amount of path multiplicity, no matter the value of  $n_{min}$ . In the large Kautz network, we can see a significant drop off in the amount of multiplicity. This drop off is less significant for  $K_{6,n}$ . For Arrangement networks, these numbers of multiplicities stay about the same. There is only a small shift, for  $n_{min} = 1$  the number of shortest path multiplicities sinks for higher N, but for  $n_{min} = 4$  or 5 they only sink slightly. Kautz graphs have a gap in the multiplicity counts. They either have a low amount (1 or 2) of shortest paths between any two nodes or a lot of overlap ( $\geq 4$ ) in the multiplicity. The analysis of the Jellyfish equivalent looks similar to the Arrangement networks. The Jellyfish equivalent of the Kautz graphs on the other hand do not have this gap in the multiplicity. For  $K_{4,n}$  and  $K_{5,n}$  the drop off in the multiplicity count can also be seen in the Jellyfish equivalent for  $N \approx 100,000$ . This cannot be seen for the Jellyfish equilibrium for  $K_{6,n}$ .

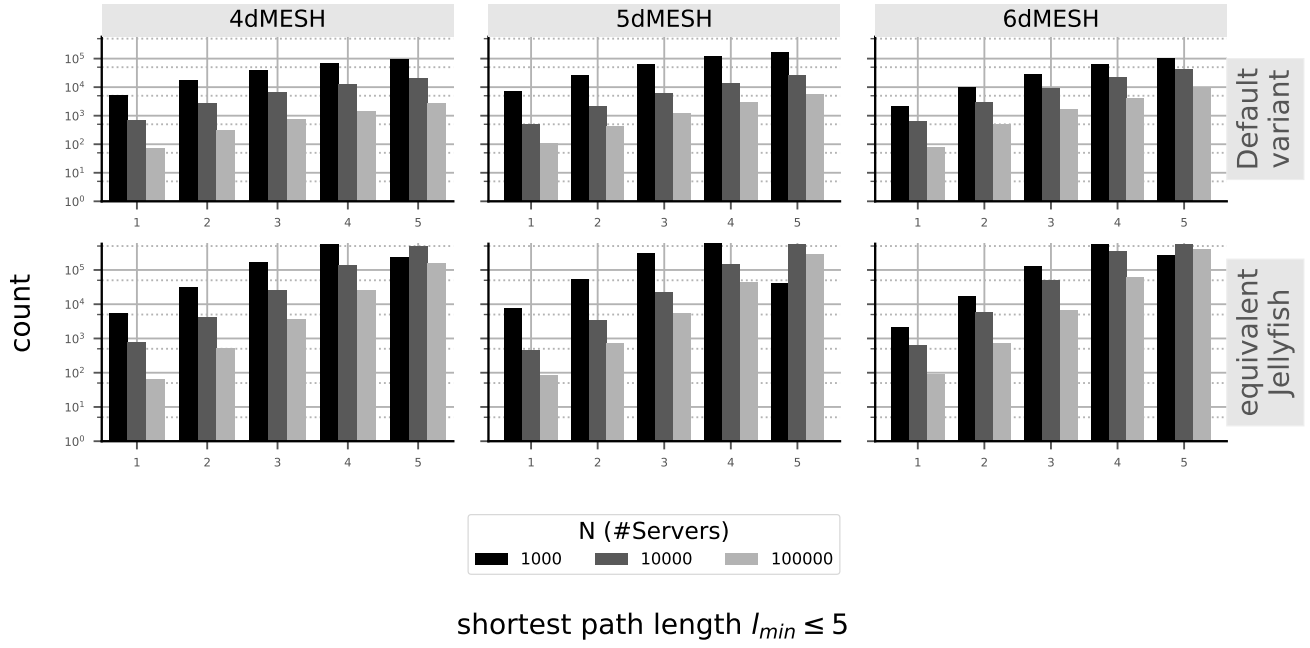


Figure 10: Shortest path plot for mesh topology of different dimensions. The number of shortest paths increases while  $l_{min}$  increases. For bigger networks, there are fewer shortest paths with a small value. The Jellyfish equivalent analysis looks fairly similar compared to the meshes. Only for  $l_{min} = 5$ , the Jellyfish network with higher  $N$  does not strictly have fewer amounts of shortest paths.

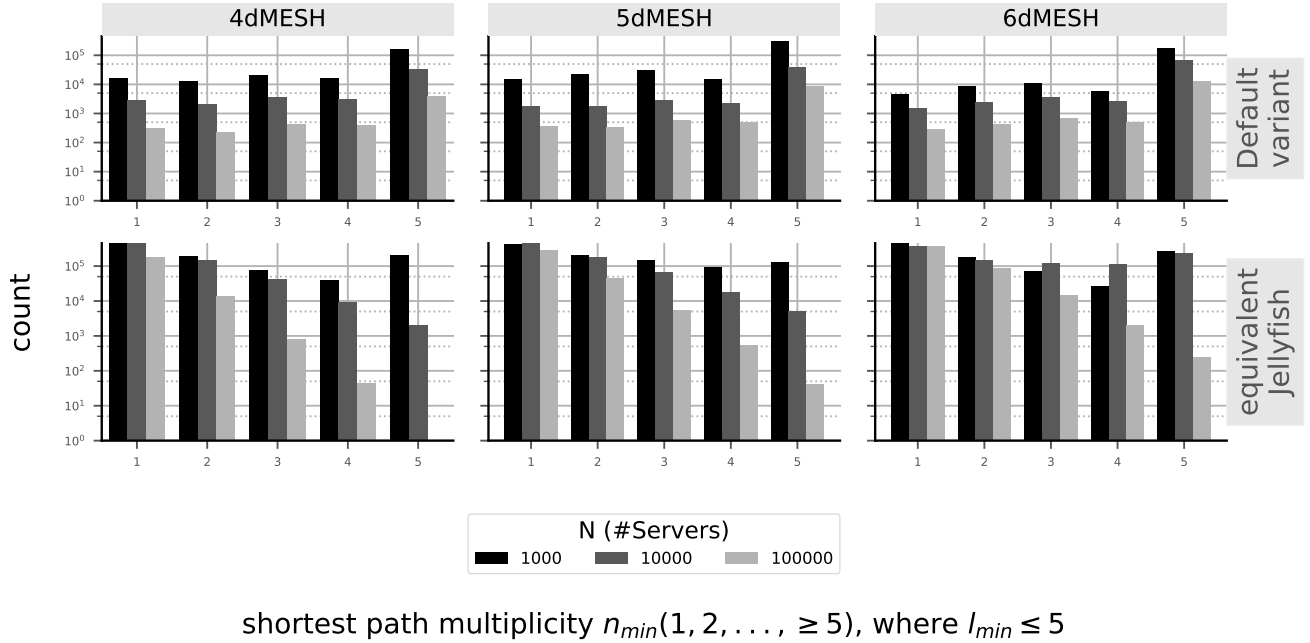


Figure 11: Multiplicity plot for meshes of different dimensions. As expected, the analysis looks fairly similar across all dimensions. For all sizes and all dimensions, the multiplicity count is the highest for  $n_{min} \geq 5$ . For bigger network sizes, the amount of shortest path multiplicity decreases with roughly the same factor, independent of  $n_{min}$ . In contrast, the Jellyfish equivalent networks have almost no difference in the multiplicity count for  $n_{min} = 1$ . For bigger values of  $n_{min}$ , the shortest path multiplicity drops even more compared to the original mesh, when we increase the network size.

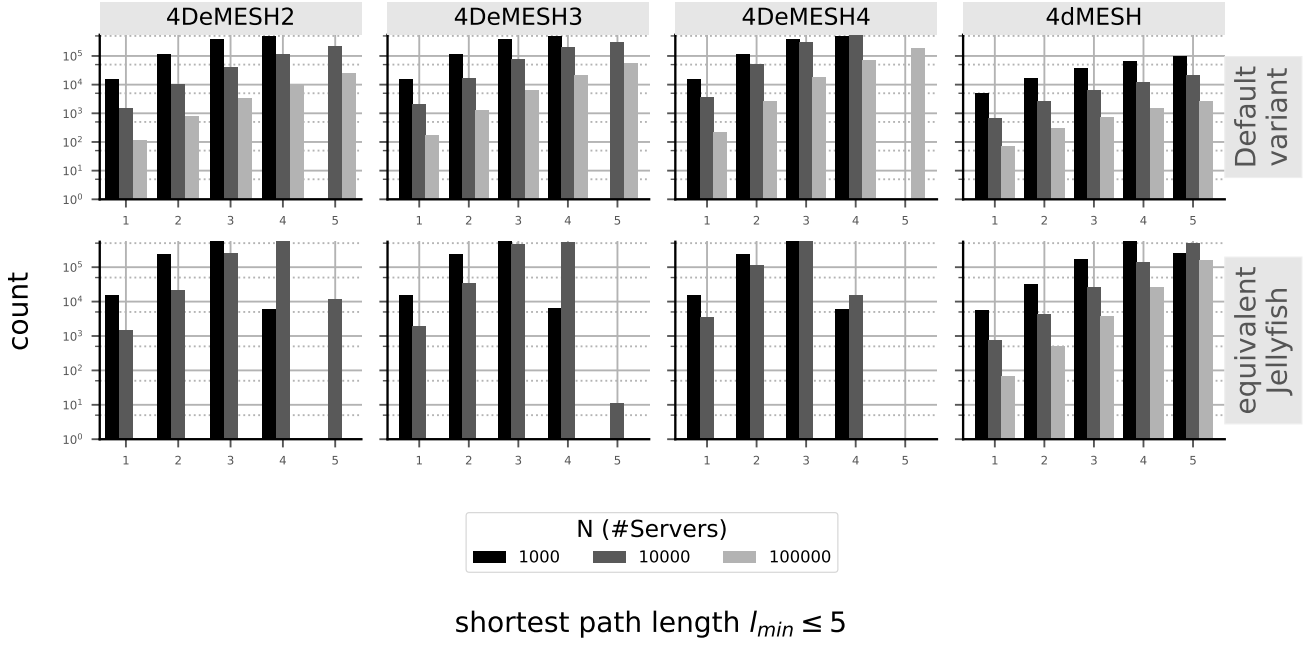


Figure 12: Shortest path plot for the 4D Express Meshes with different gap sizes. We choose the lowest  $g$ , such that for each Express Mesh 4DeMESH $x$  we have at least  $x$  additional express connections per dimension. As one can expect, the Express Meshes have lower shortest paths than the original mesh due to their express connections. For some network sizes, the Express Meshes have only a diameter of 4, while the standard 4D mesh has a diameter of  $\geq 5$ . While the number of shortest paths increase for bigger  $l$  in the Express Meshes, their equivalent Jellyfishes don't show a similar pattern. They increase first before they decrease for the biggest value of  $l$  where there still exists a shortest path.

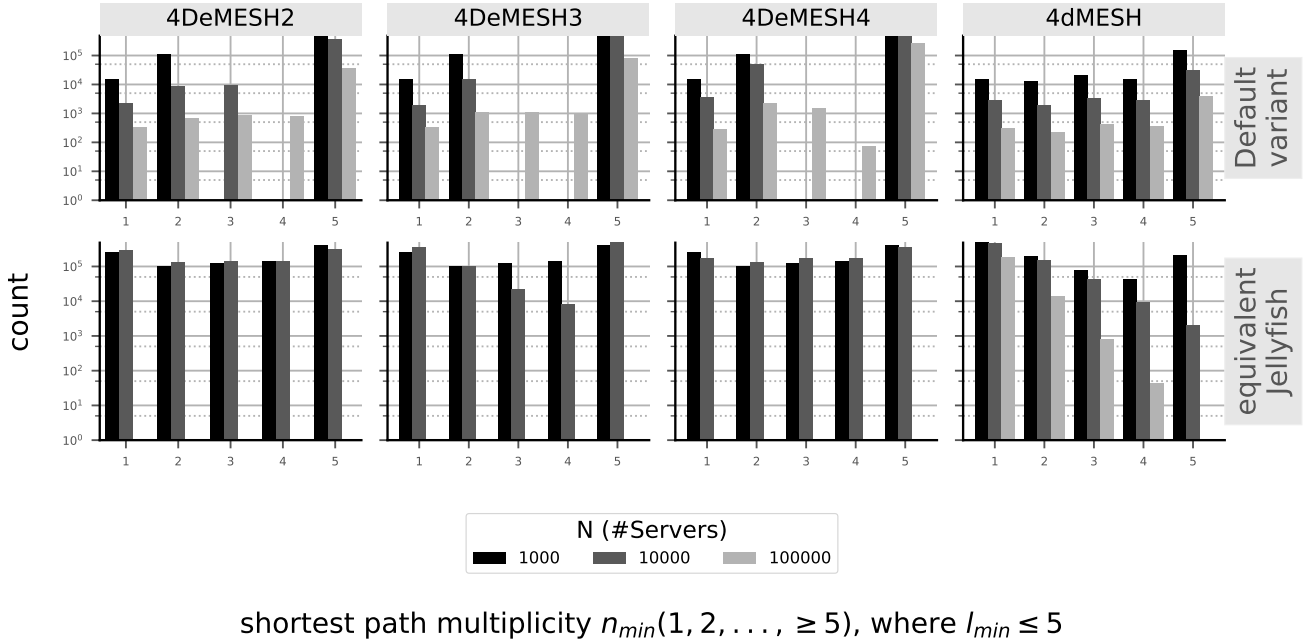


Figure 13: Multiplicity plot for Express Meshes of dimension 4. For small and medium-sized networks, there is a gap in the multiplicity counts. The length of the shortest path for any two nodes exists either once or twice, or  $\geq 5$  times. For large Express Meshes, this gap is not visible anymore. Express Meshes keep the high multiplicity of shortest paths, even for higher  $N$ , due to their express connections, especially for lower gap sizes. For all network sizes, the Express Meshes have the highest shortest paths multiplicity count for  $n_{min} \geq 5$ . There is a big difference in the multiplicity of the meshes compared to the equivalent Jellyfish. In contrast to the base Express Meshes, the Jellyfish equilibriums are well distributed along all values for  $n_{min}$  for small and medium networks.



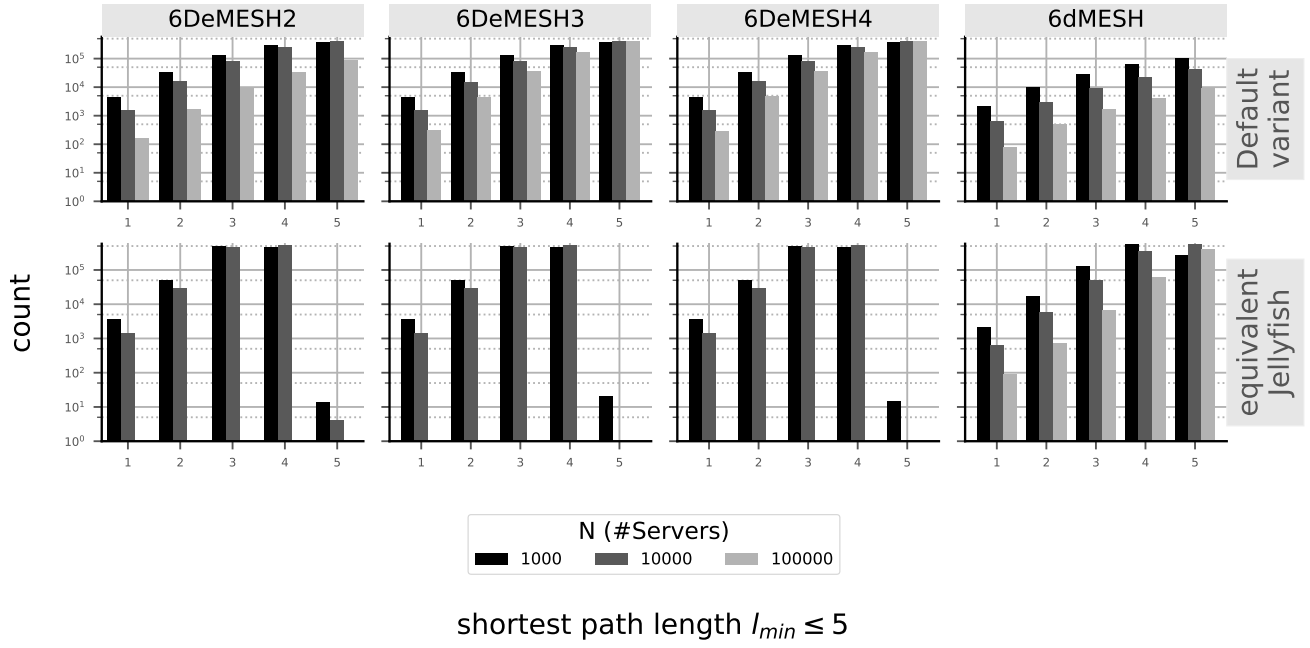


Figure 14: Shortest path plot for 6D Express Meshes with different gap sizes. We chose the lowest  $g$ , such that for each Express Mesh 6DeMESH $x$  we have at least  $x$  additional express connections per dimension. The shortest path analysis of these Express Meshes look similar to the analysis of their 4D counterpart. Express connections do not allow the 6D graph to become a diameter  $\leq 5$  network, since any path only traverses over one dimension. The Jellyfish equilibrium is also fairly similar when comparing the 6D to the 4D analysis. The amount of shortest paths grows as  $l$  grows, before dropping down for some outliers that have a shortest path length of 5.

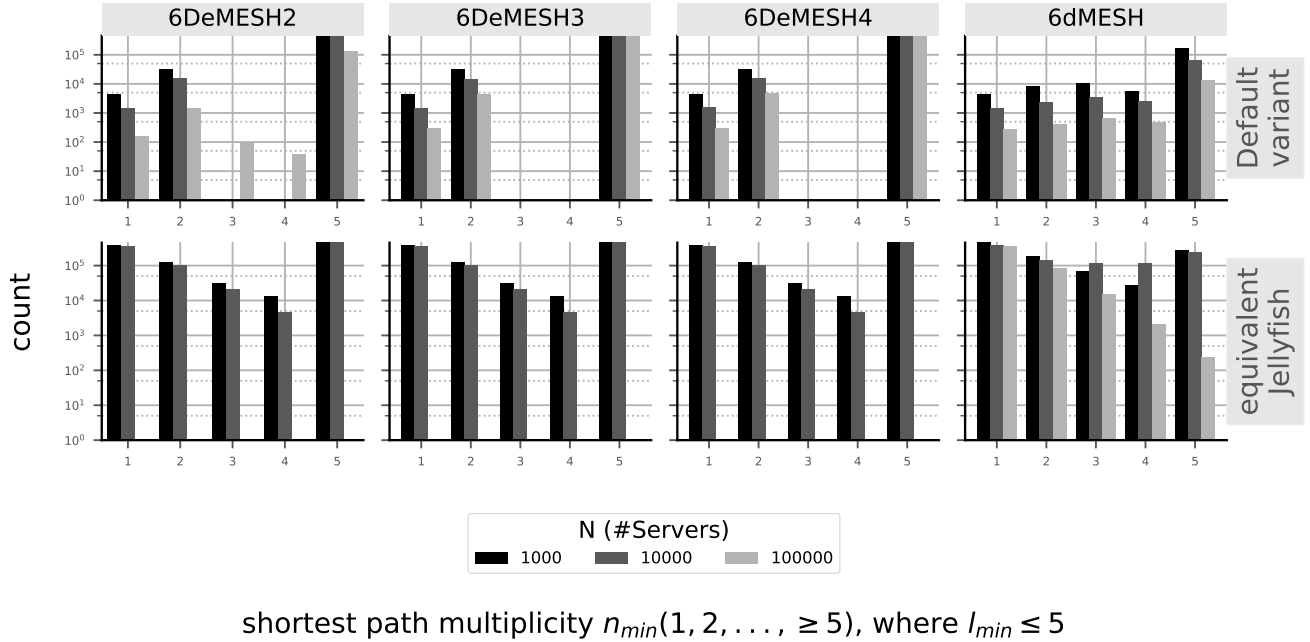


Figure 15: Multiplicity plot for Express Meshes of dimension 6. We can see again the gap in the amount of shortest path multiplicity. There either exist very few paths of length equal to the shortest path between two nodes, or there exists a lot of them (more than 4). Only the large Express Mesh with two additional express connections per dimension does not show this gap. Same as in the 4D case, most shortest paths have a multiplicity  $\geq 5$ . In case of the Jellyfish equilibrium, the shortest path multiplicity count drops at higher  $n_{min}$ . Just for  $n_{min} = 5$ , the multiplicity count increases, as they have as well the most shortest paths with multiplicity  $\geq 5$ .

