High Throughput Data Center Topology Design

Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla,
University of Illinois at Urbana–Champaign

https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/singla

This paper is included in the Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14).

April 2–4, 2014 • Seattle, WA, USA

ISBN 978-1-931971-09-6

Open access to the Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14) is sponsored by USENIX

High Throughput Data Center Topology Design

Ankit Singla, P. Brighten Godfrey, Alexandra Kolla
University of Illinois at Urbana–Champaign

Abstract

With high throughput networks acquiring a crucial role in supporting data-intensive applications, a variety of data center network topologies have been proposed to achieve high capacity at low cost. While this work explores a large number of design points, even in the limited case of a network of identical switches, no proposal has been able to claim any notion of *optimality*. The case of *heterogeneous* networks, incorporating multiple line-speeds and port-counts as data centers grow over time, introduces even greater complexity.

In this paper, we present the first non-trivial upper-bound on network throughput under uniform traffic patterns for *any* topology with identical switches. We then show that random graphs achieve throughput surprisingly close to this bound, within a few percent at the scale of a few thousand servers. Apart from demonstrating that homogeneous topology design may be reaching its limits, this result also motivates our use of random graphs as building blocks for design of heterogeneous networks. Given a heterogeneous pool of network switches, we explore through experiments and analysis, how the distribution of servers across switches and the interconnection of switches affect network throughput. We apply these insights to a real-world heterogeneous data center topology, VL2, demonstrating as much as 43% higher throughput with the same equipment.

1 Introduction

Data centers are playing a crucial role in the rise of Internet services and big data. In turn, efficient data center operations depend on high capacity networks to ensure that computations are not bottlenecked on communication. As a result, the problem of designing massive high-capacity network interconnects has become more important than ever. Numerous data center network architectures have been proposed in response to this need [2, 10–15, 20, 23, 25, 26, 30], exploiting a variety of network topologies to achieve high throughput, ranging from fat trees and other Clos networks [2, 13] to modified generalized hypercubes [14] to small world networks [21] and uniform random graphs [23].

However, while this extensive literature exposes several points in the topology design space, even in the limited case of a network of identical switches, it does not answer a fundamental question: *How far are we from throughput-optimal topology design?* The case of *heterogeneous* networks, *i.e.*, networks composed of switches or servers with disparate capabilities, introduces even greater complexity. Heterogeneous network equipment is, in fact, the common case in the typical data center: servers connect to top-of-rack (ToR) switches, which connect to aggregation switches, which connect to core switches, with each type of switch possibly having a different number of ports as well some variations in line-speed. For instance, the ToRs may have both 1 Gbps and 10 Gbps connections while the rest of the network may have only 10 Gbps links. Further, as the network expands over the years and new, more powerful equipment is added to the data center, one can expect more heterogeneity — each year the number of ports supported by non-blocking commodity Ethernet switches increases. While line-speed changes are slower, the move to 10 Gbps and even 40 Gbps is happening now, and higher line-speeds are expected in the near future.

In spite of heterogeneity being commonplace in data center networks, very little is known about heterogeneous network design. For instance, there is no clarity on whether the traditional ToR-aggregation-core organization is superior to a "flatter" network without such a switch hierarchy; or on whether powerful core switches should be connected densely together, or spread more evenly throughout the network.

The goal of this paper is to develop an understanding of how to design high throughput network topologies at limited cost, even when heterogeneous components are involved, and to apply this understanding to improve real-world data center networks. This is nontrivial: Network topology design is hard in general, because of the combinatorial explosion of the number of possible networks with size. Consider, for example, the related[1] *degree-diameter problem* [9], a well-known graph theory problem where the quest is to pack the largest possible number of nodes into a graph while adhering to constraints on both the degree and the diameter. Nontrivial optimal solutions are known for a total of only

[1] Designing for low network diameter is related to designing for high throughput, because shorter path lengths translate to the network using less capacity to deliver each packet; see discussion in [23].

seven combinations of degree and diameter values, and the largest of these optimal networks has only 50 nodes! The lack of symmetry that heterogeneity introduces only makes these design problems more challenging.

To attack this problem, we decompose it into several steps which together give a high level understanding of network topology design, and yield benefits to real-world data center network architectures. First, we address the case of networks of homogeneous servers and switches. Second, we study the heterogeneous case, optimizing the distribution of servers across different classes of switches, and the pattern of interconnection of switches. Finally, we apply our understanding to a deployed data center network topology. Following this approach, our key results are as follows.

(1) Near-optimal topologies for homogeneous networks. We present an upper bound on network throughput for any topology with identical switches, as a function of the number of switches and their degree (number of ports). Although designing *optimal* topologies is infeasible, we demonstrate that random graphs achieve throughput surprisingly close to this bound—within a few percent at the scale of a few thousand servers for random permutation traffic. This is particularly surprising in light of the much larger gap between bounds and known graphs in the related degree-diameter problem [9][2].

We caution the reader against over-simplifying this result to 'flatter topologies are better': Not all 'flat' or 'direct-connect' topologies (where all switches connect to servers) perform equally. For example, random graphs have roughly 30% higher throughput than hypercubes at the scale of 512 nodes, and this gap increases with scale [16]. Further, the notion of 'flat' is not even well-defined for heterogeneous networks.

(2) High-throughput heterogeneous network design. We use random graphs as building blocks for heterogeneous network design by first optimizing the volume of connectivity between groups of nodes, and then forming connections randomly within these volume constraints. Specifically, we first show empirically that in this framework, for a set of switches with different port counts but uniform line-speed, attaching servers to switches in proportion to the switch port count is optimal.

Next, we address the interconnection of multiple types of switches. For tractability, we limit our investigation to two switch types. Somewhat surprisingly, we find that a wide range of connectivity arrangements provides nearly identical throughput. A useful consequence of this result is that there is significant opportunity for clustering switches to achieve shorter cable lengths on average, without compromising on throughput. Jellyfish [23] demonstrated this experimentally. Our results provide the theoretical underpinnings of such an approach.

Finally, in the case of multiple line-speeds, we show that complex bottleneck behavior may appear and there may be multiple configurations of equally high capacity.

