RNG: Flat Datacenter Networks at Scale

Giacomo Bernardi Ratul Mahajan[†] C. Seshadhri[‡]
Enrico Carlesso Chinchu Merine Joseph Saurabh Kumar Pavan Manikonda
Luiza Popa Randy Ram Steven Robinson Elizabeth Tennent

Amazon Web Services

ABSTRACT

We design and deploy in production the first flat datacenter networks. Our design, called RNG, is based on quasi-random graphs. While the cost and fault-tolerance benefits of such topologies have been long known, their practical realization has been hampered by a lack of scalable routing and cabling approaches. RNG has a new distributed routing protocol that exploits the properties of random graphs to find a large number of edge disjoint paths between pairs of endpoints. It uses a novel passive optical device that internally shuffles cables, which makes its cabling complexity similar to that of fat trees. We show that RNG matches or exceeds the performance of fat trees for a range of traffic patterns, despite being up to 45% cheaper. RNG is now the default datacenter network for most workloads at Amazon.

1 INTRODUCTION

The simplicity of fat tree topologies has made them the workhorse of datacenter networks. But they offer a stark trade-off between cost and performance. Operators either build non-blocking fabrics in which any endpoint can transmit/receive at its full capacity; or they build oversubscribed fabrics that congest when a small, unfavorable set of endpoints transmit/receive at a high rate. Non-blocking fabrics are prohibitively expensive at scale, so organizations tend to build oversubscribed fabrics and risk congestion even for performance-sensitive workloads [16, 40, 43].

The root problem is that fat trees lack *capacity fungibility*. The strict hierarchical structure means traffic between pairs of endpoints is limited to small subsets of links in the topology. These links can congest while most others lie idle. The lack of capacity fungibility increases the cost of providing a consistent, congestion-free experience. One must provision ample capacity everywhere even if a small subset of endpoint pairs need high capacity at any given time.

Researchers have proposed reconfigurable topologies for this problem [11, 17, 21, 22, 30–32, 51, 60]. These designs use hardware such as optical switches, steerable wireless antennas, and free-space optical devices to dynamically change capacity between endpoints. Such hardware is not (yet) proven at scale. Most designs also use a centralized control plane to infer fabric-wide demand and configure the hardware. Such control planes are difficult to realize at scale for bursty, latency-sensitive workloads such as Web services.

A promising alternative that provides capacity fungibility is a flat topology with commodity routers connected as an expander graph [25, 26, 37, 45, 49]. In addition to their lower cost, such topologies are fault tolerant because the blast radius of any failure is small, unlike upper-layer failures in fat trees. These benefits of expander network topologies have been known for over a decade [45], but these topologies are not a reality because of three unsolved challenges [33, 43].

1. Routing. Shortest-path routing, commonly used for fat trees, is a poor fit for expanders because it cannot exploit the diversity of paths that exist. Some pairs of nodes have only one shortest path between them, so shortest path routing can cause congestion. Prior works recommend k-shortest paths routing which spreads traffic between the source and destination across k shortest paths [45, 49]. But k-shortest paths routing cannot be realized for large networks using commodity switches. It needs an order of magnitude more (fast, expensive) memory than what is available today (§3).

2. Cabling. Expanders connect pairs of devices that can be far away in the physical space. Such cabling is complex and expensive [33]. Worse, when more racks land, the process of incrementally adding new routers requires breaking existing connections. If routers have d links to other routers, each time a router lands, we need to break $d/2$ existing links whose endpoints are physically spread over the datacenter. This type of work risks collateral damage and slows down rack installs, a key concern for datacenter operators.

3. Performance predictability. Prior work on benchmarking expanders focus on concrete topologies with fixed parameters. It is unclear how a fabric might perform for other parameters, which makes it difficult for operators to design topologies that meet specific performance targets. They would need to search the combinatorial parameter space using simulations, which is infeasible for large networks.

[†]Also affiliated with the University of Washington
[‡]Also affiliated with the University of California, Santa Cruz

In contrast, fat tree designs can be automatically generated based on performance targets [2, 36, 41].

We solve these challenges and deploy (to our knowledge) the first production datacenter fabrics based on expanders. Our design, called RNG, connects routers using a mix of randomized and deterministic cabling segments. This construction yields quasi-random graphs that mimic the statistical properties of truly random graphs [8]. Our routing algorithm called *Spraypoint* is purpose-built for random graphs and permits a fully distributed implementation on commodity hardware. It finds a large number—close to the node degree—of edge disjoint paths between endpoint pairs and these paths minimally overlap across different pairs. These properties lead to capacity fungibility and high throughput.

We develop a cabling approach based on a new passive optical device called a *ShuffleBox* that mixes connections between routers internally. We place shuffle ShuffleBoxes at a small number of planned locations in the datacenter. This allows the number of physically-connected location pairs, a key driver of cabling complexity, to match that of fat trees.

The concentration properties of random graphs and the decorrelation of Spraypoint routing enable accurate modeling of RNG fabric performance as a function of topology parameters (e.g., graph size, node degree, etc.). We develop models for path length, number of edge disjoint paths, and oversubscription. The models make the impact of various parameters and design trade-offs (e.g., path length versus oversubscription) transparent. Operators can use them to design topologies that meet their performance targets.

We deploy two RNG-based production fabrics. Transport and application layer benchmarks confirm that RNG's performance is on par with fat trees, unimpacted by the peculiarities of expanders such as variable path lengths between endpoint pairs. Such end-to-end validation is a first for expander topologies.

Our analysis reveals that RNG topologies are 9–45% cheaper than fat trees with equivalent oversubscription ratio. The extent of cost reduction depends on the oversubscription ratio and is independent of network size and the number of switch ports. We also find that, for the same oversubscription ratio, RNG offers higher throughput than fat trees across a range of traffic patterns.

2 THE POTENTIAL OF FLAT EXPANDERS

The salient property of expander graphs is edge expansion [53]. For every minority subset S of nodes, the set of edges that lead out of S, called the cut, is large. Edge expansion provides capacity fungibility because every S has large bandwidth toward other nodes. In contrast, in hierarchical topologies such as fat trees, the cut of a subtree only leads to parents, so the expansion is poor. We illustrate with an example.

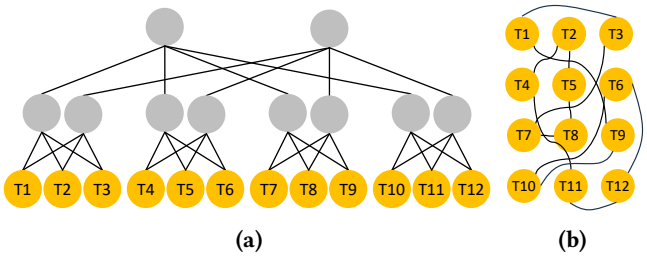

Figure 1: Example fat tree (a) and expander topologies (b). Each switch has four ports. Two ports of each ToR (T1–T12) connect to servers (not shown).

Figure 1a shows a generalized fat tree [38]. T1–T12 are ToR (top-of-rack) switches, connected via aggregation (middle) and spine (top) layers. The oversubscription ratio of this network is 3:1 since the capacity of the ToR-aggregation layer is thrice that of the aggregation-spine layer. The cut for subset of nodes with T1–T3 and their two parents is two links connecting to the spines. These nodes cannot send more than two units of traffic to T4-T12 even though most other links in the fabric are unused. Thus, capacity is stranded.

While skewed traffic patterns with a few heavy transmitters have received the most attention [25, 26], fat trees can strand capacity with uniform traffic as well. Consider an all-to-all pattern where each ToR wants to send the same data to each other ToR. The transmission rate between pairs of ToRs will be $\frac{8}{12\times9}$ of link capacity. This calculation follows from the 12×9 flows crossing the 8 aggregation-spine links. Given two units of per-ToR capacity, this traffic pattern strands roughly 60% $(2 - 11\frac{8}{12\times9})$ of the ToR uplink capacity.

Expanders, such as the random graph between T1–T12 in Figure 1b, can do significantly better. If T1–T3 are the only senders, they will be limited largely by their local capacity when other nodes are silent. Because of the expansion property, there is no small cut that constrains traffic. Even for the all-to-all traffic pattern, given a capable routing algorithm, we can effectively use all links in the topology.

Capacity fungibility lowers cost because the deployed capacity is more efficiently used, allowing smaller deployments. A simplistic view of the lower cost of flat expanders is that all upper-layer routers are removed. But an expander that is performance-equivalent to a fat tree may need more ToR uplinks (fabric-facing ports) because some uplink capacity is consumed by traffic relayed for other ToRs. It may thus need more ToRs to support the same number of servers, given a fixed number of ports per ToR. Despite this effect, expanders can reduce cost by up to 45% (§9.4).

Expander topologies are also more fault-tolerant. Edge expansion makes it difficult to partition the network, and flat

structure lowers the impact of failures. In a fat tree, upper-layer router failures have a large impact because they carry traffic for many endpoint pairs. Even a single spine router failure in Figure 1a halves the available capacity for most ToR pairs. In Figure 1b, there are no such special routers.

We focus on multi-tenant datacenters with heterogeneous workloads. In the future, we will investigate using expanders for specialized workloads such as large-scale AI training. These workloads may demand specific topologies such as Rail-optimized fat trees and may want local capacity islands such as those at the aggregation layer in Figure 1a [27, 59]. Flat, random topologies do not have aggregation islands.[1]

3 REQUIREMENTS AND CHALLENGES

Expander topologies are theoretically promising. But to be practical at scale at Amazon and supplant fat trees, they must meet several requirements. Prior expander-based designs [4, 45, 49] do not meet these requirements.

Realizable using commodity switches. The network should be realizable using commodity switches and forwarding ASICs which have limited memory. While custom hardware can be developed, it increases cost and poses substantial technical and supply-chain risks.

Prior designs cannot be realized for large networks using commodity switches. To counter the limitations of shortest paths in expander graphs, they usually propose k-shortest paths routing. Its typical implementation requires tunnels (e.g., MPLS, VLAN [35]) or forwarding based on source (in addition to destination) addresses. Suppose we want to build a network with n=10K routers and use the recommended value of k=8 [45, 49]. If each path traverses 4 routers, we need 320K forwarding entries per router on average ($4kn^2$ entries spread across n routers). Current switches support only 4-16K such entries,[2] which is 20–80x lower. State reduction techniques [47, 48] cannot bridge this gap and adding so much memory is prohibitively expensive. Harsh et al. cleverly use VRFs (virtual routing and forwarding) to create non-shortest paths, but this proposal too does not scale given limits on the number of VRFs in commodity switches [23].

Deployment and operational simplicity. Deployment and operational simplicity are critical for network reliability. Simplicity is multi-faceted; we focus on control plane and physical operations to deploy and incrementally expand the network. For the control plane, we prefer demand-oblivious, fully distributed routing (like today). For deployment and expansion, we want the number of cabling steps to be small and fast to minimize risk and expand faster.

Physical cabling is challenging for expanders because they need to connect pairs of devices uncorrelated to physical space. Jellyfish, which is based on truly random graphs, proposes to simplify cabling by removing routers (ToRs) from server racks and placing them centrally [45], but this approach increases latency between servers that share a rack, an important consideration for some applications. It also increases cabling cost because cheap, copper-based intra-rack connectivity would need to be replaced with expensive, optical connectivity. All current approaches also require changing many existing connections each time a new rack lands, a frequent operation in datacenters.

Predictable performance. Amazon operators deploy network fabrics to meet specific capacity (number of servers) and performance (e.g., oversubscription) targets. For expanders to be acceptable, they must be able to easily and confidently create topologies that meet their targets. Prior work benchmarks topologies with specific parameters and does not show how to create topologies for specific performance targets.

RNG meets these three requirements via: (1) A new routing algorithm called *Spraypoint* that finds a large number of edge-disjoint paths. Like OSPF and BGP, it is demand-oblivious and permits a fully-distributed operation on commodity hardware. (2) A cabling approach based on a passive optical device called a *ShuffleBoxes*. It limits the number of endpoints that are physically cabled/re-cabled when the datacenter expands. (3) High-fidelity models for key performance measures such as throughput and path length. We describe these elements in more detail below.

4 RNG OVERVIEW

RNG is based on a flat graph where routers interconnect through a mix of deterministic and randomized choices. Given their controlled randomness, our graphs are not truly random; they are quasi-random graphs that behave like truly random graphs and are optimal expanders [8, 12, 39][3]. RNG routers connect to each other and to external devices (e.g., servers, other fabrics). We break out each fabric-facing physical port into individual lanes (e.g., a 400 Gbps port into 4x100 Gbps lanes), each of which uses a separate fiber pair and can establish adjacency with a different remote router. Breakouts increase graph degree, lowering hop count and oversubscription. In the rest of the paper, a router uplink refers to a breakout lane.

[1] RNG does have rack-level islands like fat trees. The lack of aggregation islands is not a problem for general workloads in multi-tenant datacenters. Coordinated application deployment across topologically related racks is hard to realize here; tenants come and go continuously, with each needing different server counts and types. That is why hyperscalers aim for fabric-wide, uniform capacity pools [18].

[2] This table is different from the much larger table for destination-based lookup which uses longest-prefix matching.

[3] Further, unlike structured constructions like Xpander [49] and Slim fly [4], random graphs can support routers with different degrees [45]. RNG supports such routers, but we omit this discussion for lack of space.

Graph	n, d	# of routers, router uplinks
Spraypoint	p, h	# of waypoints, next hops
	ℓ	# of levels
ShuffleBoxes	d_r, d_c	# of r-ports, c-ports
	f_r, f_c	# of fiber pairs in r-port, c-port

Figure 2: Key design parameters of RNG

The control plane and load balancing in RNG follows today's common paradigm. The routing protocol, Spraypoint, computes next hops at each router for all destinations; and routers use ECMP (equal cost multipath) to spread traffic across all next hops for the destination. Figure 2 summarizes the key design parameters of RNG.

5 SPRAYPOINT ROUTING

Spraypoint constructs a large number of edge disjoint paths between endpoint pairs, which increases network throughput by offering more independent options to load balancing mechanisms like ECMP. While we analyze and deploy it on random graphs, Spraypoint works on any expander graph. It is based on the observation that high fan-out at the source and high fan-in at the destination suffice for creating many edge-disjoint paths in an expander. The middle has many edges because of the expansion property. Spraypoint achieves high fan-out by *spraying* packets at the source—all neighbors are eligible next hops and one is selected based on ECMP hashing. It achieves high fan-in by channeling traffic via *waypoints* that are spread all around the destination.

Forwarding paths. Spraypoint has two parameters p and h which control the number of waypoints per destination and the number of eligible next hops after the spraying step. These parameters offer a trade-off between path length and the number of edge disjoint paths. There is an auxiliary parameter ℓ which depends on these parameters and the graph size (n). To compute paths to a destination t, Spraypoint partitions all nodes into the following *levels*.

(1) $WP_0(t)$: Base waypoint level with all neighbors of t.

(2) $WP_{l\in[1,\ell]}(t)$: Higher-level waypoints. $WP_l(t)$ has p randomly-selected neighbors of each node in $WP_{l-1}(t)$. Neighbors in earlier waypoint levels and t itself are not eligible for selection.

(3) $IR(t)$: *Inner ring* with all neighbors of $WP_\ell(t)$ not in previous levels.

(4) $OR(t)$: *Outer ring* with all nodes not in any level above.

Figure 3 shows an example graph for ℓ=1 and p=2. $WP_0(t)$ has all neighbors of t: $\{v_1, v_2, v_3, v_4\}$. $WP_1(t)$ has two randomly selected neighbors of each v_i. In the example, w_1, w_2 are selected from v_1, w_3, w_4 are selected from v_2, and so on. Node v_1 ends up with three neighbors in $WP_1(t)$, because an additional neighbor (w_8) was selected via v_4. The neighbors of v_i that are not selected in $WP_1(t)$ drop into the rings, along with neighbors of $WP_1(t)$.

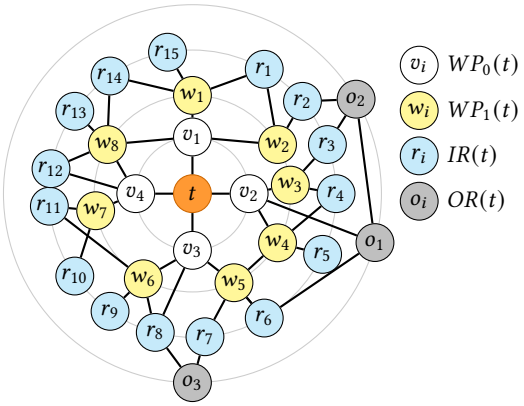

Figure 3: Example graph with Spraypoint levels for t.

The levels control traffic flow between any source s to t. The first step is *spraying*, where s forwards to a randomly chosen neighbor (i.e., ECMP). Spraying is independent of t and happens even if s and t are adjacent.[4] Post spraying, each intermediate node u follows the "pointing" rules.

(1) If $u \in WP_0(t)$, forward to t.

(2) If $u \in WP_{l\in[1,\ell]}(t)$, forward (ECMP) to one of h neighbors randomly selected from $WP_{l-1}(t)$.