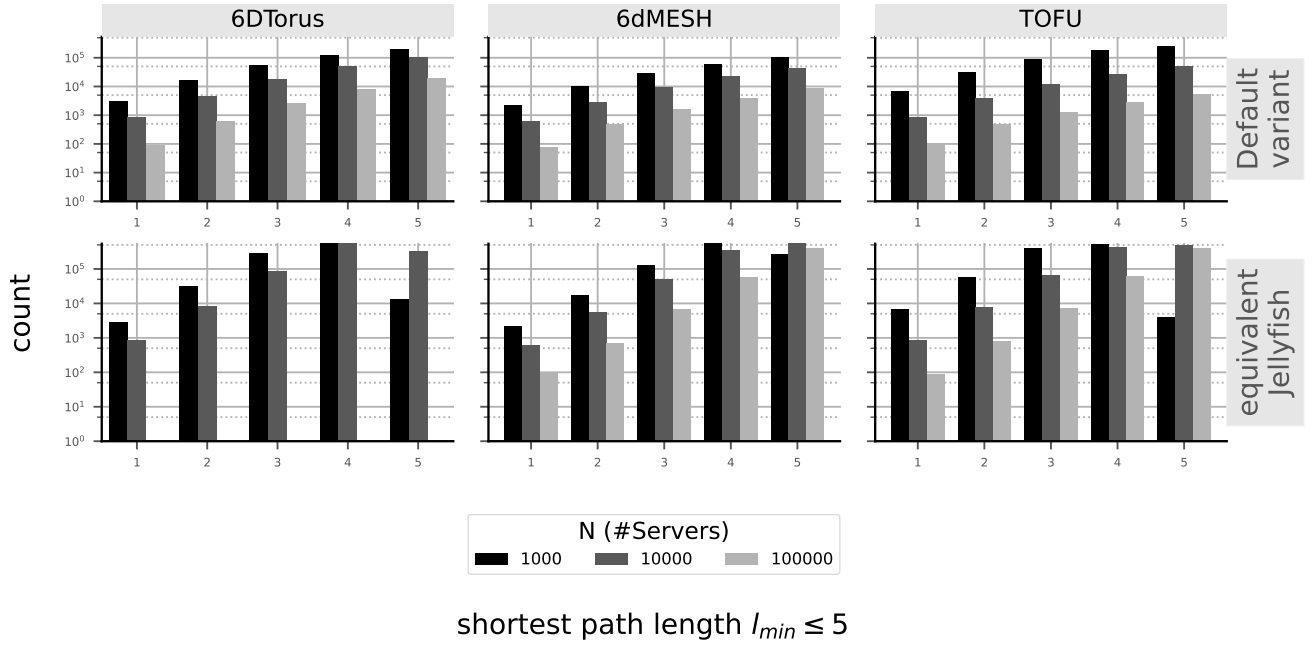


Figure 16: Shortest path plot for the Tofu topology, 6D torus and 6D mesh. As we have already seen in the analysis of the meshes, the amount of shortest paths sinks as the network size gets bigger. The bigger  $l$  gets, the more shortest paths exist of length  $l$ . All networks show similar patterns. Tofu has more shortest paths  $l_{min} \leq 5$  compared to all the other meshes for small networks, but is worse for bigger networks. The same is true when we compare Tofu to the equivalent Jellyfish.

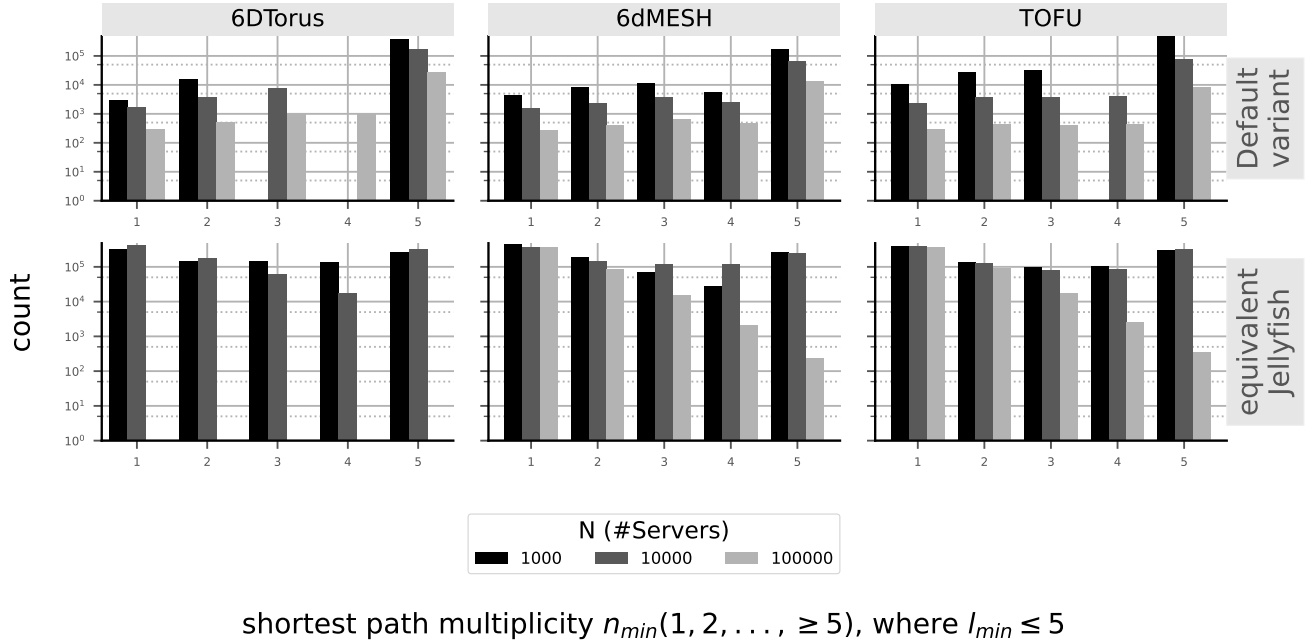


Figure 17: Multiplicity plot for the Tofu network, 6D torus and 6D mesh. Similar as for the meshes, all the topologies have the most multiplicity count where  $n_{min} \geq 5$ . The Tofu network has even the highest multiplicity count for  $n_{min} \geq 5$  for all small networks. In medium and large networks, Tofu does not have the most multiplicity count anymore. The bigger the network sizes are, the smaller the multiplicity counts get. In the analysis of the torus, we can find the already mentioned gap in the shortest path multiplicities for small and medium-size networks. Tofu has this gap only for small networks. The Jellyfish equilibrium of the Tofu network is well distributed along all values for  $n_{min}$  for small and medium networks. For the large network, there is a significant drop-off and the multiplicity is about 30 times smaller for  $n_{min} \geq 5$  compared to the small or medium network.

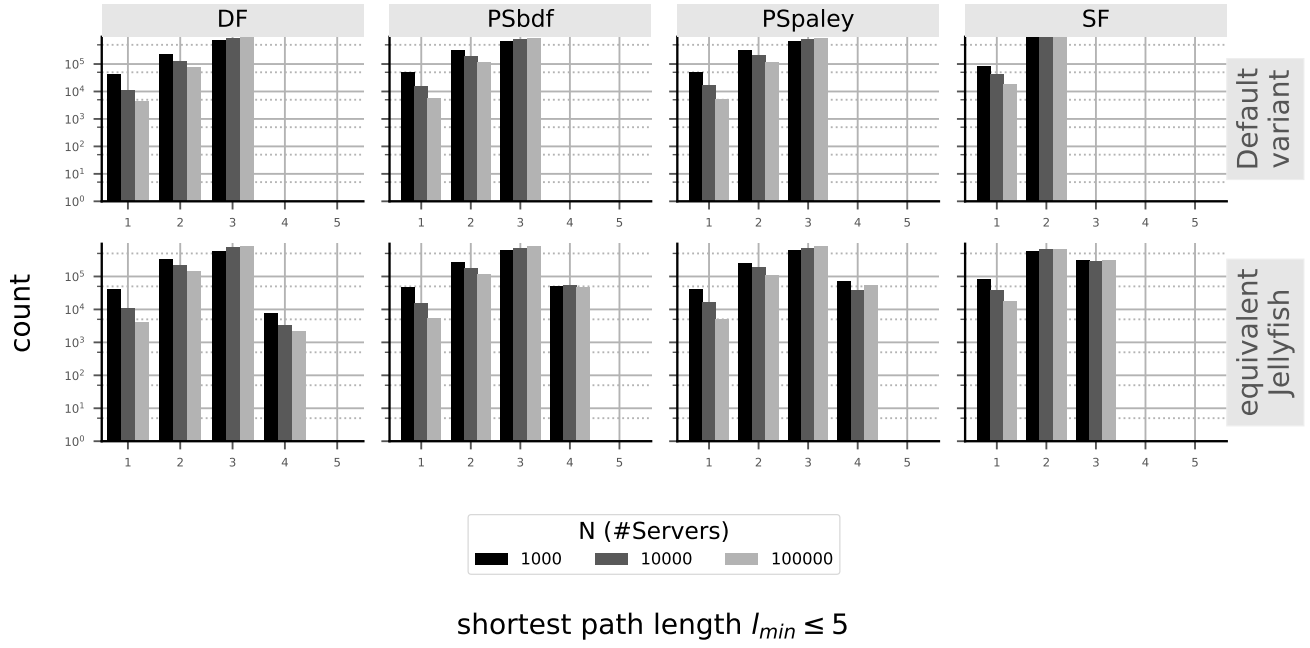


Figure 18: Shortest path plot for PolarStar compared to Dragonfly and Slim Fly. We can clearly see that all are diameter  $\leq 3$  topologies. All of the topologies have in general lower smallest paths compared to their equivalent Jellyfish graph. Slim Fly has only shortest paths of length 2. All the others look fairly similar, having fewer shortest paths with  $l_{min} = 1$  compared to  $l_{min} = 2$  or 3. The amount of shortest paths of length 1 and 2 decreases as the network size increases. The shortest paths of length 3 slightly increase though. There is almost no difference for the analysis between PolarStar with BDF or Paley subgraphs.

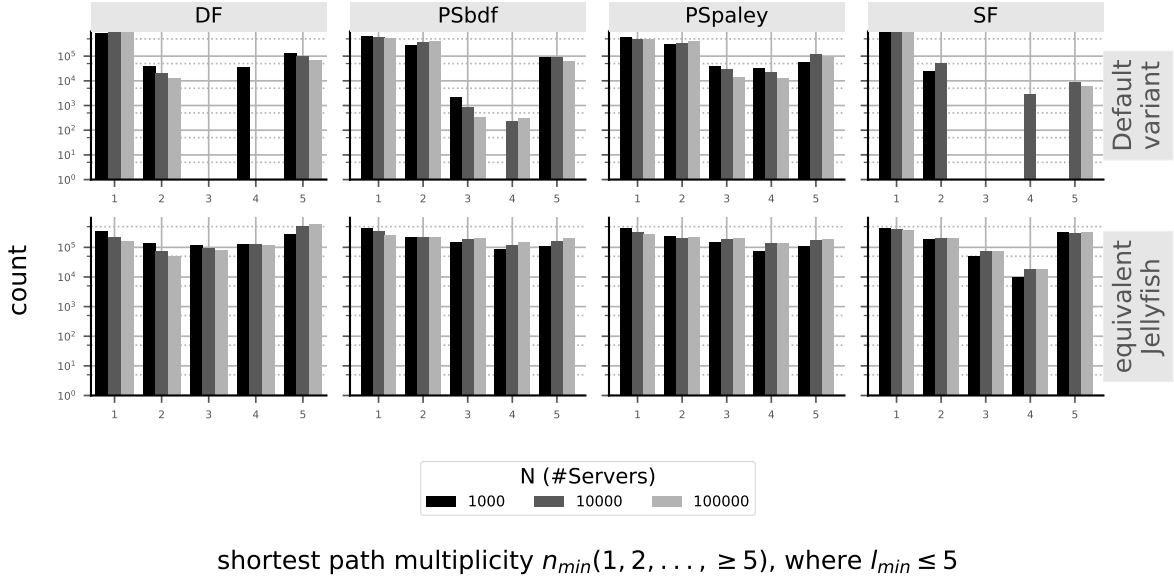


Figure 19: Multiplicity plot for PolarStar, Dragonfly and Slim Fly. Slim Fly has the least multiple shortest paths of all topologies. Only for medium and large networks, there are pairs of nodes with shortest path multiplicity bigger than 3. All topologies have the most shortest paths only once between any two nodes. PolarStar with BDF subgraphs has only a few shortest path multiplicity count for  $n_{min} = 3$  or 4, while PolarStar with the Paley subgraphs has a well-distributed count for all values of  $n_{min}$ .

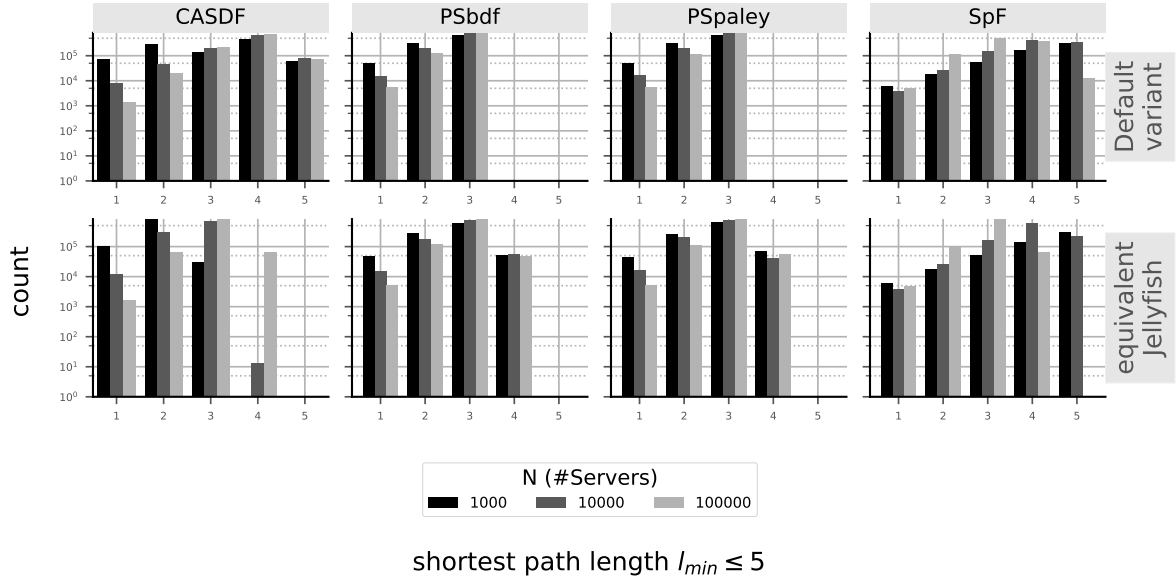


Figure 20: Shortest path plot for Cascade Dragonfly, PolarStar and Spectralfly. The amount of shortest paths of length 1 and 2 decreases, if the network gets bigger, because a single node in a group can only reach 21 nodes from another group with at most 2 jumps. Meanwhile, all 95 nodes in the same group can be reached with 2 gaps. The amount of node pairs that have a shortest path  $\geq 3$  stays almost the same. Spectralfly has the least amount of shortest paths of length 1 for all network sizes. The amount of shortest paths of low length stays the same or gets even better for higher-size networks. The Jellyfish equivalent of the Cascade Dragonfly has only a diameter of 3 or 4 (depending on the network size), while Cascade Dragonfly is a diameter 5 topology. Spectralfly's Jellyfish equilibrium on the other hand looks similar to its base counterpart. Only for large network sizes, the Jellyfish graph shows no shortest paths of length 5, while the Spectralfly graph still has some.

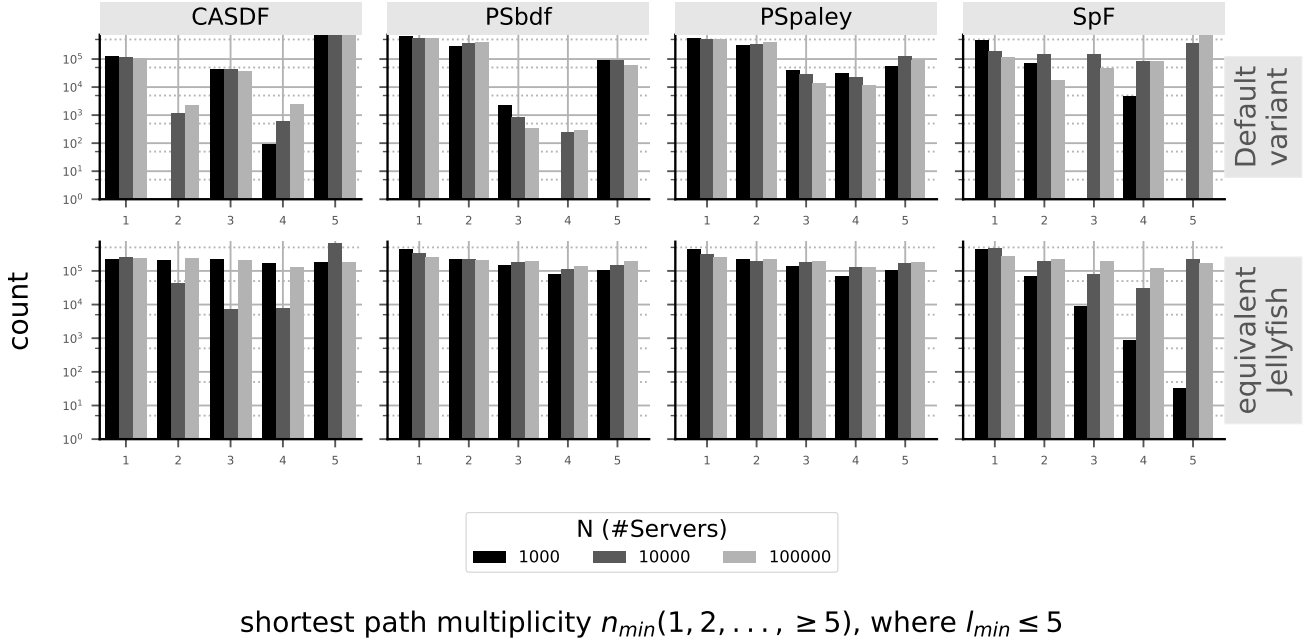


Figure 21: Multiplicity plot for the Cascade Dragonfly, PolarStar and Spectralfly networks. Most node pairs in the Cascade Dragonfly topology have multiple ( $\geq 5$ ) shortest paths. The multiplicity count either stays about the same or gets bigger, as the Cascade Dragonfly graph increases its size. The small Spectralfly network has only multiplicity counts  $\leq 4$ . For large and medium networks, the Spectralfly topology has slightly more multiplicity count for  $n_{min}$ , however only by a small margin. For small and medium networks, the equivalent Jellyfish graph of the Cascade Dragonfly has as well the most shortest path multiplicity for  $n_{min} \geq 5$ . The large Jellyfish has about the same multiplicity count over all values of  $n_{min}$ . While the equivalent Jellyfish of Spectralfly drops in multiplicity counts for higher values of  $n_{min}$ , for medium and large networks it increases.

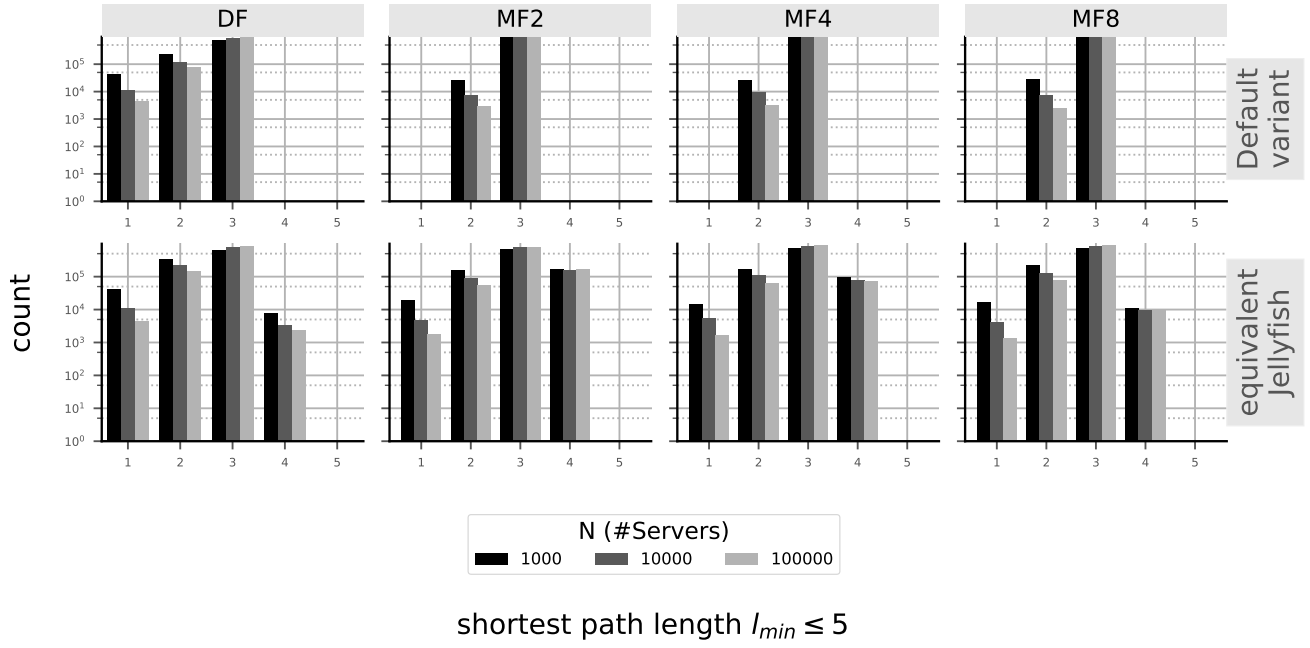


Figure 22: Shortest path plot for Megaflly with different values for  $g$  and Dragonfly. We can see, that Megaflly is a diameter 3 network as well. Megaflly has no shortest path of length 1, since every leaf router is only connected to spine routers, and not to another leaf router. Because two nodes have only shortest path of length  $l = 2$  if they are in the same group, most shortest paths are  $l_{min} = 3$  for Megaflly. All topologies have the same amount of shortest paths of length 3 for all network sizes. The analysis of the equivalent Jellyfishes looks pretty similar compared to the base Megaflly network, where Megaflly has shortest paths of length  $l$ . The Jellyfish graphs have additional shortest paths of length 1 and sometimes length 4.