(3) Applications to real-world network design. The topology proposed in VL2 [13] incorporates heterogeneous line-speeds and port-counts, and has been deployed in Microsoft's cloud data centers.[3] We show that using a combination of the above insights, VL2's throughput can be improved by as much as 43% at the scale of a few thousand servers simply by rewiring existing equipment, with gains increasing with network size.

While a detailed treatment of other related work follows in §2, the **Jellyfish** [23] proposal merits attention here since it is also based on random graphs. Despite this shared ground, Jellyfish does not address either of the central questions addressed by our work: (a) How close to optimal are random graphs for the homogeneous case? and (b) How do we network *heterogeneous* equipment for high throughput? In addition, unlike Jellyfish, by analyzing how network metrics like cut-size, path length, and utilization impact throughput, we attempt to develop an *understanding* of network design.

2 Background and Related Work

High capacity has been a core goal of communication networks since their inception. How that goal manifests in network topology, however, has changed with systems considerations. Wide-area networks are driven by geographic constraints such as the location of cities and railroads. Perhaps the first high-throughput networks not driven by geography came in the early 1900s. To interconnect telephone lines at a single site such as a telephone exchange, *nonblocking* switches were developed which could match inputs to any permutation of outputs. Beginning with the basic crossbar switch which requires $\Theta(n^2)$ size to interconnect n inputs and outputs, these designs were optimized to scale to larger size, culminating with the Clos network developed at Bell Labs in 1953 [8] which constructs a nonblocking interconnect out of $\Theta(n \log n)$ constant-size crossbars.

In the 1980s, supercomputer systems began to reach a scale of parallelism for which the topology connecting compute nodes was critical. Since a packet in a supercomputer is often a low-latency memory reference (as opposed to a relatively heavyweight TCP connection) traversing nodes with tiny forwarding tables, such

[2] For instance, for degree 5 and diameter 4, the best known graph has only 50% of the number of nodes in the best known upper bound [27]. Further, this gap grows larger with both degree and diameter.

[3] Based on personal exchange, and mentioned publicly at http://research.microsoft.com/en-us/um/people/sudipta/.

systems were constrained by the need for very simple, loss-free and deadlock-free routing. As a result the series of designs developed through the 1990s have simple and regular structure, some based on non-blocking Clos networks and others turning to butterfly, hypercube, 3D torus, 2D mesh, and other designs [17].

In commodity compute clusters, increasing parallelism, bandwidth-intensive big data applications and cloud computing have driven a surge in data center network architecture research. An influential 2008 paper of Al-Fares et al. [2] proposed moving from a traditional data center design utilizing expensive core and aggregation switches, to a network built of small components which nevertheless achieved high throughput — a folded Clos or "fat-tree" network. This work was followed by several related designs including Portland [20] and VL2 [13], a design based on small-world networks [21], designs using servers for forwarding [14, 15, 29], and designs incorporating optical switches [12, 26].

Jellyfish [23] demonstrated, however, that Clos networks are sub-optimal. In particular, [23] constructed a random degree-bounded graph among switch-to-switch links, and showed roughly 25% greater throughput than a fat-tree built with the same switch equipment. In addition, [23] showed quantitatively that random networks are easier to incrementally expand — adding equipment simply involves a few random link swaps. Several challenges arise with building a completely unstructured network; [23] demonstrated effective routing and congestion control mechanisms, and showed that cable optimizations for random graphs can make cable costs similar to an optimized fat-tree while still obtaining substantially higher throughput than a fat-tree.

While the literature on homogeneous network design is sizeable, very little is known about heterogeneous topology design, perhaps because earlier supercomputer topologies (which reappeared in many recent data center proposals) were generally constrained to be homogeneous. VL2 [13] provides a point design, using multiple line-speeds and port counts at different layers of its hierarchy; we compare with VL2 later (§7). The only two other proposals that address heterogeneity are LEGUP [11] and REWIRE [10]. LEGUP uses an optimization framework to search for the cheapest Clos network achieving desired network performance. Being restricted to Clos networks impairs LEGUP greatly: Jellyfish achieves the same network expansion as LEGUP at 60% lower cost [23]. REWIRE removes this restriction by using a local-search optimization (over a period of several days of compute time at the scale of 3200 servers) to continually improve upon an initial feasible network. REWIRE's code is not available so a comparison has not been possible. But more fundamentally, all of the above approaches are either point designs [13] or heuristics [10, 11] which by their blackbox nature, provide neither an *understanding* of the solution space, nor any evidence of near-optimality.

3 Simulation Methodology

Our experiments measure the capacity of network topologies. For most of this paper, our goal is to study topologies explicitly independent of systems-level issues such as routing and congestion control. Thus, we model network traffic using fluid splittable flows which are routed optimally. Throughput is then the solution to the standard maximum concurrent multi-commodity flow problem [18]. Note that by maximizing the minimum flow throughput, this model incorporates a strict definition of fairness. We use the CPLEX linear program solver [1] to obtain the maximum flow. Unless otherwise specified, the workload we use is a random permutation traffic matrix, where each server sends traffic to (and receives traffic from) exactly one other server.

In §8, we revisit these assumptions to address systems concerns. We include results for several other traffic matrices besides permutations. We also show that throughput within a few percent of the optimal flow values from CPLEX can be achieved *after* accounting for packet-level routing and congestion control inefficiencies.

Any comparisons between networks are made using identical switching equipment, unless noted otherwise.

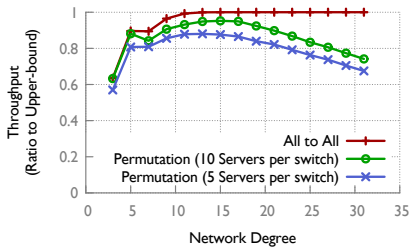
Across all experiments, we test a wide range of parameters, varying the network size, node degree, and oversubscription. A representative sample of results is included here. Most experiments average results across 20 runs, with standard deviations in throughput being $\sim$1% of the mean except at small values of throughput in the uninteresting cases. Exceptions are noted in the text.

Our simulation tools are publicly available [24].

