Xpander: Unveiling the Secrets of High-Performance Datacenters

Asaf Valadarsky
The Hebrew University of Jerusalem
asaf.valadarsky@mail.huji.ac.il

Michael Dinitz
Johns Hopkins University
mdinitz@cs.jhu.edu

Michael Schapira
The Hebrew University of Jerusalem
schapiram@huji.ac.il

ABSTRACT

Many architectures for high-performance datacenters have been proposed. Surprisingly, recent studies show that datacenter designs with *random* network topologies outperform more sophisticated designs, achieving near-optimal throughput and bisection bandwidth, high resiliency to failures, incremental expandability, high cost efficiency, and more. Unfortunately, the inherent unstructuredness and unpredictability of random designs pose serious, arguably insurmountable, obstacles to their adoption in practice. Can these guarantees be achieved by well-structured, *deterministic* datacenters? We provide a surprising affirmative answer. We show, through a combination of theoretical analyses, extensive simulations, and experiments with a network emulator, that *any* "*expander*" network topology (as indeed are random graphs) comes with these benefits. We leverage this insight to present Xpander, a novel deterministic datacenter architecture that achieves all of the above desiderata while providing a tangible alternative to existing datacenter designs. We discuss challenges en route to deploying Xpander (including physical layout, cabling costs and complexity, backwards compatibility) and explain how these can be resolved.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology

1. INTRODUCTION

The rapid growth of Internet services is placing tremendous demands on datacenters. To meet this challenge, many datacenter network architectures have been proposed [17, 18, 4, 32, 35, 16, 30], ranging from Clos networks [4, 16, 26] to hypercubes [17, 35] and small-world graphs [30]. Surprisingly, despite extensive research on high-performance datacenters, recent studies [32, 31] show that datacenters with *random* network topologies significantly outperform more sophisticated designs, achieving near-optimal throughput and bisection bandwidth, shorter path lengths, higher resiliency to failures, and better cost efficiency. While an important and thought-provoking experiment, the inherent unstructuredness of random graphs makes them hard to reason about (predict, diagnose, etc.) and to build (e.g., in terms of wiring complexity), and so poses serious, arguably insurmountable, obstacles to their adoption in practice. Hence, these findings primarily call for a principled re-inspection of high-performance datacenter design. Can well-structured, *deterministic* datacenter topologies achieve all these benefits?

We set out to answer this fundamental question. Our first step is distilling the properties of random datacenter topologies that make them so attractive. We argue that the benefits of random graphs are not the result of lack of structure but, in fact, the exact opposite: *any* network in the much broader family of *expander graphs* (aka "expanders" [19]) comes with the same benefits and more. Two important implications are that (1) random datacenter architectures (e.g., Jellyfish [32]) are but a single point in a much larger space of high-performance datacenter architectures (as random graphs are a specific class of expanders [8, 13]); and (2) the search for new and improved datacenter architectures is inextricably intertwined with the rich body of research in mathematics and computer science on constructing expander graphs. We leverage these insights to present Xpander, a deterministic datacenter architecture that matches and even surpasses random datacenter designs, while providing a tangible alternative to existing datacenter designs.

1.1 Why Expanders?

Expanders. Intuitively, in an expander 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 formalize these intuitions and

HotNets '15 November 16–17 2015, Philadelphia, PA USA

DOI: http://dx.doi.org/10.1145/2834050.2834059.

present many other benefits of expanders over traditional datacenter architectures. Specifically, we show, through a combination of theoretical analyses, extensive and highly optimized flow-level and packet-level simulations, and experiments with the mininet network emulator that:

Expanders achieve near-optimal throughput and bisection bandwidth. We show that "expander datacenters" achieve near-optimal bisection bandwidth and all-to-all throughput (used to quantify the performance of datacenter topologies in [32, 31, 20]). As random graphs are, in fact, expanders [8, 13], this explains the experimental findings in [32, 31].