(3) If $u \in IR(t)$, forward to one of h neighbors randomly selected from $WP_\ell(t)$.

(4) If $u \in OR(t)$, forward to one of h neighbors randomly selected from those with shortest paths to $IR(t)$ nodes.

Let us see Spraypoint forwarding in action for a packet from v_2 to t in Figure 3. Assume $h = 1$. First, v_2 sprays to one of t, w_3, w_4, or o_1. If the packet is sent to t, its journey is complete. If it is sent to w_3 or w_4, it will go back to v_2 per the second rule above, and then to t. Spraying happens only at the source; if a packet happens to return to the source, it follows the pointing rules. If the packet was sprayed to o_1, it reaches t via r_6, w_5, and v_3.

Why spraying. Spraying exploits the expansion property to generate many paths between source-destination pairs. In general, there may be only a few *shortest* paths from a source s to a destination t. But in an expander, there are many edge disjoint *short* paths between all pairs. A simple heuristic to use these paths is to follow shortest paths to t from all neighbors of s. Spraying implements this heuristic.

[4]Spraying is reminiscent of Valiant Load Balancing [50]. But unlike VLB, which uses arbitrary nodes as intermediaries, it uses only neighboring nodes. Its primary goal is high fan-out at the source, not load balancing, though high fan-out does aid load balancing.

Why waypoints. Spraying alone fails in some cases. Suppose s is a neighbor of t. After s sprays, its neighbors use shortest paths to t. Unfortunately, unless a neighbor is directly connected to t, which will be rare in a large graph, the shortest paths go via s. So almost all neighbors route the traffic back to s, and the $s \rightarrow t$ link will congest. Higher-level waypoints ($WP_{l \in [1,\ell]}(t)$) draw out traffic further, preventing a collapse on this link. Further, selecting p of d nodes builds a p-ary forwarding graph rooted at each neighbor of t, which helps spread traffic across all neighbors (high fan-in). Otherwise, some neighbors may not be used.

Setting ℓ. To achieve the goals of spraying and waypointing, we set ℓ=max(1, $\lceil \log_p(n/2d^2) \rceil$). The size of the set $WP_\ell(t) \cup IR(t)$ will be at least $n/2$ for this choice of ℓ (§7), which reduces path lengths because each node in $OR(t)$ will connect directly to this set with high probability.

Path length variability. Like other proposals for routing in expander graphs [23, 45, 49], Spraypoint computes paths of different lengths between a pair of endpoints. This variability will not impact single-path transport protocols like TCP which sample only one path (based on ECMP), but it could confuse multipath protocols [42, 44, 57] that spread traffic across multiple paths (by varying packet headers used by ECMP) if they use latency differential as a congestion or load balancing signal. In a datacenter that spans 300 meters and uses 100 Gbps links, a 2-hop path length difference creates a maximum latency difference of 4.4 μs (0.7 μs transmission time for a 9 KB packet, 1.5 μs propagation delay per hop). This differential is low compared to endhost latencies and a small amount of queuing (6 packets) will wipe it out. Benchmarking on production fabrics confirms the absence of performance issues for multipath protocols (§8).

Distributed implementation. Spraypoint permits a fully distributed implementation as a link-state protocol, where every node has a full view of the topology. Nodes can compute levels for each destination and then compute local next hops by applying the pointing rules. All nodes make identical waypoint selections, via deterministic hashing of a shared key, ensuring that all pointing graphs are loop free.

We implement spraying using VRFs (virtual routing and forwarding), which enable different routing logic for different interfaces. Unlike prior work that uses VRFs to create non-shortest paths [23] (which does not scale), we need only two VRFs. All server-facing interfaces of a router use a VRF that sprays incoming traffic to all fabric-facing interfaces. (One exception is traffic destined to servers connected to the same router, which is forwarded directly to those servers.) All fabric-facing interfaces use a second VRF that follows the pointing rules. This setup allows a packet to revisit its source (at most once) without looping indefinitely.

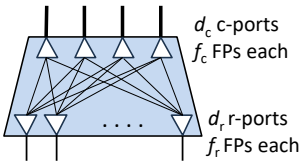

Figure 4: A ShuffleBox.

Resource requirements. Spraypoint uses two types of router memory. The first is longest-prefix matching (LPM) memory, which maps the destination address of an incoming packet to an ECMP group id. LPM memory use depends on the number of network prefixes and is similar to fat trees.[5]

The second type of memory is to map ECMP group id to next hop set. Its required size depends on how ECMP groups are setup. The two possibilities are: (1) a separate ECMP group for each destination node, which consumes $O(nh)$ memory; and (2) pre-define all possible d^h groups and use the corresponding one for each destination, which consumes $O(hd^h)$ memory. Based on n and d, we pick the method that supports the larger h while fitting in the memory. For 128-port switches, the value of d is around 64, so a minimum of h=2 is always feasible independent of fabric size ($hd^h \approx 8K$).

The computational complexity of Spraypoint is $O(n^2 d)$ per node because each node inspects nd edges per destination to compute levels. The CPUs of modern commodity switches can handle such complexity [29]. For comparison, the complexity of k-shortest-paths routing is $O(kn^2(d + \log n))$ [54].

6 CABLING QUASI-RANDOM GRAPHS

We explain how quasi-random graphs are realized in RNG after a brief background on datacenter cabling.

Datacenter cabling. Datacenter space is typically partitioned into O(10) *rooms* (not necessarily with dividing walls). Each room hosts some number of racks. Before server racks land, the room is *prepared* by installing power, cooling, and basic cabling infrastructure including patch panels and inter-room trunk cables. When server racks land, additional cables are installed to connect them.

The cost and complexity of cabling stems from three main factors: (1) the number of connectors and endpoint pairs, where a patch panel is an endpoint and so is a ToR; (2) the length of cables and the fraction that are intra- versus inter-room—inter-room cabling is more expensive because it needs extra infrastructure; and (3) the amount of cabling that can be done during room preparation versus when racks land—later cabling is riskier and slower.

[5] An exception is when hierarchy is leveraged to aggregate some prefixes. Such aggregation is not feasible in flat expanders. Because LPM memory is huge, it can support even the largest Amazon DCs without aggregation.

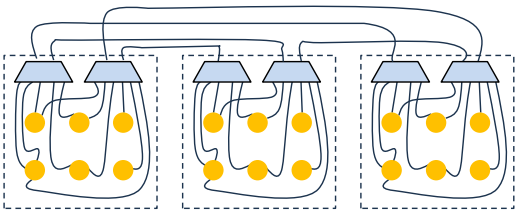

Figure 5: Cabling in a datacenter with three rooms. Routers connect to r-ports of ShuffleBoxes in the room, and ShuffleBoxes inter-connect via c-ports.

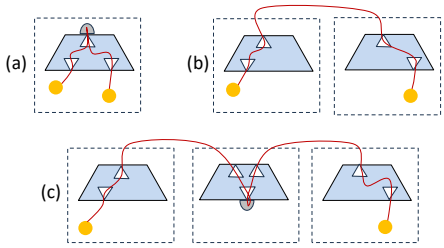

Figure 6: Physical connectivity patterns that enable logical connectivity. The hats denote ShuffleBacks.

Physical connectivity in RNG. We build a quasi-random graph using ShuffleBoxes (Figure 4). Each ShuffleBox has d_r *r-ports* that connect to routers and d_c *c-ports* that connect to other ShuffleBoxes. R-ports and c-ports terminate, respectively, f_r and f_c fiber pairs (FPs) each. An FP enables duplex communication between routers. ShuffleBoxes connect FPs between r-ports and c-ports ($d_r \times f_r = d_c \times f_c$). We describe these parameter values later.

To connect routers via ShuffleBox, we deploy a set of ShuffleBox, called a *shuffle panel*, in each room (Figure 5). Router uplinks connect to randomly selected r-ports in the panel. The c-ports of panels in different rooms are also randomly connected. If there are R panels, each with C ShuffleBoxes, we connect approximately $\frac{(R-1)Cd_c}{R}$ c-ports of each panel to other panels, essentially building a random graph between panels. Unconnected c-ports of a panel have a *ShuffleBack*, a special connector that bridges pairs of FPs coming into the c-port (from r-ports). It bridges FP1 to FP2, FP3 to FP4, and so on. Panel r-ports that are not connected to routers also have a ShuffleBack that bridges c-port FP pairs. Routers connect only to the ShuffleBoxes (never directly), and topology modifications are achieved by changing only connections between the ShuffleBoxes (as explained below).

Figure 6 shows three ways in which two routers can logically connect: (a) routers in the same room connect to the same ShuffleBoxe and their FPs are shuffled to a c-port with a ShuffleBack;[6] (b) routers in different rooms whose FPs are shuffled to connected c-ports; (c) routers whose FPs end up at the same ShuffleBacked r-port in another room. In the final pattern, the two ToRs may be in the same room.

Connecting via ShuffleBoxes has higher optical loss than direct connectivity, but this loss is within the margins of commodity transceivers. Our construction can produce even longer end-to-end paths, e.g., if the right r-port in Figure 6c had a ShuffleBack instead of connecting to a router, the path will extend and go back out via a c-port. We disable paths with over seven connectors to maintain optical signal quality.

Incrementally deploying rooms and racks. RNG's physical connectivity can be done incrementally. §A describes the process. To summarize the key steps: (1) install a new shuffle panel when a new room is prepared; (2) if it is not the first room, rebalance inter-panel connectivity by removing some existing (randomly selected) c-port connections and ShuffleBacks and using the opened c-ports to connect to the new panel; and (3) when racks land in the room, connect router uplinks to randomly selected r-ports.

ShuffleBox configuration. We use f_r=4, the number of breakout lanes for a 400 Gbps physical port. So that each FP coming into an r-port from a router can reach different places via different c-ports, d_c=f_r=4. We use f_c=32 to balance availability of commodity connectors (larger values are less common) and effort to rebalance inter-panel connectivity (small values increase effort). Finally, since $d_r \times f_r = d_c \times f_c$, we get d_r=32. With this configuration, the shuffling pattern of ShuffleBoxes is simple: each c-port FP goes to a different r-port, and each r-port FP goes to a different c-port (full bipartite).

Cabling complexity. RNG's cabling has low complexity per the three factors outlined earlier. With n routers and R rooms, the number of physically-connected endpoint pairs is $n + \frac{R(R-1)}{2}$ as opposed to n^2 logically-connected pairs. We count each shuffle panel as one endpoint because after the trunk cables reach the panel, the intra-panel distribution is simple. Inter-room cables are limited to $\frac{R(R-1)}{2}$ trunks between shuffle panels. Other than router to r-port connectivity, all cabling can be done before racks land. Along these factors, cabling complexity of RNG is on par with fat trees. Rebalancing inter-panel connectivity is a unique activity in RNG, but it is needed only a small number of times, when new rooms are prepared.

Cost. ShuffleBoxes and ShuffleBacks are novel passive optical components that RNG uses to simplify cabling. Being passive, their cost is low relative to switches and transceivers and

[6] With random mapping of router uplinks to ShuffleBox r-ports, there is a chance that two uplinks of the same router land on the same ShuffleBox and are bridged via a c-port ShuffleBack. For realistic router counts, the probability of self-edges is negligible since each ShuffleBox is small.

similar to traditional patch panels [14] and loopbacks [15] that are often used to simplify fat tree cabling.

Resulting graphs. The logical graph built using our physical construction is an optimal expander despite using deterministic router-to-ShuffleBox connections and constraining randomness (since c-port edges are balanced between rooms and bundle edges between multiple router pairs. Because each bundle has random routers and c-port connections are random, our analysis confirms that RNG topologies have the same spectral gap [53], a measure of expansion, as unconstrained random graphs.

7 MODELING PERFORMANCE

Following the standard practice for quasi-random graphs, our analysis assumes that the graphs are truly random. The combination of random graphs and spraying-induced decorrelation enables modeling of RNG's performance. Our models inevitably make simplifying assumptions but predict performance quite accurately (§9). We summarize the key results in this section and include details in the appendix. Our models give approximate formulas that give accurate predictions of the actual quantities. For convenience, we do not give formal statements here, and defer all precise mathematical statements and proofs to the appendix.

We consider operating regimes in which (i) $2(\ln(n)+5) \leq d \ll n$, (ii) $p \geq (n/d^2)^{1/\ell}$, and (iii) $h \ll d$, where ℓ is the number of waypoint levels in Spraypoint. (We used the second criterion in §5 for setting ℓ). Performance is predictable in this regime because distributional tails due to randomness are trimmed. At the same time, this regime is broad. Since commodity switches support at least 128 breakout lanes [5, 6], we can easily pick values of d that exceed the stated lower bound ($\approx$ 28 for n=10K) for even large fabrics, and we can pick compliant values of p and h . Further, ℓ=1 suffices for n=10K and typical values of d and p. Our oversubscription model focuses on that regime for simplicity.

7.1 Edge disjoint paths

Spraypoint exploits the expansion property to compute many edge disjoint paths. The formal statement and proof are given in §F.

Model 7.1. *With high probability, the number of edge-disjoint paths between s and t are approximated by:*

$$\begin{cases} d(1-\exp(-h)) & \text{if } s \notin WP_0(t) \\ \min[d-p, d(1-\exp(-(1-p/d)h))] & \text{if } s \in WP_0(t) \end{cases}$$

A detailed explanation of this model is in §F, but we summarize here. Consider $s \notin WP_0(t)$ (i.e., s is not a neighbor of t), and spraying a single packet to every neighbor of s. The number of edge disjoint paths is closely related to the number of neighbors of t reached by these packets. Modeling this process as a balls and bins calculation [52], where each ball is thrown h times, leads to the formula above.

When $s \in WP_0(t)$, p neighbors of s are in $WP_1(t)$. When traffic is sprayed by s, the traffic to these waypoints potentially comes back to s. For the other $d-p$ neighbors, we get a version of the balls and bins game.

Takeaways. (1) For both types of sources, the reduction in the number of edge disjoint paths compared to the maximum possible value of d is proportional to $\exp(-h)$. Thus, a low value of $h > 1$ suffices for good performance. The change in $\exp(-h)$ from h=1 (0.37) to h=2 (0.13) is large, but it tapers off rapidly as h increases. (2) For neighboring sources, high values of p reduce the number of edge disjoint paths.

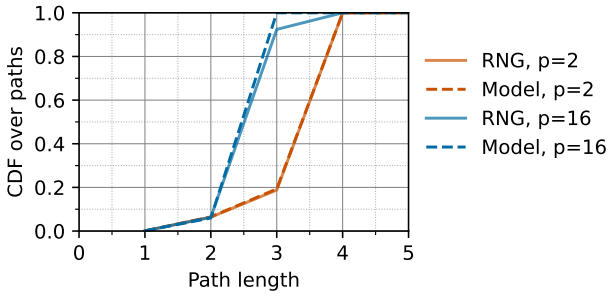

Figure 7: Path length distribution.

7.2 Path length

Path lengths in RNG depend on the sizes of various Spraypoint levels. A packet is sprayed to a random node, and the number of pointing hops depends on the node's level. We can bound the resulting path length distribution.

Model 7.2. *With high probability, the fraction of paths of length i is approximated by:*

$$\begin{cases} 1/n & i = 1 \\ p^{i-2}d/n & 2 \leq i \leq \ell+2 \\ \exp(-p^\ell d^2/n) & i = \ell+4 \\ \textit{rest} & i = \ell+3 \end{cases}$$

§E has the proof, but Figure 7 compares the model to a simulated fabric. It considers two values of p, with n=1K and d=64. The model matches the simulation well, especially for $p = 2$. When $pd > n$, as for p=16, lower-order terms that determine level sizes begin to matter. We ignore them for simplicity, but our analysis can be extended if needed.

Takeaways. (1) When $\ell = 1$ (practical regime), the maximum path length is 5, and 5-hop paths are a negligible fraction (e.g., $\exp(-pd^2/n) \approx 0.0003$ for n=1K, d=64, p=2). (2) In this regime, the average path length of RNG is less than that

of 3-tier fat trees, where the vast majority of ToR-to-ToR paths have 4 hops. (3) RNG paths have variable lengths even across the *same* endpoint pair, unlike fat trees where all paths between two endpoints are equal. (4) h does not impact path length.

7.3 Throughput (oversubscription ratio)

A key measure of a network's throughput is oversubscription ratio. It is the minimum fraction of traffic that the network can deliver, without exceeding link capacities, across all possible traffic matrices. Unlike for fat trees, one cannot determine oversubscription for expander topologies through a simple analysis of the graph structure.

Our oversubscription model uses two observations. First, the worst case traffic matrix is a *matching* where every node sends at full rate to exactly one other node and receives at full rate from exactly one other (possibly different) node [37]. We consider a matching with randomly selected source-destination pairs, which can be viewed as modeling *stochastic* oversubscription. This random traffic matrix might not be the worst case. While matchings with long paths can help find worse traffic matrices [25, 28], spraying in RNG decorrelates path lengths.

The stochastic analysis is common in random graph theory—argue that worst case for graph properties (e.g., cuts, expansion, matchings) is close to the random case, using a union bound argument [13, 34]. Second, the network throughput is maximized for a given traffic matrix, or the oversubscription ratio is minimized, when traffic takes the shortest possible paths (Lemma G.2).