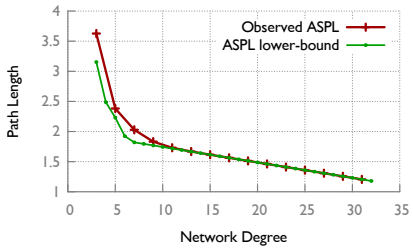
4 Homogeneous Topology Design

In this setting, we have N switches, each with k ports. The network is required to support S servers. The symmetry of the problem suggests that each switch be connected to the same number of servers. (We assume for convenience that S is divisible by N.) Intuitively, spreading servers across switches in a manner that deviates from uniformity will create bottlenecks at the switches with larger numbers of servers. Thus, we assume that each switch uses out of its k ports, r ports to connect to other switches, and $k-r$ ports for servers. It is also assumed that each network edge is of unit capacity.

The design space for such networks is the set of all subgraphs H of the complete graph over N nodes K_N, such that H has degree r. For generic, application-oblivious design, we assume that the objective is to max-

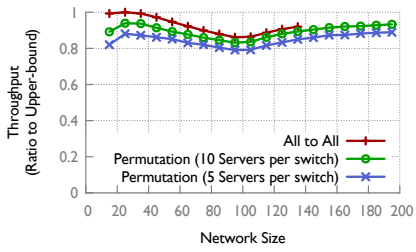


(a)

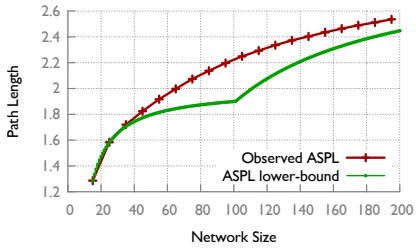


(b)

Figure 1: *Random graphs versus the bounds: (a) Throughput and (b) average shortest path length (ASPL) in random regular graphs compared to the respective upper and lower bounds for any graph of the same size and degree. The number of switches is fixed to* 40 *throughout. The network becomes denser rightward on the x-axis as the degree increases.*



(a)



(b)

Figure 2: *Random graphs versus the bounds: (a) Throughput and (b) average shortest path length (ASPL) in random regular graphs compared to the respective upper and lower bounds for any graph of the same size and degree. The degree is fixed to* 10 *throughout. The network becomes sparser rightward on the x-axis as the number of nodes increases.*

imize throughput under a uniform traffic matrix such as all-to-all traffic or random permutation traffic among servers. To account for fairness, the network's throughput is defined as the maximum value of the minimum flow between source-destination pairs. We denote such a throughput measurement of an r-regular subgraph H of K_N under uniform traffic with f flows by $T_H(N,r,f)$. The average path length of the network is denoted by $\langle D \rangle$.

For this scenario, we prove a simple upper bound on the throughput achievable by *any* hypothetical network.

Theorem 1. $T_H(N,r,f) \leq \frac{Nr}{\langle D \rangle f}$.

Proof. The network has a total of Nr edges (counting both directions) of unit capacity, for a total capacity of Nr. A flow i whose end points are a shortest path distance d_i apart, consumes at least $x_i d_i$ units of capacity in to obtain throughput x_i. Thus, the total capacity consumed by all flows is at least $\sum_i x_i d_i$. Given that we defined network throughput $T_H(N,r,f)$ as the minimum flow throughput, $\forall i, x_i \geq T_H(N,r,f)$. Total capacity consumed is then at least $T_H(N,r,f)\sum_i d_i$. For uniform traffic patterns such as random permutations and all-to-all traffic, $\sum_i d_i = \langle D \rangle f$ because the average source-destination distance is the same as the graph's average shortest path distance. Also, total capacity consumed cannot exceed the network's capacity. Therefore, $\langle D \rangle f T_H(N,r,f) \leq Nr$, rearranging which yields the result. □

Further, [7] proves a lower bound on the average shortest path length of any r-regular network of size N:

$$\langle D \rangle \geq d^* = \frac{\sum_{j=1}^{k-1} jr(r-1)^{j-1} + kR}{N-1}$$

$$\text{where} \quad R = N - 1 - \sum_{j=1}^{k-1} r(r-1)^{j-1} \geq 0$$

and k is the largest integer such that the *inequality* holds.

This result, together with Theorem 1, yields an upper bound on throughput: $T_H(N,r,f) \leq \frac{Nr}{fd^*}$. Next, we show experimentally that random regular graphs achieve throughput close to this bound.

A *random regular graph*, denoted as RRG(N, k, r), is a graph sampled uniform-randomly from the space of all r-regular graphs. This is a well-known construct in graph theory. As Jellyfish [23] showed, RRGs compare favorably against traditional fat-tree topologies, supporting a larger number of servers at full throughput. However, that fact leaves open the possibility that there are network topologies that achieve significantly higher throughput than even RRGs. Through experiments, we compare the throughput RRGs achieve to the upper bound we derived above, and find that our results eliminate this possibility.

Fig. 1(a) and Fig. 2(a) compare throughput achieved by RRGs to the upper bound on throughput for any topology built with the same equipment. Fig. 1(a) shows this comparison for networks of increasing density (*i.e.*, the

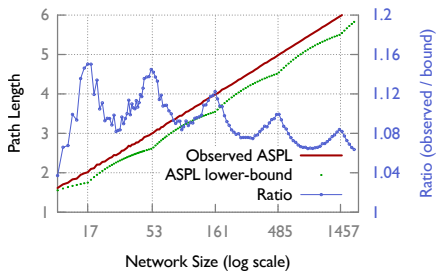


Figure 3: *ASPL in random graphs compared to the lower bound. The degree is fixed to* 4 *throughout. The bound shows a "curved step" behavior. In addition, as the network size increases, the ratio of observed ASPL to the lower bound approaches* 1. *The x-tics correspond to the points where the bound begins new distance levels.*

degree r increases, while the number of nodes N remains fixed at 40) for 3 uniform traffic matrices: a random permutation among servers with 5 servers at each switch, another with 10 servers at each switch, and an all-to-all traffic matrix. For the high-density traffic pattern, *i.e.*, all-to-all traffic, *exact optimal* throughput is achieved by the random graph for degree $r \geq 13$. Fig. 2(a) shows a similar comparison for increasing size N, with $r = 10$. Our simulator does not scale for all-to-all traffic because the number of commodities in the flow problem increases as the square of the network size for this pattern. Fig. 1(b) and 2(b) compare average shortest path length in RRGs to its lower bound. For both large network sizes, and very high network density, RRGs are surprisingly close to the bounds (right side of both figures).