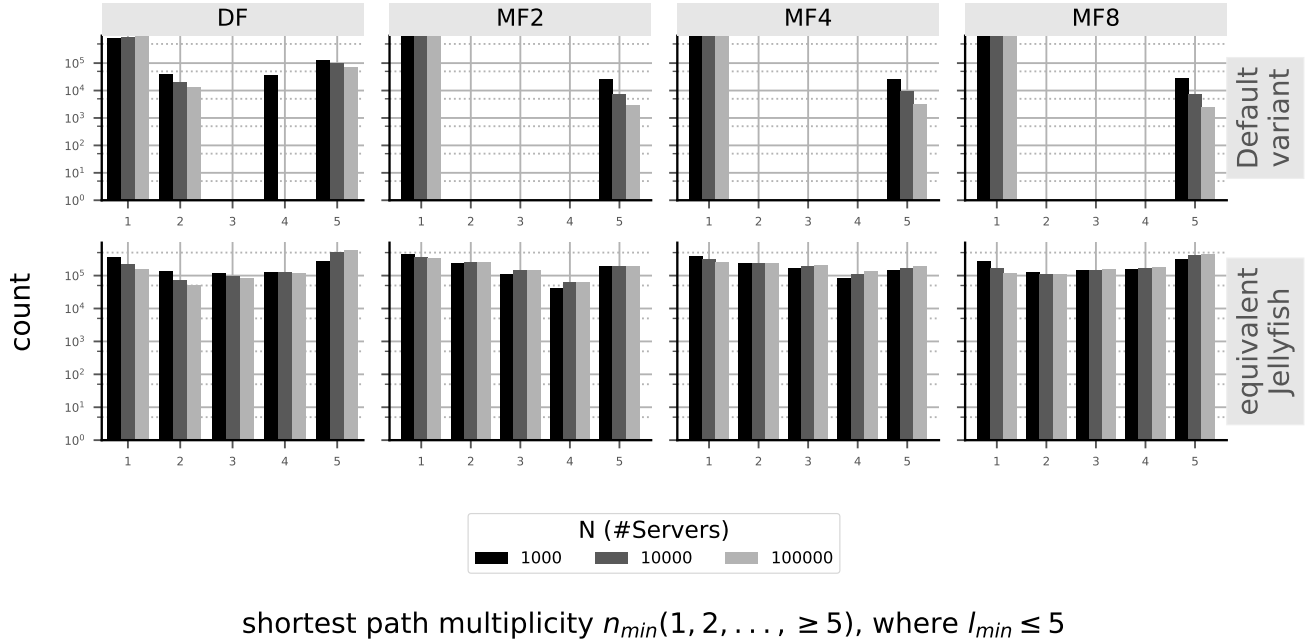


Figure 23: Multiplicity plot for the Dragonfly and Megaflly topologies with different values for  $g$ . All topologies have mostly only one shortest path between any two nodes. Between any two leaf nodes in Megaflly, there is mostly only one shortest path. There is either one shortest path between the nodes or a bunch ( $\geq 5$ ) of them. Dragonfly has some shortest paths of different multiplicity, but most lie in the extremes  $n_{min} = 1$  or  $n_{min} \geq 5$ . Their equivalent Jellyfish on the other hand have well-distributed shortest path multiplicities.



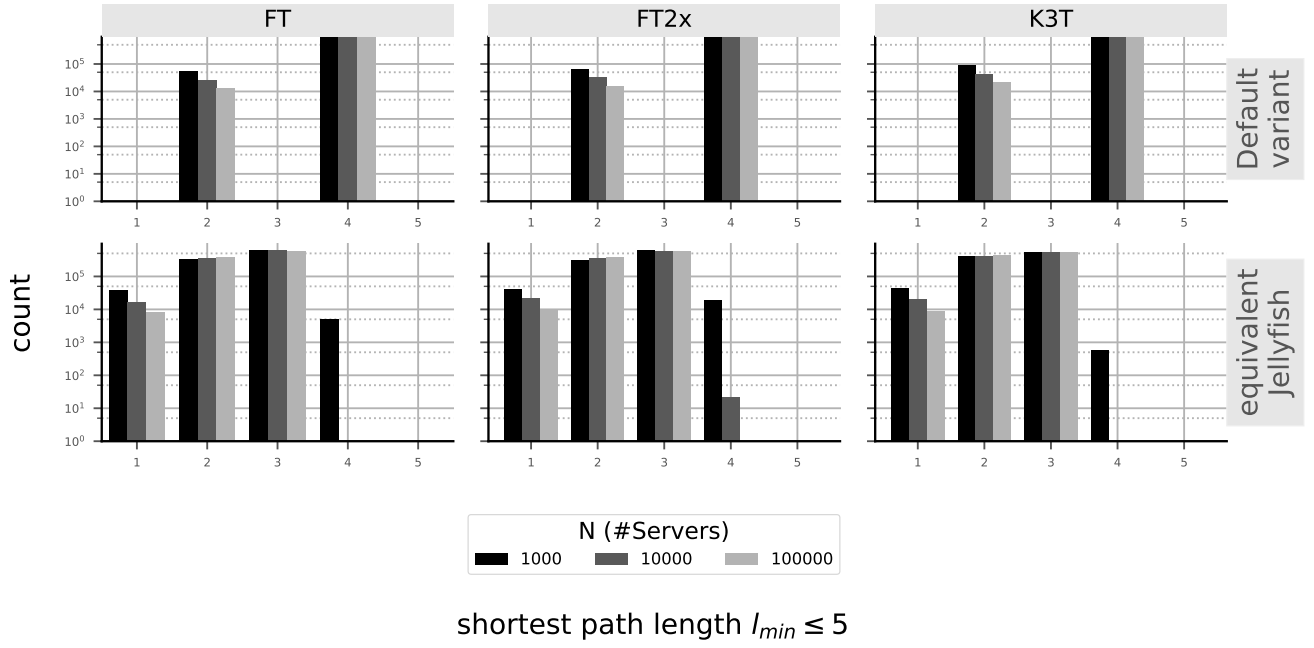


Figure 24: Shortest path plot for Fat Trees and the k-ary 3-tree. All topologies exhibit the same patterns for the shortest paths of a given length. While the shortest paths of length 4 stay the same for any network size, the number of paths of length 2 decreases as the network gets bigger.

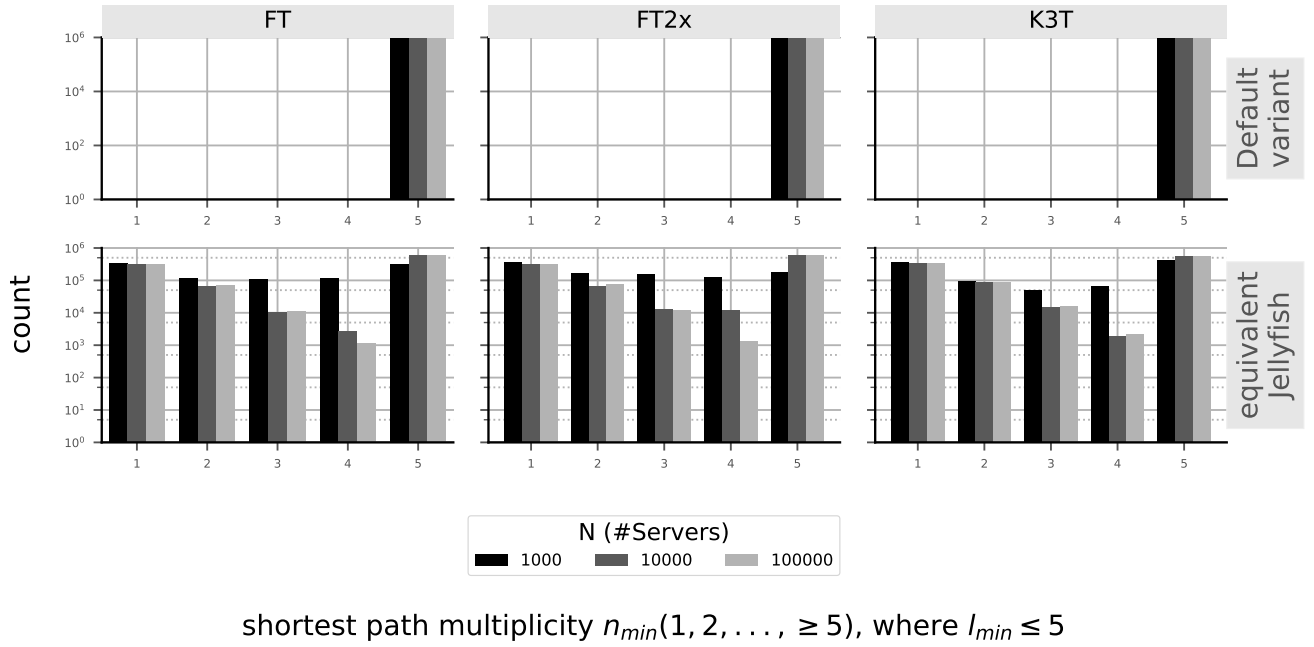


Figure 25: Multiplicity plot for Fat Trees and the k-ary 3-tree. All networks and all sizes have a path multiplicity count of 5 or higher.

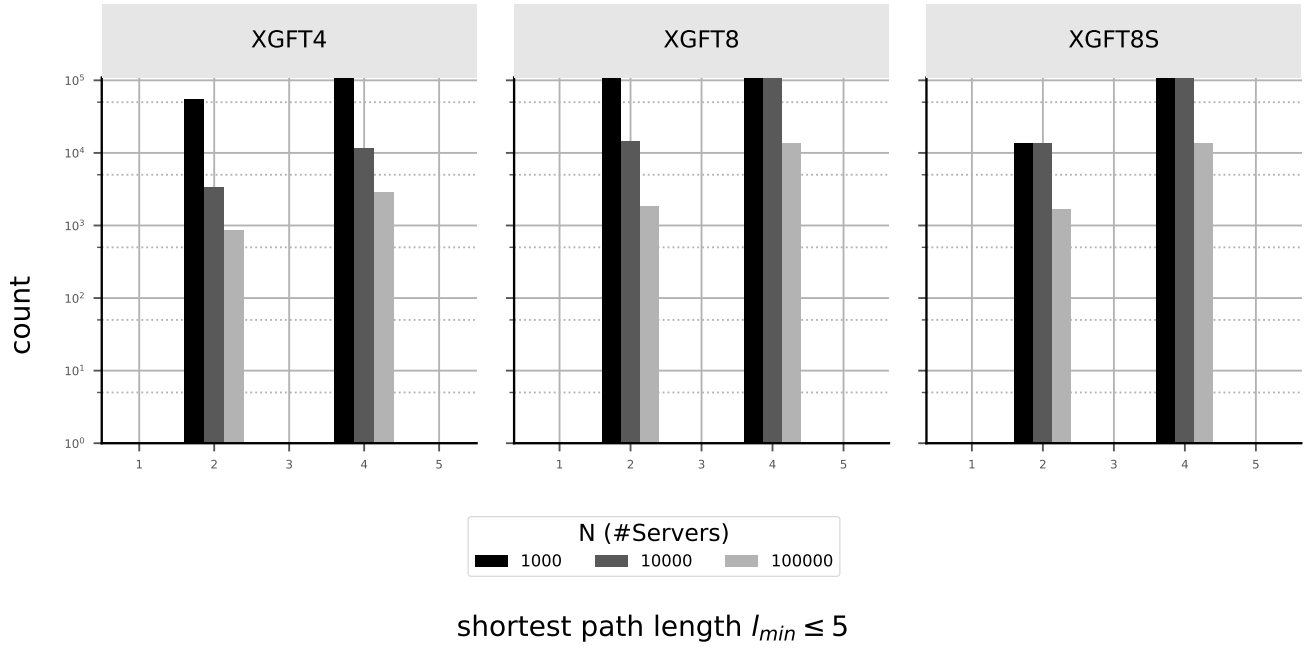


Figure 26: Shortest path plot for length  $l_{min}$  up to  $l = 5$  for multiple variants of xGFTs. There are only shortest paths of even length. There exist more shortest paths of length 4 than length 2 in general. As the network size increases, the number of shortest paths at each length decreases.

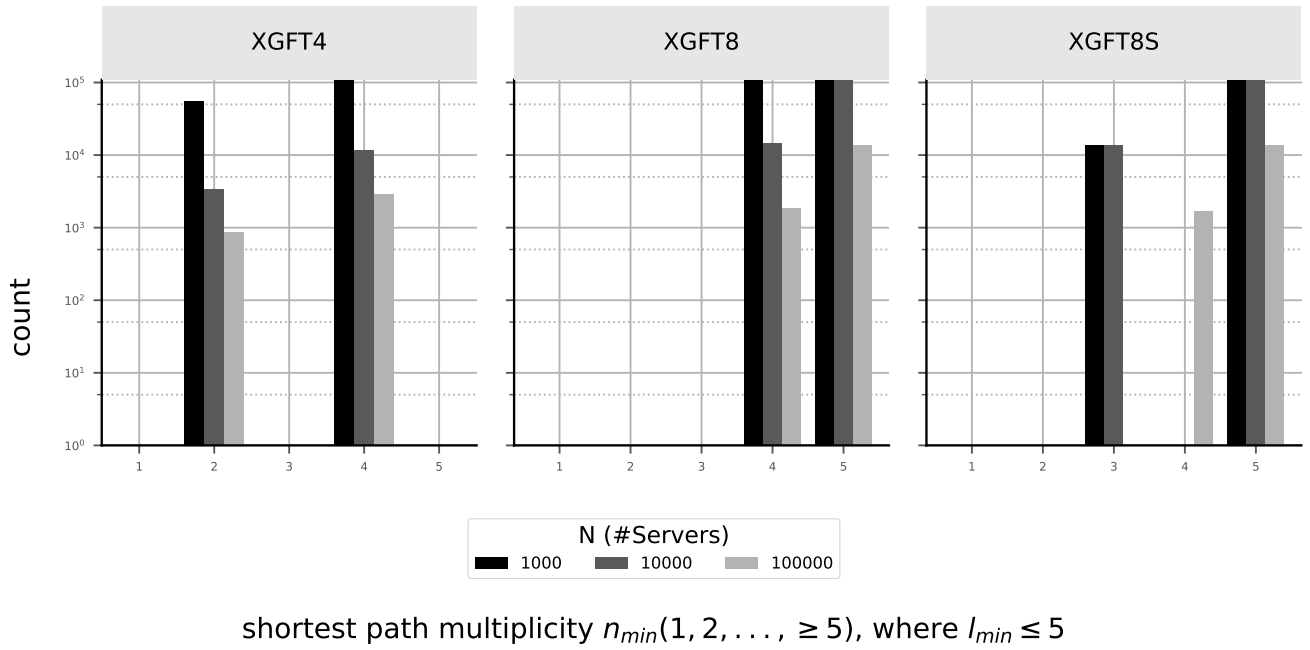


Figure 27: Multiplicity plot for multiple variants of xGFTs. Variant 4 has the lowest amount of multiplicity count. It only has a multiplicity count of 2 and 4. For large network sizes, the 8S variant has even more multiplicity than the smaller networks of that variant.

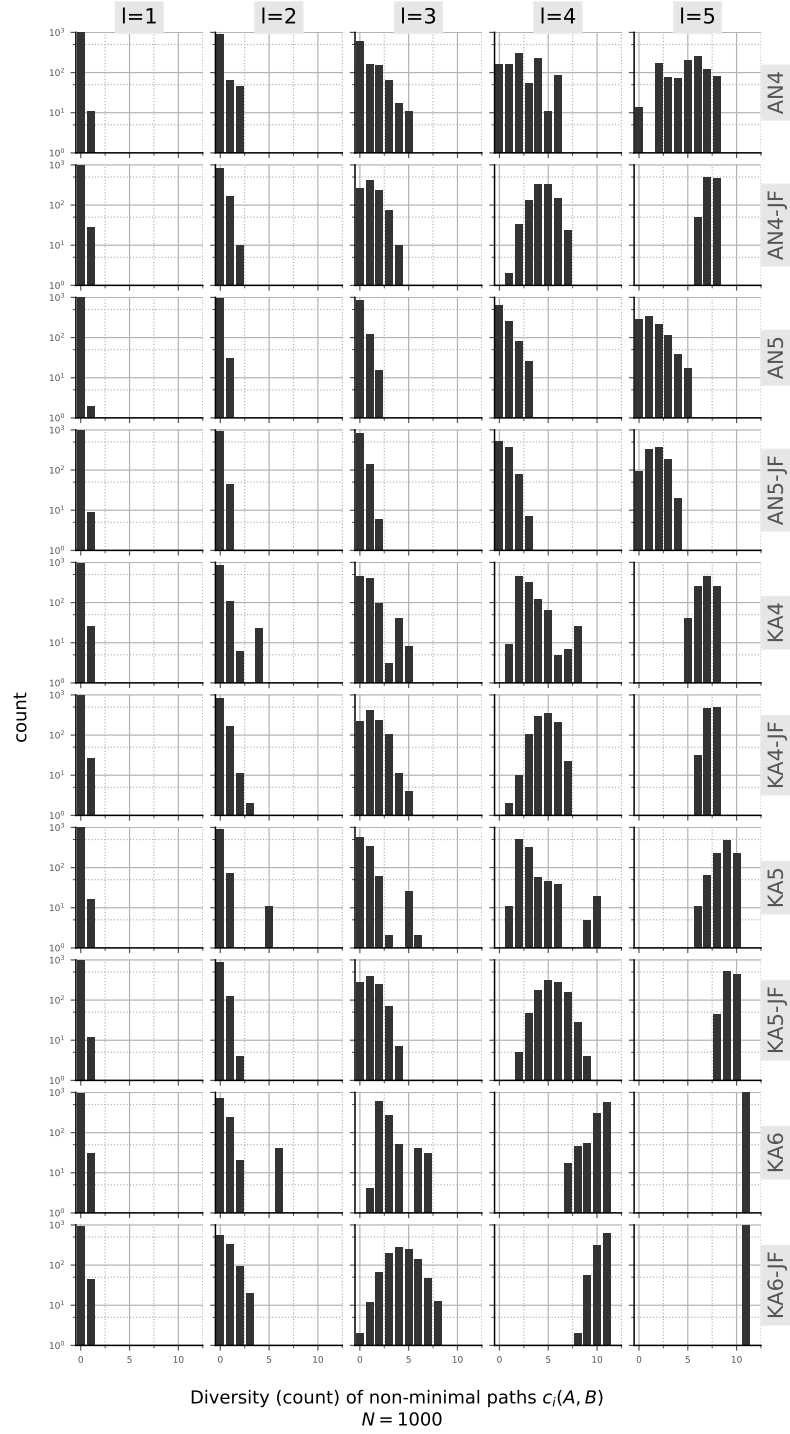


Figure 28: Edge disjoint path plots for Kautz and Arrangement graphs of small sizes. The connectivity for  $A_{n,4}$  is better than the connectivity for  $A_{n,5}$ . Both have in general fewer edge disjoint paths than the Kautz topologies.  $K_{6,n}$  has the best connectivity for these topologies. Furthermore, the Kautz networks analysis looks similar to its equivalent Jellyfish.

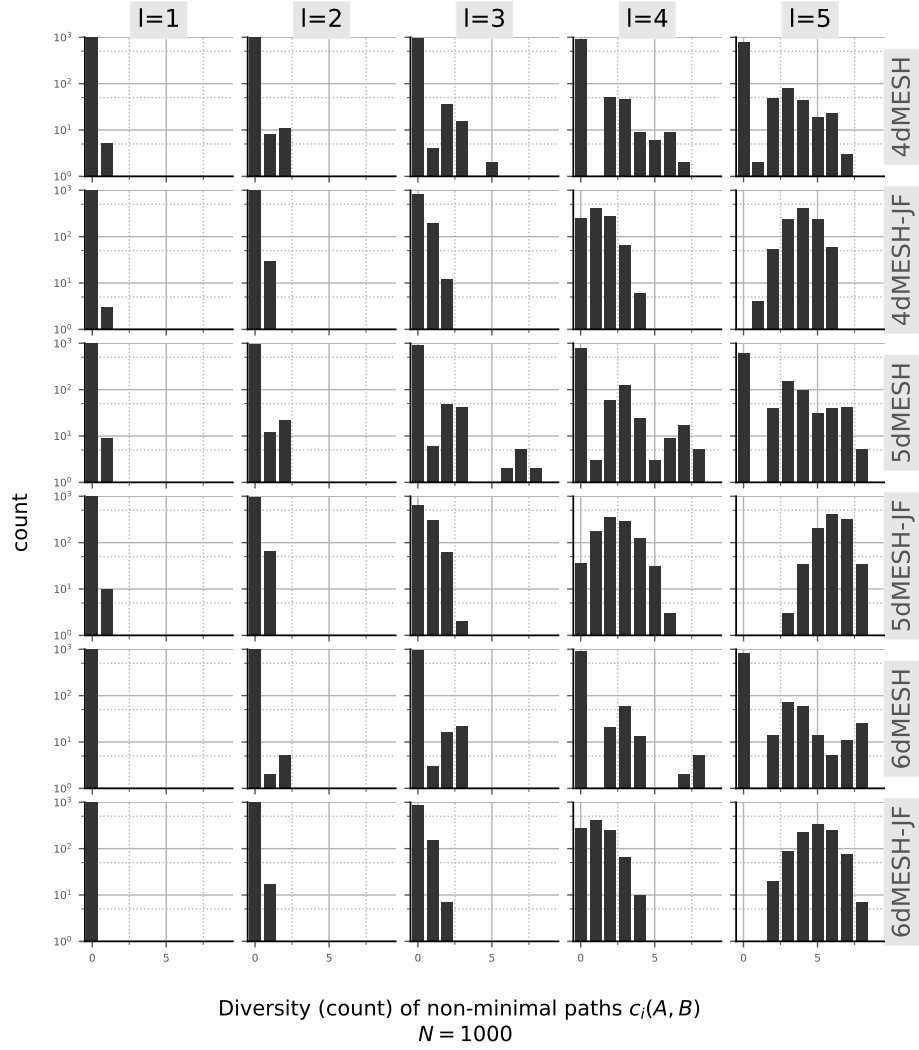


Figure 29: Edge disjoint path plots for meshes of different dimensions. For all lengths  $l$ , the meshes have mostly zero edge disjoint paths between any two nodes. Furthermore, we can see that there are almost no node pairs, where there are 2 paths of a given length  $l$ . In contrast, none of the Jellyfish equivalent graphs have zero edge disjoint paths for  $l = 5$ . The 4D mesh has a slightly worse path diversity than the 5D and 6D meshes.

