Xpander: Towards Optimal-Performance Datacenters

Asaf Valadarsky*
asaf.valadarsky@mail.huji.ac.il

Gal Shahaf†
gal.shahaf@mail.huji.ac.il

Michael Dinitz‡
mdinitz@cs.jhu.edu

Michael Schapira*
schapiram@huji.ac.il

ABSTRACT

Despite extensive efforts to meet ever-growing demands, today's datacenters often exhibit far-from-optimal performance in terms of network utilization, resiliency to failures, cost efficiency, incremental expandability, and more. Consequently, many novel architectures for high performance datacenters have been proposed. We show that the benefits of state-of-the-art proposals are, in fact, derived from the fact that they are (implicitly) utilizing "expander graphs" (aka expanders) as their network topologies, thus unveiling a unifying theme of these proposals. We observe, however, that these proposals are not optimal with respect to performance, do not scale, or suffer from seemingly insurmountable deployment challenges. We leverage these insights to present Xpander, a novel datacenter architecture that achieves near-optimal performance and provides a tangible alternative to existing datacenter designs. Xpander's design turns ideas from the rich graph-theoretic literature on constructing optimal expanders into an operational reality. We evaluate Xpander via theoretical analyses, extensive simulations, experiments with a network emulator, and an implementation on an SDN-capable network testbed. Our results demonstrate that Xpander significantly outperforms both traditional and proposed datacenter designs. We discuss challenges to real-world deployment and explain how these can be resolved.

*School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel

†Dept. of Mathematics, The Hebrew University, Jerusalem, Israel

‡Department of Computer Science, Johns Hopkins University, USA

CoNEXT '16, December 12-15, 2016, Irvine, CA, USA
 ISBN 978-1-4503-4292-6/16/12…$15.00
DOI: http://dx.doi.org/10.1145/2999572.2999580

CCS Concepts

•**Networks** → **Topology analysis and generation; Data center networks;**

1. INTRODUCTION

The rapid growth of Internet services is placing tremendous demands on datacenters. Yet, as evidenced by the extensive research on improving datacenter performance [22, 23, 6, 46, 50, 21, 43], today's datacenters often exhibit far-from-optimal performance in terms of network utilization, resiliency to failures, cost efficiency, amenability to incremental growth, and beyond.

1.1 The Secret to High Performance

We show that state-of-the-art proposals for next-generation datacenters, e.g., low-diameter networks such as Slim Fly [8], or random networks like Jellyfish [46], have an *implicit* unifying theme: utilizing an "expander graph" [24] as the network topology and exploiting the diversity of short paths afforded by expanders for efficient delivery of data traffic. Thus, our first contribution is shedding light on the underlying reason for the empirically good performance of previously proposed datacenter architectures, by showing that these proposals are specific points in a much larger design space of "expander datacenters". We observe, however, that these points are either not sufficiently close to optimal performance-wise, are inherently not scalable, or face significant deployment and maintenance challenges (e.g., in terms of unpredictability and wiring complexity).

We argue that the quest for high-performance datacenter designs is inextricably intertwined with the rich body of research in mathematics and computer science on building good expanders. We seek a point in this design space that offers *near-optimal performance* guarantees while providing a *practical* alternative for today's datacenters (in terms of cabling, physical layout, backwards compatibility with today's protocols, and more). We present Xpander, a novel expander-datacenter architecture carefully engineered to achieve both these desiderata.

Importantly, utilizing expanders as network topologies has been proposed in a large variety of contexts, ranging from parallel computing and high-performance computing [47, 14, 13, 8] to optical networks [39] and peer-to-peer networks [38, 37]. Our main contributions are examining the performance

and operational implications of utilizing expanders in the *datacenter networking context*, and seeking optimal design points in this specific domain (namely, Xpander). Indeed, despite the large body of research on expanders, many aspects of using expanders as datacenter networks (e.g., throughput-related performance measures, specific routing and congestion control protocols, deployment costs, incremental growth, etc.) remain little understood. We next elaborate on expanders, expander datacenters, and Xpander.

1.2 Why Expanders?

Intuitively, in an expander graph the total capacity from any set of nodes S to the rest of the network is large with respect to the size of S. We present a formal definition of expanders in Section 2. Since this implies that in an expander every cut in the network is traversed by many links, traffic between nodes is (intuitively) never bottlenecked at a small set of links, leading to good throughput guarantees. Similarly, as every cut is large, every two nodes are (intuitively) connected by many edge-disjoint paths, leading to high resiliency to failures. We validate these intuitions in the datacenter context.

Constructing expanders is a prominent research thread in both mathematics and computer science. In particular, building well-structured, deterministic expanders has been the subject of extensive study (see, e.g., [32, 30, 41]).

1.3 Why Expander Datacenters?

We refer to datacenter architectures that employ an expander network topology as "expander datacenters". We evaluate the performance of various expander datacenters through a combination of formal analyses, extensive flow-level and packet-level simulations, experiments with the mininet network emulator, and implementation on an SDN-capable network testbed (OCEAN [3]).

Our results reveal that expander datacenters achieve near-optimal network throughput, significantly outperforming traditional datacenters (fat trees [6]). We show, in fact, that expander datacenters can match the performance of today's datacenters with roughly $80-85\%$ of the switches. Beyond the above improvements in performance, our results establish that expander datacenters are significantly more robust to network changes than today's datacenters.

Studies of datacenter traffic patterns reveal tremendous variation in traffic over time [7]. Unfortunately, a network topology that fares well in one traffic scenario might fail miserably in other scenarios. We show that expander datacenters are robust to variations in traffic. Specifically, an expander datacenter provides close-to-optimal performance guarantees with respect to *any* (even adversarially chosen!) traffic pattern. We show, moreover, that *no* other network topology can do better. The performance of expander datacenters also degrades much more gracefully than fat trees in the presence of network failures.

Alongside the above merits, expander datacenters, unlike today's rigid datacenter networks, can be incrementally expanded to any size while preserving high performance, thus meeting the need to constantly grow existing datacenters.

1.4 Why Xpander?

While our results indicate that expander datacenters universally achieve high performance, different expander datacenters can differ greatly both in terms of the exact level of performance attained, and in terms of their deployability. We thus seek a point in the space of expander datacenters that achieves near-optimal performance while presenting a practical alternative for today's datacenters. We present Xpander, which we regard as a good candidate for such a point. We evaluate Xpander's performance guarantees, showing that it achieves the same level of performance as random networks, the current state-of-the-art with respect to performance [46, 45, 25] (but which suffer from severe impediments to deployment, as discussed later), and that it outperforms low-diameter datacenter designs (namely, Slim-Fly [8]).

We analyze the challenges facing the deployment of Xpander in practice through detailed investigations of various scenarios, from "container datacenters" to large-scale datacenters. Our analyses provide evidence that Xpander is realizable with monetary and power consumption costs that are comparable or lower than those of today's prevalent datacenter architectures and, moreover, that its inherent well-structuredness and order make wiring Xpanders manageable (avoiding, e.g., the inherent unstructuredness and complexity of random network designs a la Jellyfish [46]).

1.5 Organization

We provide a formal exposition of expanders and expander datacenters in Section 2, and of Xpander in Section 3. We show that expander datacenters such as Xpander indeed attain near-optimal performance in Section 4, matching the performance of randomly networked datacenters. We compare Xpander to fat trees [6] and Slim Fly [8] in Sections 5 and 6, respectively. We discuss deployment challenges and solutions in Section 7, and related work in Section 8. We conclude in Section 9. Due to space constraints, our proofs and some additional materials appear in the Xpander project webpage [5].

2. EXPANDER DATACENTERS

We provide below a formal exposition of expanders. We discuss past proposals for high-performance datacenters and explain why these are, in fact, "expander datacenters".

Consider an (undirected) graph $G = (V, E)$, where V and E are the vertex set and edge set, respectively. For any subset of vertices S, let $|S|$ denote the size of S, let $\partial(S)$ denote the set of edges leaving S, and let $|\partial(S)|$ denote the size of $\partial(S)$. Let n denote the number of vertices in V (that is, $|V|$). The *edge expansion* $EE(G)$ of a graph G on n vertices is $EE(G) = \min_{|S| \leq \frac{n}{2}} \frac{|\partial(S)|}{|S|}$. We say that a graph G is *d-regular* if the degree of each vertex is exactly d. We call a d-regular graph G an *expander* if $EE(G) = c \cdot d$ for some constant $c > 0$. We note that the edge expansion of a d-regular graph cannot exceed $\frac{d}{2}$.[1] Constructions of expanders

[1]This is what is achieved by a random bisection and thus the worst bisection is no better.

whose edge expansion (asymptotically) matches this upper bound exist (see more below).

We refer to datacenters that rely on an expander graph as their network topology as "expander datacenters". We consider two recent approaches to datacenter design that have been shown to yield significantly better performance than both today's datacenters and many other proposals: low-diameter networks, e.g., Slim Fly [8], and randomly networked datacenters, e.g., Jellyfish [46]. We observe that these two designs are, in fact, expander datacenters. We show in the subsequent sections that this is indeed what accounts for their good performance.