The curve in Fig. 2(b) has two interesting features. First, there is a "curved step" behavior, with the first step at network size up to $N = 101$, and the second step beginning thereafter. To see why this occurs, observe that the bound uses a tree-view of distances from any node — for a network with degree d, d nodes are assumed to be at distance 1, $d(d-1)$ at distance 2, $d(d-1)^2$ at distance 3, etc. While this structure minimizes path lengths, it is optimistic — in general, not all edges from nodes at distance k can lead outward to unique new nodes[4]. As the number of nodes N increases, at some point the lowest level of this hypothetical tree becomes full, and a new level begins. These new nodes are more distant, so average path length suddenly increases more rapidly, corresponding to a new "step" in the bound. A second feature is that as $N \to \infty$, the ratio of observed ASPL to the lower bound approaches 1. This can be shown analytically by dividing an upper bound on the random regular graph's diameter [6] (which also upper-bounds its ASPL) by the lower bound of [7]. For greater clarity, we show in Fig. 3 similar behavior for degree $d = 4$, which makes it easier to show many "steps".

[4]In fact, prior work shows that graphs with this structure do not exist for $d \geq 3$ and diameter $D \geq 3$ [19].

The near-optimality of random graphs demonstrated here leads us to use them as a building block for the more complicated case of heterogeneous topology design.

5 Heterogeneous Topology Design

With the possible exception of a scenario where a new data center is being built from scratch, it is unreasonable to expect deployments to have the same, homogeneous networking equipment. Even in the 'greenfield' setting, networks may potentially use heterogeneous equipment. While our results above show that random graphs achieve close to the best possible throughput in the homogeneous network design setting, we are unable, at present, to make a similar claim for heterogeneous networks, where node degrees and line-speeds may be different. However, in this section, we present for this setting, interesting experimental results which challenge traditional topology design assumptions. Our discussion here is mostly limited to the scenario where there are two kinds of switches in the network; generalizing our results for higher diversity is left to future work.

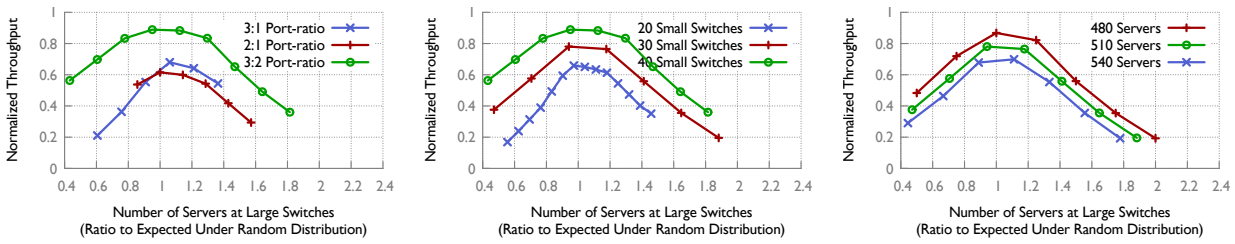
5.1 Heterogeneous Port Counts

We consider a simple scenario where the network is composed of two types of switches with different port counts (line-speeds being uniform throughout). Two natural questions arise that we shall explore here: (a) How should we distribute servers across the two switch types to maximize throughput? (b) Does biasing the topology in favor of more connectivity between larger switches increase throughput?

First, we shall assume that the interconnection is an unbiased random graph built over the remaining connectivity at the switches after we distribute the servers. Later, we shall fix the server distribution but bias the random graph's construction. Finally we will examine the combined effect of varying both parameters at once.

Distributing servers across switches: We vary the numbers of servers apportioned to large and small switches, while keeping the total number of servers and switches the same[5]. We then build a random graph over the ports that remain unused after attaching the servers. We repeat this exercise for several parameter settings, varying the numbers of switches, ports, and servers. A representative sample of results is shown in Fig. 4. The particular configuration in Fig. 4(a) uses 20 larger and 40 smaller switches, with the port counts for the three curves in the figure being 30 and 10 (3:1), 30 and 15 (2:1), and 30 and 20 (3:2) respectively. Fig. 4(b) uses 20 larger switches (30 ports) and 20, 30 and 40 smaller switches

[5]Clearly, across the same type of switches, a non-uniform server-distribution will cause bottlenecks and sub-optimal throughput.



(a) (b) (c)

Figure 4: *Distributing servers across switches: Peak throughput is achieved when servers are distributed proportionally to port counts i.e., x-axis=1, regardless of (a) the absolute port counts of switches; (b) the absolute counts of switches of each type; and (c) oversubscription in the network.*

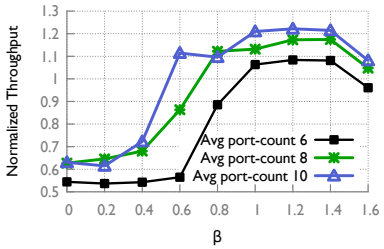


Figure 5: *Distributing servers across switches: Switches have port-counts distributed in a power-law distribution. Servers are distributed in proportion to the β^{th} power of switch port-count. Distributing servers in proportion to degree ($\beta = 1$) is still among the optimal configurations.*

(20 ports) respectively for its three curves. Fig. 4(c) uses the same switching equipment throughout: 20 larger switches (30 ports) and 30 smaller switches (20 ports), with 480, 510, and 540 servers attached to the network. Along the x-axis in each figure, the number of servers apportioned to the larger switches increases. The x-axis label normalizes this number to the *expected* number of servers that would be apportioned to large switches if servers were spread randomly across all the ports in the network. As the results show, distributing servers in proportion to switch degrees (*i.e.*, x-axis$= 1$) is optimal.

This result, while simple, is remarkable in the light of current topology design practices, where top-of-rack switches are the only ones connected directly to servers.