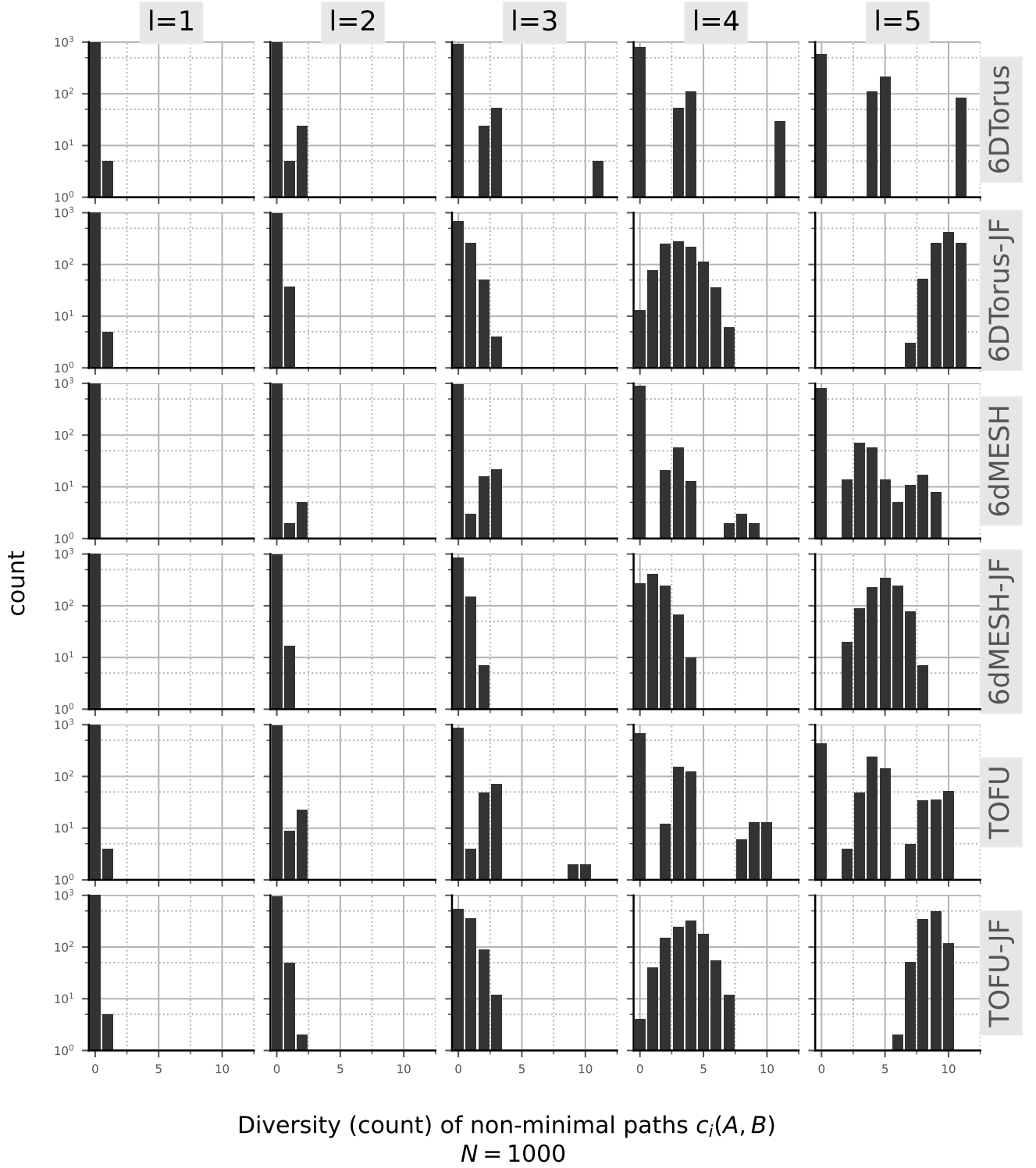


Figure 30: Edge disjoint path plots for the Tofu topology, 6D torus and 6D Express Mesh. Similar to the diversity analysis of the meshes, all the topologies have zero path diversity for most node pairs of any given length. For both Tofu and the 6D torus network, their path diversity counts lie in three groups. They have diversity counts of 1,  $\approx 4$  and  $\approx 10$ . This is even more extreme for the 6D torus network and cannot be seen for the 6D mesh graph. Same as before, the equivalent Jellyfish graphs have a well-distributed diversity count.

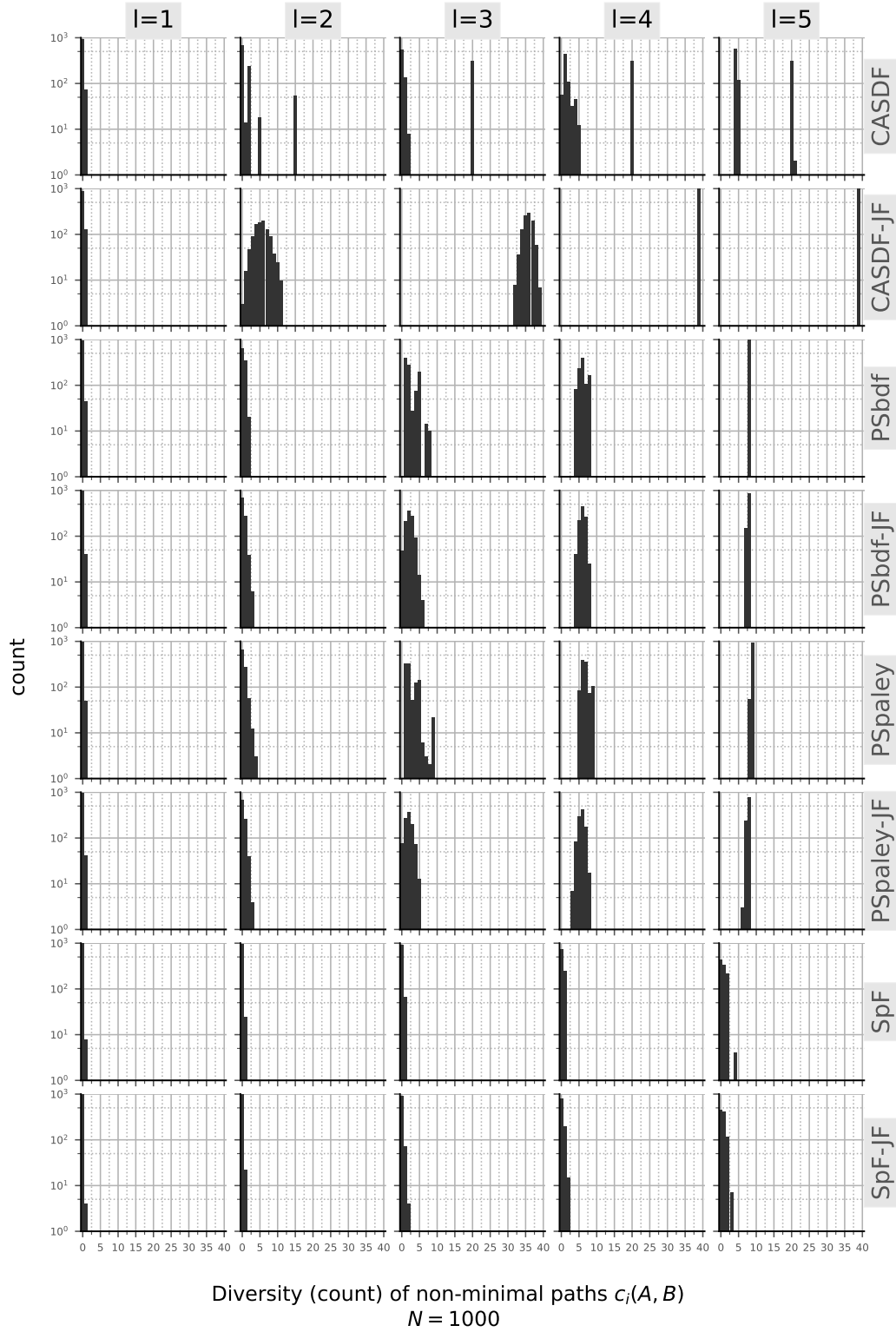


Figure 31: Edge disjoint path plots for Cascade Dragonfly, PolarStar and Spectralfly topologies. Both Spectralfly and Cascade Dragonfly have mostly a low diversity count. Cascade Dragonfly has an additional peak with node pairs of a higher diversity count. Even for length  $l = 5$ , Spectralfly has only node pairs with 4 edge disjoint paths.

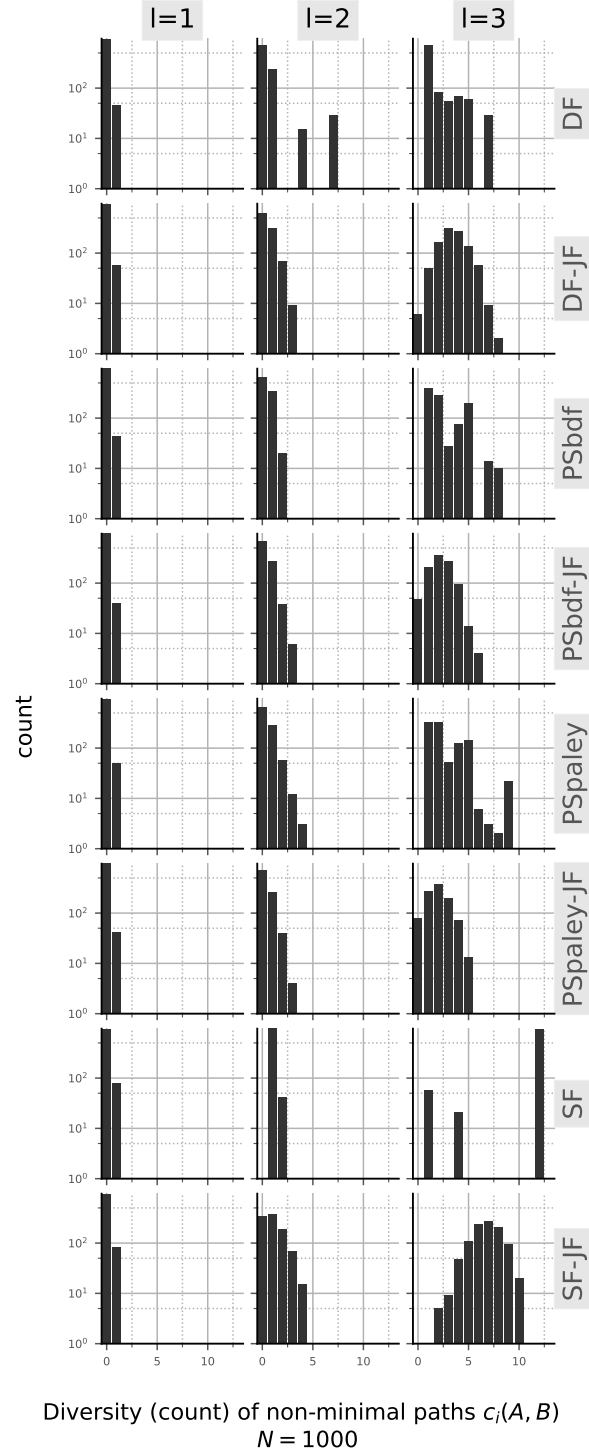


Figure 32: Edge disjoint path plots for Dragonfly, PolarStar and Slim Fly topologies. Slim Fly has the most path diversity by far. It has a diversity count  $\geq 10$  for most nodes, even for paths of length 3. The diversity analysis of PolarStar is pretty similar to Dragonfly for length 1 to 4. For  $l = 5$  we can see that Dragonfly has a higher diversity count by 2 for all node pairs. The PolarStar graph with Paley subgraphs has slightly higher edge disjoint paths for  $l = 5$ . Both variants of PolarStar have the same or better diversity compared to their Jellyfish equivalent for all lengths.

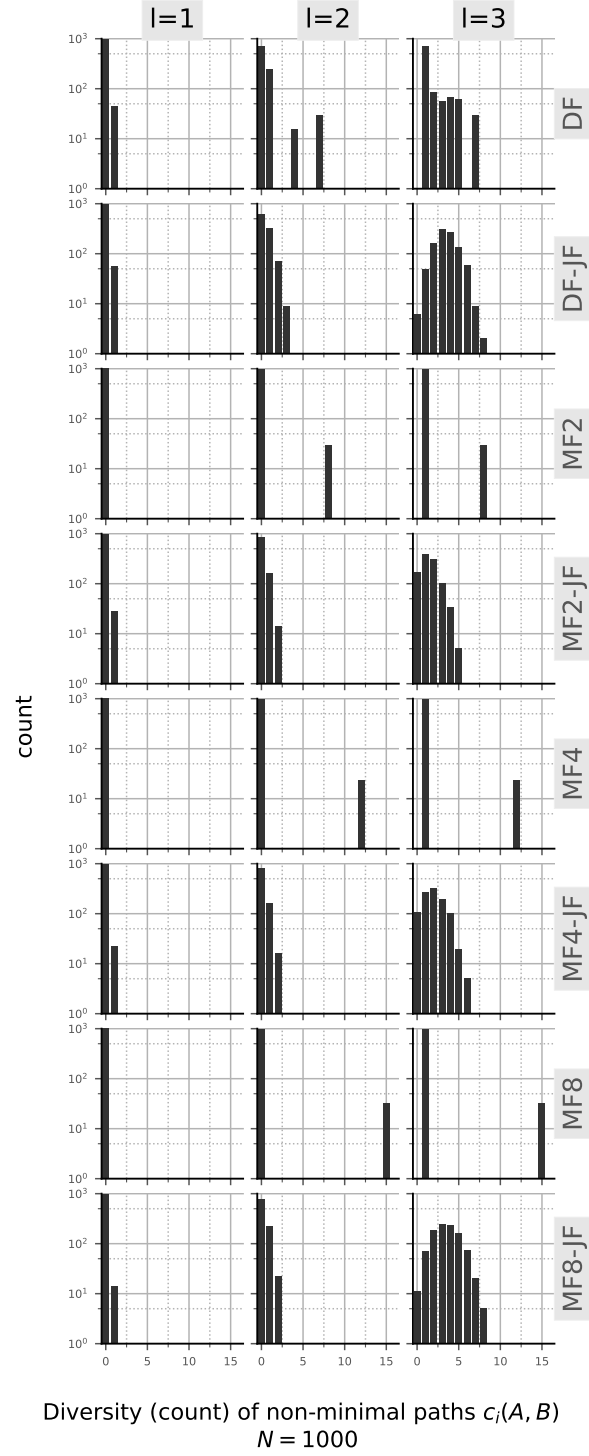


Figure 33: Edge disjoint path plots for Dragonfly and Megaflly networks with different values of  $g$ . All Megaflly networks have a low path diversity of either 0 or 1 for most node pairs and all lengths. Furthermore, all have a second peak for a very high diversity count. Depending on the value of  $g$ , this peak is between 7 and 15. Although Megaflly has node pairs with a higher diversity count, Dragonfly has a well-distributed diversity count between 1 and 5. It is difficult to compare the values with this plot. The diversity counts of all Jellyfish equilibriums show a Gauss distribution for  $l = 3$ .



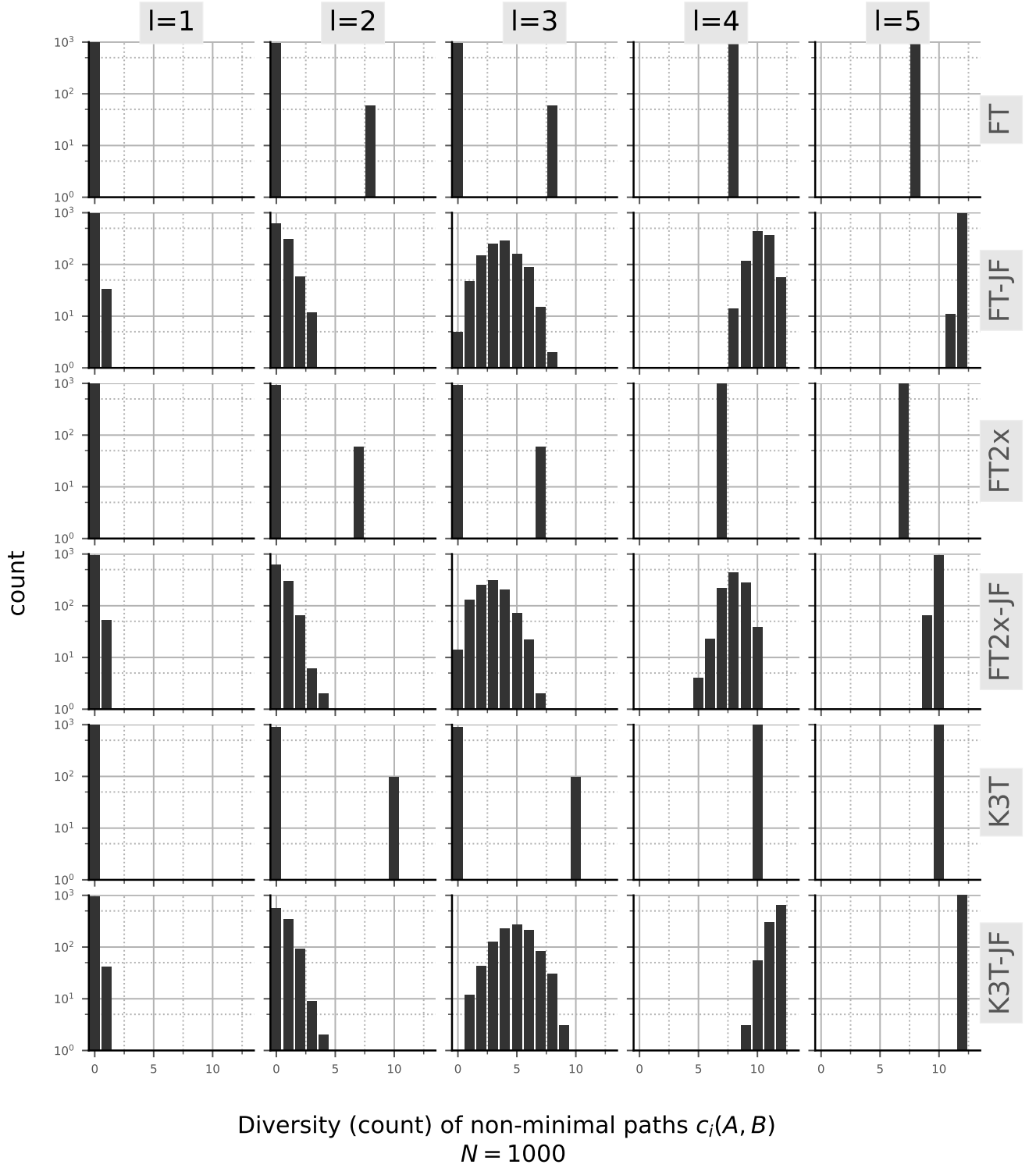


Figure 34: Edge disjoint path plots for the Fat Trees and k-ary 3-tree. All the topologies have only specific values for the path diversity. None of the topologies have any node pairs that have a diversity count of zero for  $l \geq 4$ . The k-ary 3-tree has the best path diversity among all other topologies.