We estimate oversubscription by estimating μ_i for each flow in a random matching, where μ_i is the maximum fraction of flow carried by i-hop paths. The details are in §G.2; we summarize below.

We consider $i \in [2, 5]$ because the probability of length 1 paths is negligible and the maximum path length is 5 when $\ell = 1$ (practical regime). We approximate μ_2 as d/n because each flow has d/n fraction of 2-hop paths (Model 7.2). With an assumption that paths do not overlap, we can show that almost every such path carries a single unit of flow.

Estimating μ_3 is more challenging because we cannot assume that paths do not overlap. We first estimate ϕ_3, the fraction of a source s's capacity along 3-hop paths after removing the d/n edges used by 2-hop paths, using the facts that 3-hop paths are of the form $s \rightarrow WP_1(t) \rightarrow Nbr(t) \rightarrow t$ and there are pd waypoints. We then estimate κ_3, the amount of source-destination flow that such a path can carry, using the distribution of path overlaps.

$$\begin{aligned}\phi_3 &= \min(pd/n, 1-d/n)\cdot(1-d/n)\cdot(1-(4d/n)^h)\\ \kappa_3 &= (1-\phi_3)^6/2 + (1-\phi_3^2)^3/6 + 1/3\end{aligned}$$

After similar polynomial calculations for μ_4, μ_5, we get:

Model 7.3. *The oversubscription ratio is approximated by* $(\mu_2+\mu_3+\mu_4+\mu_5)^{-1}$.

$$\begin{aligned}\mu_2 &= d/n\\ \mu_3 &= \phi_3\kappa_3 \ \textit{(see above)}\\ \mu_4 &= [1-(p+1)d/n - \exp(-pd^2/n)]\\ &\quad \cdot\Big(1-[1-(1-2d/n)(1-(4d/n)^h)]^h\Big)\\ &\quad \cdot(1-\mu_2-2\mu_3)/4\\ \mu_5 &= \exp(-pd^2/n)(1-\mu_2-2\mu_3-3\mu_4)/5\end{aligned}$$

This model characterizes the mesh-layer (between routers) oversubscription. For an end-to-end (server-to-server) analysis, oversubscription at the ToRs must also be factored (§7.4).

Takeaways. Fabric oversubscription is a complex interplay of all parameters. By capturing it for a wide range of parameter values,[7] our model enables fabric design for specific performance targets, which we discuss next.

7.4 Designing fabrics

While the exact procedure for determining the topology and routing parameters (n, d, p, and h) depends on desired tradeoffs (e.g., between oversubscription and path length), we illustrate for a common target: minimize cost (fewest ToRs) of connecting s servers with an end-to-end oversubscription ratio below r_e and ToR-layer oversubscription below r_t ($1 \le r_t \le r_e$). Since ToRs are a localized congestion risk, operators usually want to control their oversubscription [40].

To meet the target above, we binary search for the minimal viable value of d such that $d \ge \lceil \frac{P}{r_t+1} \rceil$, $d \ge 2ln(\lceil \frac{s}{P-d} \rceil) + 5)$, and $d < P$, where P is the number of router ports. The first constraint ensures that ToR layer oversubscription, which is $\frac{P-d}{d}$:1, is below r_t; the second ensures that the design is within the modeled operating regime; and the third ensures that at least one port is available for servers. Minimizing d subject to these constraints minimizes ToR count because it maximizes P-d, the ports that connect to servers.

The viability of a value of d is determined as follows. Based on d, we compute the number of required ToRs n as $\lceil \frac{s}{P-d} \rceil$. We then determine the maximum value of h that ToRs can support based on n, d, and the ECMP memory size (§5). For a given d, n, and h, the valid range of p is $[n/d^2, d]$, and different values yield different oversubscription ratios. We compute the range of oversubscription ratios using Model 7.3 for extreme values of p, and deem d as viable if r_e/r_t, which is the desired mesh-layer oversubscription, falls in this range.

[7] The formulas can be simplified for specific regimes, e.g., when a parameter is fixed. For h=2, $\log_d(n/p) + 2$ approximates Model 7.3 well.

Figure 8: (a) One of the racks hosting the emulated ShuffleBoxes. (b) Rows of emulated ShuffleBoxes. (c) A ShuffleBack dongle with 4-FPs MPO connector.

Once we have the minimal viable d, and corresponding n, h, we use the maximum $p \in [n/d^2, d]$ with oversubscription less than r_e/r_t. Maximizing p minimizes oversubscription.

8 PRODUCTION FABRICS

We have deployed the RNG design in two fabrics that carry production workloads. The first, called the "server mesh", connects ToRs as an expander. The second, called the "edge mesh," connects to the server mesh and to remote datacenters, and it provides transit between these networks. Ports that interconnect the two RNG fabrics are spread across their routers. We build separate fabrics because they have different oversubscription targets. The edge mesh is non-blocking and server mesh has the same oversubscription ratio as Amazon's latest fat tree fabrics. We elide the oversubscription level and mesh sizes for confidentiality.

At the time of deploying these fabrics, we did not have manufactured ShuffleBoxes. We emulated shuffling with a traditional patch panel that bridges individual FPs and custom ShuffleBacks. See Figure 8.

Spraypoint performance. We implemented Spraypoint by extending Amazon's shortest-paths based link-state protocol. We reused the topology dissemination component and modified next hop computation. For similarly-sized topologies, Spraypoint performs similarly to the current protocol along key metrics such as convergence time after a failure.

Cabling experience. Cabling RNG using the scheme in §6 was relatively smooth. One challenge was ensuring enough ShuffleBoxes are installed in each room. The exact number of racks, and thus router uplinks, that can land in a room varies, depending on their power consumption. We used an estimated upper bound on the number of router uplinks per room.[8]

Due to a lack of structure and physically identifiable patterns, a concern with random graphs has been operators connecting ports not intended to be connected [45, 49]. The incidence of such miscabling in RNG was below 1.5%.

Operational challenges. We give a short preview of these challenges, even though it is not a focus of this paper. Operationalizing random graphs required upgrading many software tools that manage Amazon's networks. The hierarchy of fat trees was embedded deeply into these tools, starting from device names itself to managing redundancy during maintenance. For instance, ToRs in fat trees do not rely on each other and can be upgraded based on concerns local to the rack, but in RNG we need to account for inter-ToR dependencies and cannot simultaneously upgrade too many neighbors of any given ToR. We ensured that fault localization functions properly and built new tools to easily determine the paths between ToRs for troubleshooting.

Application performance. A majority of traffic in hyperscale datacenters is moving toward multipath transport protocols [42, 44, 57]. The sender splits data across tens of "flowlets" that use different paths in the network (based on flowlet hash). The receiver assembles and orders the data, transparent to the applications. The sender uses latency as one of the signals when picking which flowlet to use.

Our main benchmarking goal is to validate that latency differential across sender-receiver paths (§7.2) does not impact performance. We study this by observing the raw throughput of the transport protocol and the performance of a latency- and throughput-sensitive block-storage application that uses the protocol. We compare RNG against a production fat tree fabric with identical oversubscription and identical server specs. We normalized results by the mean values observed for the fat tree to preserve confidentiality.

Figure 9 shows the throughput distribution based on an experiment with 127 concurrent flows with infinite demand between pairs of random servers. The results are identical for RNG and fat tree. Any differences in path latencies of the two networks are immaterial for application performance.

Figure 10 (left) compares the packets per second (PPS) for small, 64-byte packets between pairs of servers. RNG is slightly higher because it has less background traffic. Figure 10 (right) shows I/O operations per second (IOPS) for

[8] If this bound were breached, we plan to treat the remaining racks as belonging to a new logical room and perform the room-addition activity. Analogous provisioning challenges arise also for fat trees in large datacenters.

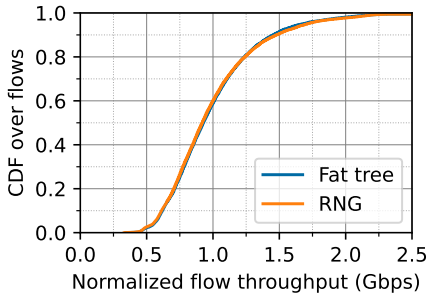

Figure 9: Per-flow throughput of multipath transport (normalized by the fat tree mean).

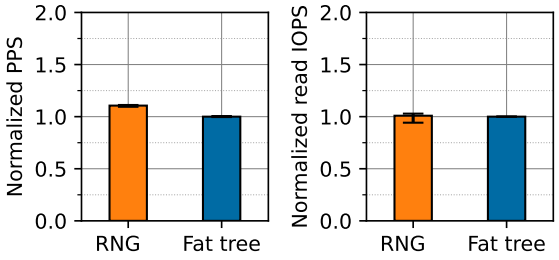

Figure 10: PPS of 64B packets (left) and IOPS of storage reads (right). The error bars are standard deviations.

storage reads from clients to multiple servers. RNG's performance matches fat trees for this workload as well.

These benchmarks, and the absence of user-reported performance issues, confirm that RNG matches fat tree performance (at lower cost and higher fault tolerance).

9 BROADER EVALUATION

Production fabric benchmarks can validate the performance of real applications but we need simulations to evaluate a broad range of design parameters and workloads. We benchmark RNG's oversubscription ratio and the number of edge disjoint paths that Spraypoint produces. We also compare its throughput and cost relative to fat trees.

9.1 Oversubscription

Oversubscription ratio characterizes worst-case throughput of a network and helps quantify congestion risk. Namyar et al. [37] prove that worst-case throughput occurs for a perfect unidirectional matching, where each node sends at full rate to its counterpart. The authors do not suggest a way to generate the worst matching among the many choices. We randomly generate 100 matchings and estimate oversubscription using the worst value, though we will see that the distribution is narrow because the topology and routing are random [13, 34]. Similar to our oversubscription model, this process characterizes stochastic oversubscription.

For a given matching, the oversubscription ratio is r if each transmitter can send at least $1/r$ fraction of its full rate without congesting any link. This measure is determined by the smallest blocked flow, irrespective of whether others can send more. We compute r by solving a linear program (LP) that encodes a multi-commodity flow problem [56]. Traffic for each flow is spread across Spraypoint paths, and the LP minimizes r under link capacity constraints. These LPs are huge and each takes multiple hours to solve, which limits the largest fabrics we can practically study.

Figure 11 shows the results of simulations ("RNG") and our model (§7.3). The line denotes the minimum value of r for each setting. There is a shadow for the max but it is barely visible because the variance is near zero. To study a broad range of settings, we vary one parameter while the others are set to default values (n=1000, d=64, p=4, h=2). This default can support 64K servers with 100 Gbps uplinks and offers an oversubscription ratio of 3.25. We see that the oversubscription ratio changes gracefully as we vary the parameters, highlighting the ability of random graphs to support fine-grained oversubscription levels. We also see that the impact of h and p flattens out beyond a point.

The model matches empirical results well, enabling performance predictability. It allows operators to plan RNG fabrics like they plan fat trees today, with the added advantage of lower cost, higher fault tolerance, and finer-grained performance tuning. Some settings in Figure 11 are technically outside the modeled regime, e.g., $d = 20 \not\geq 2(\ln(n = 1K) + 5)$. We have defined the regime conservatively to get high probability results, and the models degrade gracefully outside the regime.

9.2 Edge disjoint paths

Spraypoint aims to find many edge disjoint paths. We evaluate this ability by computing the min cut across paths between endpoint pairs, which equals the number of edge disjoint paths [55]. Figure 12 shows the CDF of min cut across endpoint pairs for the same default fabric parameters as above (n=1000, d=64, p=4, h=2). For comparison, it also shows k-shortest-path routing (which is not implementable at scale with commodity switches). We consider k=8 [45, 49] and k=64, a high value that equals the node degree (and needs 8x more resources).

For almost all endpoint pairs, Spraypoint finds over 50 edge disjoint paths. For half the pairs, it finds over 60 such paths out of the maximum possible 64. In contrast, the median for 8-shortest-paths is 5 and for 64-shortest paths is 35. This difference impacts oversubscription directly. The oversubscription for the two k-shortest-paths configurations

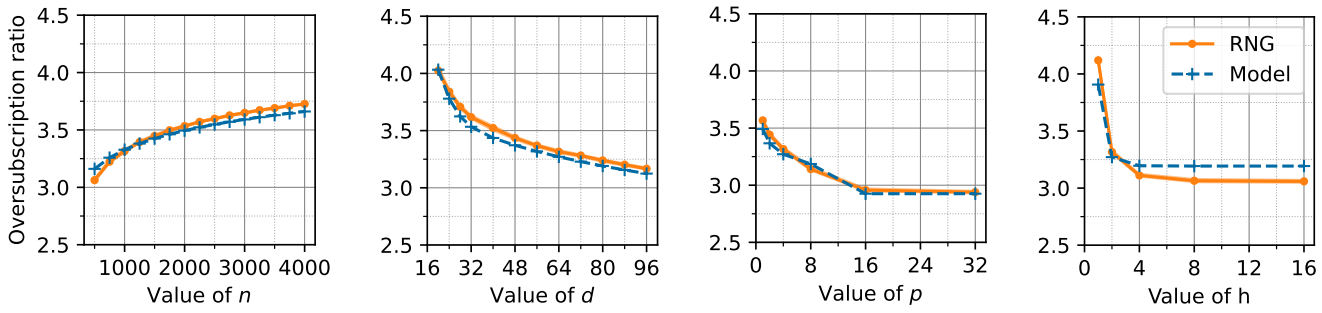

Figure 11: Oversubscription for different topology parameters. The defaults are n=1000, d=64, p=4, h=2.

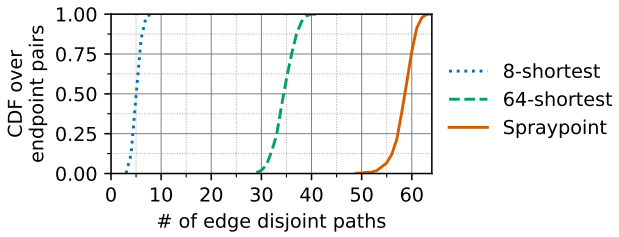

Figure 12: Edge disjoint path count.

are 21.3 and 4.7, while that for Spraypoint is 3.25. Spraypoint is easier to implement *and* finds a greater diversity of paths. It offers higher peak throughput between endpoint pairs, and offers higher network-wide throughput.

9.3 Throughput relative to fat trees

Beyond worst-case traffic patterns that help characterize oversubscription, operators want to also validate performance for other traffic patterns. This validation is hard because there are many possible traffic patterns. Our evaluation is based on the observation that any traffic pattern may be viewed as a combination of three basic types, and we can get insight into the performance of RNG for a range of traffic patterns by studying these types. The three types are:

- *Clique:* A group of nodes exchanges traffic amongst themselves (e.g., collective communication like all-reduce, replication traffic among storage servers).
- *Hubs:* Some nodes (e.g., Web or storage servers) send to and receive from all other nodes.
- *Matchings:* Nodes send all traffic to exactly one other node, as we studied earlier.

The nodes in the traffic patterns are ToRs, not individual servers, because that stresses the fabric more [37].

Each pattern type is parameterized by active fraction $f \in [0, 1]$ to capture the skew. Clique(f=0.2) means that the clique size is 20% of the nodes (randomly selected); hubs(f=0.2) means that 20% of the nodes in the network are hubs; and matchings(f=0.2) means that 20% of the nodes are involved in the matching. For each pattern type and f, all flows in the traffic matrix (between a sender-receiver pair) are equal and sum of rows and columns equal the local capacity of nodes. Thus, each matrix is routable in a non-blocking fabric.

For each value of f, we generated 100 random traffic matrices. We quantify performance for a matrix using the fraction r as before—the network can carry at least $1/r$ fraction of each flow. Thus, we are computing the oversubscription ratio of each matrix, instead of worst-case oversubscription ratio.

Figure 13 shows the results for RNG and fat tree. The lines denote the mean and the shaded area denotes the min and max. Both topologies are designed to support 61.4K servers across 960 ToRs, with a worst-case oversubscription of 3:1. In this configuration, RNG uses 45% fewer switches.

We see that RNG performs better by as much as 30% across a wide regime for clique and hubs. The exceptions are cases with f<0.1, where fat tree is 5-10% better. For these cases, shortest path routing on fat trees finds edge disjoint paths that equal the ToR degree. The equivalent number is 10% lower in RNG for h=2 (see §9.2). The number of edge disjoint paths matters most when there are few senders in the network. For matchings, fat tree is better when f<0.4 and RNG is better otherwise.

Amazon operators prefer the broadly better performance of RNG, especially given its lower cost and higher fault tolerance. The regimes with low values of f, where fat tree performs better, are acceptable because they will materialize only when all servers in the rack act in unison, a rarity for multi-tenant datacenters.