Randomly networked datacenters. Recent studies [46, 45] show that randomly networked datacenters achieve close-to-optimal performance guarantees. Indeed, to date, randomly networked datacenters remain the state-of-the-art (performance-wise) in terms of network throughput, resiliency to failures, incremental expandability, and more. Our results for expander datacenters suggest that these merits of randomly-wired, d-regular, datacenters are derived from the fact that they achieve near-optimal edge expansion (close to $\frac{d}{2}$), as established by classical results in random graph theory [10, 18].

Unfortunately, the inherent unstructuredness of random graphs makes them hard to reason about (diagnose and troubleshoot problems, etc.) and build (e.g., cable), and thus poses serious, arguably insurmountable, obstacles to their adoption. Worse yet, as with any probabilistic construct, the good traits of random graphs are guaranteed only with *some* probability. Hence, while utilizing random network topologies in datacenters is an important thought experiment, the question arises: Can a *well-structured* and *deterministic* construction achieve the same benefits *always* (and not probabilistically)? We show later that Xpander indeed accomplishes this.

Low-diameter datacenters. Slim Fly [8] (SF) leverages a graph-theoretic construction of low-diameter graphs [33] (of diameter 2 or 3). Intuitively, by decreasing the diameter, less capacity is wasted when sending data, and hence overall performance is higher. As shown in [8], SF outperforms both existing and proposed datacenter architectures, but performs worse than random network topologies. We present the following new theoretical result for SF, showing that SF is a good expander. The proof appears in [5].

THEOREM 2.1. *A Slim-Fly network of degree d and diameter 2 has edge expansion at least $\frac{d}{3} - 1$.*

This result suggests that SF's good edge expansion may, in fact, be the explanation for its good performance, and *not* its low diameter. Indeed, our findings (see Section 6) suggest that by optimizing the diameter-size tradeoff, Slim Fly sacrifices a small amount of expansion, which leads to worse performance than random networks (and Xpander) as the network gets larger. Worse yet, the low diameter of SF imposes an extremely strict condition on its size (as a function of port count of each switch), imposing, in turn, a strict limit on the scalability of Slim Fly. Xpander, in contrast, can be constructed for virtually any given switch port-count and network size, as discussed in Section 6.

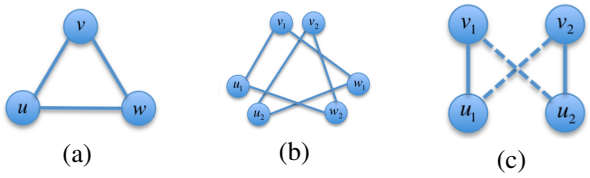


Figure 1: Illustration of 2-Lift

Other expander datacenters. A rich body of literature in mathematics and computer science deals with constructions of expanders, e.g., Margulis's construction [32], algebraic constructions [30], and constructions that utilize the so-called "zig-zag product" [41]. Such constructions can be leveraged as network topologies in the design of expander datacenters. For example, our construction of the Xpander datacenter topology utilizes the notion of 2-lifting a graph, introduced by Bilu and Linial [9, 31] (see Section 3). However, as evidenced by the above discussion, different choices of expanders can yield different performance benefits and greatly differ in terms of deployability. We next discuss the Xpander architecture, designed to achieve near-optimal performance *and* deployability.

3. XPANDER: OVERVIEW AND DESIGN

In light of the limitations of past proposals, our goal is to identify a datacenter architecture that achieves near-optimal performance yet overcomes deployment challenges. We next present the Xpander datacenter design, and show that it indeed accomplishes these desiderata.

3.1 Network Topology

Lifting a graph. Consider the graph G depicted in Figure 1(a). Our construction of Xpander leverages the idea of "lifting" a graph [9, 14]. We start by explaining *2-lifts*. A 2-lift of G is a graph obtained from G by (1) creating two vertices v_1 and v_2 for every vertex v in G; and (2) for every edge $e = \{u, v\}$ in G, inserting two edges (a matching) between the two copies of u (namely, u_1 and u_2) and the two copies of v (namely, v_1 and v_2). Figure 1(b) is an example of a 2-lift of G. Observe that the pair of vertices v_1 and v_2 can be connected to the pair u_1 and u_2 in two possible ways, described by the solid and dashed lines in Figure 1(c). When the original graph G is an expander, the 2-lift of G obtained by choosing between every two such options at random is also an expander with high probability [9]. Also, these simple random choices can be *derandomized*, i.e., the same guarantee can be derived in a deterministic manner [9].

2-lifting can be generalized to k-lifting for arbitrary values of k in a straightforward manner: create, for every vertex v in G, k vertices, and for every edge $e = \{u, v\}$ in G, insert a matching between the k copies of u and the k copies of v. As with 2-lifting, k-lifting an expander graph via random matchings results in a good expander [14, 17]. We are

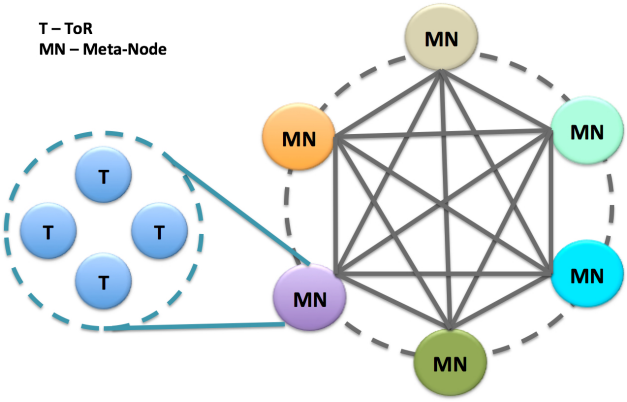


Figure 2: An Xpander topology sketch

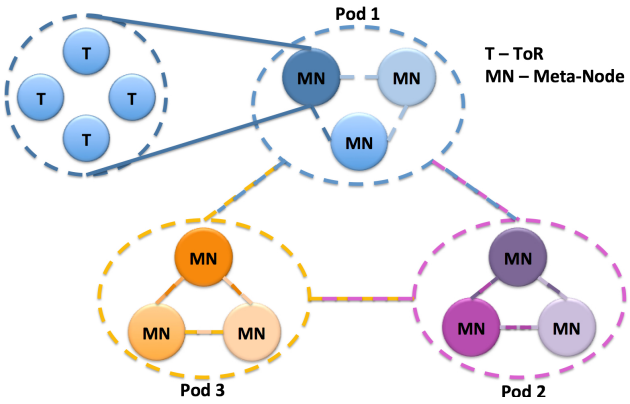


Figure 3: Division of an Xpander into Xpander-pods.

unaware of schemes for de-randomizing k-lifts for $k > 2$. However, we show empirically that k-lifts for $k > 2$ can also be derandomized (see [5]).

Xpander's network topology. To construct a d-regular Xpander network, where each node (vertex) represents a top-of-rack (ToR) switch, and d represents the number of ports per switch used to connect to other switches (all other ports are connected to servers within the rack), we do the following: start with the complete d-regular graph on $d+1$ vertices and repeatedly lift this graph in a manner that preserves expansion.

Although lifting a graph (at least) doubles the number of nodes, we show in Section 4.3 how Xpander topologies can be incrementally grown (a single node at a time) to any desired number of nodes while retaining good expansion.

Selecting the "right" Xpander. Today's rigid datacenter network topologies (e.g., fat trees) are only defined for very specific combinations of number of nodes (switches) n and per-node degree (port count) d. An Xpander, in contrast, can be generated for virtually any combination of n and d by varying the number of ports used for inter-switch connections (and, consequently, for switch-to-server connections), or through the execution of different sequences of k-lifts. The choice of specific Xpander depends on the objectives of the datacenter designer, e.g., minimizing network equipment (switches/links) while maintaining a certain level of performance. See Section 5 for a few more details. To select the "right" Xpander, the datacenter architect can play with Xpander's construction parameters to generate several Xpander networks of the desired size, number of servers supported, etc., and then measure their performance, in terms of expansion[2] and throughput, to identify the best candidate.

3.2 Xpander's Logical Organization

Observe that, as described in Figure 2, an Xpander network can be regarded as composed of multiple "meta-nodes" such that (1) each meta-node consists of the same number of ToRs, (2) every two meta-nodes are connected via the same number of links, and (3) no two ToRs within the same meta-node are directly connected. Each meta-node is, essentially, the group of nodes which correspond to one of the original $d+1$ nodes. Also, an Xpander can naturally be divided into smaller Xpanders ("Xpander-pods"), each of the form depicted in Figure 2, such that each pod is simply a collection of meta-nodes. See Figure 3 for an illustration of an Xpander with 9 meta-nodes (i.e., $d = 8$), divided into 3 equal-sized pods. Observe that division of an Xpander into Xpander-pods need not be into pods of the same size.

We show in Section 7 how this "clean" structure can be leveraged to tame cabling complexity.

3.3 Routing and Congestion Control

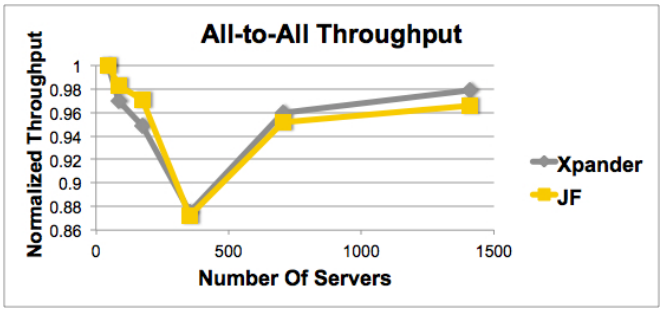
To exploit Xpander's rich path diversity, traditional routing with ECMP and TCP congestion control are insufficient [5]. Xpander thus, similarly to [46], employs multipath routing via K-shortest paths [51, 16] and MPTCP congestion control [49]. K-shortest paths can be implemented in several ways, including OpenFlow rule matching [34], SPAIN [35], and MPLS tunneling [42].