Expanders are highly robust to traffic variations. Studies of datacenter traffic patterns reveal tremendous variation in traffic over time [5]. Unfortunately, a network topology that fares well, throughput-wise, in one traffic scenario (e.g., all-to-all) might fail miserably in other scenarios. We show that *all* expanders are robust to variations in traffic. Specifically, *any* expander provides close-to-optimal guarantees with respect to *any* (even adversarially chosen!) traffic pattern. We prove, moreover, that *no* other topology can do much better.

Expanders are highly-resilient to failures. We prove that *any* expander maximizes the number of link-disjoint paths between switches, and so can withstand the maximum number of link-failures before disconnecting two communicating end-hosts. We also show that the performance of expander datacenters degrades much more gracefully with failures than that of fat-trees [4].

Expanders are cost efficient. We show that expander datacenters can, for instance, achieve the same performance as fat-trees with only 80% of the switches or, alternatively, achieve significantly higher throughput with the same equipment as a fat-tree.

Expanders are incrementally expandable. Companies such as Google, Facebook and Amazon constantly expand existing datacenters. We show that, unlike today's rigid datacenters, *any* expander datacenter topology can be incrementally expanded to any size in a *deterministic* manner while remaining an expander. Due to space constraints, there results are deferred to the full version of this paper.

Expanders have shorter average path-lengths and diameters. Our experiments show that all expanders exhibit lower average (shortest) path lengths and diameters than today's datacenters. Again, these results are deferred to the full version of this paper.

1.2 Xpander: A New Datacenter Design

We harness the above insights to propose a new *deterministic* datacenter architecture with all of these benefits, called "Xpander". Xpander relies on deterministic constructions of expanders from graph theory [7].

We grapple with challenges facing the deployment of Xpanders in practice through detailed analysis of various Xpander deployment scenarios (from "container datacenters" to large-scale datacenters). Our analyses provide evidence that Xpanders are realizable with monetary and power consumption costs that are comparable or lower to those of today's prevalent datacenter architectures and, moreoever, that their inherent well-structuredness and order render Xpanders much easier to wire than random datacenter designs.