9.4 Cost relative to fat trees

Prior works report a range of values on the cost of expander topologies relative to fat trees. These values are difficult to compare because they use different performance measures and network sizes. It is also difficult to get a holistic view

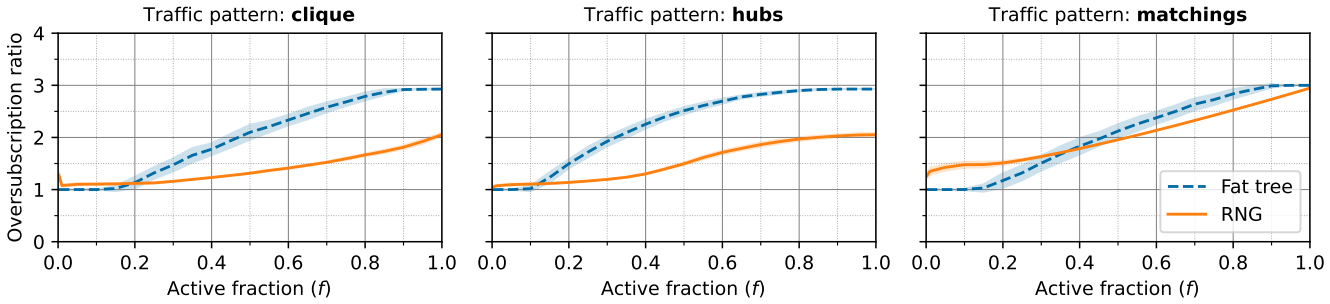

Figure 13: Oversubscription ratio (lower is better) for different traffic matrices. Both fat tree and RNG topologies are configured for worst-case oversubscription ratio of 3:1. RNG topologies use 45% fewer switches.

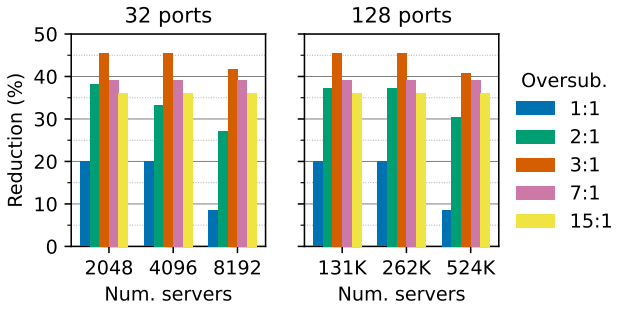

Figure 14: Reduction in number of switches in RNG relative to a 3-tier fat tree for two different port counts.

because of their experimental methodology. Our models enable systematic analysis of the relative cost as a function of oversubscription ratios and network sizes.

We use the ratio of the number of switches used by the two topologies as the measure of relative cost. This measure automatically includes transceivers, whose count is proportional to switch count. It ignores passive optical components like cables, connectors, and patch panels; the cost of these components is in the minority and varies based on specific cabling methods (e.g., if cables are trunked) in the datacenter. Switch count is a general, reproducible measure.

We compute the number of switches needed for RNG using the procedure in §7.4 and compare it to generalized 3-tier fat tree with the same oversubscription level. ToRs are not oversubscribed in either RNG or fat trees. Figure 14 shows the results for different oversubscription levels and two possible values of switch port counts. The highest server count for each port count is the maximum supported by a non-blocking fat tree [36], and the other two are 50% and 25% of the servers. Oversubscription of 2:1 cannot be precisely supported in fat trees with 32 or 128 port switches, but we assume each port is 3-way splittable for this experiment.

The cost reduction varies by 5x, between 9 and 45%. It has a similar pattern across port counts and network sizes, and oversubscription ratio is the key determiner. The reduction is low for the 1:1 oversub case because non-blocking fat trees do not strand capacity; the reduction here primarily comes from shorter paths in RNG. After increasing from oversubscription of 1:1 to 3:1, the cost advantage starts decreasing slowly. For high oversubscription, a fat tree has fewer switches at the aggregation and spine tiers, which lowers the potential of reducing cost by removing those tiers.

Our analysis makes it clear when expander topologies like RNG bring the most or least value, allowing operators to make informed choices about the type of topology to use.

10 RELATED WORK

We build on the foundations laid by a long line of prior research. The notion of an expander, or an "optimal" graph for routing purposes, was defined in the early 90s [24], and it has been long known that random graphs are nearly-optimal expanders [12]). Several researchers have proposed expander topologies for datacenters and provided insights into their performance [4, 25, 26, 37, 45, 49]. Particularly influential works for us include Jellyfish [45], which proposed (truly) random graphs for datacenters; Xpander [49], which made the connection that random graphs do well for data center workloads because they are expanders; and Namyar et al. [37], who outlined throughput bounds of expander topologies. We extend this body of work by addressing unsolved challenges related to routing, cabling, and performance predictability. We also deploy the first expander-based networks in production.

Researchers have also proposed non-expander topologies that tackle some of the challenges of the fat trees (e.g., cost), such as HyperX [1], DCell [20], and BCube [19]. We choose random graphs because they have significantly shorter path lengths. Further, cabling these other topologies for large, multi-room datacenters is still an open challenge.

Capacity fungibility, a key goal of the RNG design, can be achieved using reconfigurable hardware as well. We use expander topologies because they need neither a (logically) centralized control plane nor non-standard hardware. Most designs with reconfigurable hardware require a control plane that predicts global traffic demands and dynamically reconfigures the topology [11, 17, 21, 22, 51]. The scale of our datacenters, presence of large amounts of bursty traffic (common for Web workloads), and reconfiguration delays complicate the development and operation of such control planes.

There are "demand-oblivious" designs that do not require a centralized control plane [3, 31, 32]. They use novel optical devices that rapidly cycle through a fixed schedule. All-to-all connectivity is not available at any given time, and the systems use forwarding via intermediate nodes based on Birkhoff-von Neumann traffic matrix decomposition [7]. In addition to relying on non-commodity hardware, these designs bring a host of software risks such as time synchronization, packet reordering within a TCP flow, some throughput degradation [32], and switch-level congestion control [3].

11 CONCLUSIONS

Flat expander topologies based on quasi-random graphs can be practically realized using the routing and cabling approaches we developed. They can be designed for the desired level of performance and cost using the models we developed.

REFERENCES

[1] 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 Conference on High Performance Computing Networking, Storage and Analysis*. 1–11.

[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *Proc. ACM SIGCOMM Conference on Data Communication*. 63–74. doi:10.1145/1402958.1402967

[3] Hitesh Ballani, Paolo Costa, Raphael Behrendt, Daniel Cletheroe, Istvan Haller, Krzysztof Jozwik, Fotini Karinou, Sophie Lange, Kai Shi, Benn Thomsen, and Hugh Williams. 2020. Sirius: A Flat Datacenter Network with Nanosecond Optical Switching. In *Proc. ACM SIGCOMM Conference on Data Communication*. 782–797. doi:10.1145/3387514.3406221

[4] Maciej Besta and Torsten Hoefler. 2014. Slim fly: a cost effective low-diameter network topology. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 348–359. doi:10.1109/SC.2014.34

[5] Broadcom. 2026. Tomahawk 5 / BCM78900 series. https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm78900-series [Retrieved 6-Feb-2026].

[6] Broadcom. 2026. Tomahawk3 / BCM56980 Series. https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56980-series [Retrieved 6-Feb-2026].

[7] Cheng-Shang Chang, Duan-Shin Lee, and Yi-Shean Jou. 2002. Load balanced Birkhoff–von Neumann switches, part I: One-stage buffering. *Computer Communications* 25, 6 (2002), 611–622.

[8] F. R. K. Chung, R. L. Graham, and R. M. Wilson. 1989. Quasi-Random Graphs. *Combinatorica* 9, 4 (1989), 345–362.

[9] Devdutt Dubhashi and Alessandro Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge.

[10] Facebook. 2014. Introducing data center fabric, the next-generation Facebook data center network. https://engineering.fb.com/2014/11/14/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/ [Retrieved 6-Feb-2026].

[11] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2010. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *Proc. ACM SIGCOMM Conference on Data Communication*. 339–350.

[12] Joel Friedman. 2008. A proof of Alon's second eigenvalue conjecture and related problems. *J. of the American Mathematical Society* 195, 910 (2008).

[13] Alan M. Frieze and Páll Melsted. 2012. Maximum matchings in random bipartite graphs and the space utilization of Cuckoo Hash tables. *Random Struct. Algorithms* 41, 3 (2012), 334–364.

[14] fs.com. 2019. Fiber Distribution Panel Wiki, Types and Buying Tips. https://community.fs.com/article/fiber-distribution-panel-wiki-buying-tips.html [Retrieved: 2026-02-06].

[15] fs.com. 2026. 8 Fibers MTP Female Type 1 OM4 50/125 Multimode Fiber Loopback Module. https://www.fs.com/products/35796.html?now_cid=2686 [Retrieved: 6-Feb-2026].

[16] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, Shuqiang Zhang, Mikel Jimenez Fernandez, Shashidhar Gandham, and Hongyi Zeng. 2024. RDMA over Ethernet for Distributed Training at Meta Scale. In *Proc. ACM SIGCOMM Conference on Data Communication*. 57–70. doi:10.1145/3651890.3672233

[17] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel Kilper. 2016. Projector: Agile reconfigurable data center interconnect. In *Proc. ACM SIGCOMM Conference on Data Communication*. 216–229.

[18] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: a scalable and flexible data center network. In *Proc. ACM SIGCOMM Conference on Data Communication*. 51–62. doi:10.1145/1592568.1592576

[19] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. 2009. BCube: a high performance, server-centric network architecture for modular data centers. In *Proc. ACM SIGCOMM Conference on Data Communication*. 63–74.

[20] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. 2008. Dcell: a scalable and fault-tolerant network structure for data centers. In *Proc. ACM SIGCOMM Conference on Data Communication*. 75–86.

[21] Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall. 2011. Augmenting data center networks with multi-gigabit wireless links. In *Proc. ACM SIGCOMM Conference on Data Communication*. 38–49.

[22] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R Das, Jon P Longtin, Himanshu Shah, and Ashish Tanwer. 2014. Firefly: A reconfigurable wireless data center fabric using free-space optics. In *Proc. ACM SIGCOMM Conference on Data Communication*. 319–330.

[23] Vipul Harsh, Sangeetha Abdu Jyothi, and P. Brighten Godfrey. 2020. Spineless Data Centers. In *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*. 67–73. doi:10.1145/3422604.3425945

[24] Shlomo Hoory, Nathan Linial, and Avi Wigderson. 2007. Expander graphs and their applications. *Bulletin of the AMS* 43 (2007), 439–561.

[25] 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 (SC)*. 761–772.

[26] Simon Kassing, Asaf Valadarsky, Gal Shahaf, Michael Schapira, and Ankit Singla. 2017. Beyond fat-trees without antennae, mirrors, and disco-balls. In *Proc. ACM SIGCOMM Conference on Data Communication*. 281–294. doi:10.1145/3098822.3098836

[27] Allan Kaye. 2025. Rail-Optimised Networking: How NVIDIA is Rethinking AI Network Design in the Data Centre. https://vespertec.com/news/rail-optimised-networking-how-nvidia-is-rethinking-ai-network-design-data-centre/ [Retrieved 3-Feb-2026].

[28] Murali Kodialam, TV Lakshman, and Sudipta Sengupta. 2011. Traffic-oblivious routing in the hose model. *IEEE/ACM Transactions on Networking* 19, 3 (2011), 774–787.

[29] Alexander Krentsel, Nitika Saran, Bikash Koley, Subhasree Mandal, Ashok Narayanan, Sylvia Ratnasamy, Ali Al-Shabibi, Anees Shaikh, Rob Shakir, Ankit Singla, and Hakim Weatherspoon. 2024. A Decentralized SDN Architecture for the WAN. In *Proc. ACM SIGCOMM Conference on Data Communication*. 938–953. doi:10.1145/3651890.3672257

[30] Hong Liu, Ryohei Urata, Kevin Yasumura, Xiang Zhou, Roy Bannon, Jill Berger, Pedram Dashti, Norm Jouppi, Cedric Lam, Sheng Li, Erji Mao, Daniel Nelson, George Papen, Mukarram Tariq, and Amin Vahdat. 2023. Lightwave Fabrics: At-Scale Optical Circuit Switching for Datacenter and Machine Learning Systems. In *Proc. ACM SIGCOMM Conference on Data Communication*. 499–515. doi:10.1145/3603269.3604836

[31] William M Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C Snoeren, and George Porter. 2020. Expanding across time to deliver bandwidth efficiency and low latency. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 1–18.

[32] William M. Mellette, Alex Forencich, Rukshani Athapathu, Alex C. Snoeren, George Papen, and George Porter. 2024. Realizing RotorNet: Toward Practical Microsecond Scale Optical Networking. In *Proc. ACM SIGCOMM Conference on Data Communication*. 392–414. doi:10.1145/3651890.3672273

[33] Jeffrey C. Mogul and John Wilkes. 2023. Physical Deployability Matters. In *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*. 9–17. doi:10.1145/3626111.3628190

[34] Rajeev Motwani and Prabhakar Raghavan. 2001. *Randomized Algorithms*. Cambridge University Press.

[35] Jayaram Mudigonda, Praveen Yalagandula, Mohammad Al-Fares, and Jeffrey C. Mogul. 2010. SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 18.

[36] Jayaram Mudigonda, Praveen Yalagandula, and Jeffrey C. Mogul. 2011. Taming the Flying Cable Monster: A Topology Design and Optimization Framework for Data-Center Networks. In *USENIX Annual Technical Conference*. https://api.semanticscholar.org/CorpusID:2989246

[37] Pooria Namyar, Sucha Supittayapornpong, Mingyang Zhang, Minlan Yu, and Ramesh Govindan. 2021. A throughput-centric view of the performance of datacenter topologies. In *Proc. ACM SIGCOMM Conference on Data Communication*. 349–369. doi:10.1145/3452296.3472913

[38] Sabine R Ohring, Maximilian Ibel, Sajal K Das, and Mohan J Kumar. 1995. On generalized fat trees. In *Proc. International Parallel Processing Symposium*. 37–44.

[39] Doron Puder. 2014. Expansion of random graphs: new proofs, new results. *Inventiones mathematicae* 201, 3 (Dec. 2014), 845–908. doi:10.1007/s00222-014-0560-x

[40] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, Chao Wang, Peng Wang, Pengcheng Zhang, Xianlong Zeng, Eddie Ruan, Zhiping Yao, Ennan Zhai, and Dennis Cai. 2024. Alibaba HPN: A Data Center Network for Large Language Model Training. In *Proc. ACM SIGCOMM Conference on Data Communication*. 691–706. doi:10.1145/3651890.3672265

[41] Brandon Schlinker, Radhika Niranjan Mysore, Sean Smith, Jeffrey C. Mogul, Amin Vahdat, Minlan Yu, Ethan Katz-Bassett, and Michael Rubin. 2015. Condor: Better Topologies Through Declarative Design.. In *Proc. ACM SIGCOMM Conference on Data Communication*. 449–463. http://dblp.uni-trier.de/db/conf/sigcomm/sigcomm2015.html#SchlinkerMSMVYK15

[42] Leah Shalev, Hani Ayoub, Nafea Bshara, and Erez Sabbag. 2020. A Cloud-Optimized Transport Protocol for Elastic and Scalable HPC. *IEEE Micro* 40, 6 (2020), 67–73. doi:10.1109/MM.2020.3016891

[43] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *Proc. ACM SIGCOMM Conference on Data Communication*. 183–197. doi:10.1145/2785956.2787508

[44] Arjun Singhvi, Nandita Dukkipati, Prashant Chandra, Hassan M. G. Wassel, Naveen Kr. Sharma, Anthony Rebello, Henry Schuh, Praveen Kumar, Behnam Montazeri, Neelesh Bansod, Sarin Thomas, Inho Cho, Hyojeong Lee Seibert, Baijun Wu, Rui Yang, Yuliang Li, Kai Huang, Qianwen Yin, Abhishek Agarwal, Srinivas Vaduvatha, Weihuang Wang, Masoud Moshref, Tao Ji, David Wetherall, and Amin Vahdat. 2025. Falcon: A Reliable, Low Latency Hardware Transport. In *Proc. ACM SIGCOMM Conference on Data Communication* (São Francisco Convent, Coimbra, Portugal). 248–263. doi:10.1145/3718958.3754353

[45] Ankit Singla, Chi-Yao Hong, Lucian Popa, and Philip Brighten Godfrey. 2012. Jellyfish: Networking Data Centers Randomly. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 225–238. https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/singla

[46] snaketray.com. 2026. The Essential Guide to Data Center Cabling. https://www.snaketray.com/data-center-cabling-guide/ [Retrieved 6-Feb-2026].

[47] F. Solano, R. Fabregat, Y. Donoso, and J.L. Marzo. 2005. Asymmetric tunnels in P2MP LSPs as a label space reduction method. In *IEEE International Conference on Communications (ICC)*, Vol. 1. 43–47 Vol. 1. doi:10.1109/ICC.2005.1494318

[48] F. Solano, R. Fabregat, and J.L. Marzo. 2005. Full label space reduction in MPLS networks: asymmetric merged tunneling. *IEEE Communications Letters* 9, 11 (2005), 1021–1023. doi:10.1109/LCOMM.2005.11016

[49] Asaf Valadarsky, Gal Shahaf, Michael Dinitz, and Michael Schapira. 2016. Xpander: Towards Optimal-Performance Datacenters. In *Proc. International on Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (Irvine, California, USA). Association for Computing Machinery, New York, NY, USA, 205–219. doi:10.1145/2999572.2999580

[50] Leslie G Valiant and Gordon J Brebner. 1981. Universal schemes for parallel communication. In *Proc. ACM Symposium on Theory of Computing (STOC)*. 263–277.