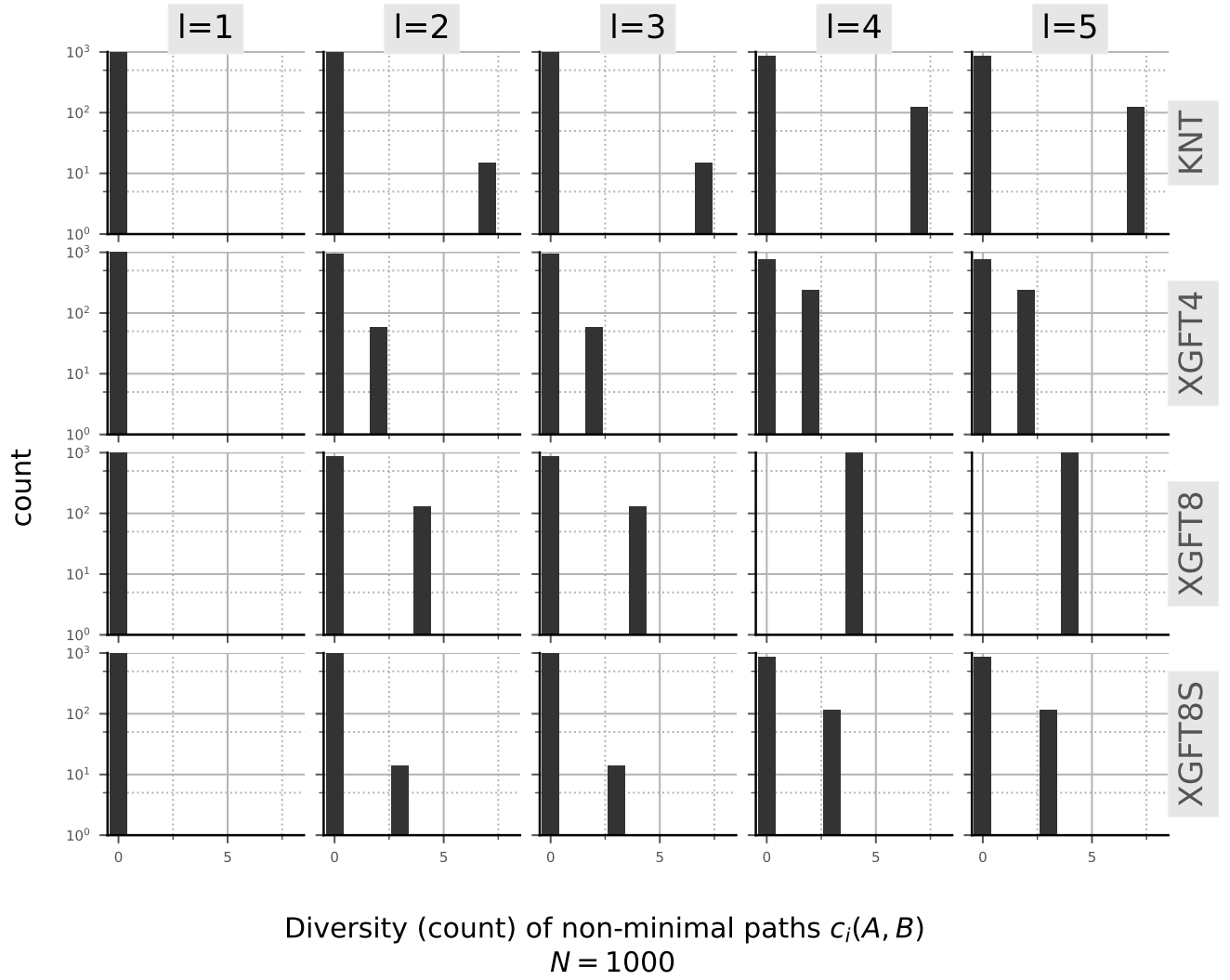
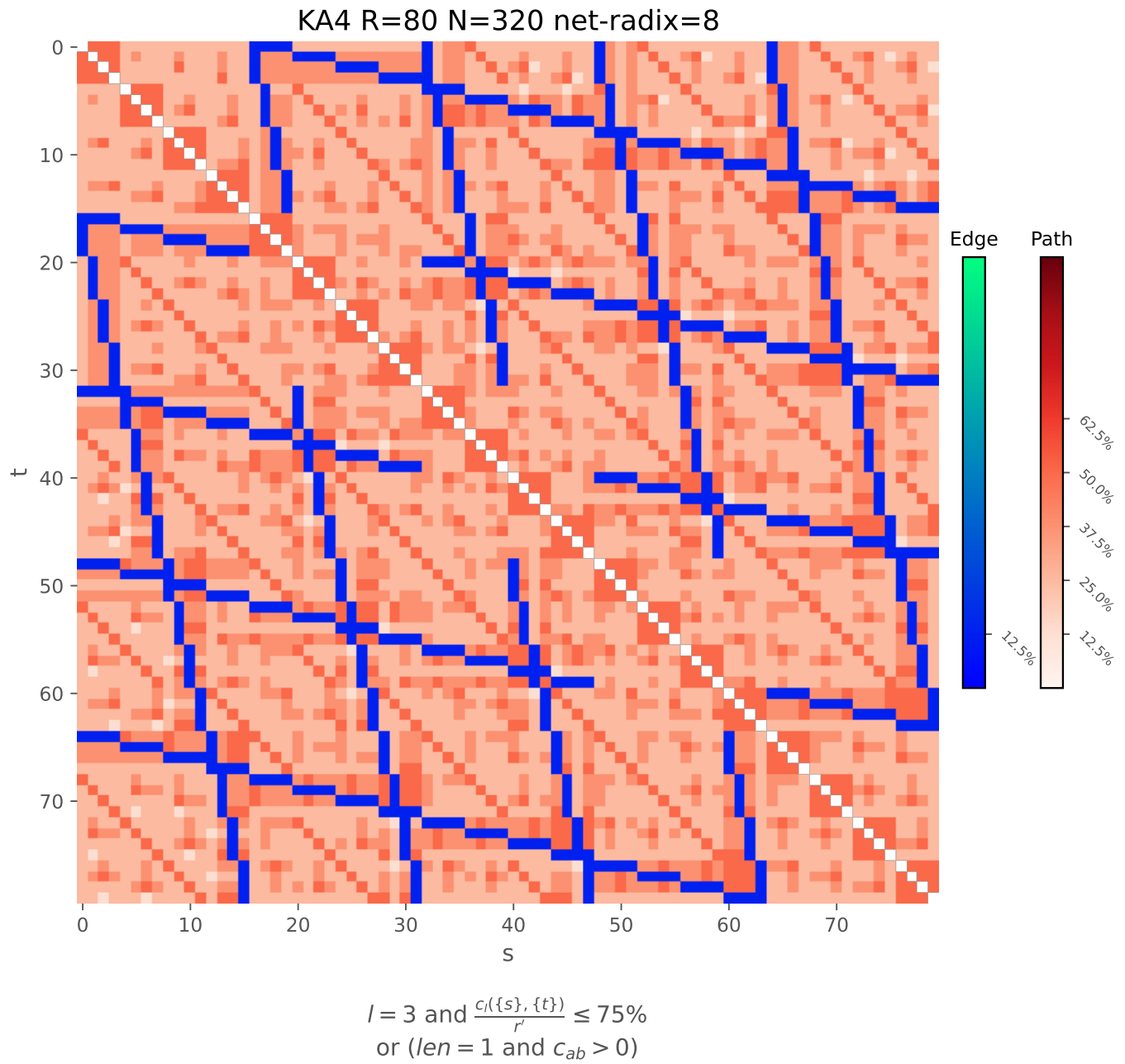


Figure 35: Edge disjoint path plots for the xGFTs of different variants. There exist only edge disjoint paths of specific values. For  $l \geq 4$ , xGFT8 does not have any edge pairs with a diversity count of zero anymore. The k-ary n-tree has edge nodes with a higher diversity count than any other topology, but it has still node pairs with a diversity count of zero for  $l = 5$ .



**Figure 36:** Low connectivity plot for  $K_{4,3}$  Kautz graph of length  $l = 3$ . All router pairs have a connectivity  $\leq 75\%$  for  $l = 3$ . We can see the segments (blue) of length 4 connecting a node to its neighbours. This can be seen horizontally as well as vertically. Due to this mesh of edges, all nodes are connected and there exist only a couple of node pairs with connectivity  $< 25\%$ .

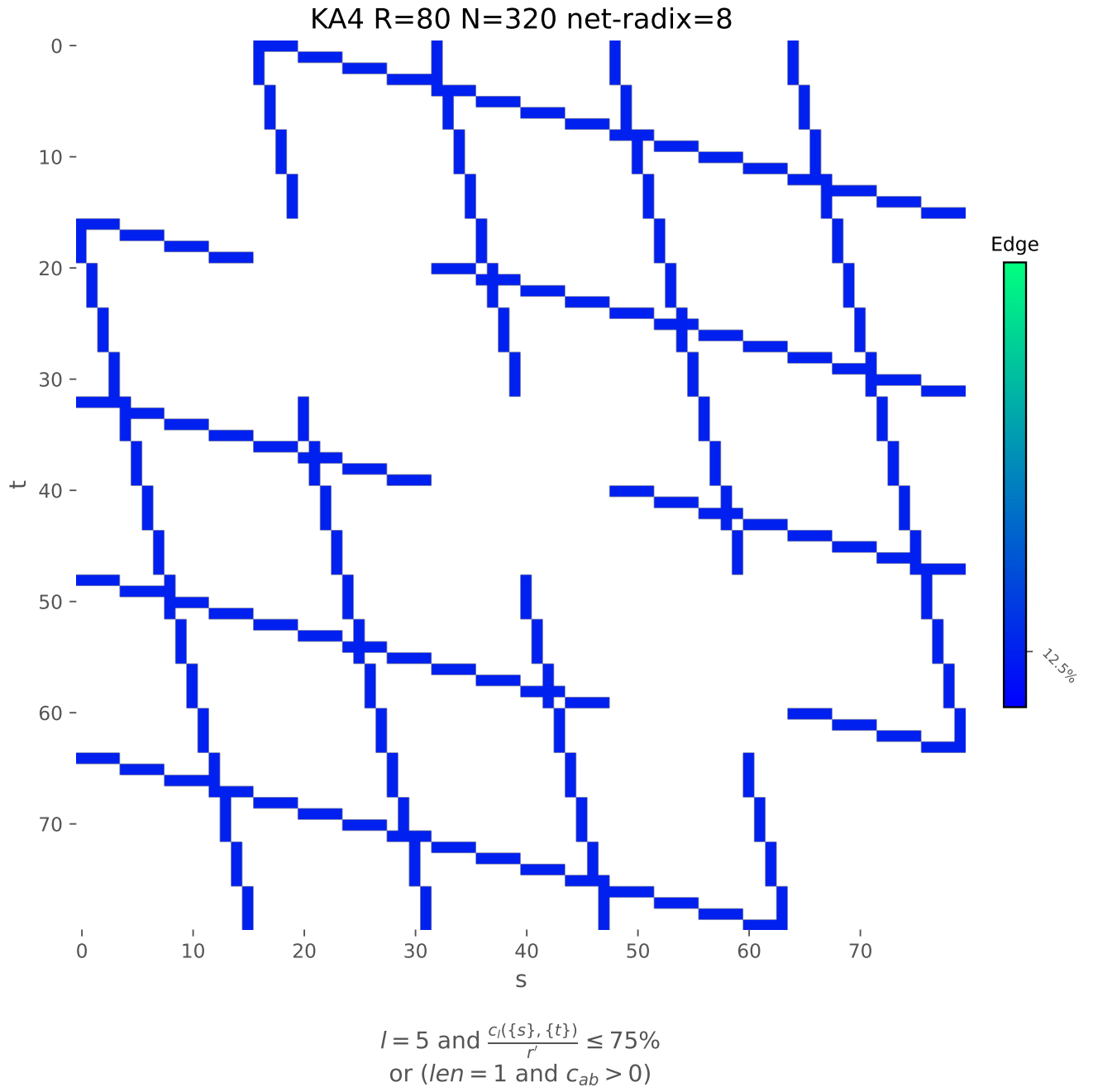


Figure 37: Low connectivity plot for  $K_{4,3}$  Kautz graph of length  $l = 5$ . All router pairs have a connectivity  $\geq 75\%$  for  $l = 5$ .

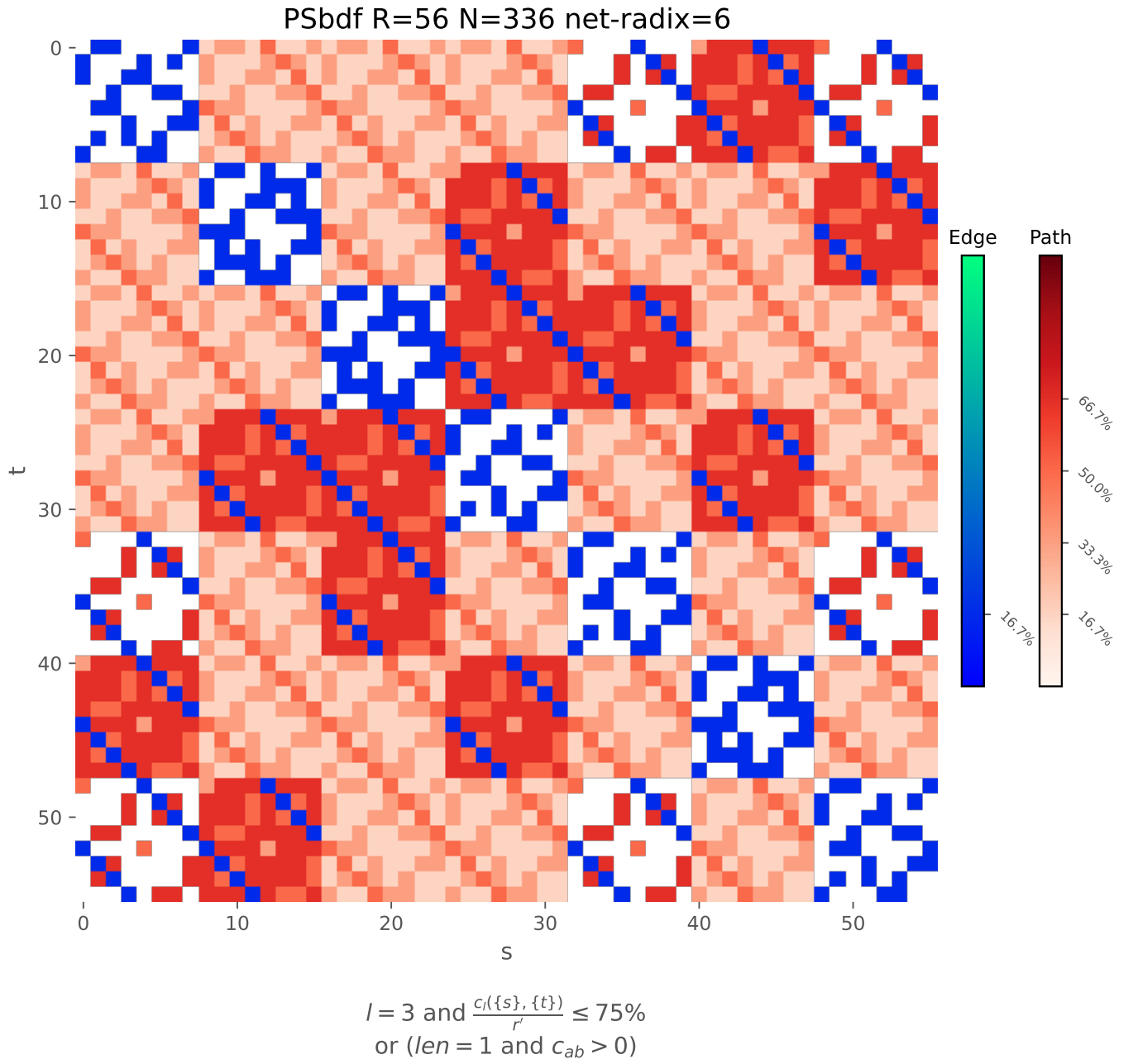


Figure 38: Low connectivity plot for the PolarStar graph of length  $l = 3$ . The structure graph is an  $ER_2$  graph and the subgraph is a  $BDF_3$  graph. We can see the 7 well-connected  $BDF_3$  clusters of each 8 nodes in the diagonal. The other well-connected areas exist due to the connection of the  $ER_2$  graph. Every node in 0 - 7 is connected to a node in 31 - 39, 40 - 47 and 48 - 55.

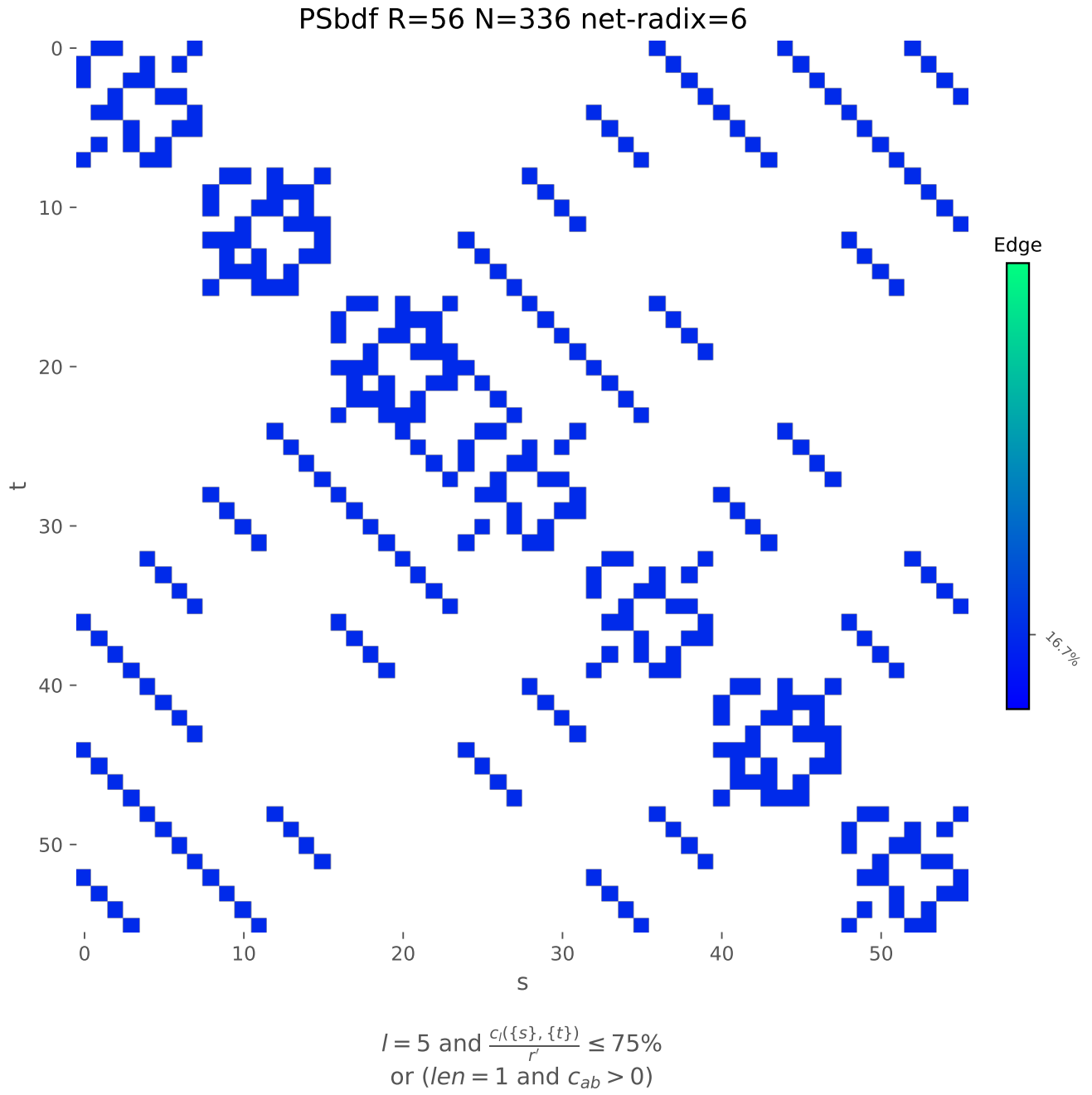


Figure 39: Low connectivity plot for the PolarStar graph for the length  $l = 5$ . The structure graph is an  $ER_2$  graph and the subgraph is a  $BDF_3$  graph. We can clearly see, that for  $l = 5$  this PolarStar graph has a good connectivity  $\geq 75\%$  for all router pairs.

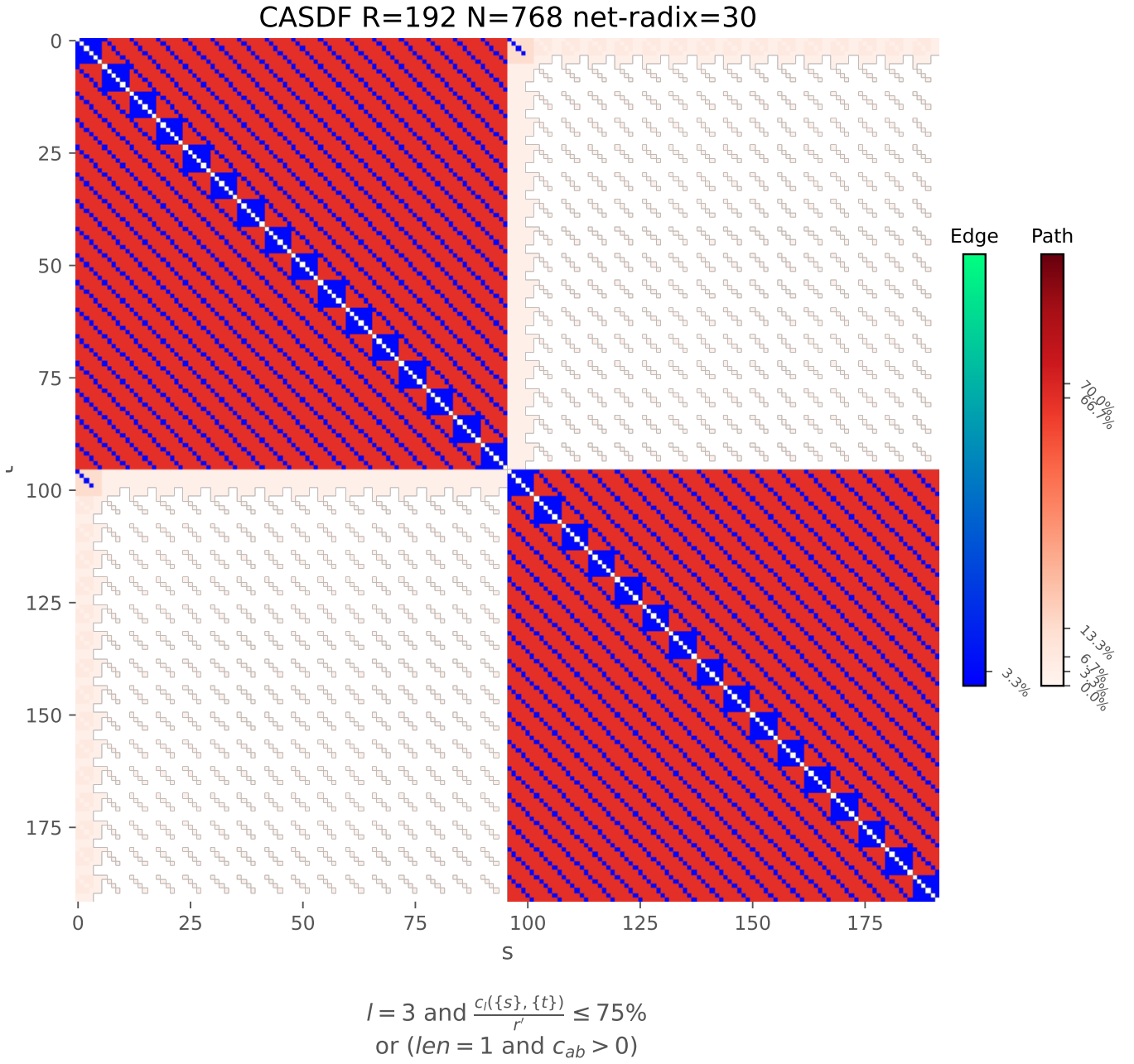


Figure 40: Low connectivity plot for the Cascade Dragonfly topology with two clusters for the length  $l = 3$ . Each cluster has a good connectivity within itself. On the diagonal, we can also see the 16 chassis for the 5 levels. The connectivity between the two clusters is very low and only about half of the nodes in each cluster have a path to nodes in the other cluster.