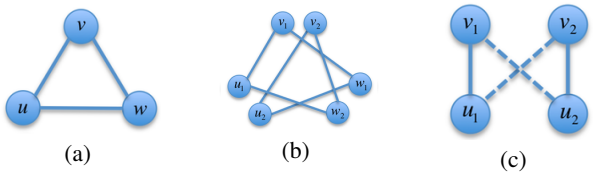


Figure 1: Illustration of 2-Lift

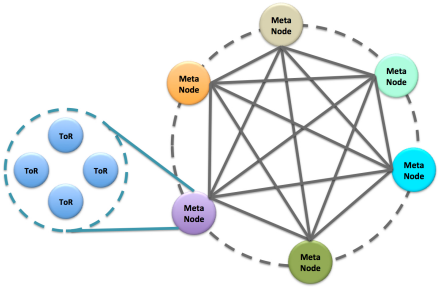


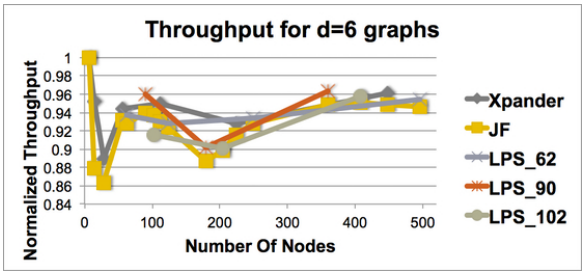
Figure 2: An Xpander topology sketch

2. EXPANDERS AND XPANDERS

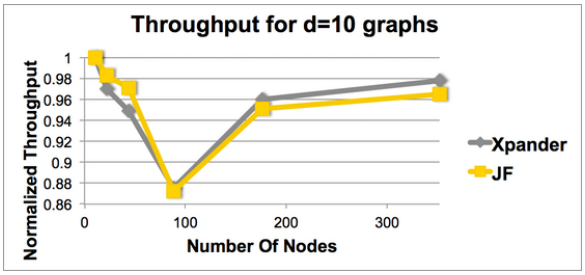
Expanders. Consider an (undirected) graph $G = (V, E)$ (where V and E are the vertex set and edge set, respectively). For any subset of vertices S, let $|S|$ denote the size of S, let $\partial(S)$ denote the set of edges leaving S, and let $|\partial(S)|$ denote the size of $\partial(S)$. Let n denote the number of vertices in V (that is, $|V|$). The *edge expansion* $EE(G)$ of a graph G on n is $EE(G) = \min_{|S| \leq \frac{n}{2}} \frac{|\partial(S)|}{|S|}$. We say that graph G is *d-regular* if each vertex has degree d. We call a d-regular graph G an *expander* if $EE(G) = c \cdot d$ for some constant $c > 0$.

Xpander network topology. Consider the graph G depicted in Figure 1(a). 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). [7] proves that if 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. [7] also shows how these simple random choices can be *derandomized*, i.e., how to achieve the same guarantee in a deterministic manner. The notion of 2-lifting a graph can be extended to k-lifting for arbitrary $k > 0$ in a straightforward manner: create k duplicates of each original vertex and replace each original edge with a k-matching.

To construct a d-regular Xpander network, where each node



(a) d=6



(b) d=10

Figure 3: Results for all-to-all throughput

(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), the following steps are executed: Start with the complete d-regular graph on $d+1$ vertices. Then, repeatedly lift this graph. As illustrated 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. We show in Section 6 how this structure can be leveraged to tame cabling complexity.

Routing and congestion control in Xpander. To exploit Xpander's great path diversity traditional routing with ECMP and TCP congestion control are insufficent. Xpander thus, similarly to [32], employs multipath routing via K-shortest paths [36, 12] and MPTCP congestion control [34]. K-shortest path can be implemented, e.g., via OpenFlow rule matching [24], SPAIN [25], or MPLS tunneling [29].

3. THROUGHPUT ANALYSIS

We show that *any* expander datacenter and, in particular, Xpander, achieves near-optimal throughput and bisection bandwidth guarantees. We defer our results for bisection bandwidth to the full version of this paper and next turn out attention to analyzing throughput directly.

3.1 Throughput Analysis

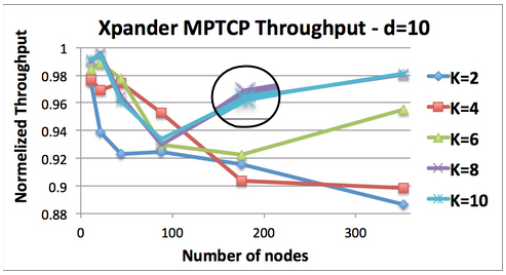
Simulations. All-to-all throughput is the maximum amount of traffic α that can be simultaneously sent between every pair of ToRs without exceeding the link capacities (see formal definition below). We show that all evaluated expanders exhibit close-to-optimal throughput. Figure 3 describes our representative results for all-to-all throughput (the y-axis) as a function of the number of nodes in the network (the x-axis) when the node inter-switch degree (ports used to connect to other switches) is $d = 6$ (left) and $d = 10$ (right). We also ran simulations for many other choices of number of nodes n and node-degree d: every even degree in the range $6-30$ and up to 600 switches (and so thousands of servers). The results are normalized by the theoretical (possibly unattainable) upper bound on the throughput of *a*ny network [31]. We plot, for $d = 6$, the all-to-all throughput of several expanders: (1) random graphs (Jellyfish); (2) Xpander; and (3) 2-lifts of the algebraic construction of expanders due to Lubotzky et al. [23], called "LPS" (the subscript specifies the number of nodes in the initial LPS expander, before 2-lifting). The sudden dip in performance is a byproduct of how the upper bound from [31] is calculated, as explained in [31].