[51] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. 2010. c-Through: part-time optics in data centers. In *Proc. ACM SIGCOMM Conference on Data Communication*. 327–338. doi:10.1145/

1851182.1851222
[52] Wikipedia. 2026. Balls into bins problem. https://en.wikipedia.org/wiki/Balls_into_bins_problem [Retrieved 6-Feb-2026].
[53] Wikipedia. 2026. Expander graph. https://en.wikipedia.org/wiki/Expander_graph [Retrieved 6-Feb-2026].
[54] Wikipedia. 2026. k-shortest path routing. https://en.wikipedia.org/wiki/K_shortest_path_routing [Retrieved: 6-Feb-2026].
[55] Wikipedia. 2026. Menger's Theorem. https://en.wikipedia.org/wiki/Menger%27s_theorem [Retrieved: 6-Feb-2026].
[56] Wikipedia. 2026. Multi-commodity flow problem. https://en.wikipedia.org/wiki/Multi-commodity_flow_problem [Retrieved: 6-Feb-2026].
[57] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. https://www.usenix.org/conference/nsdi11/design-implementation-and-evaluation-congestion-control-multipath-tcp
[58] Nicholas Wormald. 1999. *Models of Random Regular Graphs.* Cambridge. https://web.williams.edu/Mathematics/sjmiller/public_html/ntprob19/handouts/graphs/Womald_ModelsRandomGraphs.pdf
[59] Eitan Zahavi. 2012. Fat-tree routing and node ordering providing contention free traffic for MPI global collectives. *J. Parallel and Distrib. Comput.* 72, 11 (2012), 1423–1432. doi:10.1016/j.jpdc.2012.01.018 Communication Architectures for Scalable Systems.
[60] Xia Zhou, Zengbin Zhang, Yibo Zhu, Yubo Li, Saipriya Kumar, Amin Vahdat, Ben Y Zhao, and Haitao Zheng. 2012. Mirror mirror on the ceiling: Flexible wireless links for data centers. *ACM SIGCOMM Computer Communication Review* 42, 4 (2012), 443–454.

A INCREMENTAL CABLING

The physical connectivity of RNG described in §6 can be achieved incrementally. While the first room of the datacenter is being prepared, we deploy its patching panel. The number of ShuffleBoxes in the panel is based on the number of ToR uplinks expected in the room. Initially, all r-ports and c-ports have ShuffleBacks in them. When racks land in the room, ToRs are connected to random r-ports, as part of which the ShuffleBacks of selected r-ports are removed. The ToRs start forming a random graph via c-port ShuffleBacks, as in Figure 6(a). Not all ToR uplinks may find a match during this process (more on this below).

While the second room is being prepared, we land its shuffle panel and connect those c-ports to the panel in the first room. If panels are equal-sized, half of the c-ports in each must connect to the other panel; otherwise, we adjust proportionally. The ports are picked randomly. The c-port connections are enabled by the trunk cables between the two rooms. Making these connections requires removing ShuffleBacks from c-ports, as a result of which the connectivity pattern for some ToRs may go from Figure 6(a) to Figure 6(c), except that both ToRs are in the same room.

When racks can land in the second room, those ToRs connect to random r-ports in the second room's panel. Some of these ToR uplinks will connect to ToRs in the first room via c-port connections, as in Figure 6(b); some will connect to other ToRs in the same room via c-port ShuffleBacks; and, at least initially, some will fail to find a match.

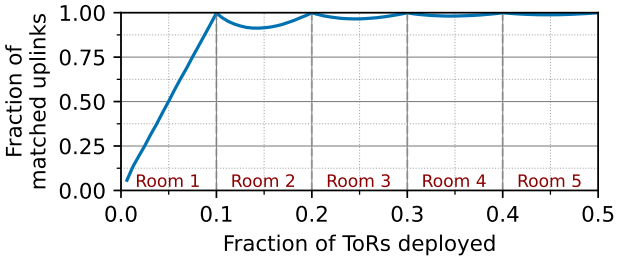

Figure 15: Matched uplinks as the datacenter grows.

When the third room starts, we land its panel and rebalance c-port connectivity. When there were two rooms, half of the first room's c-ports connected to the second room. When the third room starts, a third must go to the second room and a third to the third room (and a third have ShuffleBacks). To rebalance c-port connectivity, we break some existing c-port connections (selected randomly) between the first two rooms and use the freed up c-ports to connect to the third room. We follow the same process of landing panels and rebalancing connectivity for all subsequent rooms.

Rebalancing connectivity breaks logical links that carry live traffic. We thus drain impacted links to minimize impact. Strictly speaking, logical connectivity changes when racks land and r-port ShuffleBacks are removed, but the blast radius of that operation is much smaller (f_r=4 versus f_c=32).

A.1 Growing pains

An artifact of our incremental construction is that early in the growth of the datacenter, not all ToR uplinks form valid adjacencies. For an uplink in room 1 to form an adjacency, it must connect to an r-port FP that is bridged (by the ShuffleBox's c-port ShuffleBack) to an r-port connected to another uplink. When only a small fraction of r-ports are populated, this probability is low. A similar effect occurs for the second room but is less severe because half of ToR uplinks connect to r-port FPs that are connected to the first room via c-ports and find matches there.

Figure 15 plots the fraction of deployed uplinks with valid adjacencies as we deploy more ToRs. This simulated datacenter has 10 rooms with 100 ToRs each. We see the fraction of uplinks in room 1 that find a match increases linearly. After room 2 opens, there is an initial dip in the fraction of matched uplinks because room 2 ToRs have fewer matched uplinks and their ratio in the population is growing. But the lowest point in the dip is still above 90%. The impact is negligible beyond room 2.

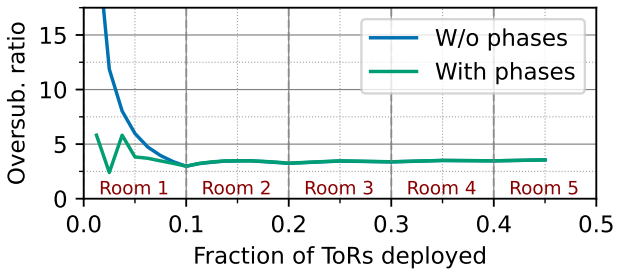

Figure 16: Oversubscription as the datacenter grows.

We present a model of datacenter expansion that accurately predicts the average degree at each point in the datacenter growth. Operators can use the model to predict the average degree during expansion and plan accordingly.

As the "W/o phases" curve of Figure 16 shows, these growing pains lead to poor early throughput for room 1 (and room 1 only). Amazon operators view this behavior as acceptable for fast-growing datacenters. The window of poor performance is short and they can delay on-boarding applications.

But the poor early performance can be problematic for slow-growing datacenters. For these cases, we propose a modified cabling scheme. We partition the ShuffleBoxes of the first room's shuffle panel into two or more phases. Initial ToRs connect only to phase 1 ShuffleBoxes. When these ShuffleBoxes are full, we connect the c-ports of phase 1 and phase 2 ShuffleBoxes and start mapping future ToRs to phase 2 ShuffleBoxes. This process is repeated for future phases. It essentially divides a room into smaller, logical rooms and shrinks the poor performance window.

The "With phases" curve of Figure 16 shows the impact of phased cabling. We partitioned room 1 into two phases, and the size of the first phase was 30% of the room. This configuration is optimal when operators want average degree of the datacenter to always stay above 0.8 as soon as 25% of the racks in room 1 are deployed. Next, we show how to derive optimal phase sizes for user-defined performance criteria.

A.2 Modeling expansion

We present a model of datacenter expansion that predicts the average degree of the fabric at each point in time. The expansion occurs in *stages*, where each stage corresponds to the landing of new panels of ShuffleBoxes and associated re-cabling. Routers of a stage only connect to ShuffleBoxes of that stage. A stage could be a room or a phase within a room, and the analysis makes no assumption about the number or relative sizes of various stages.

We model datacenter expansion using "time" t as the primary variable, where $t \in [0, 1]$ is the fraction (relative to the full DC) of routers currently landed. We denote a stage that begins at t_1 and ends at t_2 as $[t_1, t_2]$. Abusing notation, we will simply denote a router by its timestamp of arrival. Each router has d uplinks.

During the first stage, when there are no routers from previous stages, the probability of a ToR uplink finding a match increases linearly because the probability depends on bridging to an occupied r-port which grows linearly with t. The average degree of the fabric at time t is $\mathrm{d}\frac{t}{T_0}$, where T_0 is the size of the first stage.

For subsequent stages, the following invariants hold about RNG's datacenter expansion:

(1) At the end of any stage, the graph degree is d because all routers would have found a match.
(2) During a stage, all routers from previous stages maintain their degree of d.
(3) When a router on a stage lands, only some of its uplinks find a match. On expectation, that fraction is the fraction of routers present. Formally, for stage $[t_1, t_2]$, there are t_1 routers from previous stages, and we will land routers $t_1 + 1, t_1 + 2, \ldots, t_2$. The i-th router in this stage will make (on expectation) a $\frac{t_1+i}{t_2}$ fraction of connections when it lands.

Based on these invariants, we can prove:

Model A.1. *The average degree during the stage* $[t_1, t_2]$ $(t_1 > 0)$ *is* $d\left[\frac{t_1}{t} + \frac{t-t_1}{t_2}\right]$

Proof. The stage $[t_1, t_2]$ begins with a t_1 fraction of routers connected (with degree d) and ends with a t_2 fraction of routers. At time $t \in [t_1, t_2]$, all routers with timestamp at most t_1 always have a degree of d, and the routers of the current stage will have, on average, a degree of dt/t_2. The fraction of routers with timestamp at most t_1 is t_1/t, and the fraction of routers of the current stage is $(t - t_1)/t$.

Thus, the average degree at time t is:

$$\frac{t_1}{t} \cdot d + \frac{t - t_1}{t} \cdot \frac{dt}{t_2} = d\left[\frac{t_1}{t} + \frac{t - t_1}{t_2}\right] \quad (1)$$

□

Figure 17 compares Model A.1 with a simulated fabric from Figure 16. We see that the model matches simulations well, and the curves are virtually indistinguishable.

A.3 Computing optimal phases

The expansion model enables us to find optimal phases. We formulate the problem as: Find the minimum number of phases and their sizes such that the average degree is at least αd when at least a β-fraction of the first room is deployed. The parameters α and β are operator-provided inputs.

We solve the problem by applying Model A.1 to the first room, treating each phase as a stage. We know that the average degree increases linearly in the first stage. So, if

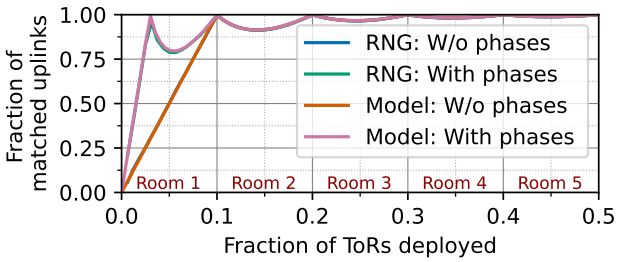

Figure 17: Comparison of fraction of uplinks that find a match based on simulation of Figure 15 versus what is predicted by the model.

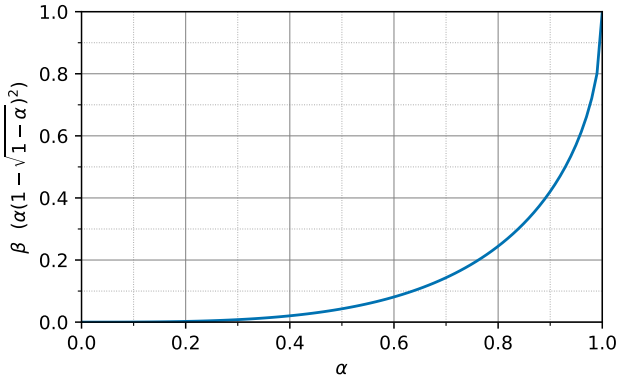

Figure 18: The minimum β achievable, for a given α in a two-phase deployment

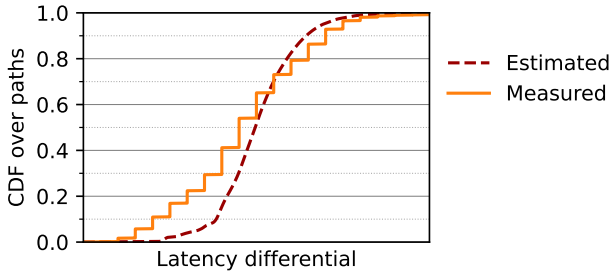

Figure 19: Comparison between estimated latency based on datacenter layout and measurements on the production fabric. Units omitted for confidentiality.

$\alpha \leq \beta$, we only require a single phase (i.e., not using phases at all). If $\alpha > \beta$, our aim is to maximize α for a given β, or equivalently, minimize β for a given choice of α.

Let us now consider the case of two phases in a room.

THEOREM A.1. *With two phases in a room, for a given α, the minimum possible β is $\alpha(1-\sqrt{1-\alpha})^2$. Further, the minimum occurs when the size of the first phase is β/α.*

PROOF. Let us first observe that the formula in Model A.1 is a hyperbola ($t_1/t + t/t_2$). Its minimum is at the geometric mean $\sqrt{t_1 t_2}$, so the minimum average degree is:

$$d(2\sqrt{t_1/t_2} - t_1/t_2) \tag{2}$$

Let us now denote the minimum possible β as β^*. In the first phase, the average degree increase linearly. At time β^*, it needs to be at least αd. This phase ends when the average degree is d, which happens when time is β^*/α, which we denote using x for convenience. At this point, the next phase begins and continues to the end of the room.

If the end of room deployment as $t = 1$, we have the stage being $[x, 1]$. Applying Equation 2, the minimum average degree is $d(2\sqrt{x} - x)$. Since we require the average degree to be αd at all times, $2\sqrt{x} - x \geq \alpha$.

Consider the quadratic $f(x) = x - 2\sqrt{x} + \alpha$. We need to choose x so that $f(x) \leq 0$, which happens after the first root of the quadratic. The roots of $f(x)$ are

$$\frac{2 \pm \sqrt{4-4\alpha}}{2} = 1 \pm \sqrt{1-\alpha}$$

The smaller root is $1-\sqrt{1-\alpha}$, and we need x to be at least this value. Plugging in the expression for x,

$$\sqrt{\frac{\beta^*}{\alpha}} \geq 1 - \sqrt{1-\alpha} \quad \Longrightarrow \quad \beta^* \geq \alpha(1-\sqrt{1-\alpha})^2$$

□

For a better understanding of the formula in Theorem A.1, Figure 18 plots the optimal β versus α. We see if we want the degree to be at least $0.8d$, we can get that as soon as $\beta \approx 25\%$ of the room is deployed. This performance can be obtained if the first phase is ≈30% of the first room.

If the β^* of the two-phase deployment does not meet the operator criterion, we can perform a similar analysis for successively higher number of phases until we find the number of phases where the criterion is met. These iterations are guaranteed to terminate. In the extreme, each router is its own phase and the graph degree is always d.

B LATENCY REDUCTION

While expander topologies have fewer hops than fat trees, we find that when deployed in a large datacenter their latency can be worse because hops can be long (half the datacenter length on average). The higher propagation delay exceeds what is saved in switching costs due to fewer hops.

To demonstrate this challenge, we simulate a datacenter with a span of 300 meters and 4K ToRs spread across 10 equally-sized rooms. For RNG, the patching panels are located in the center of each room. The fat tree has 4 tiers. ToRs connect to aggregation pods with two tiers of switches,

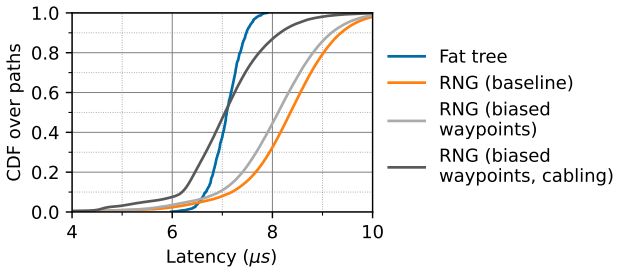

Figure 20: Latency distribution of ToR-to-ToR paths

where the bottom tier connects to ToRs and the top tier to spine routers. Aggregation pods are located in the center of each room to serve the ToRs in the room, and all spine routers are in one room in the middle of the datacenter [10]. For both topologies, the link speed is 100 Gbps and cables travel along cable trays [46]. Figure 19 validates the accuracy of our latency estimation methodology by comparing what it estimates for the production server mesh (§8) with measurements on the fabric.

While the fat tree has 6 hops and RNG has 4 hops for the vast majority of ToR pairs in the simulated datacenter, Figure 20 shows that the median one-way latency of ("RNG (baseline)") is 15% higher (8.4 vs. 7.1 μs). This difference may seem small, but it compounds over round trips for applications and so datacenter operators are sensitive to it.