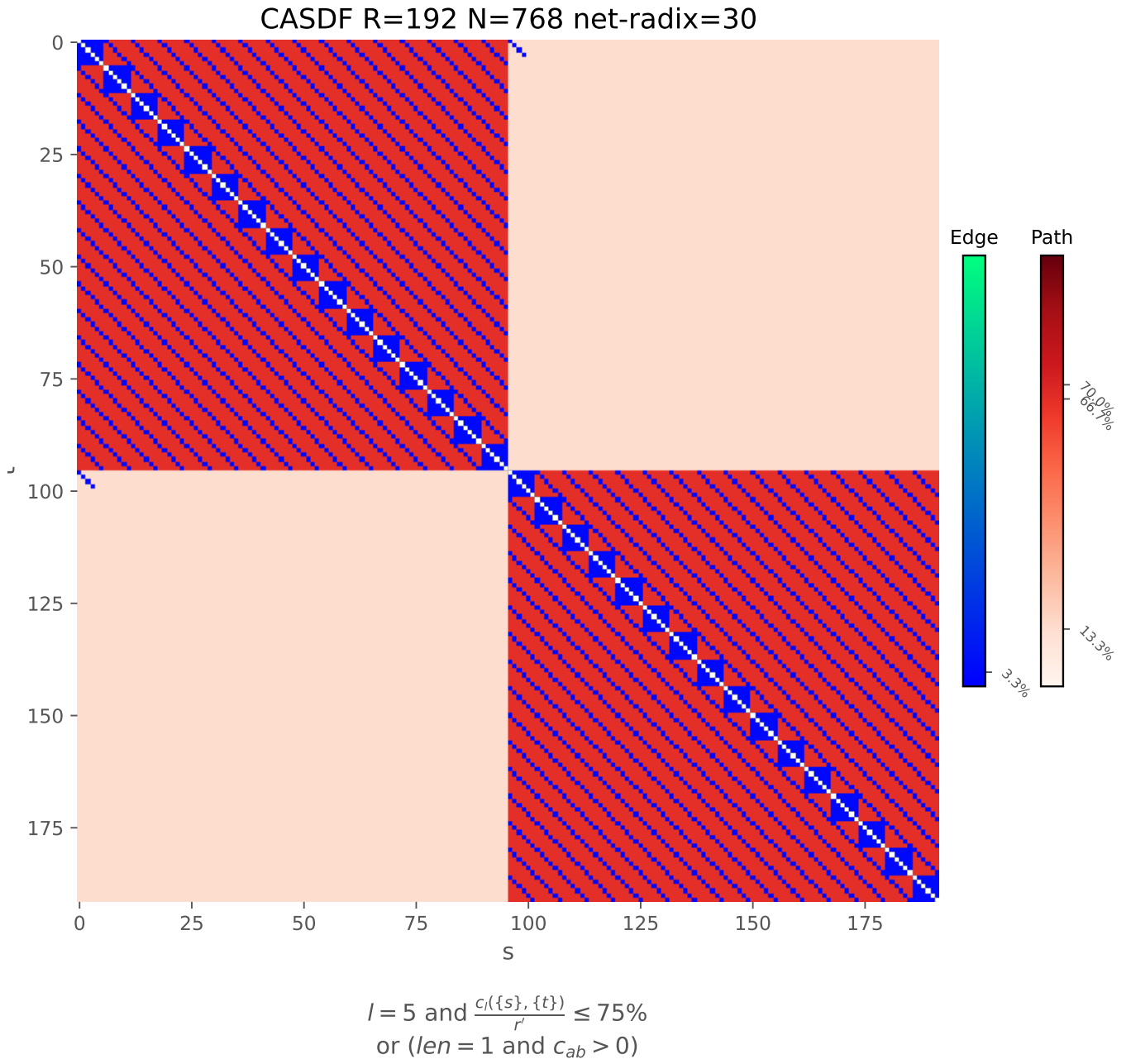


Figure 41: Low connectivity plot for the Cascade Dragonfly topology with two clusters for the length  $l = 5$ . Right underneath the first cluster, we can see the four node pairs that connect the two clusters. Even for paths of  $l = 5$ , the connectivity between the two clusters stays low at only 10%.



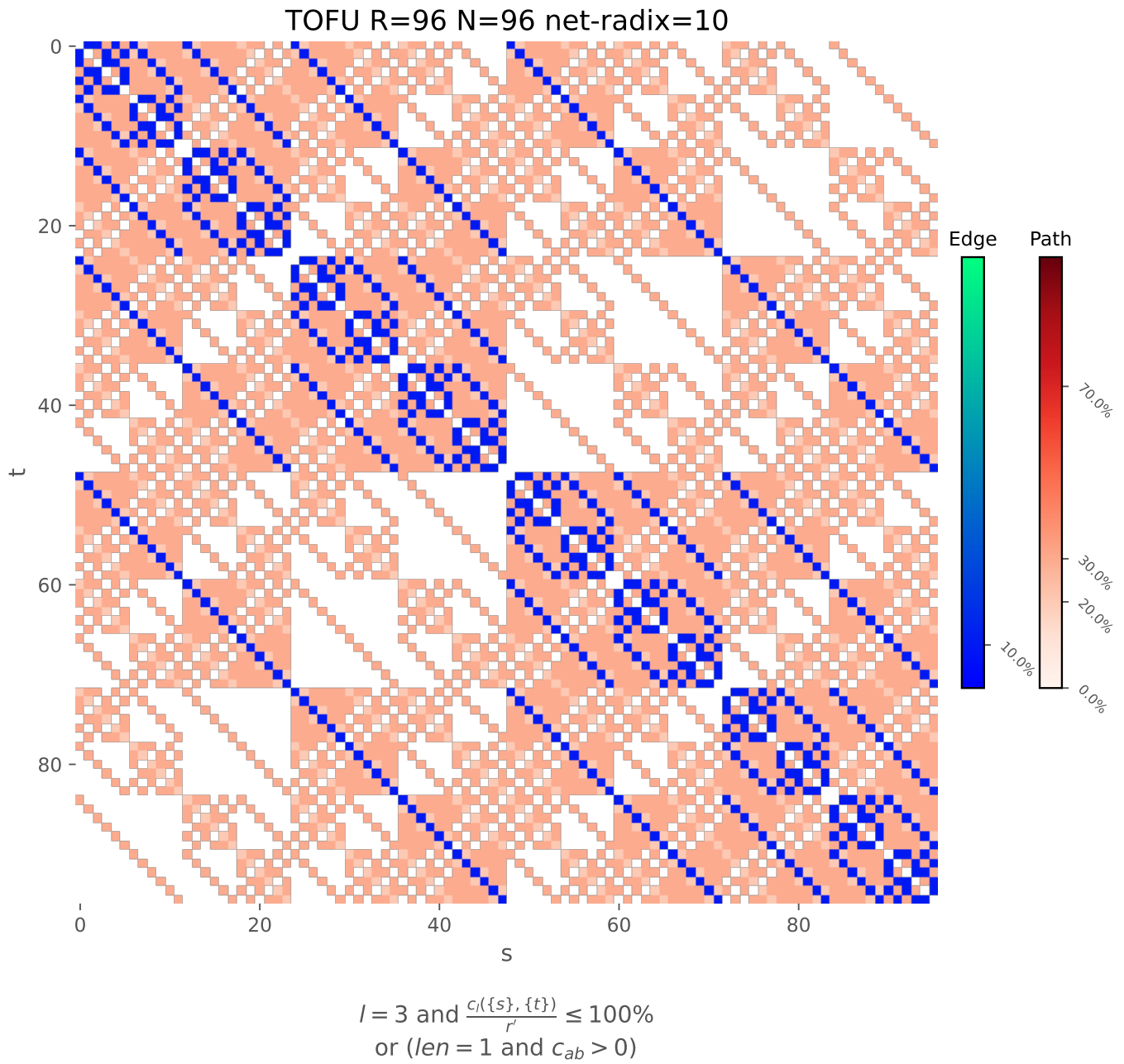


Figure 42: Low connectivity plot for the Tofu topology with  $X = Y = Z = 2$  and  $l = 3$ . Not all nodes are connected with each other at length  $l = 3$ . The 8 clusters of 12 nodes each are visible in the diagonal. The small blue diagonal lines show the edges between the nodes in different clusters. Each row is intersected by 3 of these lines. This is the number of connections each node has to other clusters.

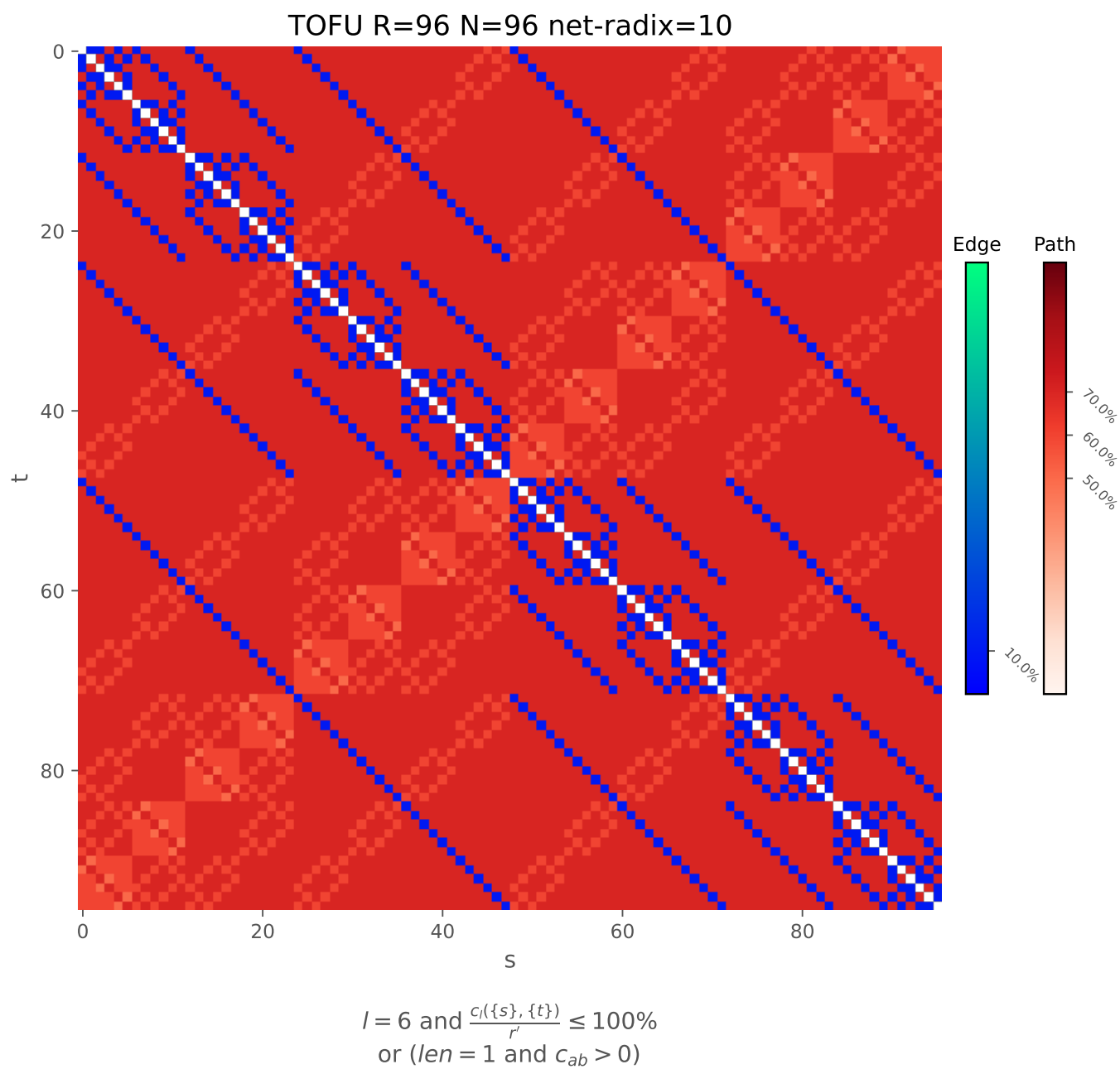


Figure 43: Low connectivity plot for the Tofu topology with  $X = Y = Z = 2$  and length  $l = 6$ . There is a good connectivity  $\geq 70\%$  for most node pairs.

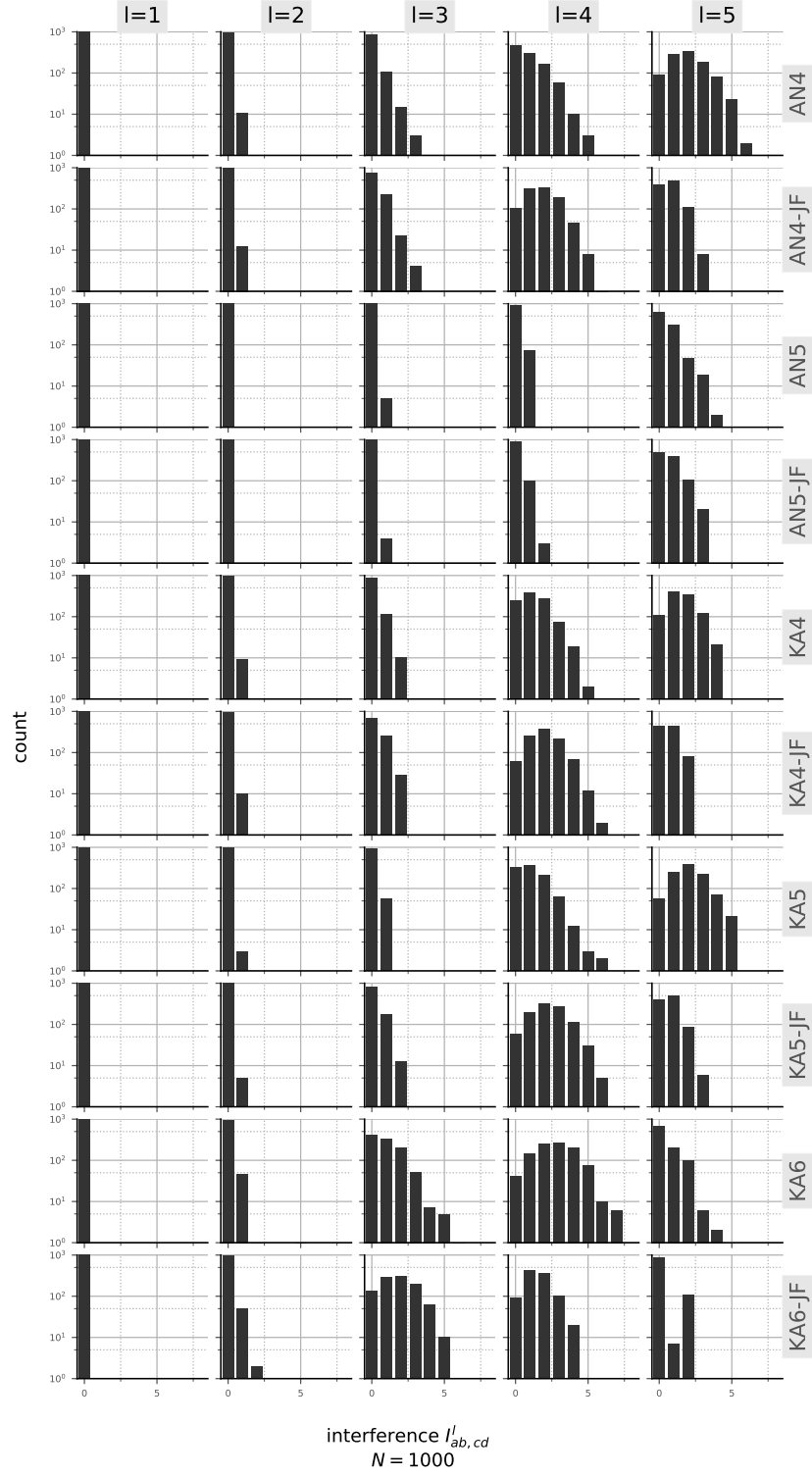


Figure 44: Interference plots for Kautz and Arrangement graphs of length  $l = 5$ . The Arrangement networks have the same or less interference than their Jellyfish equivalent counterpart for  $l \leq 5$ . For  $l = 5$ , both have slightly more interference. The same can be seen for  $K_{4,n}$  and  $K_{5,n}$  Kautz graphs, but not for  $K_{6,n}$ , which has the most interference for  $l = 3$  and  $l = 4$ , but the least interference for  $l = 5$ .

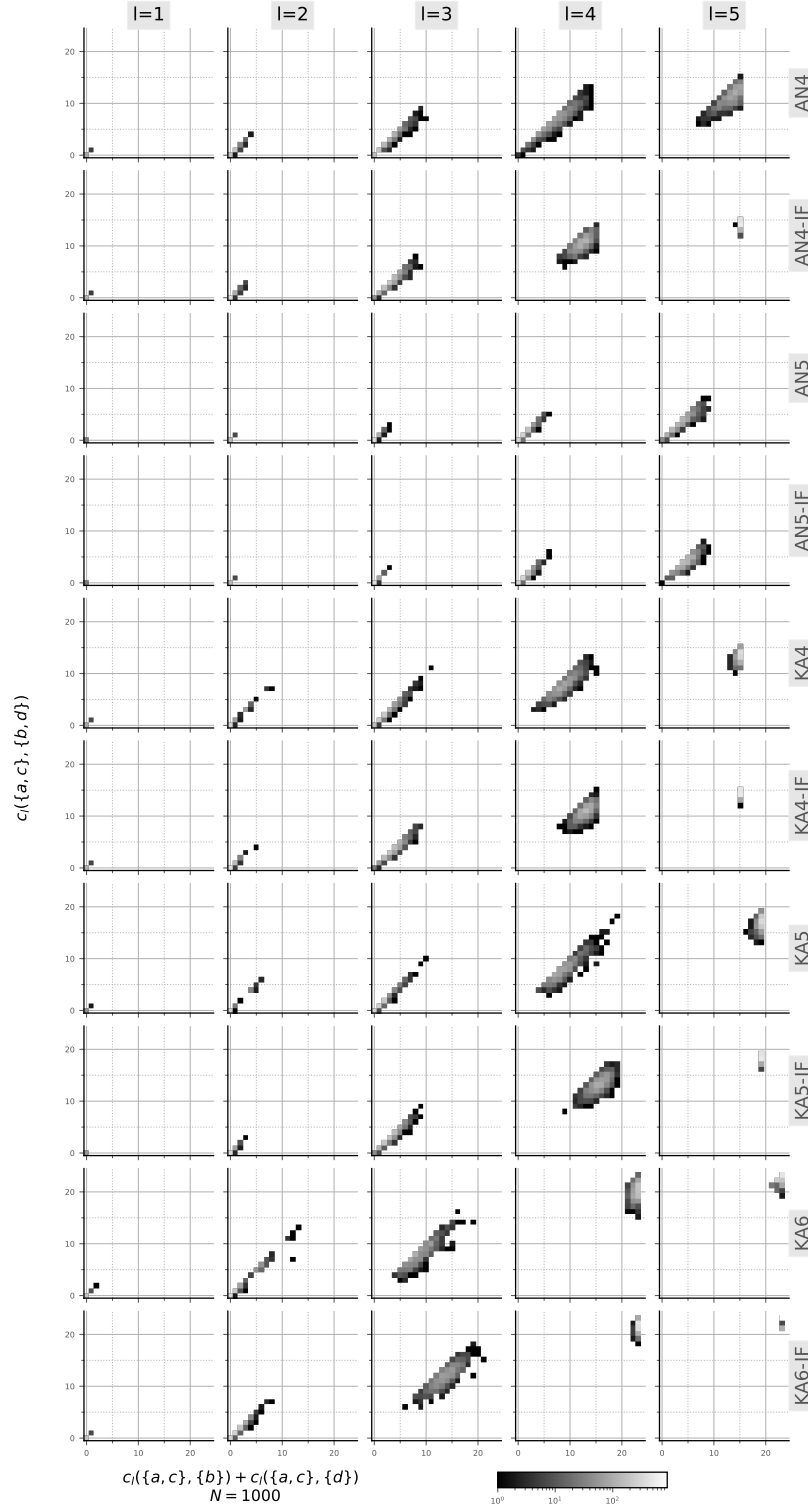


Figure 45: Two-dimensional detailed interference plots for Kautz and Arrangement graphs of length  $l$  up to 5.  $A_{n,4}$  has more interference than  $A_{n,5}$ . The different values of  $k$  for  $K_{b,n}$  Kautz graphs have less of an impact on the amount of interference, but at which length  $l$  they will occur. Most of these networks have the most interference at  $l = 4$ . Their equivalent Jellyfish have similar interference analysis, but in general a bit less.