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%

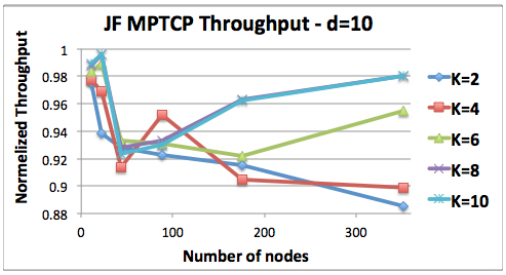
Table 1: Distance of throughput from the (unattainable) optimum for various combinations of β, K, n.

We also evaluated the throughput of Xpander and Jellyfish for "skewed traffic matrices", where each of K 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 $K \in \{1, 6, 11, \ldots, 46\}$ and $\beta \in \{4, 40, 400\}$. We computed, for each choice of parameters, 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 *a*ny network's throughput for these traffic demands (calculation omitted). Our simulation results for skewed traffic matrices show that the throughputs achieved by both Xpander and Jellyfish are almost always (over 96% of results) within 15% of the (unattainable) upper bound on the optimum throughput, and typically within $5-10\%$ from the theoretical optimum. See Table 1 for a breakdown of all the results.

Theory. We consider the following simple fluid-flow model of network throughput [20]: A network is represented as a capacitated graph $G = (V, E)$, where vertex set V represents (ToR) switches, and edge set E represents switch-to-



(a) Xpander



(b) Jellyfish

Figure 4: Results for K-Shortest & MPTCP with $\#Subflows$ fixed at 8 and different values of K

Fat-Tree Degree	#Switches (Xpander)	#Servers (Xpander)	#Throughput (Xpander)
8	80%	100%	121%
10	100%	100%	157%
24	80%	100%	111%

Table 2: Xpanders vs. fat-trees.

switch links. All edges have capacity 1. A *traffic matrix* T specifies, for every two vertices $u, v \in V$, the total amount of requested flow $T_{u,v}$ from u 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* between every two vertices without exceeding 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 that $T_{u,v} = 1$ for every $u, v \in V$ as the "*all-to-all setting*".

We prove that *any* d-regular expander is close-to-optimal (a constant factor away) with respect to all-to-all throughput.

THEOREM 3.1. *In the all-to-all setting, the throughput of any 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.*

Our next two results, when put together, show that expanders are, in a sense, the network topology most resilient to adversarial traffic scenarios.

THEOREM 3.2. *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 3.3. *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 $\alpha(G^*, T) \geq \Omega(\log_d n) \cdot \alpha(G, T)$.*

3.2 Realizing Near-Optimal Flows

We evaluate all-to-all throughput under ECMP (and TCP) for many choices of n and d. Our results (deferred to the full version of this paper) show that while deterministic constructions consistently outperform random graphs (JF) by $\sim$10%, ECMP fails to exploit the great path diversity of expanders. Indeed ECMP utilizes $\sim$80% of the network capacity at best, and only 60-70% most of the time.

We show below, through simulations and experiments with mininet (see Section 4), that one approach to overcoming ECMP's low throughput guarantees is using the K-shortest paths algorithm [36, 12] and MPTCP [34]. We evaluate the all-to-all throughput under K-shortest paths and MPTCP [34]. We measured, for different choices of K, (1) the throughput when the number of MPTCP subflows is 8 (the recommended value [32, 34]), and (2) when the number of subflows is K. To compute the throughput under K-shortest paths and MPTCP we use the MPTCP packet-level simulator [2]. We later (Section 4) validate these results using the mininet network emulator [21]. Our results show that when $K \geq 6$ the throughput is very close to the (possibly unattainable) theoretical upper bound on all-to-all throughput in [31]. We present our results for $d = 10$, averaged over 10 runs for Jellyfish, in Figure 4. The results for other values of d exhibit the same trends. As before, the sudden dip in performance in the figure is due to how the upper bound from [31] is calculated.