4. NEAR-OPTIMAL PERFORMANCE

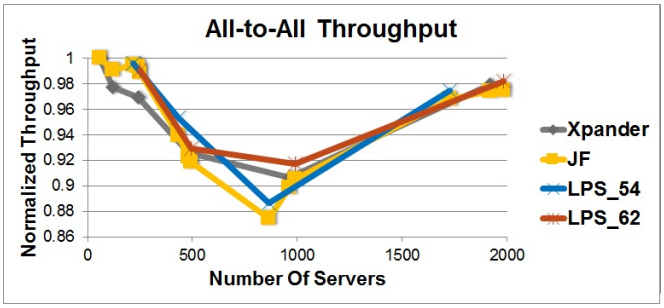
We show that Xpander achieves near-optimal performance in terms of throughput and bisection bandwidth guarantees, robustness to traffic variations, resiliency to failures, incremental expandability, and path lengths. Both our simulations and theoretical results benchmark Xpander against a (possibly unattainable) theoretical upper bound on performance for *any* datacenter network. To benchmark also against the state-of-the-art, we show that Xpander matches the near-optimal performance of random datacenter architectures, demonstrated in [46, 45]. We will later explain how Xpander's design mitigates the deployment challenges facing randomly networked datacenters (Section 7).

We note that Xpander's near-optimal performance benefits are derived from utilizing a nearly optimal expander network topology [9]. Indeed, our results for Xpander's performance extend to all expander datacenters whose d-regular network topology exhibits near-optimal edge expansion (i.e.,

[2]Importantly, computing edge expansion is, in general, computationally hard [28], yet edge expansion can be approximated via the tractable notion of spectral gap [24].

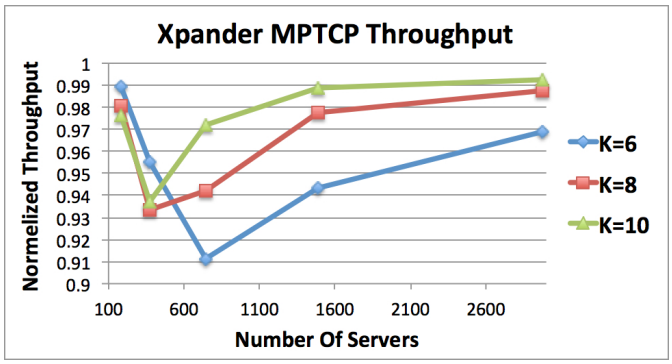


(a) $k = 14$. There are 4 servers placed under each switch

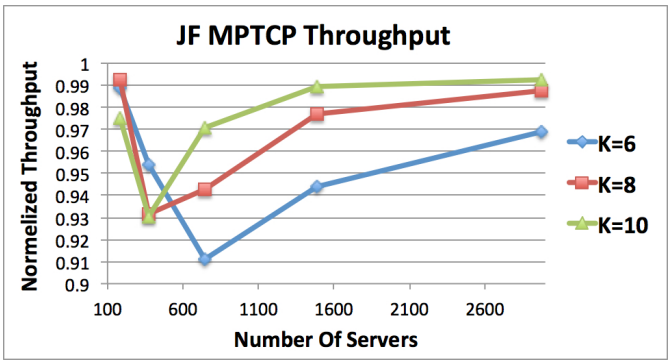


(b) $k = 18$. There are 4 servers placed under each switch.

Figure 4: Results for all-to-all throughput



(a) Xpander



(b) Jellyfish

Figure 5: Results for K-Shortest & MPTCP with $K = \#Subflows$. 6 servers placed under each 36-port switch.

edge expansion close to $\frac{d}{2}$), such as random networks and algebraic constructions of expanders [30]. To simplify exposition, our discussion below focuses on Xpander and Jellyfish [46]. We refer the reader to [5, 48] for more results for other expander datacenters.

4.1 Near-Optimal Throughput

4.1.1 Simulation Results

Simulation framework. We ran simulations on Xpander networks for many choices of number of nodes n and node-degree d. The node degree d in our simulations refers only to the number of ports of a switch used to connect to other switches and not to switch-to-server ports. We henceforth use k to refer to the total number of ports at each switch (used to connect to either other switches or servers). We tested every even degree in the range 6-30 and up to 600 switches (and so thousands of servers) using a flow-level simulator described below. As our simulations show the exact same trends for all choices of parameters, we display figures for selected choices of n and d. To validate that Xpanders indeed achieve near-optimal performance, we benchmarked Xpanders also against Jellyfish's performance (averaged over 10 runs). We also simulate large Xpander networks, the largest supporting 27K servers, using the MPTCP packet simulator [2].

We compute the following values for every network topology considered: (1) the maximum all-to-all throughput, that is, the maximum amount of flow-level traffic α that can be concurrently routed between every two switches in the network without exceeding link capacities (see formal definition in Section 4.1.2); (2) the flow-level throughput under skewed traffic matrices (elephants and mice); and (3) the throughput under K-shortest paths [51] combined with MPTCP. Intuitively, (1)+(2) capture the maximum flow achievable when the full path diversity of Xpanders and Jellyfish can be exploited, whereas (3) captures the packet-level performance under Xpander's routing and congestion control. Thus, our simulations quantify both the best (flow-level) throughput achievable *and* how closely Xpander's routing and congestion control protocols approach this optimum.

To compute (1)+(2), our simulations ran the CPLEX optimizer [1] on a 64GB, 16-core server. Our simulation framework is highly-optimized, allowing us to investigate datacenter topologies with significantly higher parameter values (numbers of switches n, servers, and per switch port counts d) than past studies. To compute the throughput under K-shortest paths and MPTCP we use the MPTCP packet-level simulator [2]. We later validate these results using the mininet network emulator [27] (Section 5.3).

Results. Figure 4 describes our representative results for all-to-all throughput (the y-axis) as a function of the number of servers in the network (the x-axis) when the switch's inter-switch degree (ports used to connect to other switches) is $d = 10$ (left) and $d = 14$ (right). The results are normalized by the theoretical upper bound on the throughput of *any* network [45]. To show that our results are not specific to Xpander of Jellyfish, but extend to all network topologies with comparable edge expansion, Figure 4 also plots the performance of 2-lifts of the algebraic construction of expanders due to Lubotzky et al. [30], called "LPS". To evaluate LPS-based expanders of varying sizes, we generated an LPS expander and then repeatedly 2-lifted it to generate larger expanders (the subscript in the figure specifies the number of nodes in the initial LPS expander, before 2-lifting). More results for other expander datacenters can be found in [5].

Clearly, the achieved performance for all evaluated expander datacenters is essentially identical and is close to the (possibly unattainable) theoretical optimum. Note that there is a dip in the performance, but as explained in [45], this is a function of how the upper bound, which at this point becomes unattainable, is calculated. Even here, however, both Xpander and Jellyfish remain fairly close to the (unattainable) upper bound. We show in the subsequent sections that this level of performance is above that of both today's datacenters and Slim Fly [8].

We also measured, using the MPTCP packet-level simulator [2], the average per-server throughput (as a percentage of the servers' NIC rate) for different choices of parameter K in K-shortest paths. Specifically, we measured (1) the throughput when the number of MPTCP subflows is 8 (the recommended value [46, 49]), and (2) the throughput when the number of subflows equals K. We present our representative results in Figure 5, where we use $d = 30$ and up to 496 switches (or nearly 3K servers). The results for other choices of d and n exhibit the same trends. Our results show that when $K \geq 6$ and the number of MPTCP subflows equals K, the server average throughput is very close to its maximum outgoing capacity.

We also evaluated the throughput of Xpander for skewed traffic matrices, where each of T randomly-chosen pairs of nodes wishes to transmit a large amount of traffic, namely β units of flow, and all other pairs only wish to exchange a single unit of flow (as in the all-to-all scenario). We simulated this scenario for network size $n = 250$, every even $d = 2, 4, \ldots, 24$, and all combinations of $T \in \{1, 6, 11, \ldots, 46\}$ and $\beta \in \{4, 40, 400\}$.

For each choice of parameters, we computed the network throughput α, that is, the maximum fraction of each traffic demand that can be sent through the network concurrently without exceeding link capacities. The results are again normalized by a simple theoretical (and unattainable) upper bound on *any* network's throughput for these traffic demands (calculation omitted). Our simulation results for skewed traffic matrices show that the throughput achieved by Xpander is almost always (over 96% of results) within 15% of the optimum throughput, and typically within $5-10\%$ from the theoretical optimum. See Table 1 for a breakdown of the results. We use the MPTCP simulator to compare Xpander to fat trees in Section 5, showing that Xpander provides the same level of performance with much fewer switches.

Distance from Optimum	Xpander	JellyFish
throughput<80%	<1%	<1%
80% $\leq$ throughput <85%	2.3%	2.3%
85% $\leq$ throughput <90%	16.14%	16.14%
90% $\leq$ throughput <95%	44.48%	48.03%
95% $\leq$ throughput	36.61%	32.67%

Table 1: Distance of throughput from the (unattainable) optimum for various combinations of β, T, d.