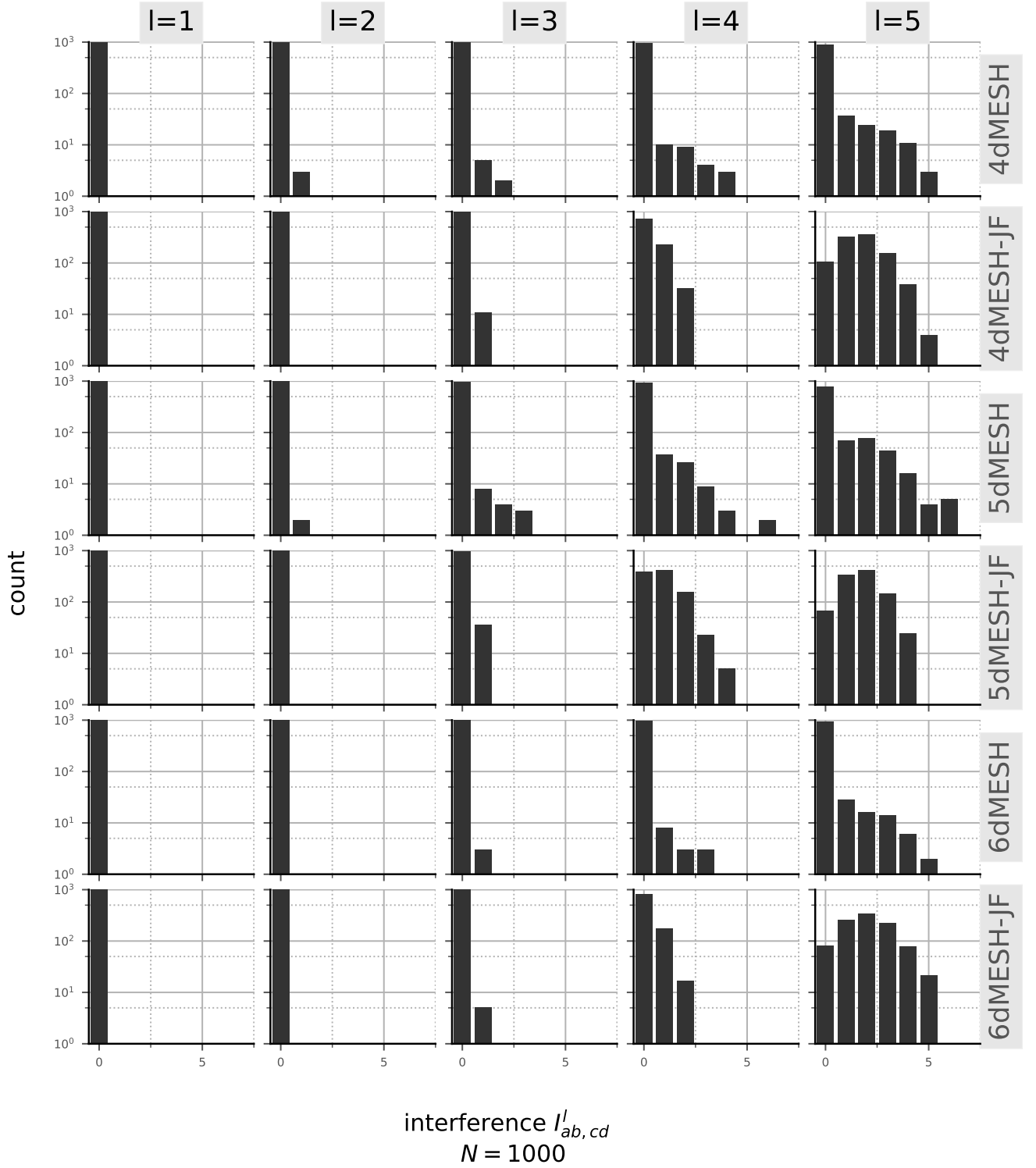


Figure 46: Interference plots for meshes of different dimensions of lengths up to  $l = 5$ . While 4D and 6D meshes have a pretty similar interference analysis, the 5D mesh has more interference. Furthermore, for  $l \leq 4$ , the base networks of all the meshes have higher or equal interference than their equivalent Jellyfish. 4D and 6D meshes have better interference for  $l = 5$ , which is not true for the 5D mesh. For  $l = 5$ , all meshes have more node pairs with zero interference compared to their equivalent Jellyfish graphs.

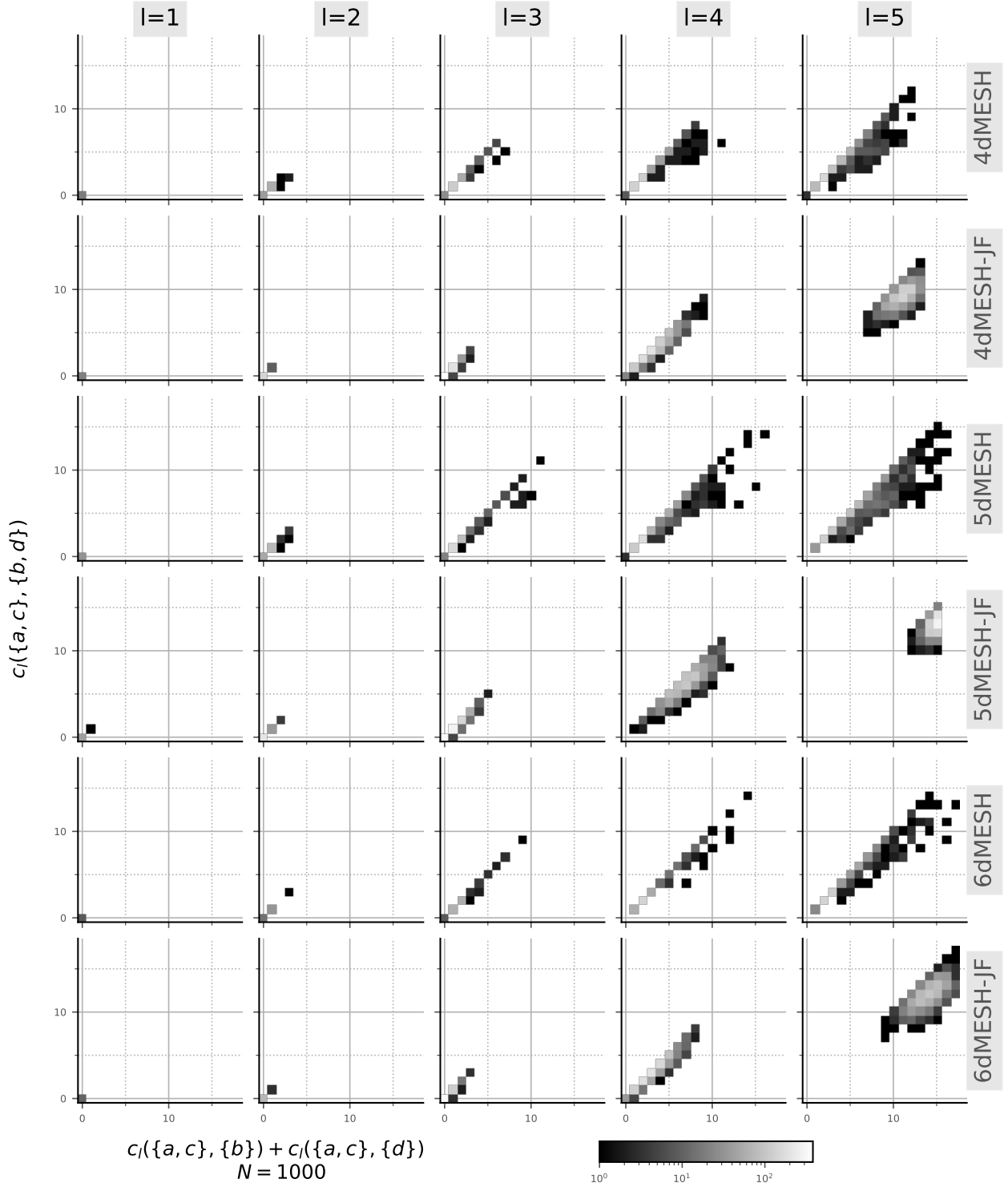


Figure 47: Two-dimensional detailed interference plots for meshes of different dimensions of lengths up to  $l = 5$ . We can see, that all meshes have similar interferences analysis. For the base mesh networks and  $l = 5$ , all components are affected by interference, while for the equivalent Jellyfish, only a subset of them are.

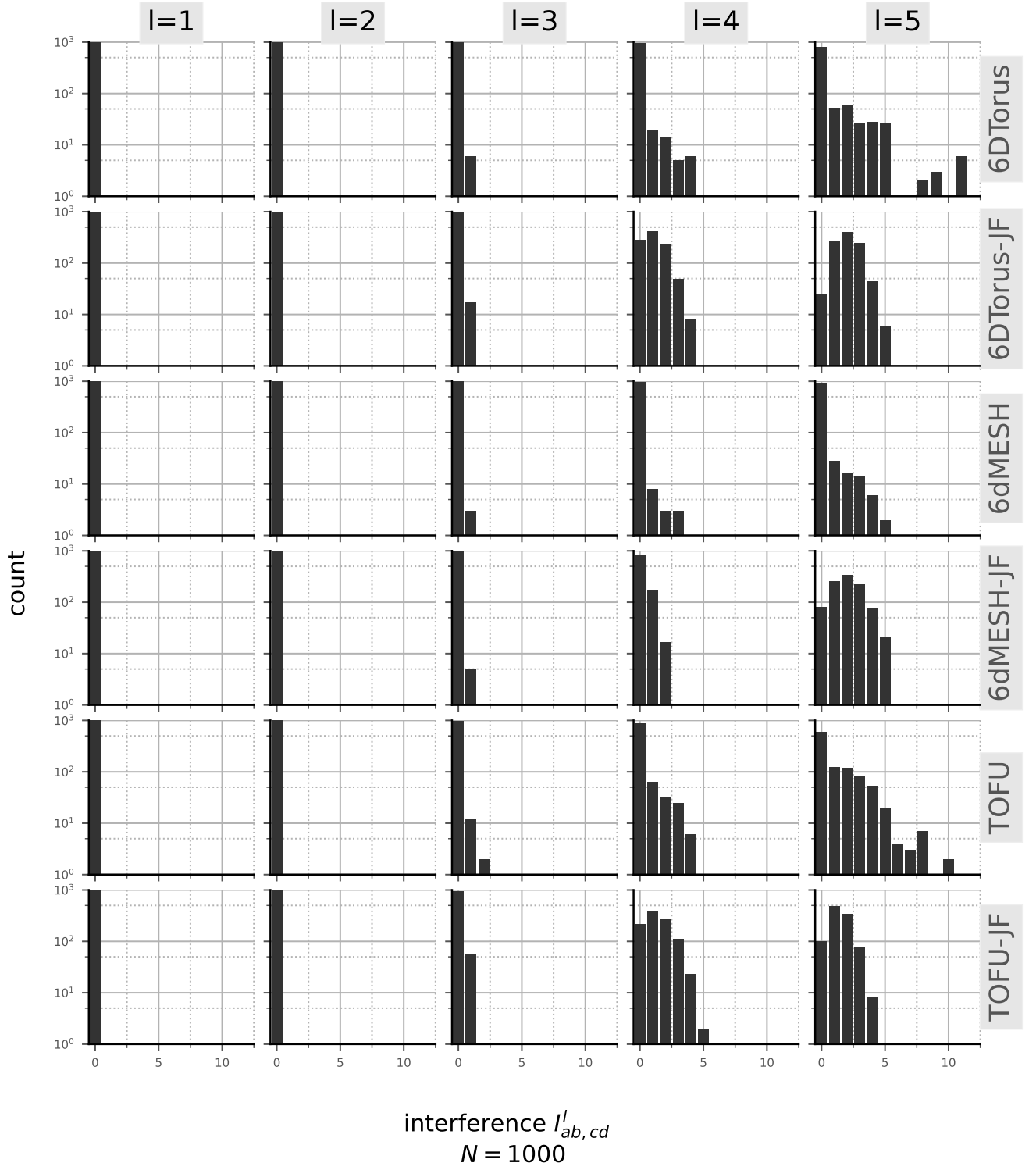


Figure 48: Interference plots for Tofu, 6D mesh and 6D torus with lengths up to  $l = 5$ . The Tofu topology has about the same amount of interference as the 6D torus network, but more than the regular 6D mesh. For  $l = 5$ , both torus and the Tofu graph have more interference than the 6D mesh and more than any of the Jellyfish equivalents.

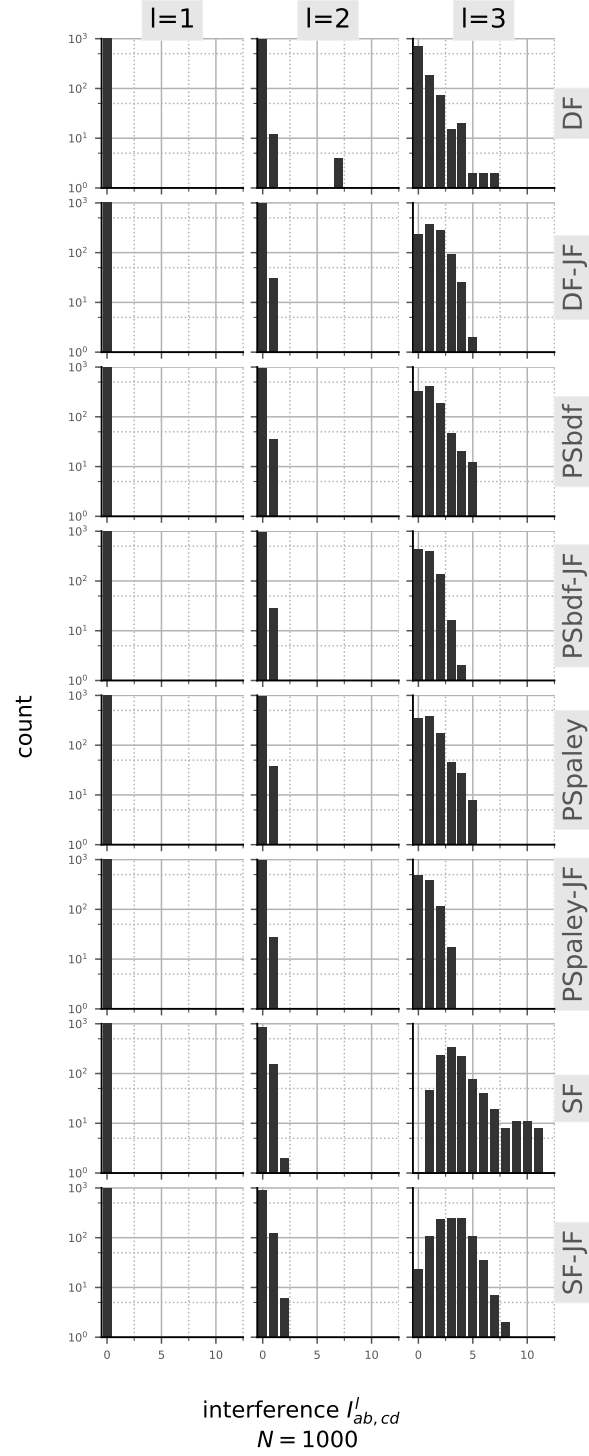


Figure 49: Interference plots for PolarStar topology with both subgraphs, Dragonfly and Slim Fly of length  $l = 1$  up to  $l = 5$ . Slim Fly has the most interference at  $l = 3$ , but less interference for all other values of  $l$ . The other topologies have the most interference for  $l = 4$ . The interference analysis looks similar for the topologies when comparing them to their equivalent Jellyfish, although all of the base topologies have slightly more interference.



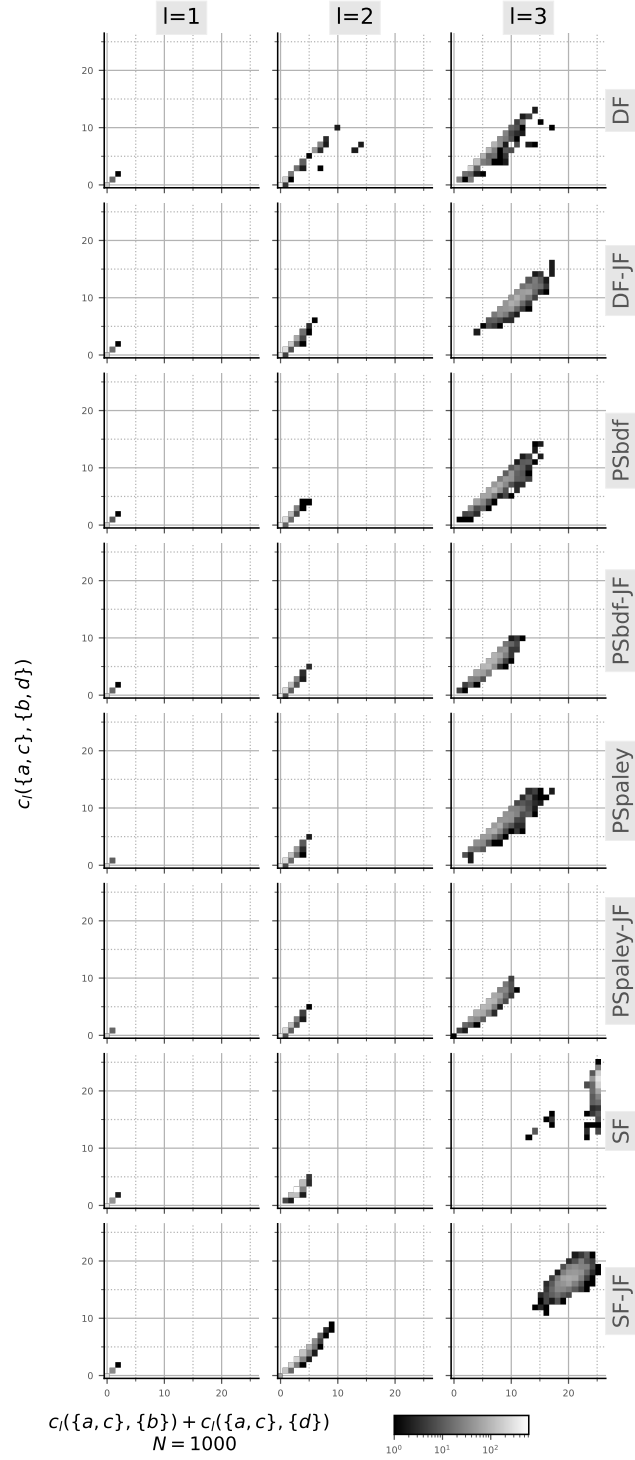


Figure 50: Two-dimensional detailed interference plots for PolarStar topology with both subgraphs, Dragonfly and Slim Fly of length  $l \leq 5$ . The detailed analysis of both PolarStar graphs are fairly similar. While for Dragonfly, the interference for  $l = 4$  is contributed by a lot of components, only a small amount of components contribute to the high amount of interference for PolarStar.

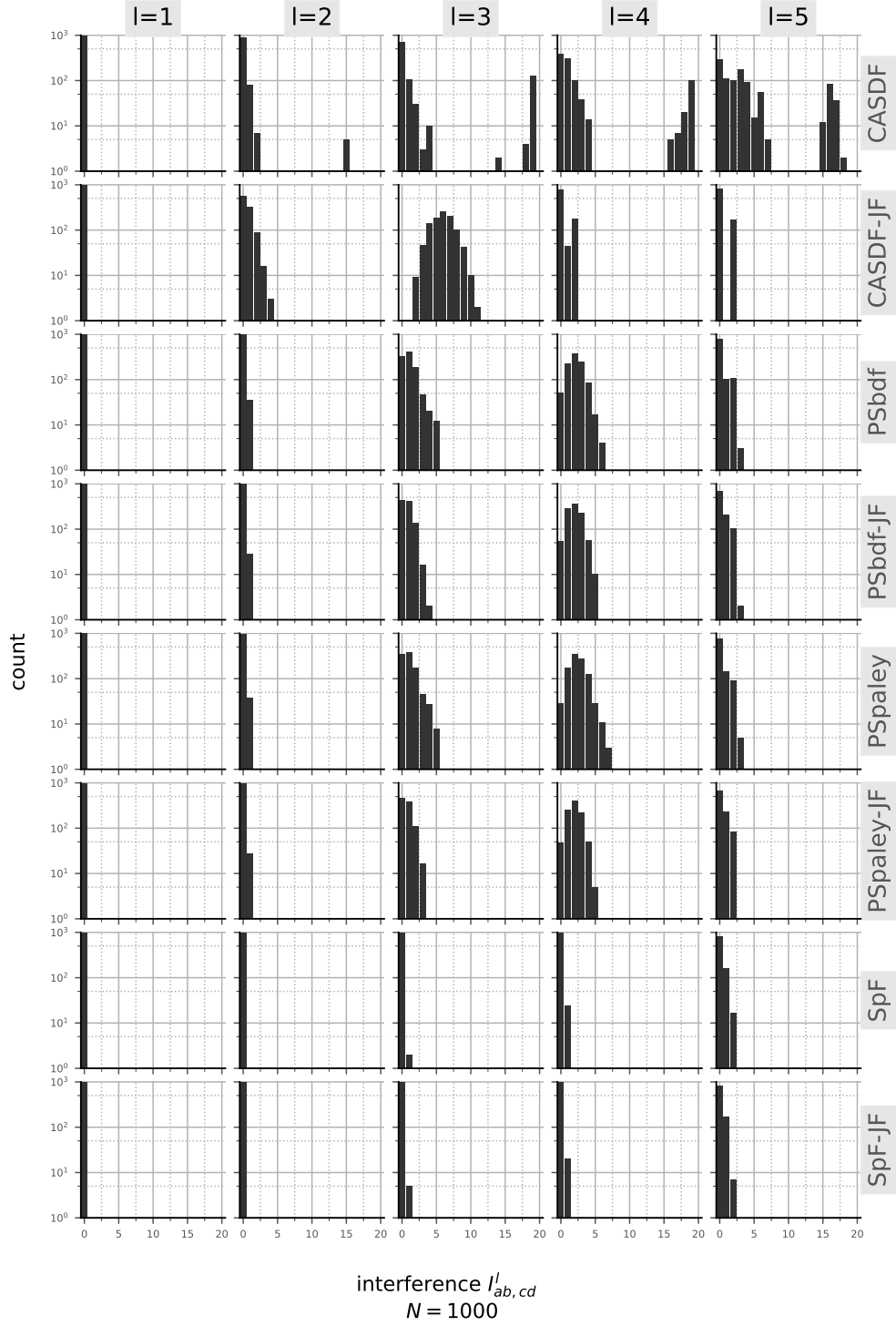


Figure 51: Interference plots for Cascade Dragonfly, Spectralfly and PolarStar with both subgraphs of length  $l = 5$ . Spectralfly has little interference compared to the other topologies for all lengths. It has even about the same amount of interference as its equivalent Jellyfish topology. In contrast to all the other topologies, Cascade Dragonfly has two groups of node quadruples: the ones with little interference and the other group with high interference. Due to the second group of nodes with high interference, Cascade Dragonfly has the most interference of all the topologies. The most extreme difference when comparing to its equivalent Jellyfish, can be seen for  $l = 4$  and  $l = 5$ . While the Jellyfish graph has at most interference 3 for any node quadruple, there are groups of nodes with interference values  $> 15$ .