Simple modifications to routing and cabling reduce the latency of RNG without compromising performance.

Biased waypoint selection. When selecting waypoints, instead of picking p random neighbors of $v \in Nbr(t)$, prefer waypoints closer to t. We implement this preference by ranking each candidate waypoint w using $room\_dist(w, v) + room\_dist(v, t)$, where $room\_dist()$ is the distance between rooms of the two nodes. Using coarse, room-level distances, instead of actual fiber runs, avoids systematically preferring advantageously-located waypoints (e.g., close to cable trays).

Biased cabling. We lower the number of inter-room connections, which are longer than intra-room ones. Specifically, a patching panel makes $\alpha \times \frac{1}{R} \times C \times d_c$ c-port connections to another panel, where C, R, d_c are the number of ShuffleBoxes, rooms, and c-ports per ShuffleBoxes, and $\alpha \in [0, 1]$ is a parameter that controls the amount of inter-room connections. For the baseline construction, $\alpha = 1$. We use $\alpha = 0.5$, which reduces number of inter-room connections by half. It reduces latency without hurting throughput because there is still enough inter-room capacity. Another positive side-effect of this optimization is that it reduces (by $1-\alpha$) the number of inter-room cables, an important factor behind cabling costs.

Figure 20 shows that the combination of these modifications makes the median latency of RNG similar to that of the fat tree. Biased waypoint selection reduces the median latency by 0.3 μs and biased cabling by a further 1 μs.

Latency can be further reduced by limiting spraying to a subset of neighbors that have shorter paths to the destination and by selecting the h next hops in a distance-aware manner. But, unlike the two modifications above, these modifications reduce, respectively, the number of edge disjoint paths and throughput for some traffic matrices. They must thus be used only in settings where reduced performance is acceptable.

We experimented with a few other ideas, such as spraying to only a subset of close neighbors and biasing the selection of h random next hops, but they either reduced throughput or did not bring substantial benefit.

C MODELING OVERVIEW

Our modeling of RNG prioritizes simpler formulas over mathematical precision. It makes numerous modeling assumptions and performs asymptotic analysis (as n gets large) that ignores lower order terms in some situations. We constrain the operating regime to where the inaccuracy of these assumptions and asymptotics is minimal. Recall from §7 that this regime is (i) $2(\ln(n) + 5) \leq d \ll n$, (ii) $p \geq (n/d^2)^{1/\ell}$, and (iii) $h \ll d$, where n and d are the node count and degree. Further, given random topology and routing, the analysis is inherently probabilistic, i.e., results hold with high probability and can fail with low probability.

The following sections derive the performance models in §7 in a stepwise manner.

Random graph preliminaries (§D): The first step of the modeling is to define a probabilistic process that generates the random network topology. We give a precise formulation, since this forms the foundation of the analysis. We prove some basic theorems about the graph construction. This section is purely using random graph theory, and does not involve any asymptotics or modeling of Spraypoint/routing.

Modeling path length (§E): We begin the actual analysis by giving formulas for the sizes of various Spraypoint levels. The random graph theorems from §D are used to determine the structure of the Spraypoint levels and their sizes. The path length distributions follow quite directly from these formulas. These formulas use asymptotics to get convenient expressions. This yields Model 7.2.

Modeling edge disjoint paths (§F): Armed with the tools from the previous sections, we can analyze the mincut properties between a source-destination pair. For this analysis, we make some minor modeling assumptions, to represent the flow as a tractable balls and bins problem. A detailed, tight analysis of these problems has been done before [13], but we observe that a simpler upper bound formula is close enough. This bound is given in Model 7.1.

Modeling oversubscription (§G): The most difficult part of our analysis is bounding the oversubscription ratio. For this analysis, we use many modeling assumptions (detailed in §G.1). The oversubscription ratio is the output of a multi-commodity flow linear program, so it is much harder to get a simple formula. Moreover, the mathematics of interacting flows is complex and we resort to various simplifications based on intuitions on random graphs. Our final formula is given by a collection of polynomial equations in the various parameters, as given in §7.3.

D RANDOM GRAPH PRELIMINARIES

The router graph $G = (V, E)$ has n nodes and degree d. We model its construction as a random configuration graph (Chap. 2 of [58]). There are various ways to construct graphs in the configuration model, all of which are statistically equivalent and provide convenient analytical tools.

- The standard method: consider each node as incident to d "half-edges". Paired half-edges result in a proper edge. We generate a pairing with a uniform random permutation of all half-edges, and matching the first to the second, the third to the fourth, and so on.
- Getting the neighborhood of a set S: Condition on some edges within S. Each remaining half-edge in S picks *independently* another uar (uniformly at random) half-edge in graph to pair with. If multiple S half-edges pick the same partner to pair with, only one them (picked randomly) will pair up. The others remain unpaired. At this stage, some of the half-edges in S have paired up. Now, we simply pair up all remaining half-edges in the graph using the standard method (which has dependencies). The first part makes a collection of independent decisions, allowing for easier analysis.

D.1 Probability preliminaries

We list some probability theory definitions and theorems used in the analysis. In what follows, capital letters X, Y, Z denote non-negative integer-valued random variables; $\mathbf{E}[X]$ denotes the expectation of X. script letters $\mathcal{E}, \mathcal{F}, \ldots$ denote events; and $\Pr[\mathcal{E}]$ denotes the probability of event $\mathcal{E}$.

The first theorem will help us derive expected values of a collection of random variables. It makes no independence assumption on the random variables.

Theorem D.1. *[Linearity of Expectation] For any finite collection of random variables $X_1, X_2, \ldots, X_k$, $\mathbf{E}[\sum_{i\le k} X_i] = \sum_{i\le k} \mathbf{E}[X_i]$.*

The next theorem is convenient for dealing with "error bounds" and bounding the probability of bad events.

Theorem D.2. *[Union bound] Given a finite collection of events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k$, $\Pr[\bigcup_{i\le k} \mathcal{E}_i] \le \sum_{i\le k} \Pr[\mathcal{E}_i]$.*

The following states classic *concentration inequalities* that upper bound the probability of a random variable deviating significantly from its expectation.

Theorem D.3. *[The Chernoff bound] (Theorem 1.1 of [9]) Let $X = \sum_{i\le k} X_i$, where the X_i's are independent random variables in $[0, 1]$. Then,*

- *For all $\varepsilon \in (0, 1)$, $\Pr[X < (1-\varepsilon)\mathbf{E}[X]]$, $\Pr[X > (1+\varepsilon)\mathbf{E}[X]] \le \exp(-\varepsilon^2\mathbf{E}[X]/3)$.*
- *For all t, $\Pr[X > \mathbf{E}[X] + t]$, $\Pr[X < \mathbf{E}[X] - t] \le \exp(-2t^2/k)$.*
- *If $t > 2e\mathbf{E}[X]$, then $\Pr[X > t] \le 2^{-t}$.*

We use the phrase *with high probability* to denote probability at least $1 - o(1)$, with asymptotics over increasing n.

D.2 Random graph tools

We state and prove some probabilistic graph theorems that will be useful in the analysis.

Claim D.4. *Consider a node set S. Condition on any choice of edges within S that leave k half-edges unpaired. The probability that a vertex $v \notin S$ is a neighbor of S is at least $1 - \exp(-k/n)$.*

Proof. Conditioned on the current choice of edges in S, let us consider the process of building the neighbors of S. As described above, in the first step of this process, each of the k half-edges incident to S picks a uar unpaired half-edge to pair with. Fix $v \notin S$. Let us look at the probability that a single iteration connects with one of the d half-edges incident to v. This probability is at least $d/nd = 1/n$.

Since there are k independent iterations, one for each unpaired half-edge incident to S, the probability that no iteration connects with v is at most $(1-1/n)^k$. Applying the inequality $1 - x \le \exp(-x)$, we upper bound by $\exp(-k/n)$. Thus, the probability of v connecting to S is at least $1 - \exp(-k/n)$. □

Let us unpack this expression. Using the approximation (for small x), $\exp(-x) \approx 1 - x$, roughly speaking, we can approximate as $1 - (1 - k/n) = k/n$.

The following claim shows that all sets of sufficiently large size connect with every other node. It is a special case of the proof that the configuration model generates expanders.

Claim D.5. *Fix subset $S \subseteq V$ and condition on any choice of edges within S that leaves at least $nd/4$ unpaired half-edges incident to S. With probability at least $1 - n^{-4}$, every $v \in V \setminus S$ has an edge to S.*

Proof. By Claim D.4, the probability that $v \notin S$ is *not* a neighbor of S is at most $\exp(-k/n) \le \exp(-d/4)$. Let event $\mathcal{E}_v$ occur when v does not connect to S. By the union bound $\Pr[\bigcup_{v\notin S} \mathcal{E}_v] \le \sum_{v\notin S} \Pr[\mathcal{E}_v] \le n\exp(-d/4) \le n^{-4}$ (since

$d \gg \ln n$). The event $\overline{\bigcup_{v \notin S} \mathcal{E}_v}$ occurs precisely when all vertices outside S have an edge to S. □

E MODELING PATH LENGTH

This section models the path lengths in the RNG fabrics. We begin by bounding the sizes of Spraypoint levels.

E.1 Level sizes

Let us fix a destination t, and for convenience, denote $WP_i(t)$ as WP_i. We imagine constructing the levels incrementally, together with the graph. So $WP_{\le i} := \bigcup_{j \le i} WP_j$ is constructed, and then we choose the randomness to construct WP_{i+1}. Conditioning on $WP_{\le i}$ implies fixing the graph *inside* $WP_{\le i}$. So we have fixed all the edges between the waypoints up to this level. Then WP_i chooses some of its edges to determine WP_{i+1}. It is convenient to set WP_0 to be $Nbr(t)$.

We state and prove our main theorem about level sizes. Recall that $\ell := \max(1, \lceil \log_p(n/2d^2) \rceil)$.

Theorem E.1. *Fix destination t. The following bounds hold with probability $\ge 1 - o(1)$. We set $\lambda := p^\ell d^2/n$.*

- *For $0 \le i \le \ell$, $|WP_i(t)| = (1 \pm o(1))dp^i$.*
- *$|OR(t)| \le (\exp(-\lambda) + o(1))n$.*
- *All nodes in $OR(t)$ have an edge to a node in $IR(t)$.*

The proof has a few moving parts, so we separate them out. For any set of nodes S, we use $N(S)$ to denote the neighborhood of S (which may include nodes in S).

Claim E.2. *For any i, condition on $|WP_{\le i}| \le n/d$. The probability that any node v has more than $d/2$ edges to $WP_{\le i}$ is at most $1/n$.*

Proof. The node v makes d connections. Consider choosing these randomly as follows. First, each half-edge incident to v chooses to connect with a node in $WP_{\le i}$ with probability $|WP_{\le i}|/n \le 1/d$. Once these choices are made, then the half-edges are paired to a random half-edge in $WP_{\le i}$ or $\overline{WP_{\le i}}$ respectively. Each of these choices can be represented as a Bernoulli X_i, and the sum $X = \sum_{i \le d} X_i$ is the number of edges v has into $WP_{\le i}$.

We note that $\mathbf{E}[X] \le 1$. By the Chernoff bound Theorem D.3, $\Pr[X \ge d/2] \le 2^{-d/2}$. By a union bound over all nodes, the probability that any node has more than $d/2$ edges is at most $n2^{-d/2}$. Since $d \ge 2(\ln n + 5)$, $n2^{-d/2} \le n^{-5}$. □

The following lemma encapsulates a key calculation.

Lemma E.3. *Condition on a set S of nodes with at least half of its incident half-edges unpaired. Let $|S| \ge d/2$. Consider pairing all the half-edges incident to S. Then, with probability at least $1 - n^{-10}$, $|N(S)| \ge \min(n/4, |S|d/4)$. Moreover, if $|S| \le n/d$, at least $(d-4)|N(S)|$ half-edges incident to $N(S)$ are left unpaired.*

Proof. We will perform a coupling of the random variable $|N(S)|$ as follows. We initially mark an arbitrary set of $n/4$ nodes. We pair the half-edges incident to S iteratively. If a pairing connects to an unmarked node *and* the number of current neighbors of S is $< n/4$: we pick an marked non-neighbor of S, unmark it, and marked the (new) unmarked neighbor. (Otherwise, we do not change the marking.) Thus, if there are fewer than $n/4$ neighbors of S, then all neighbors of S are marked.

For the rth half-edge paired, let X_r be the indicator of the rth half-edge pairing with an unmarked node. Since the number of marked nodes is exactly $n/4$, the number of unmarked nodes is also exactly $3n/4$. Hence, $\Pr[X_r = 1] = 3/4$. Recall that there are at least $k \ge |S|d/2$ unpaired half-edges incident to S Let $X = \sum_{r \le k} X_r$, so $\mathbf{E}[X] = 3k/4$. By the Chernoff bound of Theorem D.3, $\Pr[\sum_r X_r \le k/2] \le \exp(-\mathbf{E}[X]/(3 \cdot 3^2)) \le \exp(-|S|d/54) \le \exp(-d^2/108)$. Since $d \ge \ln n$, this probability can be bounded by n^{-10} (for sufficiently large n).

So, with high probability, $X \ge k/2 \ge |S|(d/4)$. This means that either each iteration connected to an unmarked node (thereby creating a new neighbor) or the number of neighbors was at least $n/4$. Thus, $|N(S)| \ge \min(n/4, |S|d/4)$.

If $|S| \le n/d$, then $|S|/|N(S)| \le 4/d$. We paired up at most $|S|d$ half-edges in the above random process. Hence, there are at least $|N(S)|d - |S|d = |N(S)|d - 4|N(S)| \ge (d-4)|N(S)|d$ half-edges incident to $N(S)$ that are unpaired. □

E.1.1 *The first bullet point of Theorem E.1.*

We prove the first bullet point by an induction over i. We require a stronger induction hypothesis. We will prove that, conditioned on a choice of WP_i that leaves at least $d|WP_i|/2$ half-edges unpaired, with high probability, the construction of WP_{i+1} ensures that $|WP_{i+1}| \ge (1 - o(1))pd^{i+1}$ and at least $d|WP_{i+1}|/2$ half-edges of WP_{i+1} are unpaired.

We now perform the induction. We first deal with the base case WP_0, the neighbor set. Consider the pairing the d half-edges incident to t, one by one. Take the rth half-edge being paired. There are at most d neighbors of t already constructed, by the pairing of previous half-edges. The probability that the rth half-edge connects to one of these neighbors is at most d/n, regardless of all the previous choices. Let Z be the random variable denoting the number of half-edges connecting to an existing neighbor. The upper tail probabilities of Z is dominated by the corresponding tail of $B(d/n, d)$. (Here, $B(k, \alpha)$ denotes the binomial of k independent trials with success probability α.) We can apply the Chernoff bound to $B(d/n, d)$. Let $Y \sim B(d/n, d)$. Note that $\mathbf{E}[Y] = (d/n) \cdot d = d^2/n$. Since $d \ll n$, for sufficiently small $\varepsilon > 0$, $\varepsilon d \ge 2ed^2/n$. By Theorem D.3, $\Pr[Y \ge \varepsilon d] < 2^{-\varepsilon d} = o(1)$. Thus, $\Pr[Z \ge \varepsilon d] \le \Pr[Y \ge \varepsilon d] = o(1)$. In words, the overall probability of the half-edges connecting to εd existing

neighbors is $o(1)$. So, with probability $1 - o(1)$, the size of the neighbor set WP_0 is at least $(1 - o(1))d$.

At this stage, there are $(d-1)|WP_0|$ unpaired half-edges incident to WP_0. The only paired half-edges (formed edges) are those incident to t, so each neighbor in WP_0 is incident to $(d-1)$ unpaired half-edges.

We split into two cases, depending on ℓ.

Case 1, $\ell = 1$: In this case, there is just WP_1 to consider. As argued above, with high probability, $|WP_0| = d - o(1)$ and it is incident to $|WP_0|(d-1)$ unpaired edges. Apply Lemma E.3 with $S = WP_0 = Nbr(t)$. So probability $> n^{-10}$, $|N(S)| \geq \min(n/4, |S|d/4)$. Observe that $pd \leq n/4$ and $p \leq d/4$. So, $pd \leq |N(S)|$. There are enough candidates to construct WP_1 as desired. There is no induction necessary, since this is the last waypoint level. This ends the proof for this case.

Case 2, $\ell > 1$: Note that $\ell = \lceil \log_p(n/2d^2) \rceil$. Pick some $i < \ell$. We inductively assume that for all $j \leq i$, $|WP_j| = (1 - o(1))pd^j$ and that there are at least $|WP_i|d/2$ unpaired edges in the construction thus far. We just proved this for the base case. At each step of the induction, there is a small probability of error ($< 1/n^5$). We will simply union bound all these errors over the at most $\ell = \Theta(\ln n)$ levels.

Observe that $|WP_i| \leq dp^i \leq dp^{\log_p(n/2d^2)} \leq n/2d$. Also, $|WP_{\leq i}| \leq \sum_{j \leq i} dp^j = d(p^{i+1} - 1)/(p-1)$. Since $p \geq 2$, $p - 1 \geq p/2$. Hence, $\sum_{j \leq i} |WP_i| \leq dp^{i+1}/(p/2) \leq 2dp^i \leq (2 + o(1))|WP_i|$.