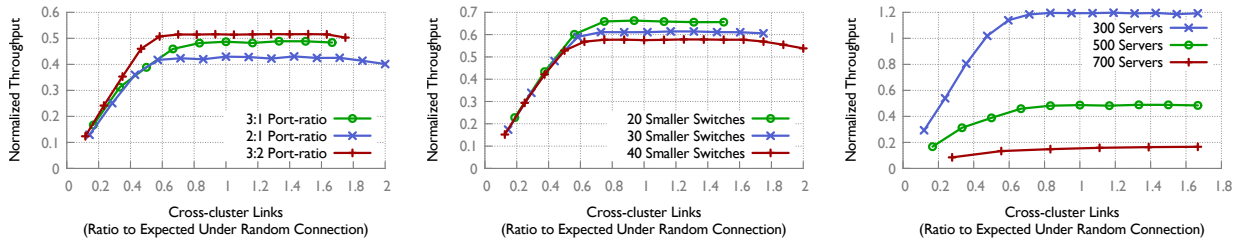
Next, we conduct an experiment with a diverse set of switch types, rather than just two. We use a set of switches such that their port-counts k_i follow a power law distribution. We attach servers at each switch i in proportion to k_i^β, using the remaining ports for the network. The total number of servers is kept constant as we test various values of β. (Appropriate distribution of servers is applied by rounding where necessary to achieve this.) $\beta = 0$ implies that each switch gets the same number of servers regardless of port count, while $\beta = 1$ is the same as port-count-proportional distribution, which was optimal in the previous experiment. The results are shown in Fig. 5. $\beta = 1$ is optimal (within the variance in our data), but so are other values of β such as 1.2 and 1.4. The variation in throughput is large at both extremes of the plot, with the standard deviation being as much as 10% of the mean, while for $\beta \in \{1, 1.2, 1.4\}$ it is $< 4\%$.

Switch interconnection: We repeat experiments similar to the above, but instead of using a uniform random network construction, we vary the number of connections across the two clusters of (large and small) switches[6]. The distribution of servers is fixed throughout to be in proportion to the port counts of the switches.

As Fig. 6 shows, throughput is surprisingly stable across a wide range of volumes of cross-cluster connectivity. x-axis $= 1$ represents the topology with no bias in construction, *i.e.*, vanilla randomness; $x < 1$ means the topology is built with fewer cross-cluster connections than expected with vanilla randomness, etc. Regardless of the absolute values of the parameters, when the interconnect has too few connections across the two clusters, throughput drops significantly. This is perhaps unsurprising – as our experiments in §6.1 will confirm, the cut across the two clusters is the limiting factor for throughput in this regime. What *is* surprising, however, is that across a wide range of cross-cluster connectivity, throughput remains stable at its peak value. Our theoretical analysis in §6.2 will address this behavior.

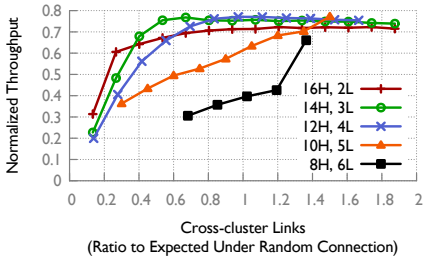
Combined effect: The above results leave open the possibility that joint optimization across the two parameters (server placement and switch connectivity pattern) can yield better results. Thus, we experimented with varying both parameters simultaneously as well. Two representative results from such experiments are included here. All the data points in Fig. 7(a) use the same switching equipment and the same number of servers. Fig. 7(b), likewise, uses a different set of equipment. Each curve in these figures represents a particular distribution of servers. For instance, '16H, 2L' has 16 servers attached to each larger

[6]Note that specifying connectivity across the clusters automatically restricts the remaining connectivity to be within each cluster.

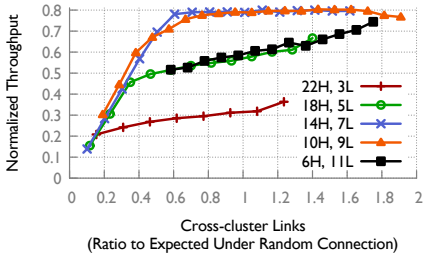


(a) (b) (c)

Figure 6: *Interconnecting switches: Peak throughput is stable to a wide range of cross-cluster connectivity, regardless of (a) the absolute port counts of switches; (b) the absolute counts of switches of each type; and (c) oversubscription in the network.*



(a)



(b)

Figure 7: *Combined effect of server distribution and cross-cluster connectivity: Multiple configurations are optimal, but proportional server distribution with a vanilla random interconnect is among them. (a)* 20 *large,* 40 *small switches, with* 30 *and* 10 *ports respectively. (b)* 20 *large,* 40 *small switches, with* 30 *and* 20 *ports respectively. Results from* 10 *runs.*

switch and 2 to each of the smaller ones. On the *x*-axis, we again vary the cross-cluster connectivity (as in Fig. 6(a)). As the results show, while there are indeed multiple parameter values which achieve peak throughput, a combination of distributing servers proportionally (corresponding to '12H, 4L' and '14H, 7L' respectively in the two figures) and using a vanilla random interconnect is among the optimal solutions. Large deviations from these parameter settings lead to lower throughput.

5.2 Heterogeneous Line-speeds

Data center switches often have ports of different line-speeds, *e.g.*, tens of 1GbE ports, with a few 10GbE ports. How does this change the above analysis change?

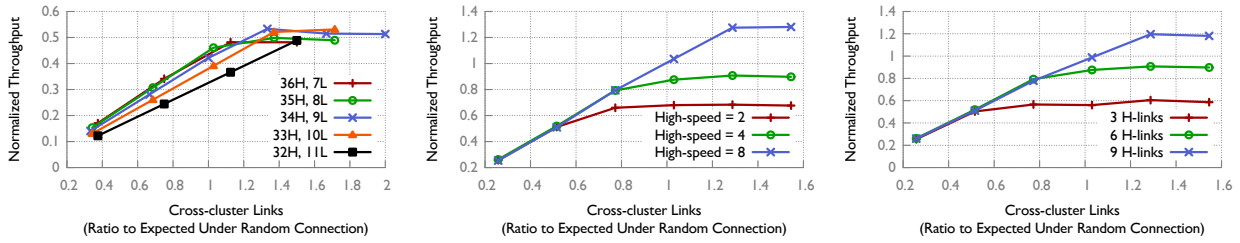
To answer this question, we modify our scenario such that the small switches still have only low line-speed ports, while the larger switches have both low line-speed ports and high line-speed ports. The high line-speed ports are assumed to connect only to other high line-speed ports. We vary both the server distribution and the cross-cluster connectivity and evaluate these configurations for throughput. As the results in Fig. 8(a) indicate, the picture is not as clear as before, with multiple configurations having nearly the same throughput. Each curve corresponds to one particular distribution of servers across switches. For instance, '36H, 7L' has 36 servers attached to each large switch, and 7 servers attached to each small switch. The total number of servers across all curves is constant. While we are unable to make clear qualitative claims of the nature we made for scenarios with uniform line-speed, our simulation tool can be used to determine the optimal configuration for such scenarios.