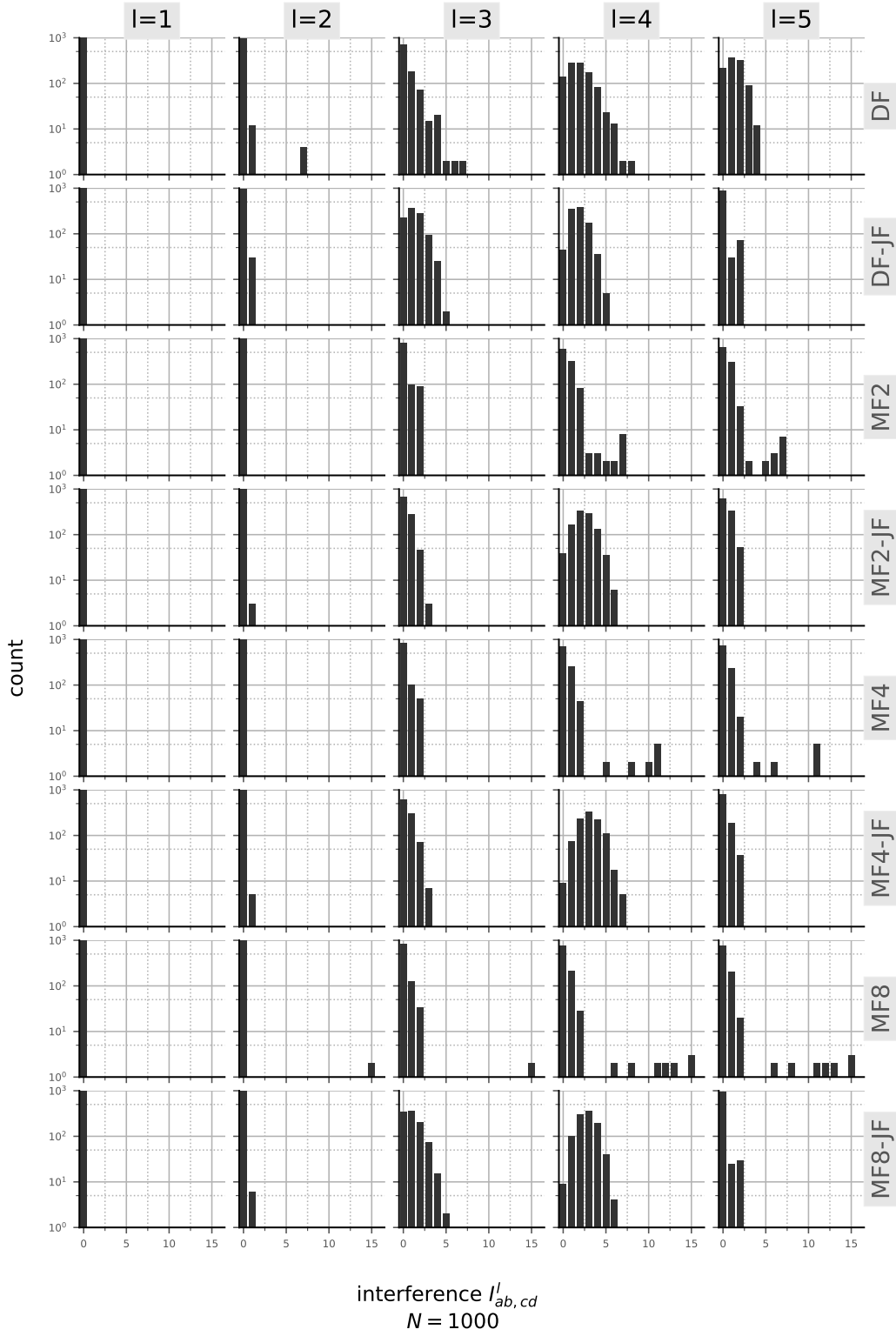


Figure 52: Interference plots for Megaflly with different gap sizes and Dragonfly of length  $\leq 5$ . In general, Megaflly has less interference than Dragonfly for length  $l \leq 3$ . Only MF8 has a few outlying node pairs with a high interference at  $l = 3$ . All Megaflly variants have most node pairs with interference between 0 and 3. For  $l \geq 4$ , all variants start to have a few  $\approx 10$  outliers with higher interference. This cannot be seen in the Dragonfly topology.

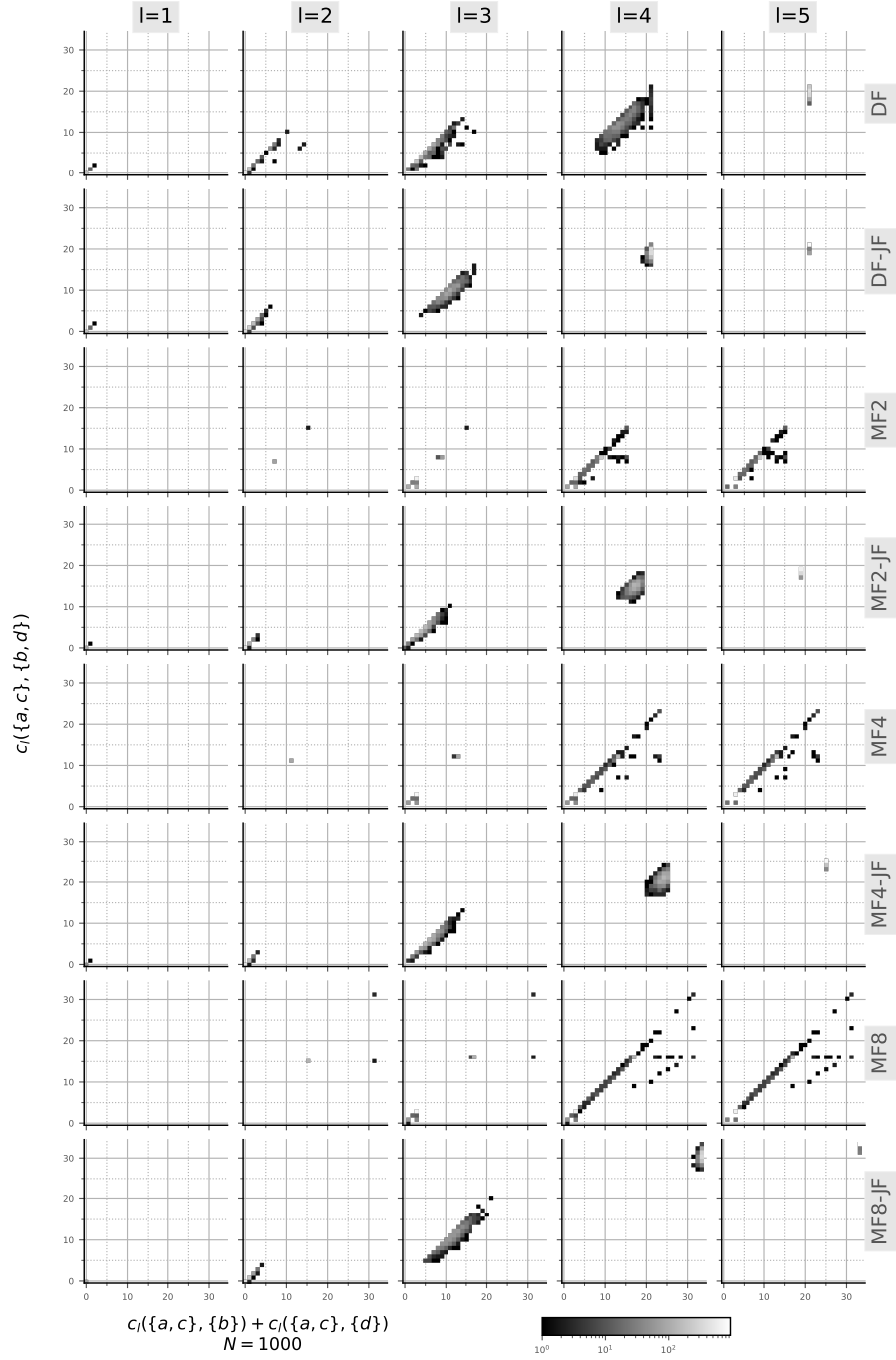


Figure 53: Two-dimensional detailed interference plots for Megaflly with different gap sizes and Dragonfly. For lengths of 2 and 3, only a few components contribute to the total interference in the Megaflly networks. This changes for  $l > 3$ , where almost all values for  $c_l(\{a, c\}, \{b\}) + c_l(\{a, c\}, \{d\})$  are possible. The same cannot be said for the analysis of the Dragonfly graph. For all lengths only a subset of possible values for the components contribute the total interference. This is also true for all Jellyfish equivalent graphs.

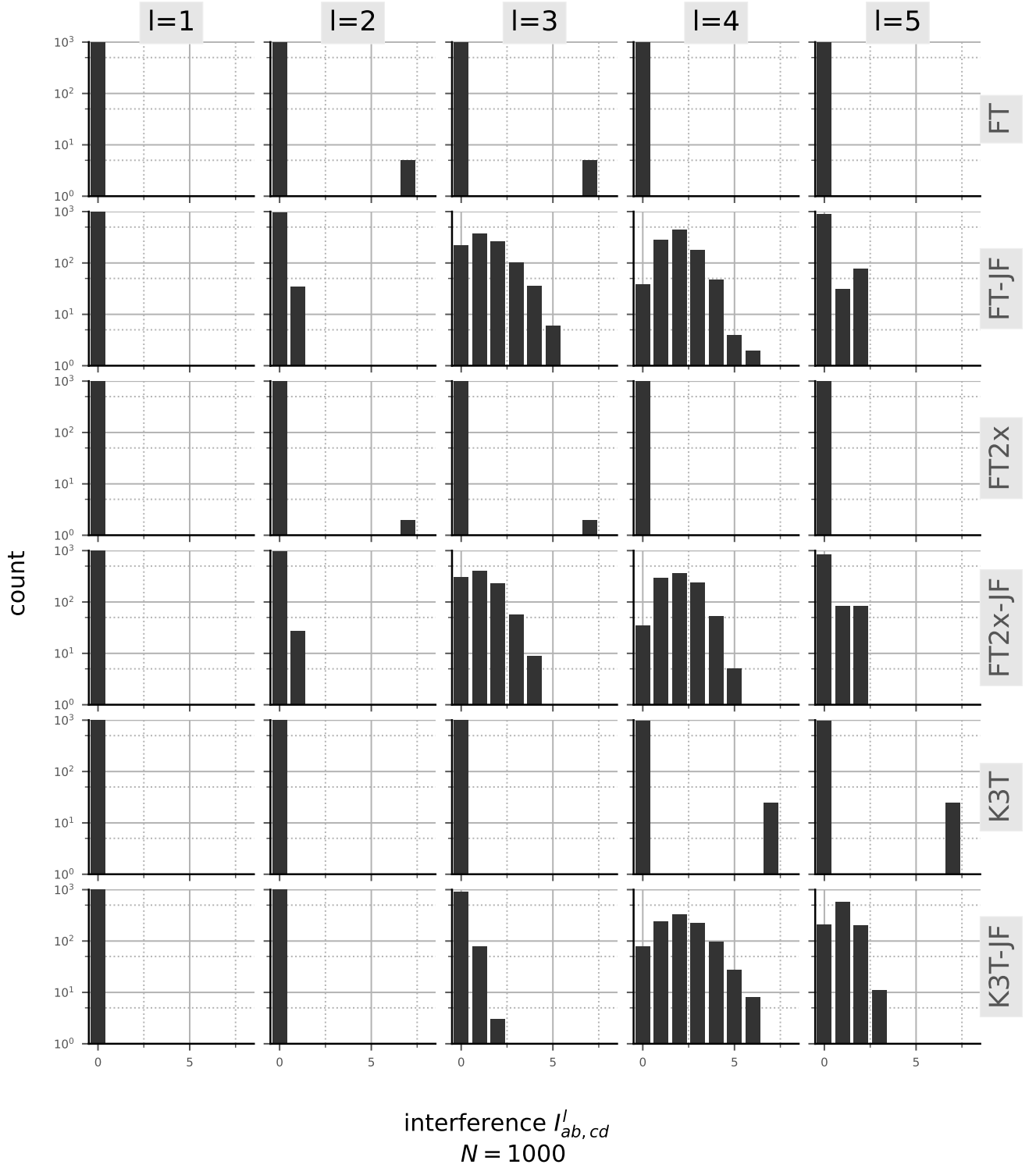


Figure 54: Interference plots for  $k$ -ary  $n$ -trees for different values for  $k$ , Fat Trees and Fat Trees2x. For all base topologies, most node pairs have an interference of zero. Both Fat Tree variants have a few outliers that have a higher interference for length 2 and 3. The  $k$ -ary 3-tree has these outliers also, although only for  $l \geq 4$ . There are also more outlying node pairs than in the Fat Tree variants.

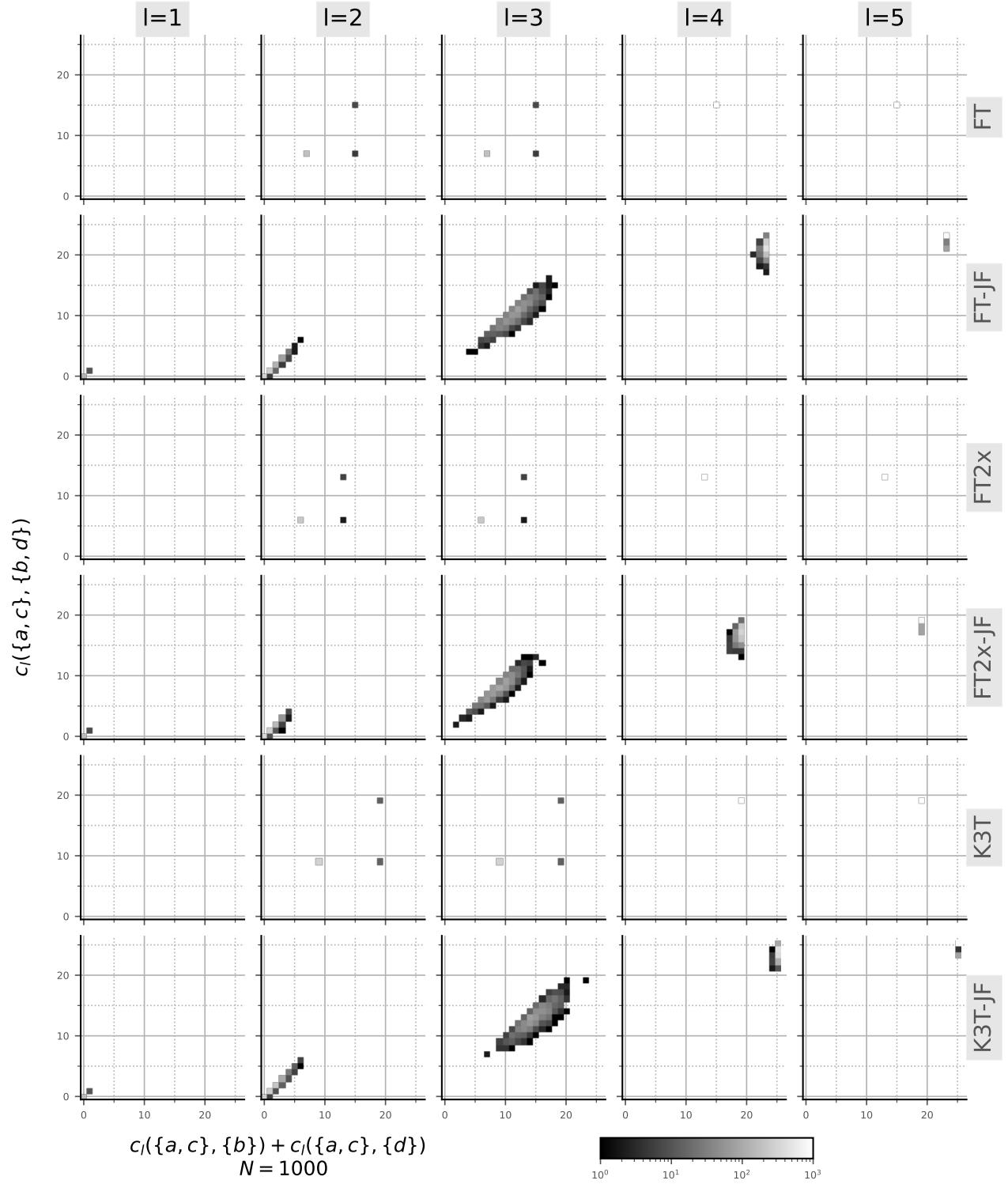


Figure 55: Two-dimensional detailed interference plots for the k-ary 3-tree, Fat Tree and Fat Tree2x. For both components, there exist only specific values they can get. Since values in the diagonal cancel each other out, we can see that there is only one peak next to interference zero for all base topologies.

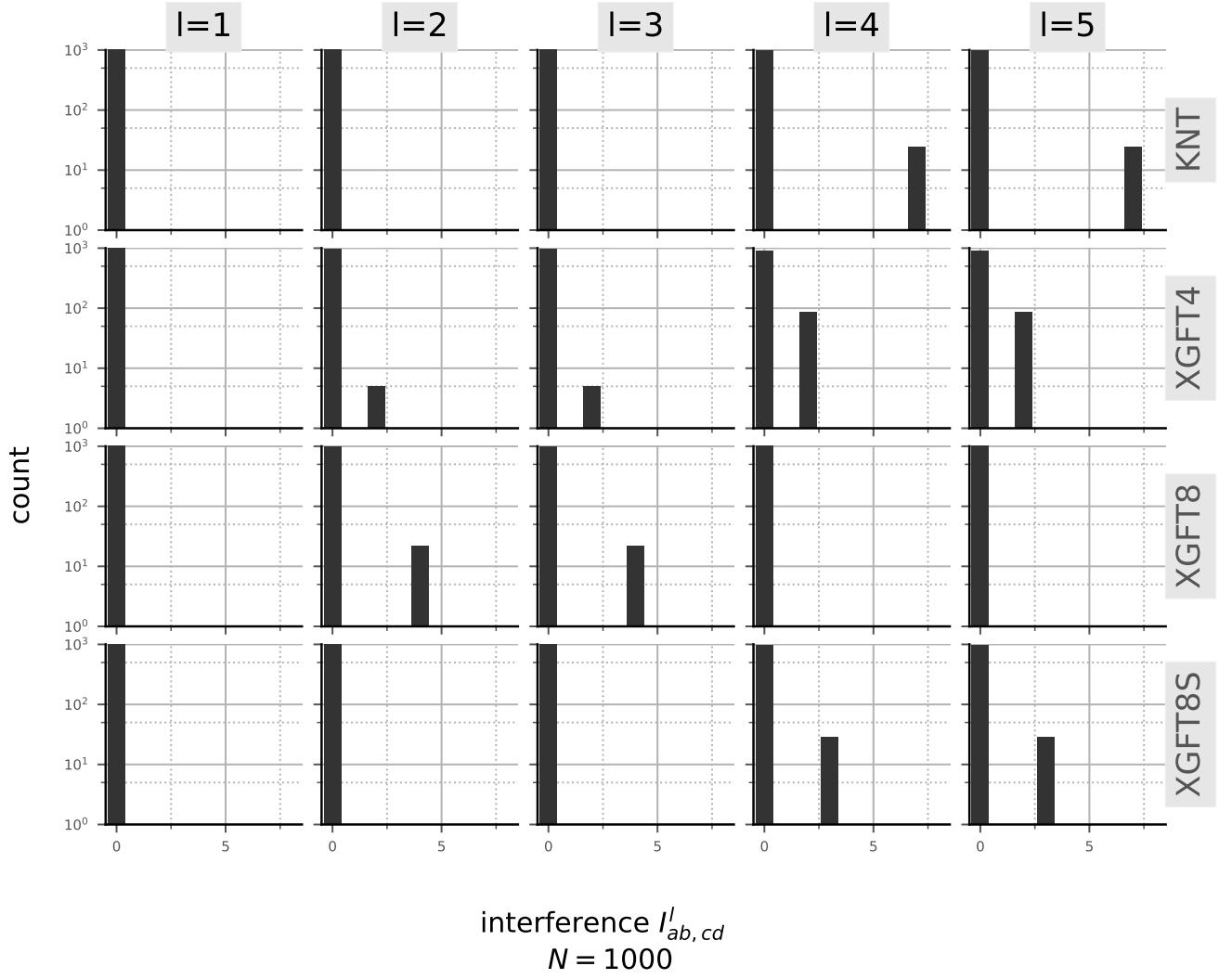


Figure 56: Interference plots for xGFTs of different variants and k-ary n-tree. Similar to before, all topologies have interference of zero for most node pairs. We can also see the peaks at a specific interference value. xGFT<sub>8S</sub> has the lowest interference of all of them. It has only interference higher than zero for  $l > 3$  and the peaks have only interference of 3. xGFT<sub>8</sub> has only an interference  $> 0$  at length 2 and 3.