We apply Lemma E.3 with $S = WP_i$. Note that $|S| \geq d$ and $|S| \leq n/2d$. At least half of the incident half-edges are unpaired. Hence, with probability $> 1 - n^{-10}$, $|N(WP_i)| \geq |WP_i|d/4$. Since $|WP_{\leq i}| \leq 2|WP_i|$, there are at least $(d/4 - 2)|WP_i| \geq |WP_i|p$ (since $p + 2 \leq d/4$) nodes that are neighbors of WP_i and *not* in $WP_{\leq i}$. These are all candidates for WP_{i+1}. Thus, we can construct WP_{i+1} as desired. We get that $|WP_{i+1}| = p|WP_i| = (1 \pm o(1))dp^{i+1}$.

At least $(d-4)|N(WP_i)|$ half-edges incident to $N(WP_i)$ are unpaired. It remains the bound the number of unpaired half-edges incident to WP_{i+1}. Observe that a uniform random node of $N(WP_i)$ has at least $(d-4)$ unpaired half-edges. Let Z_r be the random variable denoting the fraction of unpaired edges on the rth node of WP_{i+1}. We have $\mathbf{E}[Z_r] \geq 1 - d/4$. Setting $Z = \sum_r Z_r$, $\mathbf{E}[\sum_r Z_r] \geq (1 - d/4)|WP_{i+1}|$ by linearity of expectation. Applying Theorem D.3, $\Pr[Z < \mathbf{E}[Z]/2] \leq \exp(-(1/3 \cdot 2^2) \cdot (1 - d/4)|WP_{i+1}|)$. Since $|WP_{i+1}| \geq (1 - o(1))pd^{i+1}$ and $i \geq 0$, the probability is $\exp(-\Theta(d)) \leq n^{-4}$. Thus, we have at least $d|WP_{i+1}|/2$ unpaired half-edges incident to WP_{i+1}. We union bound over all the errors. This completes the induction, and the proof of the first bullet.

E.1.2 The other bullets of Theorem E.1. With high probability, $|WP_i| = (1 \pm o(1))dp^i$. We will simply assume this property, and union bound the errors. Hence, the set WP_ℓ is incident to at least $(1 - o(1))d^2p^\ell$ half-edges. By Claim D.4, the probability that a node is *not* a neighbor of WP_ℓ is at most $\exp(-(1 - o(1))d^2p^\ell/n) = \exp(-\lambda(1 - o(1))) = (1 + o(1))\exp(-\lambda)$. For node v, let X_v be the indicator random variable that v is not a neighbor of WP_ℓ. Note that $\mathbf{E}[X_v] \leq (1 + o(1))\exp(-\lambda)$.

Consider the graph construction process where each v first independently chooses to connect to WP_ℓ. Then, if possible, the graph is formed conditioned on these choices. Setting $X = \sum_v X_v$, we can apply the Chernoff bound of Theorem D.3. So $\Pr[|X - \mathbf{E}[X]| > n/\ln n] \leq 2\exp(-2n/\ln^2 n)$, which is a tiny probability. Therefore, with high probability, the number of neighbors of WP_ℓ is close to the expectation, and hence, the graph can be feasibly formed conditioned on the the choices of X_v.

Thus, with high probability, the graph construction process is feasible, and the number of non-neighbors of WP_ℓ is at most $(1 + o(1))\exp(-\lambda)n + n/\ln n = \exp(-\lambda)n + o(n)$. Observe that all nodes in $OR(t)$ are non-neighbors of WP_ℓ, so this completes the proof of the second bullet.

Now for the third bullet. We can bound

$$\lambda \geq p^{\lceil \log_p(n/2d^2) \rceil} d^2/n \geq 1/2$$

As shown earlier, with high probability, $|OR(t)| \leq (\exp(-\lambda) + o(1))n \leq 0.7n$. Hence, at least $0.3n \geq n/4$ nodes lie in the remaining levels, waypoints or $IR(t)$. Claim D.5 asserts that every node is a neighbor of these remaining levels with high probability. So every node is either a neighbor of a waypoint or $IR(t)$. Nodes in $OR(t)$ are by definition not neighbors of waypoints, and hence must be neighbors of $IR(t)$. We union bound over all the errors, each of which was $< n^{-5}$. We perform the union bound at most $n^2 \log n$ times (once for each source, once for each destination, and once for each level). That bounds the error probability as $n^{-3} \log n < n^{-2}$.

E.2 Path length distribution

We now compute the distribution of Spraypoint path lengths.

Theorem E.4. *Fix a destination t. Consider the distribution of path lengths seen by a packet sprayed uniformly at random from a uniform random source s. With high probability, the distribution of Spraypoint paths to a destination t is as follows:*

- *There are d paths of length 1.*
- *For every $2 \leq i \leq \ell + 2$, there are $(1 \pm o(1))p^{i-2}d^2$ paths of length i.*
- *There are at most $(1 + o(1))\exp(-\lambda)nd$ paths of length $\ell + 4$.*
- *All other paths have length $\ell + 3$.*

Proof. Consider the pointing structure. Theorem E.1 gives the sizes of each level in this structure. We will assume the

bounds and statements of Theorem E.1. Imagine routing traffic from every possible source to the destination t. The spraying step would send traffic along all edges. For each edge, let us track the length to the destination.

Any edge incident to t leads to a path of just length 1, which gives d paths of length 1. If an edge incident to $WP_i(t)$, the path to the destination has length $i+2$ (one step for spraying and $(i+1)$ steps along the waypoints leading to t). Edges incident to $IR(t)$ lead to paths of length $\ell+3$. Finally, by Theorem E.1, observe that all nodes in $OR(t)$ have an edge to $IR(t)$ but not to any waypoints. Hence, edges incident to $OR(t)$ give Spraypoint paths of length $\ell+4$. We can use the level size bounds of Theorem E.1 to complete the proof. □

The bounds in Model 7.2 are expressed as fractions. For a destination t, there are nd possible paths to consider, one per source and spray operation. So we divide all the bounds in Theorem E.4 by nd to get the fractions in Model 7.2.

F MODELING EDGE DISJOINT PATHS

In this section, we estimate the number of edge disjoint paths between a source s and destination t by estimating the min cut of Spraypoint paths between s and t. We first set up some notation.

Definition F.1. For every $v \neq t$, we define the t-parent as:

- For $v \in Nbr(t)$, the parent is t.
- For $v \in WP_i(t)$ $(i \geq 1)$: we pick h uar neighbors in $WP_{i-1}(t)$ as the parents.
- For $v \in IR(t)$: we pick up to h uar neighbors of v that are waypoints (excluding $\{t\} \cup Nbr(t)$).
- For $v \in OR(t)$: we pick h random neighbors in $IR(t)$.

(If h selections cannot be made, we make the largest possible.)

Recall that the routing algorithm is simple. The first step is Spraying. After that, as part of pointing, a packet is forwarded to a uar parent. Eventually, the packet reaches t. This leads to some natural definitions of Spraypoint paths, from which we can define the graph between the endpoints.

Definition F.2. Fix a destination t. For any v, the Spraypoint path to t is any path obtained by repeatedly taking t-parents until ending at t.

- The **pointing graph** $\mathcal{P}_t$ has all edges that forward traffic to t.
- For a source-destination pair (s,t), the **Spraypoint graph** $\mathcal{S}_{s,t}$ has all incident edges to s and the union of Spraypoint paths from $N(s)$ to t.

We work with $\ell = 1$ setting. We make some asymptotic assumptions consistent with our parameters. The assumptions allow for simpler formulas. We assume:

- $pd/n \ll 1$
- $\lambda = pd^2/n \gg 1$
- $d \gg \ln^2 n$
- $h \ll \lambda$

Claim F.3. *Assume that $(p+1)d < n/\ln^2 n$. With probability $> 1 - n^{-2}$, all nodes pick at most $o(d)$ neighbors outside $IR(t)$.*

Proof. Fix a node v. We will treat each of the at most d neighbors that v has to pick as independent choices among $Nbr(t) \cup WP_1(t) \cup R_1(t)$. The probability that a neighbor lies in $Nbr(t) \cup WP_1(t)$ is at most $|Nbr(t) \cup WP_1(t)|/n \leq d(p+1)/n \leq 1/\ln^2 n$. Let X_v be the random variable denoting the number of neighbors of v in $Nbr(t) \cup WP_1(t)$, which is a sum of iid Bernoullis. Note that $\mathbf{E}[X_v] \leq d/\ln^2 n$. By Theorem D.3, $\Pr[X_v > d/\ln n] \leq 2^{-d/\ln n}$. By a union bound over all vertices, the probability that any $X_v > d/\ln n$ is at most $n2^{-d/\ln n} \leq n^{-2}$.

By Theorem E.4, with probability $> 1 - n^{-4}$, the size of $OR(t)$ is at most $(\exp(-\lambda) + o(1))n$. The probability that v has a single neighbor in this set is $o(1)$. Using a Chernoff bound identical to the one above, the probability that v has more than δn neighbors (for any constant $\delta > 0$) is $n^{-\Theta(1)}$.

Take a union bound, with high probability, all nodes pick at most $o(d)$ neighbors outside $IR(t)$. □

F.1 The matching connection

For any subset $S \subseteq IR(t)$, let us construct the bipartite graph $D_S = (S, Nbr(t), E)$ as follows. We connect $s \in S$ to $v \in Nbr(t)$ if s is a descendant of v in the pointing graph $\mathcal{P}_t$.

Claim F.4. *For any $S \subseteq IR(t)$: connect a source s to all nodes in S and consider the resulting pointing graph. The size of the s-t mincut is the size of the largest matching in D_S.*

Proof. We first prove that the largest matching in D_S is at least the mincut size (say r). By the maxflow-mincut theorem, there is a flow of value r from S to t, with unit edge capacities. By the integrality of flow, this flow consists of r edge disjoint paths from s to t. Note that each node in S has a single edge to s, and every node in $Nbr(t)$ has a single edge to t. So a node in $S \cup Nbr(t)$ participates in at most one flow path. Thus, the r flow paths provide edge disjoint paths from S to $Nbr(t)$ in the pointing graph $\mathcal{P}_t$, which constitute a matching in D_S.

Now, we prove that the mincut is at least the size of the largest matching in D_S. Consider a matching given by $\phi : S \to Nbr(t)$. For each $s' \in S$, follow a parent to $WP_1(t)$, and then a parent to get to $\phi(s) \in Nbr(t)$. For two nodes $s' \neq s''$ in S, $\phi(s') \neq \phi(s'')$. So the corresponding paths are edge disjoint. Hence, we can send one unit of flow on each such path, and get a flow with value at least the matching size. The maxflow value (and hence the mincut value) is at least the matching size. □

We can now connect the s-t mincut of the Spraypoint graph $\mathcal{S}_{s,t}$ to matching sizes. Note that $\mathcal{S}_{s,t}$ is formed by adding the edges connecting s to its neighborhood to $\mathcal{P}_t$. For convenience, we will only add the edges connecting s to $IR(t)$. By Claim F.3, this reduces the mincut by at most $o(d)$.

To bound the matching sizes, we use the following notation, based on random matching sizes.

Definition F.5. For positive integers ℓ, r, h, consider the random bipartite graph $H(L, R, E)$ formed as follows. We have $|L| = \ell$, $|R| = r$, and each node in L picks h uar neighbors in R with replacement. We use $\mu(\ell, r; h)$ to denote the expected maximum matching in H.

We now state our main theorem relating s-t mincuts in $\mathcal{S}_{s,t}$ to matching sizes.

Theorem F.6. *Assume $h = o(d)$. The expected s-t mincut size of $\mathcal{S}_{s,t}$ has the following values:*

- *If $s \in Nbr(t)$: the size is at least $\mu(d-p, d; h) \pm o(d)$.*
- *If $s \notin Nbr(t)$: the size is at least $\mu(d, d; h) \pm o(d)$.*

Proof. Consider $s \in Nbr(t)$. As discussed earlier, it picks $d-p-1$ random neighbors after conditioning on the pointing structure. By Claim F.3, at most $o(d)$ of these neighbors lie outside $IR(t)$. So, it picks $d - p - 1 - o(d) = d - p - o(d)$ neighbors in $IR(t)$. Call this set S. By Claim F.4, the s-t mincut size is the size of the largest matching in D_S. Each node s' in S (which is in $IR(t)$) picks h random parents in $WP_1(t)$, each of which has a parent in $Nbr(t)$. (We ignore the fact that nodes in $WP_1(t)$ might have multiple parents.) Thus, each s' effectively picks h random ancestors in $Nbr(t)$. So D_S is distributed exactly as the random bipartite graph in Definition F.5, and the expected matching size is $\mu(d - p - o(d), d; h)$. We can express this quantity as $\mu(d-p, d; h) \pm o(d)$, since changing any of the sets (in a bipartite graph) by $o(d)$ can affect the matching size by at most $o(d)$.

Consider $s \in WP_1(t)$. It picks $d - o(d)$ random neighbors in $IR(t)$. Applying the same logic as above, the expected mincut has value $\mu(d, d; h) \pm o(1)$. For $s \in IR(t)$, it picks $d - h - o(d)$ random neighbors in $IR(t)$. Since $h = o(d)$, s basically picks $d - o(d)$ random neighbors, and the value is the same as for $s \in WP_1(t)$. □

This theorem gives a lower bound on the size of the mincut. As mentioned in the proof, we ignore the fact that $WP_1(t)$ has potentially h neighbors into $Nbr(t)$, and only use one of the neighbors for routing. We believe this only loses a lower order term, and so we ignore it for the sake of cleaner expressions.

F.2 Expressions for random matching sizes

A good rule of thumb is that $\mu(d, d; h) \approx d(1 - \exp(-h))$. If $d - p = \alpha d$, then:

$$\mu(d-p, d; h) = \mu(\alpha d, d; h) = \min\left[\alpha d, d[1-\exp(-\alpha h)]\right] \quad (3)$$

For $h = 1$, these bounds are tight. For larger h, there are only upper bounds. Nevertheless, they match up quite accurately with experiments. The exact lower bound is a more complex expression given by Frieze-Mellsted [13], as explained later.

We show the *upper* bound in the next lemma, and then give the complicated exact expression. (The proof for $\mu(\alpha d, d; h)$ is analogous and omitted.)

Lemma F.7. *For all h, $\mu(d, d; h) \le d(1 - \exp(-h)) - o(d)$. Also, $\mu(d, d; 1) = d(1 - 1/e) - o(d)$.*

Proof. Recall that $\mu(d, d; h)$ is the expected maximum matching in the following random bipartite graph. There are two sets L, R of size d. Each node in L makes h random connections with R.

Consider a vertex $r \in R$. The probability that an edge does not connect with r is $(1 - 1/d)$. The probability that no edge makes a connection is $(1-1/d)^{dh}$, since there are dh random (multi)edges in the graph. Since $1 - x \ge \exp(-x - x^2)$, we lower bound this probability by $\exp(-(1/d + 1/d^2) \cdot dh) = \exp(-h)(1 - \Theta(h/d))$.

Consider indicator random variable X_r for r having degree zero. By linearity of expectation and the calculation above, $\mathbf{E}[\sum_{r\in R} X_r] = \sum_{r\in R} \mathbf{E}[X_r] \ge d\exp(-h) - o(d)$. We can also upper bound $(1-1/d) \le exp(-1/d)$, and get the lower bound $d\exp(-h)$. Thus, the expected number of degree zero nodes in R is $d\exp(-h) \pm o(d)$. These nodes cannot be matched, and hence the matching size is at most $d(1-\exp(-h)) \pm o(d)$.

When $h = 1$, we can match this bound. Take the iterative process that picks an arbitrary node in R with non-zero degree, and matches it arbitrarily to some neighbor in L. Then, we delete this node and its neighbors from the graph, and iterate. When $r \in R$ is removed, it removes its neighbors in L. Since they all have degree 1, all edges removed are incident to r. Hence, degrees of other nodes in R are unaffected. All in all, every node of non-zero degree in R gets matched. □

The Frieze-Mellsted bound: The optimal value of $\lim_{d\to\infty} \mu(d, d; h)/d$ was computed in [13]. We state the bound here, and show that it is fairly close to the simpler expression of Lemma F.7.

(Refer to Thm. 3 of [13].) Let z^* be the largest non-negative solution of $(z/h)^{1/(h-1)} + \exp(-z) + 1 = 0$. Then,

$$\begin{aligned} \lim_{d\to\infty} \mu(d, d; h)/d &= 2 - (1 - \exp(-z^*))^h \\ &\quad -(1 + z^*)\exp(-z^*) \quad (4) \\ &\approx 1 - (1 + z^* - h)\exp(-z^*) \quad (5) \end{aligned}$$

For $h = 2$, the bound is ≈ 0.838, while the bound of Lemma F.7 is ≈ 0.865. For $h = 4$, the bound is ≈ 0.979, while the bound of Lemma F.7 is ≈ 0.982. As h becomes larger, z^* tends to h, and the bound above tends to $1 - \exp(-h)$.