We also investigate the impact of the number and the line-speed of the high line-speed ports on the large switches. For these tests, we fix the server distribution, and vary cross-cluster connectivity. We measure throughput for various 'high' line-speeds (Fig. 8(b)) and numbers of high line-speed links (Fig. 8(c)). While higher number or line-speed does increase throughput, its impact diminishes when cross-cluster connectivity is too small. This is expected: as the bottlenecks move to the cross-cluster edges, having high capacity between the large switches does not increase the *minimum* flow.

In the following, we attempt to add more than just the intuition for our results. We seek to explain throughput behavior by analyzing factors such as bottlenecks, total network utilization, shortest path lengths between nodes, and the path lengths actually used by the network flows.

6 Explaining Throughput Results

We investigate the cause of several of the throughput effects we observed in the previous section. First, in §6.1,



(a) (b) (c)

Figure 8: *Throughput variations with the amount of cross-cluster connectivity: (a) various server distributions for a network with* 20 *large and* 20 *small switches, with* 40 *and* 15 *low line-speed ports respectively, with the large switches having* 3 *additional* 10× *capacity connections; (b) with different line-speeds for the high-speed links keeping their count fixed at* 6 *per large switch; and (c) with different numbers of the high-speed links at the big switches, keeping their line-speed fixed at* 4 *units.*

we break down throughput into component factors — network utilization, shortest path length, and "stretch" in paths — and show that the majority of the throughput changes are a result of changes in utilization, though for the case of varying server placement, path lengths are a contributing factor. Note that a decrease in utilization corresponds to a saturated bottleneck in the network.

Second, in §6.2, we explain in detail the surprisingly stable throughput observed over a wide range of amounts of connectivity between low- and high-degree switches. We give an upper bound on throughput, show that it is empirically quite accurate in the case of uniform line-speeds, and give a lower bound that matches within a constant factor for a restricted class of graphs. We show that throughput in this setting is well-described by two regimes: (1) one where throughput is limited by a sparse cut, and (2) a "plateau" where throughput depends on two topological properties: total volume of connectivity and average path length $\langle D \rangle$. The transition between the regimes occurs when the sparsest cut has a fraction $\Theta(1/\langle D \rangle)$ of the network's total connectivity.

Note that bisection bandwidth, a commonly-used measure of network capacity which is equivalent to the sparsest cut in this case, begins falling as soon as the cut between two equal-sized groups of switches has less than $\frac{1}{2}$ the network connectivity. Thus, our results demonstrate (among other things) that bisection bandwidth is not a good measure of performance[7], since it begins falling asymptotically far away from the true point at which throughput begins to drop.

6.1 Experiments

Throughput can be exactly decomposed as the product of four factors:

$$T = \frac{C \cdot U}{\langle D \rangle \cdot AS} = C \cdot U \cdot \frac{1}{\langle D \rangle} \cdot \frac{1}{AS}$$

where C is the total network capacity, U is the average link utilization, $\langle D \rangle$ is the average shortest path length, and AS is the average stretch, i.e., the ratio between average length of routed flow paths[8] and $\langle D \rangle$. Throughput may change due to any one of these factors. For example, even if utilization is 100%, throughput could improve if rewiring links reduces path length (this explained the random graph's improvement over the fat-tree in [23]). On the other hand, even with very low $\langle D \rangle$, utilization and therefore throughput will fall if there is a bottleneck in the network.

We investigate how each of these factors influences throughput (excluding C which is fixed). Fig. 9 shows throughput (T), utilization (U), inverse shortest path length ($1/\langle D \rangle$), and inverse stretch ($1/AS$). An increase in any of these quantities increases throughput. To ease visualization, for each metric, we normalize its value with respect to its value when the throughput is highest so that quantities are unitless and easy to compare.

Across experiments, our results (Fig. 9) show that high utilization best explains high throughput. Fig. 9(a) analyzes the throughput results for '480 Servers' from Fig. 4(c), Fig. 9(b) corresponds to '500 Servers' in Fig. 6(c), and Fig. 9(c) to '3 H-links' in Fig. 8(c). Note that it is not obvious that this should be the case: Network utilization would also be high if the flows took long paths and used capacity wastefully. At the same time, one could reasonably expect 'Inverse Stretch' to also correlate with throughput well — if the paths used are close to shortest, then the flows are not wasting capacity. Path lengths do play a role — for example, the right end of Fig. 9(a) shows an increase in path lengths, explaining why throughput falls about 25% more than utilization falls — but the role is less prominent than utilization.

Given the above result on utilization, we examined where in the network the corresponding bottlenecks occur. From our linear program solver, we are able to obtain the link utilization for each network link. We

[7] This result is explored further in followup work [16], where we point out problems with bisection bandwidth as a performance metric.

[8] This average is weighted by amount of flow along each route.

averaged link utilization for each link type in a given network and flow scenario *i.e.*, computing average utilization across links between small and large switches, links between small switches only, etc. The movement of under-utilized links and bottlenecks shows clear correspondence to our throughput results. For instance, for Fig. 6(c), as we move leftward along the x-axis, the number of links across the clusters decreases, and we can expect bottlenecks to manifest at these links. This is exactly what the results show. For example, for the leftmost point ($x = 1.67$, $y = 1.67$) on the '500 Servers' curve in Fig. 6(c), links inside the large switch cluster are on average $< 20\%$ utilized while the links between across clusters are close to fully utilized ($> 90\%$ on average). On the other hand, for the points with higher throughput, like ($x = 1$, $y = 0.49$), all network links show uniformly high utilization ($\sim$100%). Similar observations hold across all our experiments.