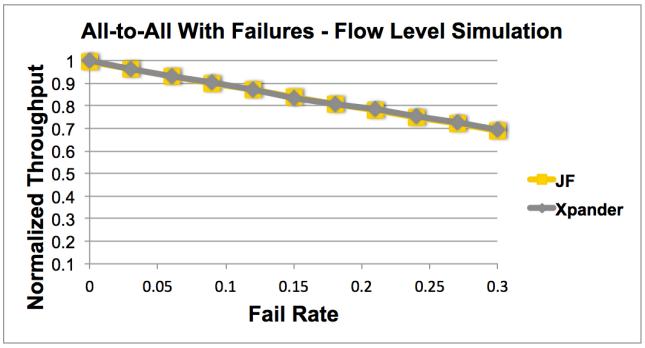


Figure 6: Throughput under link failures.

4.1.2 *Theoretical Results*

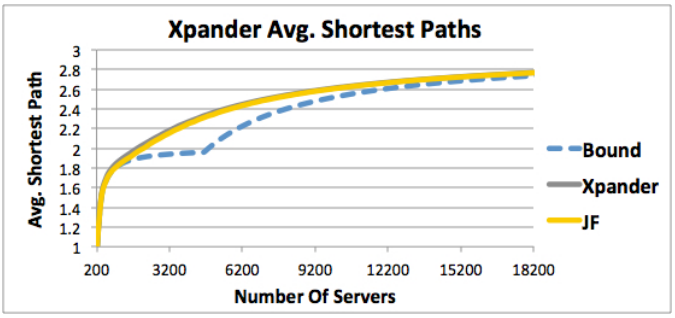
Near-optimal bisection bandwidth. Recall that bisection bandwidth is the minimum number of edges (total capacity) traversing a cut whose sides are of *equal* size [52], or formally $\min_{S:|S|=\frac{n}{2}} |\partial(S)|$ using the notation from Section 2. Since in an expander intuitively *all* cuts are large, not surprisingly Xpander indeed achieves near-optimal bisection bandwidth. Note that the bisection bandwidth of any d-regular graph is at most $\frac{n}{2} \cdot \frac{d}{2} = \frac{nd}{4}$.[3]

THEOREM 4.1. *An Xpander graph on n vertices has bisection bandwidth at least $\frac{n}{2}\left(\frac{d}{2} - O(d^{3/4})\right)$.*

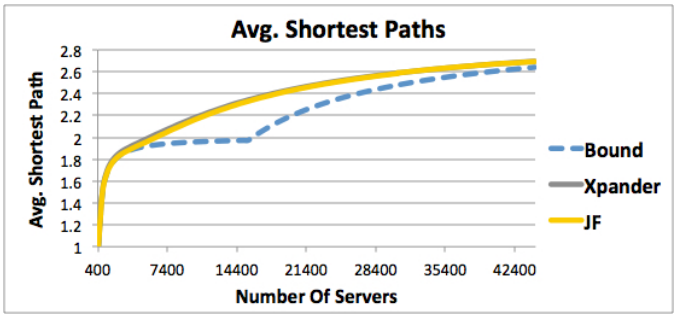
Moreover, the definition of edge expansion of a graph G, $EE(G)$, immediately implies the bisection bandwidth of an arbitrary d-regular graph is at least $\frac{n}{2} \cdot EE(G)$, and so even in an arbitrary d-regular expander the bisection bandwidth must be quite high.

Near-optimal throughput. We consider the following simple fluid-flow model of network throughput [25]: A network is represented as a capacitated graph $G = (V, E)$, where vertex set V represents (top-of-rack) switches and edge set E represents switch-to-switch links. All edges have a capacity of 1. A *traffic matrix* T specifies, for every two vertices (switches) $u, v \in V$, the total amount of requested flow $T_{u,v}$

[3]This is what is achieved by a random bisection and thus the worst bisection is no better.



(a) $k = 32$. There are 8 servers placed under each switch.



(b) $k = 48$. There are 12 servers placed under each switch.

Figure 7: Results for avg. path length

from servers connected to u to servers connected to v. The *network throughput* under traffic matrix T is defined as the maximum value α such that $\alpha \cdot T_{u,v}$ flow can be routed *concurrently* from each vertex u to each vertex v without exceeding the link capacities. For a graph G and traffic matrix T, let $\alpha(G, T)$ denote the throughput of G under T. We refer to the scenario in which $T_{u,v} = 1$ for every $u, v \in V$ (i.e. every node aims to send 1 unit of flow to every other node) as the "*all-to-all setting*". We will slightly abuse notation and let $\alpha(G)$ denote the throughput of G in the all-to-all setting.

We present several simple-to-prove results on the throughput guarantees of expander datacenters. Beyond accounting for Xpander's good performance, these results also account for the good performance of other expander datacenters, e.g., Jellyfish and Slim Fly. While possibly folklore, to the best of our knowledge, these results do not appear to have been stated or proven previously. We thus include them here for completeness.

THEOREM 4.2. *In the all-to-all setting, the throughput of a d-regular expander G on n vertices is within a factor of $O(\log d)$ of that of the throughput-optimal d-regular graph on n vertices.*

The next two results, when put together, show that expanders (and so also Xpander) are, in a sense, the network topology most resilient to adversarial traffic scenarios.

THEOREM 4.3. *For any traffic matrix T and d-regular expander G on n vertices, $\alpha(G, T)$ is within a factor of $O(\log n)$ of that of the throughput optimal d-regular graph on n vertices with respect to T.*

THEOREM 4.4. *For any d-regular graph G on n vertices, there exists a traffic matrix T and a d-regular graph G^* on n vertices such that $\alpha(G^*, T) \geq \Omega(\log_d n) \cdot \alpha(G, T)$.*

4.2 Near-Optimal Resiliency to Failures

It is easy to prove that in any d-regular expander datacenter, the number of edge-disjoint paths between any two vertices is exactly d [5]. Since this is the maximum possible number of such paths in a d-regular graph, Xpander's network topology provides optimal connectivity between any two communicating end-points and can thus withstand the maximum number of link-failures (specifically, $d - 1$) without disconnecting two switches.

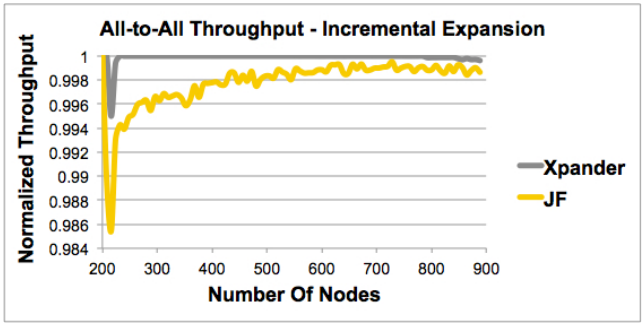


Figure 8: Throughput under incremental expansion for $k = 32$, incrementally adding 1 switch and 8 servers at each step.

We compute the all-to-all server-to-server throughput in Xpander and Jellyfish after failing X links uniformly at random, where X ranges from 0% to 30% in increments of 3%. We repeated this for Xpander networks of many sizes and node-degrees. Figure 6 shows our (representative) results for Xpander and Jellyfish of 708 servers and 236 14-ports switches (with one switch port left unused). As shown in the figure, the throughput of Xpander degrades linearly with the failure rate. We show in Section 5 through both flow-level and packet-level simulations that the throughput of Xpanders indeed dominates that of fat trees (both with and without failures).

Intuitively, this linear dependence on the failure rate is natural. If the probability of link failure is p, then after failures the graph is similar to a $((1-p)d)$-regular Xpander. This is because for each cut S the number of edges across it ($|\partial(S)|$) before failure was large, owing to G being an expander, and so after failure the number of edges across the cut is tightly concentrated around its expectation, which is $(1-p)|\partial(S)|$.

4.3 Incremental Expandability

Companies such as Google, Facebook and Amazon constantly expand existing datacenters to meet ever-growing de-

mands. A significant advantage of random graphs (i.e., Jellyfish [46]) over traditional datacenter topologies (e.g., fat trees) is the ability to incrementally expand the network without having to leave many ports unused etc.

We present a deterministic heuristic for incrementally expanding a d-regular expander datacenter with few wiring changes when adding a new node (ToR): To add a new node to the datacenter, disconnect $\frac{d}{2}$ existing links and connect the d incident nodes (ToRs) to the newly added node (recall that d is the number of ports used to connect to other switches). Observe that this is indeed the minimal rewiring needed as at least $\frac{d}{2}$ links must be removed to "make room" for the new switch. The key challenge is selecting the links to remove. Intuitively, our heuristic goes over all links and quantifies the loss in edge expansion from removing each link, and then removes the $\frac{d}{2}$ links whose removal is the least harmful in this regard. Importantly, since computing edge expansion is, in general, computationally hard [28], our heuristic relies on the tractable notion of spectral gap [24], which approximates the edge expansion. We refer the reader to [5] for a technical exposition of this heuristic.

We compute the all-to-all throughput of topologies that are gradually expanded from the complete d-regular graph using our deterministic incremental growth algorithm, and compare them to the theoretical upper bound on the throughput of *any* d-regular datacenter network.