4. COST EFFICIENCY

Deciding which specific Xpander to deploy (in terms of size, switch-to-switch port count, etc.) depends on the specific optimization goal, e.g., minimizing the number of switches while achieving the same (or better) performance[1] as a fat-tree, or, alternatively, maximizing the performance-level with the *same* equipment. We examined uniform fat-tree topologies for every even degree (number of ports per switch) between 8 and 24. We identified, for each fat-tree in this range, an Xpander with either much fewer switches, much better server-to-server throughput, or both. See Table 2 for results for $d = 8, 10, 24$. We analyze costs associated with physical layout and cabling in Section 6.

Experiments with mininet. To show that an Xpander with significantly less switches can achieve performance comparable to a fat-tree, we used mininet [21] and the RipL-POX [3]

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

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

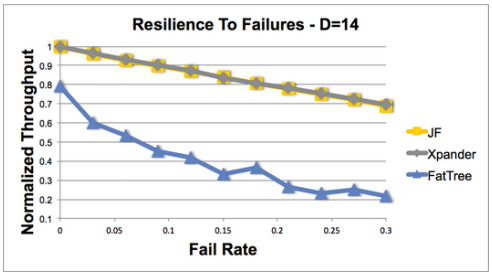
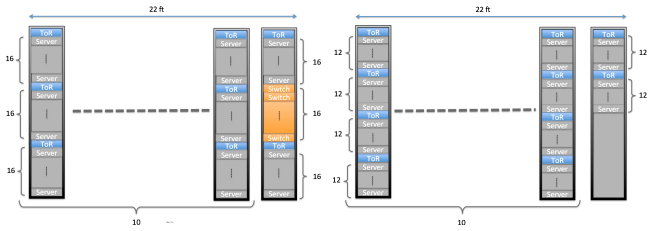


Figure 5: Server-to-server throughput degradation with failures ($d = 14$).



(a) 2-FCN (b) Xpander

Figure 6: A $d = 32$ 2-FCN network topology and the matching Xpander

controller to simulate both 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. These simulations were performed on a VM running Ubuntu 14.04 with MPTCP kernel version 3.5.0-89 [1]. 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 1Mb file to 10 other randomly chosen (unique) hosts, (3) Many-to-One, the reversed scenario, 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 3. Observe that even with 80% of the switches, Xpander provides comparable or better performance.

5. ROBUSTNESS TO FAILURES

We prove (proof omitted) that expander datacenters provide optimal connectivity guarantees, in the sense that in any d-regular expander with edge expansion at least 1, any two vertices are connected by exactly d edge-disjoint paths.

We compute the all-to-all server-to-server throughput in fat-trees, random graphs, and Xpanders after failing X links uniformly at random, where X ranges 0% to 30% in increments of 3%. We repeated this for fat-trees of all even degrees in the range 8-24 and the corresponding Xpanders from Table 2. Figure 5 describes our (representative) results for a fat-tree of degree $d = 14$. As shown in the figure, the throughput of the random network and (deterministic) expander topology degrade linearly with the failure rate whereas the throughput in a fat-tree both exhibits erratic behaviour and degrades at a much steeper rate.

6. COSTS AND COMPLEXITY

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 many of the intricacies of building an actual datacenter. Our aim is merely to illustrate our main insights regarding the deployability of Xpanders.

Physical layout and cabling complexity. As illustrated in Figure 2, 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: (1) placing all ToRs in a meta-node in close proximity (the same rack / row(s)) enables bundling cables between every two meta-nodes, and (2) 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.