F.3 The mincut bound, summarized

We summarize the bounds in this section. Fix a destination t. The s-t mincut value of the Spraypoint graph $\mathcal{S}_{s,t}$ is given by the following formulas:

- For most s, the average mincut is $d(1 - \exp(-h))$.
- The lowest (expected) mincut occurs for $s \in Nbr(t)$. In that case, setting $d - p = \alpha d$, the bound is $\min\left[\alpha d, d[1 - \exp(-\alpha h)]\right]$. When α is close to 1, the bound is close to $d(1 - \exp(-h))$.

To get a sense of the benefits of h, we plug in different values. The best possible mincut is d. Let us see what fraction of d is achieved. For $h = 1$, we get a $1 - 1/e \approx 0.63$ fraction. For $h = 2$, we get a $1 - \exp(-2) \approx 0.86$ fraction. For $h = 4$, we get a $1 - \exp(-4) \approx 0.98$ fraction. At this point, the statistical variation of the mincut value is bigger than any potential improvements.

The following table gives the various fractions. Note that $p = 0$ gives the bound for most s.

p	$h = 1$	$h = 2$	$h = 4$
0	0.63	0.86	0.98
$d/4$	0.53	0.75	0.75
$d/3$	0.49	0.66	0.66
$d/2$	0.39	0.5	0.5

G MODELING OVERSUBSCRIPTION

To help us characterize stochastic oversubscription, we set up some notation and definitions. We set up a multicommodity flow problem. Each edge of the graph $G = (V, E)$ is treated as two directed arcs, one in each direction. The capacity of each arc is one. There is a doubly stochastic demand matrix M that specifies that amount of flow that each source needs to send to each destination. So $M_{s,t}$ is the amount of flow that s needs to send to t. The total flow in or out of any node is at most one since M is doubly stochastic.

Definition G.1. (1) A demand matrix N is a **feasible matrix** if the multicommodity flow specified by N can be routed while honoring link capacity constraints. (2) The **demand multiplier** of a doubly stochastic demand matrix M, denoted $c(M)$, is the largest c such that cM can be satisfied. (3) The **oversubscription ratio (oversub)** is the value of $\max_M d/c(M)$.

Thus, the oversub is the scaled reciprocal of the lowest possible demand multiplier.

Our first lemma connects the oversub to the fraction of optimal flow along different lengths.

Lemma G.2. *Consider the traffic matrix M that achieves the oversub bound. For the corresponding optimal flow, let δ_i be the fraction of the flow that takes length i paths. Then, the oversub is at least $\sum_i i\delta_i$.*

Proof. The proof goes via a simple "total capacity" argument. Suppose the oversub is ω. This means that the corresponding demand multiplier is d/ω, which is the total flow that each node sends/receives. The total flow in the network is nd/ω, since each node is a source.

Let us bound the sum $F := \sum_{\text{arc } e} f_e$, where f_e is the total flow on the arc e. By definition, the amount of flow along length i paths is $\delta_i nd/\omega$. Observe that this flow takes up $i\delta_i nd/\omega$ capacity. Thus, the contribution of this flow to the sum F is $i\delta_i nd/\omega$, and we bound $F = \sum_i i\delta_i nd/\omega$.

Since each arc has unit capacity and there are nd arcs, $F \le nd$. So $\sum_i \delta_i nd/\omega \le nd$ and $\omega \ge i \sum_i \delta_i$. □

G.1 Principles underlying the model

Based on Lemma G.2, we will devise a model for oversub in the next section as a function of system parameters, n, d, p, h. The model is based on a number of principles.

The greedy shorter path principle: The oversub lower bound of Lemma G.2 is $\sum_i i\delta_i$, so it is preferable to have larger values of δ_i for small i, rather than the other way around. Based on this intuition, we construct the demand multipliers for a demand matrix by greedily maximizing the amount of flow on shorter paths first. For the $\ell = 1$ setting, all flows paths have length between 1 to 5 (Theorem E.4). We can ignore path length 1, which form a neglible fraction of paths. The main calculations compute the maximum amount of flow that goes along paths of length 2, then conditioned on that, maximize flow along paths of length 3, so on and so forth. We compute μ_2, μ_3, μ_4, and μ_5 defined as follows. The average amount of flow that a source sends along length i paths is $\mu_i d$. The total demand multiplier is $(\mu_2+\mu_3+\mu_4+\mu_5)d$, and hence the oversub is $(\mu_2 + \mu_3 + \mu_4 + \mu_5)^{-1}$.

The average case principle: By Theorem 2.1 of [37], the worst-case traffic matrix is a permutation matrix, which is a perfect matching of source to sinks. We will actually analyze the average demand multiplier, and hence oversub, of a *random* permutation matrix. In many problems in random graph theory, the worst-case behavior can be shown to be close to the average-case behavior. Specifically, suppose an adversary picks a traffic matrix and then we choose the graph G. We can compute the probability that the demand multiplier deviates significantly from the average value. If this probability is small enough (like $< n^{-n}$), we could union bound over all permutation matrices. This means, that with non-trivial

probability, a random graph will have high demand multipliers for *all* permutation matrices. This is a common paradigm in random graph/matrix theory: show that the probability of large deviations from average are exponentially small, and then union bound over all the exponential possibilities (proof of expansion of random graphs, Chap. 5 of [34]).

The random deletion principle: Once (say) we have determined μ_2, this flow uses up some capacity. When computing μ_3, the capacity constraints have changed. We consider a simplified model for the change. The length 2 flow uses up $2\mu_2 nd$ capacity from the network, so we assume that each edge is fully congested with probability $2\mu_2 nd/nd = 2\mu_2$. We delete these edges, since they are unavailable to carry flow. This deletion principle is less accurate for larger length flows. Larger length flows do not use up the capacity of constituent edges. Hence, for most calculations, we will only assume this principle for smaller length flows.

The viable path principle: For any length i, we estimate the number of paths of length i that can be used for routing. For concreteness, for the remaining principles, we will consider length 3 (the first non-trivial case). Such a path goes $s \to WP_1(t) \to Nbr(t) \to t$. Theorem E.4 tells us how many neighbors of s lead to such paths (in this case, it is pd^2/n for an average source s). Because of the deletion principle, some of these edges (from s to $WP_1(t)$) get deleted. Consider some $v \in WP_1(t)$ such that (s, v) is not deleted. The pointing paths from v to t may get removed *after* the deletions. We need to model the probability of this event. The parameter h ensures that each v has h (edge disjoint) pointing paths to t. We calculate the probability that some path survives the deletion. Putting it all together, we can estimate the number of paths of length 3 that allow for routing. The next principles account for the intersections/congestion among these paths, from which we compute the flow that can be sent along these paths.

The unique position principle: By the mincut bounds of Lemma F.7, there should be enough edge disjoint paths length between a given source-destination pair to carry the desired flow. (We do not expect an oversub less than 2, so even an mincut of $d/2$ suffices for a source-destination pair.) Consider the entire collection of edge disjoint length 3 paths, over all source-destination pairs. Congestion occurs because some edges are present among length 3 paths for multiple source-destination pair.

The concept of *position* helps us bound the congestion. For a path, an edge has one of three positions: first, middle, or last. *An edge (u, v) cannot be in the first position for more than one path.* When (u, v) is the first edge, u must be the source, and all length 3 paths from u (in the collection) are edge disjoint. Similarly, an edge cannot be in the last position for more than one path. Extrapolating, we assume that no edge can be in a specific position for more than one path. (This is technically false, since an edge could be in the middle for multiple paths, but we believe this only leads to a lower order term.) Overall, for length 3, an edge can participate in at most 3 flow paths of length 3. Similarly, we assume that for length i, an edge participates in at most i flow paths, one in each position.

The binomial congestion principle: We want to compute the amount of flow along length 3 paths to estimate μ_3. Using the viable path principle, suppose we determine that there are $\phi_3 d$ edge disjoint paths of length 3 between every source-destination pair. Consider the total collection of $\phi_3 nd$ paths. We want to model, for a given edge, the number of paths that this edge participates in. This is the congestion on that edge, focusing only on length 3 flow. By the unique position principle, the congestion is at most 3. We will assume that an edge is present in a particular position independently with probability ϕ_3. Consider an edge on a particular length 3 path. It already occupies a position on this path, and can take up two other positions (on different paths). The congestion on the edge is distributed as the binomial $B(2, \phi_3) + 1$ (the binomial $B(2, \phi_3)$ is the sum of 2 independent 0-1 random variables with expectation ϕ_3) plus one.

The reciprocal principle: On each edge of the path, using the binomial model, we have a distribution for the number of *other* paths that use this edge. The maximum value among all the edges of the path is the congestion of the path. We use the following simple claim to get a flow bound.

Claim G.3. *Consider a family $\mathcal{P}$ of paths. For each path in $P \in \mathcal{P}$, let $c(P)$ be the congestion of P among the paths $\mathcal{P}$. (So $c(P)$ is the maximum, among all edges of P, of the number of paths that use the edge.) Then each path $P \in \mathcal{P}$ can simultaneously route $1/c(P)$ units of flow without violating the (unit) capacity constraints.*

Proof. Suppose every path $P \in \mathcal{P}$ routes $1/c(P)$ units of flow. Consider any edge e that is contained in path $P_1, P_2, \ldots, P_k$. The congestion of e, denoted $c(e)$, is k. Observe that for every path P_i, $c(P_i) \geq c(e) = k$. The total amount of flow in e is $\sum_{i \leq k} 1/c(P_i) \leq \sum_{i \leq k} 1/c(e) = \sum_{i \leq k} 1/k = 1$. □

This claim allows us to get the total flow that can be sent along, say, length 3 paths. Over these paths, we have computed the distribution of congestion. The average value of the reciprocal of the congestion is the average flow that can be sent along paths of this length. We multiply this average by the total number of paths to get the total flow.

To move to the next path length, we use the random deletion principle to delete edges that are used by this flow. We run this entire calculation again for the next path length. For larger paths lengths i (like 4 and above), the congestion values of the paths are quite close to their maximum value i. So we just send $1/i$ units of flow along paths of these length.

This simplification helps us avoid complicated probability calculations that are lower order terms.

G.2 The oversub formula

We now compute the formula based on the principles above. Given the parameters, n, d, p, and h, the final formula is a polynomial function of p and d/n, with exponential dependencies on h. There are minor corrections needed for extreme parameter settings (when d is comparable to n, or p is close to d). As discussed earlier, we will estimate μ_2, μ_3, μ_4, and μ_5.

μ_2: By Theorem E.4, for a given destination t, a typical source s would have a d/n fraction of its neighbors in $Nbr(t)$. The total number of such paths from s to t is d^2/n and the total number of edges on these paths is $2d^2/n$. As a fraction of the total number of arcs (directed edges), we get $2d/n$ (= $(2d^2/n) \cdot n/nd$). We typically assume that $2d \ll n$. Assuming that an edge is present on these paths with probability $2d/n$, the probability that an edge is on two (or more) paths is $(2d/n)^2$, which is negligible. So we assume that no edge is present on two of these paths, or equivalently, the congestion of every path/edge is at most one. So all the length 2 paths across all source-destination pairs are edge disjoint. Each such path can carry one unit of flow, so s can send d^2/n units of flow along length 2 paths to t. Thus, $\mu_2 = d/n$

μ_3: This is where most of the work goes. By the unique position principle and the random deletion principle, each edge has probability μ_2 of being in the first position of a length 2 flow path. (Same for the last position.) So we will delete a μ_2 fraction of edges (for the first position), and delete another μ_2 fraction for the last position.

We now estimate the number of length 3 paths that can be used to route flow. We compute a fraction ϕ_3 such that each source-destination pair has $\phi_3 d$ viable length 3 paths to route flow. By Theorem E.4, a typical source s has a $pd/n = p\mu_2$ fraction of neighbors in $WP_1(t)$. (For extreme large values of p, $p\mu_2$ could be larger than one, so we "correct" by making this fraction $\min(p\mu_2, 1-\mu_2)$. The latter term just assumes that everything that is not in $Nbr(t)$ is in $WP_1(t)$.) Of these corresponding edges incident to s, some of them might be in the last position of a length 2 flow. They cannot be in the first position, since all first position edges go from s to $Nbr(t)$, while these edges under consideration go from s to $WP_1(t)$. By the deletion principle, we expect a μ_2-fraction of the edges to be deleted.

So we get that a $\min(p\mu_2, 1-\mu_2) \cdot (1-\mu_2)$ fraction of neighbors of a source s lead to length 3 paths to the corresponding destination t. Consider such a neighbor v of s that is in $WP_1(t)$. It has h pointing (likely edge disjoint) paths of length 2 to the destination t. Each edge gets deleted with independent probability $2\mu_2$, so a path survives with probability $(1-2\mu_2)^2 \approx 1 - 4\mu_2$ (assuming $\mu_2 << 1$). The path gets removed with probability $\approx 4\mu_2$. The probability that at least one of h edge disjoint paths survives is $1-(4\mu_2)^h$. Overall, the probability that $v \in WP_1(t)$ still has a path to t is $1-(4\mu_2)^h$.

Combining with the number of such v's, we get that there are $\phi_3 d$ neighbors of s that lead to viable 3 length flows, where:

$$\phi_3 = \min(p\mu_2, 1-\mu_2) \cdot (1-\mu_2) \cdot (1-(4\mu_2)^h) \quad (6)$$

Our next step is to deal with the congestion. Consider a length 3 flow path. Each edge on this path could potentially take up two other positions (on different paths), using the unique position principle. The congestion on the path is the maximum of three independently chosen random variables according to the shifted binomial $B(2, \phi_3) + 1$.

Claim G.4. *Define $Y := \max(X_1, X_2, X_3)$, where each X_i is independently chosen from $B(2, \phi_3)$. Then, $\Pr[Y = 0] = (1-\phi_3)^6$, $\Pr[Y = 1] = [(1-\phi_3^2)^3 - (1-\phi_3)^6]$, and $\Pr[Y = 2] = [1-(1-\phi_3^2)^3]$.*

Proof. We can treat each binomial as the sum of two independent Bernoullis, each with parameter ϕ_3. For Y to be zero, all the Bernoullis need to take value zero. This happens with probability $(1-\phi_3)^6$.

Each X_i takes value 2 with probability ϕ_3^2. The probability that no X_i takes value 2 is $(1-\phi_3^2)^3$. Thus, $\Pr[Y = 2] = [1-(1-\phi_3^2)^3]$.

We deduce $\Pr[Y = 1]$ as $1 - \Pr[Y = 0] - \Pr[Y = 2]$, and applying the above formulas. □

The congestion on a path is distributed as $Y + 1$, where Y is distributed as in Claim G.4. As discussed in the earlier section, the average flow that can be sent is $\mathbf{E}[1/(Y+1)]$. Using Claim G.4, this average can be computed easily:

$$\begin{aligned} \kappa_3 &= (1-\phi_3)^6 + [(1-\phi_3^2)^3 - (1-\phi_3)^6]/2 \\ &\quad + [1-(1-\phi_3^2)^3]/3 \end{aligned} \quad (7)$$

Finally, we set $\mu_3 = \phi_3 \kappa_3$.

μ_4 and μ_5: We create variables for the relative sizes of the Spraypoint levels, based on Theorem E.4.

$$\sigma_4 = 1-(p+1)d/n - \exp(-pd^2/n) \quad \sigma_5 = \exp(-pd^2/n) \quad (8)$$

Let us start with μ_4. The quantity σ_4 is the fraction of s's neighbors that lead to a length 4 flow path. The probability that these corresponding edges are removed by previous flows is $\mu_2 + 2\mu_3$. (This edge can take up the last position in length 2 flows, and the middle or last position in length 3 flows.)

Consider a neighbor v of s leading to length 4 path. As in the previous μ_3 calculations, we need to compute the probability that v still has a path to t after the deletions. We will model v's pointing paths as follows. It has h neighbors in $WP_1(t)$, each of which has h edge disjoint paths to t. Since

we are considering longer paths, we will go with the weaker deletion principle of removing edges with probability $2\mu_2$. The probability that v still has a path to t can be approximated as β, defined below.

$$\beta = 1 - [1 - (1 - 2\mu_2)(1 - (4\mu_2)^h)]^h \tag{9}$$

The fraction of viable paths is $\phi_4 = \sigma_4\beta$. For paths of length 4, we will simply assume the worst case congestion of 4, so we can only send $1/4$ units of flow on each path. So we set $\kappa_4 = 1/4$ and $\mu_4 = \sigma_4\kappa_4 = \sigma_4\beta(1 - \mu_2 - 2\mu_3)/4$.

For μ_5, we only look at deletions incident to the source s. This is because there will be many paths (polynomial in h) from a neighbor of s (in $OR(t)$) to the destination t. The odds are quite high that some path will still be available for routing, so we do not bother to model it carefully. Following the logic for μ_4, we estimate $\mu_5 = \sigma_4(1 - \mu_2 - 2\mu_3 - 3\mu_4)/5$.

Let $\mu_i^+ = \max(\mu_i, 0)$. This deals with extreme situations (n is too small, p or d are large). Finally, our oversub bound is

$$(\mu_2^+ + \mu_3^+ + \mu_4^+ + \mu_5^+)^{-1} \tag{10}$$