Figure 8 shows the all-to-all throughput results for an Xpander with 32-port switches. At each size-increment-step, one switch (and 8 servers) is added, thus gradually growing the datacenter network from 200 servers to 900 servers. As shown in Figure 8, all Xpander datacenter networks obtained in this manner achieve near-optimal throughput. We compare Xpander against the incremental growth of Jellyfish, as presented in [46].

4.4 Short Path-Lengths and Diameter

As shown in Figure 7, all evaluated Xpander (and Jellyfish) networks exhibit the same average shortest path lengths and are, in fact, usually within 5% of the lower bound on average path length in [12]. (Results for many other choices of n and d lead to the same conclusion.) Thus, Xpander effectively minimizes the average path length between switches.

A straightforward argument (omitted) shows that $\lceil \log_d n \rceil$ is a lower bound on the diameter of a d-regular graph. All Xpanders evaluated are within just 1 hop from this theoretical lower bound. See [5] for an exposition of our results for network diameter.

5. XPANDER *VS.* FAT TREE

We next detail the significant performance benefits of Xpander datacenters over fat trees. We show that Xpander can support the same number of servers as a fat tree at the same (or better) level of performance with only about $80 - 85\%$ of the switches. We also show that Xpander is more resilient to failures.

fat tree Degree	#Switches	#Servers	Throughput
8	80%	100%	121%
10	100%	100%	157%
12	80.5%	100%	103%
14	96%	103%	122%
16	80%	100%	120%
18	90%	100%	137%
20	80%	100%	118%
22	89%	102%	121%
24	80%	100%	111%

Table 2: Xpanders vs. fat trees (FT), flow-level simulations. Percentages are Xpander/FT.

5.1 Better Performance, Less Equipment

We examine uniform fat tree topologies for every even degree (number of ports per switch) between 8 and 24. Unlike a fat tree, which is uniquely defined for a given switch port-count, Xpander gives the datacenter designer much greater flexibility (see discussion in Section 3.1). We identify, for each fat tree in the above range, an Xpander with much fewer (roughly 80-85%) switches that supports the same number of servers with at least the same level of server-to-server throughput.[4] See details in Table 2.

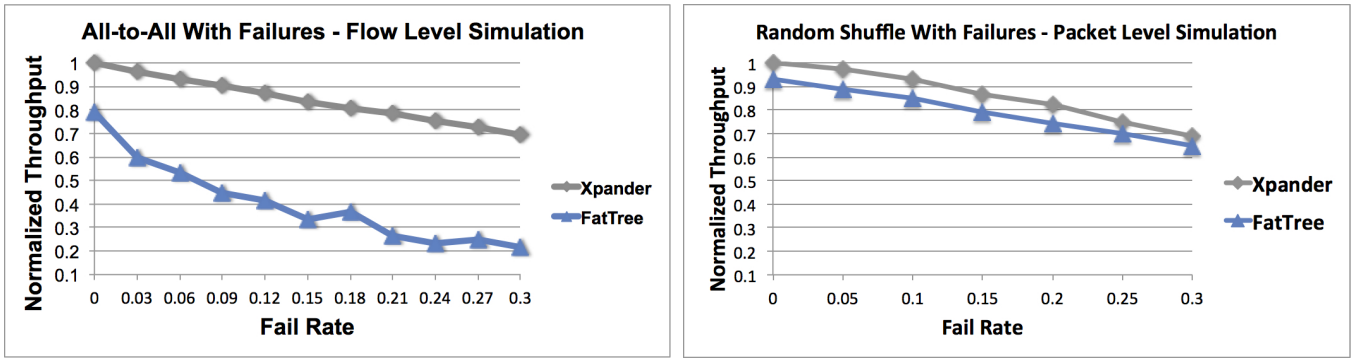
5.2 Simulation Results

Throughput. We evaluate the all-to-all throughput of Xpander and fat trees. We show in Table 2 the all-to-all server-to-server throughput in fat trees and Xpanders without any link failures. As can be seen in Table 2, Xpander is able to achieve comparable, if not better, server-to-server throughput than fat trees *even* with as few as 80% of the switches. Table 3 shows the results of extensive packet based-simulations using the MPTCP network simulator for Xpander and fat trees with $k = 32$ and $k = 48$, containing 8K and 27K servers, respectively. Again, Xpander networks achieve similar or better performance to that of fat trees, with significantly fewer switches.

To explore how Xpander and fat trees compare for other traffic patterns, we also simulate a fat tree of 8K servers (i.e., $k = 32$) and its matching Xpander under the following "Many-to-One" scenario using the MPTCP packet level simulator. We randomly select 10% of the servers as our destinations and for each such server we select at random x% of the servers to generate traffic destined for that server, where x=1%, 1.5%, 2%, 2.5% and 3% (1% is roughly 80 servers). Table 5 presents the averaged results of 4 such simulations. We conclude that, once again, Xpander provides the same level of performance with less network equipment.

Robustness to failures. We compute the all-to-all server-to-server throughput in fat trees and Xpanders after failing X links uniformly at random, where X ranges 0% to 30% in

[4]We now consider server-to-server all-to-all throughput and not switch-to-switch all-to-all throughput, since in a fat tree not all switches originate traffic.



(a) d=14

(b) d=32

Figure 9: Server-to-server throughput degradation with failures. All-to-All flow level (on the left) and Random-Shuffle packet level (on the right

Tested Topology	Random Shuffle		One-to-Many		Many-to-One		Big-and-Small	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
Xpander	19.66	58.86	79.52	104.03	70.09	90.88	28.66	120.21
FatTree (TCP+ECMP)	26.7	102.86	80.72	89.94	80.79	91.51	42.72	220.1
FatTree (MPTCP+ECMP)	17.94	105.71	78.18	138.5	69.56	91.31	31.75	180.64

Table 4: Xpander and fat trees under various traffic matrices. Values are given in seconds and indicate the average finishing time for transmission.

Fat Tree Degree	#Switches	#Servers	Packet Simulation Throughput
32	90%	98.5%	110%
48	88%	100%	102%

Table 3: Xpanders vs. fat trees, packet-level simulation results. Percentages are Xpander/FT.

Percentage Of Servers Routing To Each Destination	Packet Simulation Throughput
1%	99.6%
1.5%	99.3%
2%	101%
2.5%	103%
3%	103%

Table 5: Xpanders vs. fat trees, 10% of the servers are selected as destinations. Percentages for throughput are Xpander/FT.

increments of 3%. We repeated this for fat trees of all even degrees in the range 8-24 and the corresponding Xpander networks from Table 2. Figure 9(a) describes our (representative) results for fat trees of degree $k = 14$ (vs. Xpander). We further simulate a fat tree with $k = 32$, containing 1280 switches and 8192 servers, against an Xpander containing 90% of the switches and 98.5% of the servers, under a random-shuffle permutation matrix after failing S links uniformally at random, where S ranges from 0% to 30% in increments of 5%. The results of this simulation are described in Figure 9(b). Both results (flow-level and packet-level alike) show that Xpander indeed exhibits better resiliency to failures than fat tree. We note that the smaller gap between Xpander and fat tree in the MPTCP simulations can be explained by the fact that the per switch degree is higher and so naturally routing is less affected by failures.

5.3 Xpander Emulation

To show that an Xpander with significantly less switches can achieve comparable performance to fat trees, we used mininet [27] and the RipL-POX [4] controller to simulate fat tree [6] networks under various workloads, and for two routing & congestion control schemes: (1) ECMP with TCP and (2) K-shortest-paths with $K = 8$ and MPTCP with 8 subflows. We also simulated Xpanders for the same workloads under K-shortest-paths with $K = 8$ and MPTCP with 8 subflows. These simulations were performed on a VM running Ubuntu 14.04 with MPTCP kernel version 3.5.0-89 [11]. We chose, for compute and scalability constraints, to evaluate a fat tree network of degree 8, which contains 80 switches and 128 servers. We tested against this fat tree topology the corresponding Xpander datacenter from Table 2, which contains only 64 switches and the same number of servers. All links between switches in both networks are of capacity 1Mbps.

The workloads considered are: (1) Random Shuffle, where the 128 servers are divided into two halves, and each member of the first half sends a 1Mb file to a (unique) randomly chosen member of the other half; (2) One-to-Many, where 4 servers are randomly selected and these servers send a

#Ports per Switch (Sw2Sw / Total)	#Switches (XPNDR / SF)	#Servers (XPNDR / SF)	Bisection Bandwidth	Cost per node	Power per node	Expansion
5 / 8	18 / 18 (100%)	54 / 54 (100%)	86%	123%	114%	77%
7 / 11	48 / 50 (96%)	192 / 200 (96%)	67%	79%	81%	72%
11 / 17	96 / 98 (98%)	576 / 588 (98%)	98%	106%	104%	84%
17 / 26	234 / 242 (96%)	2,106 / 2,178 (96%)	102%	102%	100%	92%
19 / 29	340 / 338 (104%)	3,400 / 3,380 (104%)	103%	95%	94%	96%
25 / 38	572 / 578 (99%)	7,436 / 7,541 (98%)	109%	95%	94%	102%
29 / 44	720 / 722 (99%)	10,800 / 10,830 (99%)	118%	104%	102%	103%
35 / 51	1080 / 1058 (102%)	17,280 / 16,928 (102%)	119%	102%	100%	101%
43 / 65	1672 / 1682 (99%)	36,784 / 37,004 (99%)	117%	99%	96%	107%
47 / 71	1920 / 1922 (99%)	46,080 / 46,128 (99%)	122%	103%	101%	108%
55 / 83	2688 / 2738 (96%)	75,264 / 76,664 (98%)	119%	91%	88%	112%

Table 6: Xpander vs. Slim-Fly. The first column specifies the number of ports per switch used for switch-to-switch and the total port count (for both Xpander and SF). Percentages are Xpander/SF.