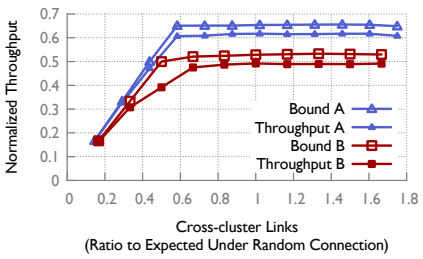
6.2 Analysis

Fig. 6 shows a surprising result: network throughput is stable at its peak value for a wide range of cross-cluster connectivity. In this section, we provide upper and lower bounds on throughput to explain the result. Our upper bound is empirically quite close to the observed throughput in the case of networks with uniform line-speed. Our lower bound applies to a simplified network model and matches the upper bound within a constant factor. This analysis allows us to identify the point (*i.e.*, amount of cross-cluster connectivity) where throughput begins to drop, so that our topologies can avoid this regime, while allowing flexibility in the interconnect.

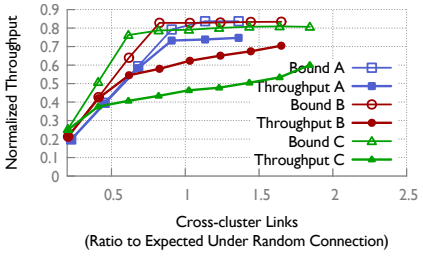
Upper-bounding throughput. We will assume the network is composed of two "clusters", which are simply arbitrary sets of switches, with n_1 and n_2 attached servers respectively. Let C be the sum of the capacities of all links in the network (counting each direction separately), and let $\bar{C}$ be that of the links crossing the clusters. To simplify this exposition, we will assume the number of flows crossing between clusters is exactly the expected number for random permutation traffic: $n_1 \frac{n_2}{n_1+n_2} + n_2 \frac{n_1}{n_1+n_2} = \frac{2n_1n_2}{n_1+n_2}$. Without this assumption, the bounds hold for random permutation traffic with an asymptotically insignificant additive error.

Our upper bound has two components. First, recall our path-length-based bound from §4 shows the throughput of the minimal-throughput flow is $T \leq \frac{C}{\langle D \rangle f}$ where $\langle D \rangle$ is the average shortest path length and f is the number of flows. For random permutation traffic, $f = n_1 + n_2$.

Second, we employ a cut-based bound. The cross-cluster flow is $\geq T \frac{2n_1n_2}{n_1+n_2}$. This flow is bounded above by the capacity $\bar{C}$ of the cut that separates the clusters, so we must have $T \leq \bar{C} \frac{n_1+n_2}{2n_1n_2}$.



(a)



(b)

Figure 10: *Our analytical throughput bound from Eqn. 1 is close to the observed throughput for the uniform line-speed scenario (a) for which the bound and the corresponding throughput are shown for two representative configurations A and B, but can be quite loose with non-uniform line-speeds (b).*

Combining the above two upper bounds, we have

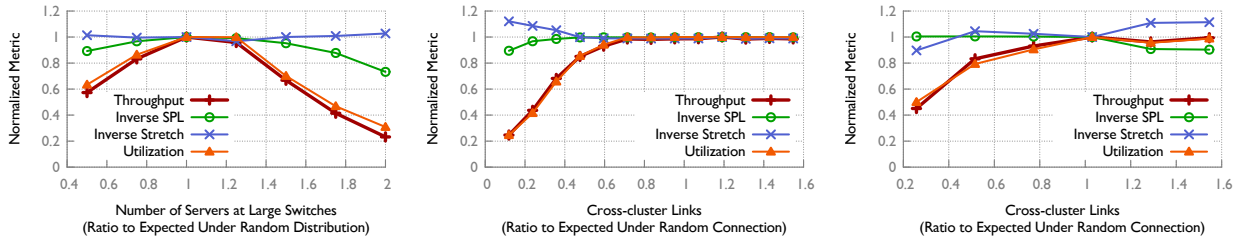
$$T \leq min \left\{ \frac{C}{\langle D \rangle (n_1+n_2)}, \frac{\bar{C}(n_1+n_2)}{2n_1n_2} \right\} \qquad (1)$$

Fig. 10 compares this bound to the actual observed throughput for two cases with uniform line-speed (Fig. 10(a)) and a few cases with mixed line-speeds (Fig. 10(b)). The bound is quite close for the uniform line-speed setting, both for the cases presented here and several other experiments we conducted, but can be looser for mixed line-speeds.

The above throughput bound begins to drop when the cut-bound begins to dominate. In the special case that the two clusters have equal size, this point occurs when

$$\bar{C} \leq \frac{C}{2\langle D \rangle}. \qquad (2)$$

A drop in throughput when the cut capacity is inversely proportional to average shortest path length has an intuitive explanation. In a random graph, most flows have many shortest or nearly-shortest paths. Some flows might cross the cluster boundary once, others might cross back and forth many times. In a uniform-random graph with large $\bar{C}$, near-optimal flow routing is possible with any of these route choices. As $\bar{C}$ diminishes, this flexibility means we can place some restriction on the choice of routes without impacting the flow. However, the flows



(a) (b) (c)

Figure 9: *The dependence of throughput on all three relevant factors: inverse path length, inverse stretch, and utilization. Across experiments, total utilization best explains throughput, indicating that bottlenecks govern throughput.*

which cross clusters must still utilize at least one cross-cluster hop, which is on average a fraction $1/\langle D\rangle$ of their hops. Therefore in expectation, since $\frac{1}{2}$ of all (random-permutation) flows cross clusters, at least a fraction $\frac{1}{2\langle D\rangle}$ of the total traffic volume will be cross-cluster. We should therefore expect throughout to diminish once less than this fraction of the total capacity is available across the cut, which recovers the bound of Equation 2.

However, while Equation 2 determines when the *upper bound* on throughput drops, it does not bound the point at which *observed* throughput drops: since the upper bound might be loose, throughput may drop earlier or later. However, given a peak throughput value, we can construct a bound based on it. Say the peak throughput in a configuration is T^*. $T^* \leq \bar{C}\frac{n_1+n_2}{2n_1n_2}$ implies throughput must drop below T^* when $\bar{C}$ is less than $C^* := T^*\frac{2n_1n_2}{n_1+n_2}$. If we are able to empirically estimate T^* (which is not unreasonable, given its stability), we can determine the value of $\bar{C}^*$ below which throughput *must* drop.