	#Switches	#Servers	#Physical Racks	#Cables	Cable Length
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
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

Table 4: Xpander vs. 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%)	3,240 vs. 4,045 (80%)

Table 5: Xpander vs. FatTree

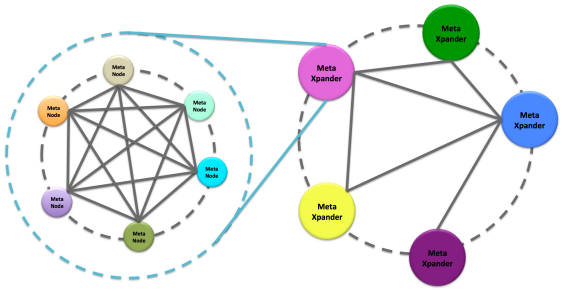


Figure 7: A sketch of an Xpander of Xpander graphs

Equipment, cabling costs, and power consumption. As shown in Table 2, and verified in Section 4, Xpanders can support the same number of servers at the same (or better) level of performance as traditional fat-tree networks, with as low as 80% of the switches. This, of course, has important implications for equipment (switches/servers) 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. Importantly, as the marginal cost active optical cables (AOCs) decreases with length, this reduction in the number of links can translate to potentially greater reductions in costs.

Analyzing deployment scenarios. We analyze two case studies: (1) small clusters, e.g., "container datacenters", and (2) large-scale datacenters. Specifically, we inspected container datacenters in the form of 2-layered folded-clos (2-FCN) networks of degrees 32 and 48 and matching Xpanders. See physical layouts of the $k = 32$ 2-FCNs and Xpander networks in Figure 6 and the results of our analysis in Table 4. The detailed calculations are deferred to the full version of this paper. We also analyzed floor plans for deploying a uniform-degree fat-tree with port-count $k = 32$ and a matching Xpander network with $k = 30$ ports. See side by side comparison of the two networks in Table 5. 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. Large-scale datacenters dispersed over several rooms/locations can be designed a a 2-layered Xpander: an overlay Xpander network interconnecting Xpander networks, as sketched in Figure 7. We defer the details to the full version of this paper.

7. RELATED WORK

Datacenter networks. Datacenters have been extensively researched from many angles, e.g., throughput optimization [31, 28, 20], failure resiliency [22, 15, 14], and expandability [32, 11]. In particular, many datacenter topologies have been proposed, e.g., Clos networks [4, 16, 26], hypercubes [17, 35], small-world graphs [30], and random graphs [32, 31].

Expanders. See [19] for a survery on expander constructions. Expanders play a key role in a host of applications, ranging from networking (e.g., high-performance computing [33, 10, 9, 6] and optical networks [27]) to complexity theory and coding. Our focus is on datacenter networking and the challenges that arise in this context (e.g., specific performance measures, protocols, costs, incremental growth).

8. CONCLUSION

We showed that Xpander datacenters offer many valuable advantages over traditional datacenter designs and suggested practical approaches to building such datacenters. We believe that Xpanders provide an appealing, tangible alternative to traditional datacenter designs.

Acknowledgments

We thank Brighten Godfrey, Nati Linial, and Jonathan Perry, for helpful discussions. This work was supported by NSF awards CCF-1464239 and CCF-1535887, ISF grant 420/12, the Israel Ministry of Science, the Israeli Center for Research Excellence in Algorithms, and a MSR PhD Scholarship.

9. REFERENCES

[1] C. Paasch, S. Barre, et al., Multipath TCP in the Linux Kernel. `http://www.multipath-tcp.org`.

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

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

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

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

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

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

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

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

[10] 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.

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

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

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

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

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

[16] 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).

[17] 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).

[18] 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).

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

[20] JYOTHI, S. A., SINGLA, A., GODFREY, B., AND KOLLA, A. Measuring and understanding throughput of network topologies. *CoRR* (2014).

[21] LANTZ, B., HELLER, B., AND MCKEOWN, N. A network in a laptop: Rapid prototyping for software-defined networks. ACM.

[22] LIU, V., HALPERIN, D., KRISHNAMURTHY, A., AND ANDERSON, T. F10: A fault-tolerant engineered network. NSDI.

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

[24] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev. 38*, 2 (Mar. 2008), 69–74.

[25] MUDIGONDA, J., YALAGANDULA, P., AL-FARES, M., AND MOGUL, J. C. Spain: Cots data-center ethernet for multipathing over arbitrary topologies.

[26] 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).

[27] 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).

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

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

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

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

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

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

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

[35] 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).

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