Scenario	Min	Max	Average
Random Shuffle	234%	83%	135%
One-to-Many	94%	138%	123%
Many-to-One	115%	86%	92%

Table 7: Results for the physical simulations, all values are the averaged value of Xpander/FatTree.

1Mb file to 10 other randomly chosen (unique) hosts; (3) Many-to-One, where 40 different servers send 1Mb file each to 4 servers (10 sending servers per each receiving server); and (4) Big-And-Small, which is similar to Random Shuffle, only in addition each of 8 randomly chosen servers sends a 10Mb file to a unique other server. Our results are summarized in Table 4. Observe that even with 80% of the switches, Xpander provides comparable or better performance.

5.4 Implementation on a Network Testbed

To validate the above findings, we use OCEAN [3], an SDN-capable network testbed composed of 13 Pica8 Pronto 3290 switches with 48 ports eac. We used the OCEAN platform to experimentally evaluate two datacenter networks: (1) a fat tree composed of 20 4-port switches, of which 8 are top-of-rack switches, each connected to 2 servers (16 servers overall), and (2) an Xpander consisting of 16 4-port switches, each connected to a single server (again, 16 servers overall). Observe that while both networks support the same number of servers, the number of switches in the Xpander is 80% of the number of switches in the fat tree (16 vs. 20). Similarly, the number of links connecting switches to other switches in the Xpander is 75% that of the fat tree (24 vs. 32). Note, however, that the network capacity *per server* in Xpander is much higher as the number of ToRs is higher (16 vs. 8), and each ToR is connected to less servers (1 vs. 2) and to more other switches (3 vs. 1). Thus, intuitively, the Xpander can provide comparable or better performance with less network equipment. Our experiments on OCEAN confirm this intuition.

ECMP routing and TCP are used to flow traffic in the fat tree, whereas for the Xpander we use K-Shortest Paths, with $k = 3$, and MPTCP with 3 subflows (and so on each of the 3 distinct paths between a source and a destination there is a separate MPTCP subflow). We evaluate the min, max, and average throughput under three different traffic scenarios: (1) random shuffle, in which every server routes to a single random destination, (2) many-to-one, in which a randomly chosen server receives traffic from the other 15 servers, and (3) one-to-many, in which a randomly chosen server sends traffic to all other servers. To compute the throughput for two communicating servers, an "endless" flow between these servers is generated using of an iperf client and server and the averaged throughput is computed after 10 minutes. Our results for each simulation setting are averaged over 5 independent runs.

Table 7 shows our results for the above three traffic scenarios (the rows) and for min, max, and average throughput (the columns), where % are Xpander/FT. Xpander achieves comparable or better results in each of the evaluated scenarios (again, using only 80% of the network equipment).

6. XPANDER *VS.* SLIM FLY

We also contrast Xpander with Slim Fly, a recent proposal from the world of high-performance computing (HPC). SF leverages a graph-theoretic construction of low-diameter graphs [33] (either diameter of 2, the situation most explored in [8], or diameter of 3).

Intuitively, by decreasing the diameter, less capacity is wasted when sending data, and hence overall performance is higher. Indeed, [8] shows that SF outperforms existing and proposed datacenter architectures, but performs worse than random topologies, e.g., in terms of bisection bandwidth and resiliency to failures.

When comparing SF to Xpander, we first note that the low diameter of SF imposes an extremely strict condition on the relationship between the per node degree d and the number of nodes n: by requiring diameter 2, SF requires

$d \geq \Omega(\sqrt{n})$. (Importantly, d here represents only the ports used to connect a switch to other switches, and so, to support servers, the actual port count must be even higher.) This imposes a strict limit on the scalability of Slim Fly. Xpander topologies, on the other hand, exist for essentially any combination of n and d and, in particular, can be used for arbitrarily large datacenters even with a fixed per-switch degree d.

Not only is Xpander more flexible than SF in supporting more nodes with smaller degrees, but it exhibits better performance than SF as the network grows, even in the degree/node regimes in which SF is well-defined. We used the simulation framework published in [8] to compare SF to Xpander in terms of performance and costs. The METIS partitioner [26] was used for approximating bisection bandwidth (as in [8]) and the code from [8] for cost and power consumption analysis (using the switch/cable values in [8]). We also computed the expansion for both categories of graphs using spectral gap computation, which approximates edge expansion. See our results for Xpanders in Table 6 and for Jellyfish in [5].

Our findings suggest that, in fact, the good performance of SF can be attributed to the fact that it is an expander datacenter. We back these empirical results with the new theoretical result presented in Section 2, which shows that a Slim-Fly network of degree d and diameter 2 has edge expansion at least $\frac{d}{3} - 1$, and is thus not too far from the best achievalbe edge expansion of $\frac{d}{2}$.

We argue, however, that by optimizing the diameter-size tradeoff, Slim Fly sacrifices a small amount of expansion leading to worse performance than random networks and Xpander as the network gets larger. Our results reveal that for networks with less than 100 switches, SF is a better expander than both Xpander and Jellyfish and exhibits better bisection bandwidth. This advantage is reversed as the network size increases and in turn Xpander and Jellyfish become better expanders. Our results thus both validate and shed light on the results in [8], showing why random graphs (and Xpander) outperform SF for large networks.

We also show (Table 6) that Xpander's cost and power consumption are comparable to those of SF.

7. DEPLOYMENT

We grapple with important aspects of building Xpander datacenters: (1) equipment and cabling costs; (2) power consumption; and (3) physical layout and cabling complexity. We first present a few high-level points and then the results of a detailed analysis of Xpander deployment in the context of both small-scale (container-sized) datacenters and large-scale datcenters. We stress that our analyses are straightforward and, naturally, do not capture all the intricacies of building an actual datacenter. Our aim is to illustrate our main insights regarding the deployability of Xpanders, demonstrating, for instance, its significant deployability advantages over random datacenter networks like Jellyfish.

Physical layout and cabling complexity. As illustrated in Figures 2 and 3, an Xpander consists of several meta-nodes, each containing *the same* number of ToRs and connected to each other meta-node via *the same* number of cables. No two ToRs within the same meta-node are connected. This "clean" structure of Xpanders has important implications: First, placing all ToRs in a meta-node in close proximity (the same rack / row(s)) enables bundling cables between every two meta-nodes. Second, a simple way to reason about and debug cabling is to color the rack(s) housing each meta-node in a unique color and color the bundle of cables interconnecting two meta-nodes in the respective two-color stripes. See the illustration in Figure 2.

Thus, similarly to today's large-scale datacenters [44], Xpander's inherent symmetry allows for the taming of cabling complexity via the bundling of cables. Unlike Xpander networks, whose construction naturally induces the above cabling scheme, other expander datacenters (e.g., Jellyfish [46]) do not naturally lend themselves to such homogeneous bundling of cables. Our strong suspicion, backed by some experimental evidence, is that Jellyfish does not (in general) allow for a clean division of ToRs into equal-sized meta-nodes where every two meta-nodes are connected by the same number of cables (forming a matching), as Xpander does. Unfortunately, proving this seems highly nontrivial. Identifying practical cabling schemes for other expander datacenters is an important direction for further research.

Equipment, cabling costs, and power consumption. As shown in Table 2 and validated in Sections 5.3 and 5.4, Xpanders can support the same number of servers at the same (or better) level of performance as traditional fat tree networks with as few as 80% of the switches. This has important implications for equipment (switch/serve) costs and power consumption. We show, through analyzing cable numbers and lengths, that the reduced number of inter-ToR cables in Xpanders, compared to Clos networks/fat trees, translates to comparable or lower costs. In addition, beyond the positive implications for cabling complexity, the bundling of cables afforded by Xpander also has implications for capex and opex costs. As discussed in [44], manufacturing fiber in bundles can reduce fiber costs considerably (by nearly 40%) and expedite deployment of datacenter fabric by multiple weeks.

Analyzing deployment scenarios. We analyze below two case studies: (1) small clusters ("container datacenters"), and (2) large-scale datacenters.

7.1 Scenario I: Container Datacenters

As several ToR switches can sometimes be placed in the same physical rack along with the associated servers, we distinguish between a Virtual Rack (VR), i.e., a ToR switch and the servers connected to it, and a Physical Rack (PR), which can contain several VRs. Our analyses assume that all racks are 52U (this choice is explained later) and are of standard dimensions, switches are interconnected via Active-Optical Cables (AOC), and servers are connected to ToR switches via standard copper cables.

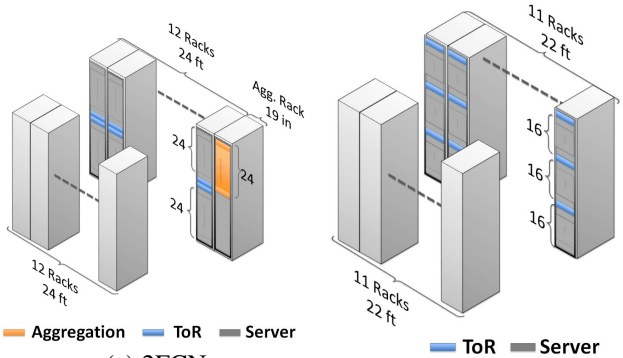
We inspect 2-layered folded-clos network (FCN) of degrees 32 and 48 (see [5]). We select, for each of these two

	#Switches	#Servers	#Physical Racks	#Cables	Cable Length	All-to-All Server to Server Throughput
k=32	42 vs. 48 (87.5%)	504 vs. 512 (98.44%)	11 vs. 11 (100%)	420 vs. 512 (82%)	4.2km vs. 5.12km (82%)	109%
k=48	66 vs. 72 (91.76%)	1,056 vs. 1,152 (91.67%)	22 vs. 25 (88%)	1056 vs. 1152 (91.6%)	10.56km vs. 11.52km (91.6%)	142%

Table 8: Xpander vs. 2-FCN. Percentages are Xpander/2-FCN

Switch Degree	#Switches	#Servers	#Physical Racks	#Cables	Cable Length (m)	Ttl. Space (ft^2)
30 vs. 32 (93.75%)	1,152 vs. 1,280 (90%)	8,064 vs. 8,192 (98.44%)	192 vs. 221 (86.87%)	13,248 vs. 16,348 (80.85%)	220.8k vs. 174k (127%)	3.24k vs. 4k (81%)

Table 9: Xpander vs. fat tree. Percentages are Xpander/fat tree



(a) 2FCN (b) Xpander

Figure 10: A $k = 48$ 2FCN network topology and the matching Xpander

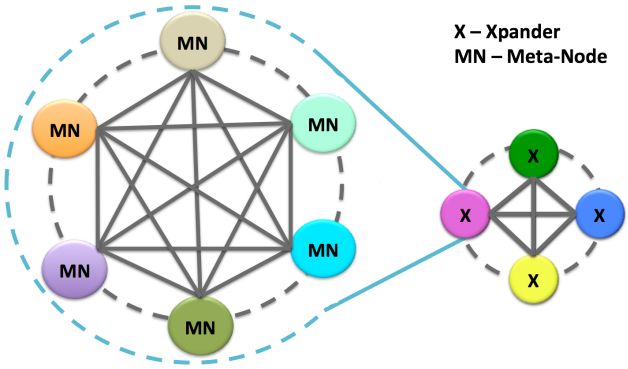


Figure 11: A sketch of an Xpander of Xpander graphs

topologies, a matching Xpander with better performance. We consider 52U racks as these provide the best packing of VRs into PRs for Clos networks. Specifically, 3 VRs fit inside each physical rack for the $k = 32$ Clos network and 2 VRs fit into a PR for $k = 48$ Clos network. The two matching Xpanders are created via a single 2-lift. As each VR in an Xpander contains less servers than that of the comparable 2-FCN network, more VRs can reside in each physical rack (for both degrees). We present the physical layouts of both the 2-FCNs and Xpander networks in Figure 10 and [5], and our analysis in Table 8.

Clearly, the use of less switches in Xpanders immediately translates to a reduction in costs. An Xpander network of switch-to-switch degree d, where each meta-node contains x ToRs, requires $x \cdot \frac{d \cdot (d+1)}{2}$ AOC cables, whereas for a 2-FCN of switch degree (total port count) k there are $\frac{k^2}{2}$ cables. The lower number of AOC cables in Xpanders, assuming 10m-long AOC cables, yields the cable lengths in Table 8. Importantly, the marginal cost of AOC cables greatly decreases with length and so the reduction in number of cables translates to potentially greater savings in costs.

A detailed analysis appears in [5].

7.2 Scenario II: Large-Scale Datacenters

We now turn our attention to large-scale datacenters. Specifically, we analyze the cost of building a uniform-degree fat tree with port-count $k = 32$ per switch (and so of size 1280 switches and 8192 servers) vs. a matching Xpander. We first present, for the purpose of illustration, the physical layout of each network in a single floor. We point out, however, that while deploying a fat tree (or the matching Xpander) of that scale in a single room might be physically possible, this might be avoided in the interest of power consumption and cooling considerations. We hence also discuss large-scale datacenters that are dispersed across multiple rooms/floors.

7.2.1 Single Floor Plan

A fat tree with total port count $k = 32$ per node contains 32 pods and $\frac{d^2}{4} = 256$ core switches, where each pod contains 16 ToRs, 512 servers and 16 aggregation switches, totaling in 8192 servers, 512 ToRs, and 512 aggregation switches. We present a straightfoward, hypothetical floor plan for deploying such a fat tree in [5]. Again, we assume 52U physical racks as this is the most suitable for packing VRs in the fat tree, allowing us to fit 3 VRs in each PR and consequently an entire pod (including the aggregation switches)

in a row of 6 PRs. We can place 2 such pods (rows) inside a hot/cold-aisle containment enclosure, resulting in 16 such enclosures for the entire datacenter. We end up with 12 rows, each containing exactly 18 physical racks (which, in turn, can house 3 pods), and core switches placed in 5 additional physical racks. We assume that, within a pod, 5m AOC cables are used to connect each ToR to its aggregation layer switches. Each of the 2-pod enclosures can be connected to the row of core switches using combination of 10m/15m/20m/25m AOC cables, depending of their proximity.

We compare this fat tree network with an Xpander network of degree 23 ($k = 30$-port switches instead of $k = 32$), constructed using four 2-lifts and another 3-lift. See the side by side comparison of the two networks in Table 9. This specific Xpander houses 8064 servers under 1152 ToRs and consists of 24 meta-nodes, each containing 48 VRs with 7 servers per VR. We present a possible floor plan for deploying this Xpander in [5]. Using 52U racks, 6 VRs can be packed into a physical rack, resulting in a total of 8 racks per meta node. Again, each hot/cold-aisle containment enclosure houses 2 rows, resulting in 16 52U racks (8 in each row). We present the physical layout analysis for both networks in Table 9. See a more detailed analysis in [5].

7.2.2 *Xpander of Xpanders*

So far, our analysis of large-scale Xpanders assumed that the whole datacenter network fits in a single floor. This might not be feasible due to power supply and cooling constraints. To this end, the Xpander must be "broken" into several, dispersed, components. One approach to do so is to place a single Xpander-pod, or several such pods, in a separate floor. Another possible approach is constructing an Xpander network interconnecting smaller Xpander networks, as illustrated in Figure 11: (1) each smaller Xpander will be housed in a single container/room/floor and be constructed as illustrated above; (2) several (higher-degree) switches in each of these Xpanders will be designated as "core switches"; (3) these core switches will then be interconnected through an Xpander overlay network. Since, as evidenced by our results, an Xpander provides near-optimal performance guarantees (throughput, failure resiliency, average path length, etc.), this construction can yield good peformance both within each smaller Xpander and between the Xpander networks.

8. RELATED WORK

Datacenter networks. Datacenters have been extensively researched from many different angles, e.g., throughput optimization [45, 40, 25], failure resiliency [29, 20, 19], and expandability [46, 15]. In particular, many datacenter topologies have been proposed, including Clos networks [6, 21, 36], hypercubes [22, 50], small-world graphs [43], and random graphs [46, 45].

Expanders. Expanders play a key role in a host of applications, ranging from networking to complexity theory and coding. See the survey of Hoory, Linial, and Wigderson [24]. A rich body of literature in mathematics and computer science deals with constructions of expanders, e.g., Margulis's construction [32], algebraic constructions [30], and constructions that utilize the so-called "zig-zag product" [41]. Our construction of the Xpander datacenter topology utilizes the notion of 2-lifting a graph, introduced by Bilu and Linial [9, 31]. Utilizing expanders as network topologies has been proposed in the context of parallel computing and high-performance computing [47, 14, 13, 8], optical networks [39] and also for peer-to-peer networks and distributed computing [38, 37]. Our focus, in contrast, is on datacenter networking and on tackling the challenges that arise in this context (e.g., specific, throughput-related performance measures, specific routing and congestion control protocols, costs, incremental growth, etc.).

Relation to [48]. Our preliminary results on Xpander appeared at HotNets 2015 [48]. Here, we provide a deeper and much more detailed evaluation of the merits of expander datacenters in general, and of Xpander in particular, including (1) implementation and evaluation on the OCEAN SDN testbed [3], (2) comparison of Xpander to Slim-Fly including theoretical results and simulations, (3) many additional simulation results with the MPTCP simulator for Xpander and fat tree, (4) results for path-lengths, diameter and and incremental growth, (5) results for bisection bandwidth of Xpander, and (6) a detailed discussion of Xpander's deployment scenarios.

9. SUMMARY

We showed that expander datacenters offer many valuable advantages over traditional datacenter designs and that this class of datacenters encompasses state-of-the-art proposals for high-performance datacenter design. We suggested practical approaches for building such datacenters, namely, the Xpander datacenter architecture. We view Xpander as an appealing and practical alternative to traditional datacenter designs.

Acknowledgements

We thank Nati Linial for many insightful conversations about expander constructions and Daniel Bienstock for his help in optimizing linear programs for throughput computation. We thank Brighten Godfrey and Ankit Singla for sharing the Jellyfish code with us and for helpful discussions. We thank Soudeh Ghorbani for helping with running experiments on the OCEAN platform. We thank Noga Alon, Alex Lubotzky, and Robert Krauthgamer for useful discussions about expanders. We also thank Jonathan Perry for suggesting Xpander's color coding, presented in Section 7. Finally, we thank our shepherd, Siddhartha Sen, and the anonymous CoNEXT reviewers, for their valuable feedback. The 1st author is supperted by a Microsoft Research Ph.D. Scholarship. The 2nd and 4th authors are supported by the Israeli Center for Research Excellence (I-CORE) in Algorithms. The 1st and 4th authors are supported by the PetaCloud industry-academia consortium. The third author is supported in part by NSF awards 1464239 and 1535887.

10. REFERENCES

[1] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html.

[2] MPTCP Simulator v0.2. http://nets.cs.pub.ro/~costin/code.html.

[3] Ocean cluster for experimental architectures in networks (ocean). http://ocean.cs.illinois.edu/.

[4] RipL-POX, simple datacenter controller build on RipL. https://github.com/brandonheller/riplpox.

[5] Xpander Project Page. http://husant.github.io/Xpander.

[6] AL-FARES, M., LOUKISSAS, A., AND VAHDAT, A. A scalable, commodity data center network architecture. In *SIGCOMM* (2008).

[7] AL-FARES, M., RADHAKRISHNAN, S., RAGHAVAN, B., HUANG, N., AND VAHDAT, A. Hedera: Dynamic flow scheduling for data center networks. In *NSDI* (2010).

[8] BESTA, M., AND HOEFLER, T. Slim Fly: A cost effective low-diameter network topology. In *SC14* (2014).

[9] BILU, Y., AND LINIAL, N. Lifts, discrepancy and nearly optimal spectral gap. *Combinatorica* (2006).

[10] BOLLOBÁS, B. The isoperimetric number of random regular graphs. *Eur. J. Comb.* (1988).

[11] C. PAASCH, S. BARRE, ET AL. Multipath TCP in the Linux Kernel. http://www.multipath-tcp.org.

[12] CERF, V. G., COWAN, D. D., MULLIN, R. C., AND STANTON, R. G. A lower bound on the average shortest path length in regular graphs. *Networks* (1974).

[13] CHONG, F. T., BREWER, E. A., LEIGHTON, F. T., AND KNIGHT, T. F., J. Building a better butterfly: the multiplexed metabutterfly. In *ISPAN* (1994).

[14] CHONG, F. T., BREWER, E. A., LEIGHTON, F. T., AND KNIGHT, T. F., J. Scalable expanders: Exploiting hierarchical random wiring. In *Parallel Computer Routing and Communication*. 1994.

[15] CURTIS, A. R., KESHAV, S., AND LÓPEZ-ORTIZ, A. Legup: using heterogeneity to reduce the cost of data center network upgrades. In *CoNEXT* (2010).

[16] DE QUEIRÓS VIEIRA MARTINS, E., AND PASCOAL, M. M. B. A new implementation of yen's ranking loopless paths algorithm. *4OR* (2003).

[17] FRIEDMAN, J. Relative expanders or weakly relatively ramanujan graphs. *Duke Math. J. 118*, 1 (05 2003), 19–35.

[18] FRIEDMAN, J. *A Proof of Alon's Second Eigenvalue Conjecture and Related Problems*. Memoirs of the American Mathematical Society. American Mathematical Soc., 2008.

[19] GEORGE B. ADAMS, I., AND SIEGEL, H. J. The extra stage cube: A fault-tolerant interconnection network for supersystems. *IEEE Trans. Computers* (1982).

[20] GILL, P., JAIN, N., AND NAGAPPAN, N. Understanding network failures in data centers: measurement, analysis, and implications. In *SIGCOMM* (2011).

[21] GREENBERG, A. G., HAMILTON, J. R., JAIN, N., KANDULA, S., KIM, C., LAHIRI, P., MALTZ, D. A., PATEL, P., AND SENGUPTA, S. Vl2: a scalable and flexible data center network. In *SIGCOMM* (2009).

[22] GUO, C., LU, G., LI, D., WU, H., ZHANG, X., SHI, Y., TIAN, C., ZHANG, Y., AND LU, S. Bcube: a high performance, server-centric network architecture for modular data centers. In *SIGCOMM* (2009).

[23] GUO, C., WU, H., TAN, K., SHI, L., ZHANG, Y., AND LU, S. Dcell: a scalable and fault-tolerant network structure for data centers. In *SIGCOMM* (2008).

[24] HOORY, S., LINIAL, N., AND WIGDERSON, A. Expander graphs and their applications. *Bull. Amer. Math. Soc.* (2006).

[25] JYOTHI, S. A., SINGLA, A., GODFREY, P. B., AND KOLLA, A. Measuring throughput of data center network topologies. In *SIGMETRICS* (2014).

[26] KARYPIS, G., AND KUMAR, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.* (1998).

[27] LANTZ, B., HELLER, B., AND MCKEOWN, N. A network in a laptop: Rapid prototyping for software-defined networks. In *HOTNETS* (2010).

[28] LINIAL, N., LONDON, E., AND RABINOVICH, Y. The geometry of graphs and some of its algorithmic applications. *Combinatorica* (1995).

[29] LIU, V., HALPERIN, D., KRISHNAMURTHY, A., AND ANDERSON, T. F10: A fault-tolerant engineered network. In *NSDI* (2013).

[30] LUBOTZKY, A., PHILLIPS, R., AND SARNAK, P. Ramanujan graphs. *Combinatorica* (1988).

[31] MARCUS, A., SPIELMAN, D. A., AND SRIVASTAVA, N. Interlacing families I: Bipartite ramanujan graphs of all degrees. In *FOCS* (2013).

[32] MARGULIS, G. A. Explicit constructions of expanders. *Problemy Peredači Informacii* (1973).

[33] MCKAY, B. D., MILLER, M., AND ÃĚÂĂIRÃČÂĄÃĚÂĹ, J. A note on large graphs of diameter two and given maximum degree. *Journal of Combinatorial Theory, Series B* (1998).

[34] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. Openflow: Enabling innovation in campus networks. In *SIGCOMM* (2008).

[35] MUDIGONDA, J., YALAGANDULA, P., AL-FARES, M., AND MOGUL, J. C. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *NSDI* (2010).

[36] MYSORE, R. N., PAMBORIS, A., FARRINGTON, N., HUANG, N., MIRI, P., RADHAKRISHNAN, S., SUBRAMANYA, V., AND VAHDAT, A. Portland: a

scalable fault-tolerant layer 2 data center network fabric. In *SIGCOMM* (2009).

[37] NAOR, M., AND WIEDER, U. Novel architectures for p2p applications: The continuous-discrete approach. In *SPAA* (2007).

[38] PANDURANGAN, G., ROBINSON, P., AND TREHAN, A. DEX: Self healing expanders. In *SPAA* (2014).

[39] PATURI, R., LU, D.-T., FORD, J. E., ESENER, S. C., AND LEE, S. H. Parallel algorithms based on expander graphs for optical computing. *Appl. Opt.* (1991).

[40] POPA, L., RATNASAMY, S., IANNACCONE, G., KRISHNAMURTHY, A., AND STOICA, I. A cost comparison of datacenter network architectures. In *CoNEXT* (2010).

[41] REINGOLD, O., VADHAN, S., AND WIGDERSON, A. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *FOCS* (2000).

[42] ROSEN, E., VISWANATHAN, A., AND CALLON, R. Multiprotocol Label Switching Architecture. RFC 3031, 2001.

[43] SHIN, J.-Y., WONG, B., AND SIRER, E. G. Small-world datacenters. In *SoCC* (2011).

[44] SINGH, A., ONG, J., AGARWAL, A., ANDERSON, G., ARMISTEAD, A., BANNON, R., BOVING, S., DESAI, G., FELDERMAN, B., GERMANO, P., KANAGALA, A., PROVOST, J., SIMMONS, J., TANDA, E., WANDERER, J., HÖLZLE, U., STUART, S., AND VAHDAT, A. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (2015), SIGCOMM '15, ACM.

[45] SINGLA, A., GODFREY, P. B., AND KOLLA, A. High throughput data center topology design. In *NSDI* (2014).

[46] SINGLA, A., HONG, C.-Y., POPA, L., AND GODFREY, P. B. Jellyfish: Networking data centers randomly. In *NSDI* (2012).

[47] UPFAL, E. An o(log n) deterministic packet-routing scheme. *J. ACM* (1992).

[48] VALADARSKY, A., DINITZ, M., AND SCHAPIRA, M. Xpander: Unveiling the Secrets of High-Performance Datacenters. In *HOTNETS* (2015).

[49] WISCHIK, D., RAICIU, C., GREENHALGH, A., AND HANDLEY, M. Design, implementation and evaluation of congestion control for multipath TCP. In *NSDI* (2011).

[50] WU, H., LU, G., LI, D., GUO, C., AND ZHANG, Y. Mdcube: a high performance network structure for modular data center interconnection. In *CoNEXT* (2009).

[51] YEN, J. Y. Finding the k shortest loopless paths in a network. *Management Science* (1971).

[52] ZOLA, J., AND ALURU, S. *Encyclopedia of Parallel Computing*. Springer.