Fig. 11 has 18 different configurations with two clusters with increasing cross-cluster connectivity (equivalently, $\bar{C}$). The one point marked on each curve corresponds to the $\bar{C}^*$ threshold calculated above. As predicted, below $\bar{C}^*$, throughput is less than its peak value.

Lower-bounding throughput. For a restricted class of random graphs, we can lower-bound throughput as well, and thus show that our throughput bound (Eqn. 1), and the drop point of Eqn. 2, are tight within constant factors.

We restrict this analysis to networks $G = (V,E)$ with n nodes each with constant degree d. All links have unit capacity in each direction. The vertices V are grouped into two equal size clusters V_1, V_2, i.e., $|V_1| = |V_2| = \frac{1}{2}n$. Let p, n be such that each node has pn neighbors within its cluster and qn neighbors in the other cluster, so that $p+q = d/n = \Theta(1/n)$. Under this constraint, we choose the remaining graph from the uniform distribution on all d-regular graphs. Thus, for each of the graphs under consideration, the total inter-cluster connectivity is $\bar{C} = 2q\cdot|V_1|\cdot|V_2| = q\cdot\frac{n^2}{2}$. Decreasing q corresponds to decreasing the cross-cluster connectivity and increasing

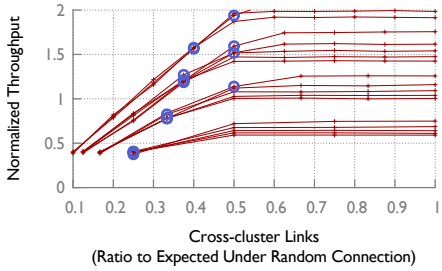


Figure 11: *Throughput shows a characteristics profile with respect to varying levels of cross-cluster connectivity. The one point marked on each curve indicates our analytically determined threshold of cross-cluster connectivity below which throughput must be smaller than its peak value.*

the connectivity within each cluster. Our result below holds with high probability (w.h.p.) over the random choice of the graph. Let $T(q)$ be the throughput with the given value of q, and let T^* be the throughput when $p = q$ (which will also be the maximum throughput).

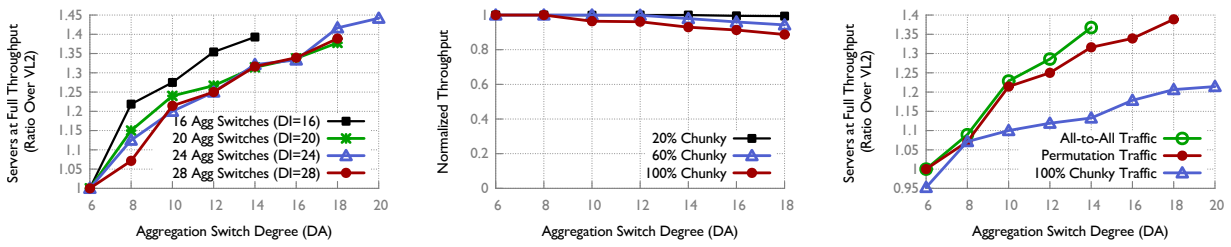
Our main result is the following theorem, which explains the throughput results by proving that while $q \geq q^*$, for some value q^* that we determine, the throughput $T(q)$ is within a constant factor of T^*. Further, when $q < q^*$, $T(q)$ decreases roughly linearly with q. We refer the reader to our technical report [22] for the proof.

Theorem 2. *There exist constants c_1, c_2 such that if $q^* = c_1\frac{1}{\langle D\rangle}p$, then for $q \geq q^*$ w.h.p. $T(q) \geq c_2T^*$. For $q < q^*$, $T(q) = \Theta(q)$.*

7 Improving VL2

In this section, we apply the lessons learned from our experiments and analysis to improve upon a real world topology. Our case study uses the VL2 [13] topology deployed in Microsoft's data centers. VL2 incorporates heterogeneous line-speeds and port-counts and thus provides a good opportunity for us to test our design ideas.

VL2 Background: VL2 [13] uses three types of switches: top-of-racks (ToRs), aggregation switches, and



(a) (b) (c)

Figure 12: *Improving VL2: (a) The number of servers our topology supports in comparison to VL2 by rewiring the same equipment; (b) Throughput under various chunky traffic patterns; and (c) The number of servers our topology can support in comparison to VL2 when we require it to achieve full throughput for all-to-all traffic, permutation traffic, and chunky traffic.*

core switches. Each ToR is connected to 20 1GbE servers, and has 2 10GbE uplinks to different aggregation switches. The rest of the topology is a full bipartite interconnection between the core and aggregation switches. If aggregation switches have *DA* ports each, and core switches have *DI* ports each, then such a topology supports $\frac{DA.DI}{4}$ ToRs at full throughput.

Rewiring VL2: As results in §5.1 indicate, connecting ToRs to only aggregation switches, instead of distributing their connectivity across all switches is sub-optimal. Further, the results on the optimality of random graphs in §4 imply further gains from using randomness in the interconnect as opposed to VL2's complete bipartite interconnect. In line with these observations, our experiments show significant gains obtained by modifying VL2.

In modifying VL2, we distribute the ToRs over aggregation and core switches in proportion to their degrees. We connect the remaining ports uniform randomly. To measure our improvement, we calculate the number of ToRs our topology can support at full throughput compared to VL2. By 'supporting at full throughput', we mean observing full 1 Gbps throughput for each flow in random permutation traffic across each of 20 runs. We obtain the largest number of ToRs supported at full throughput by doing a binary search. As Fig. 12(a) shows, we gain as much as a 43% improvement in the number of ToRs (equivalently, servers) supported at full throughput at the largest size. Note that the largest size we evaluated is fairly small – 2,400 servers for VL2 – and our improvement increases with the network's size.

8 In Practice

In this section, we address two practical concerns: (a) performance with a more diverse set of traffic matrices beyond the random permutations we have used so far; and (b) translating the flow model to packet-level throughput without changing the results significantly.

8.1 Other Traffic Matrices

We evaluate the throughput of our VL2-like topology under other traffic matrices besides random permutations. For these experiments, we use the topologies corresponding to the '28 Agg Switches (*DI*=28)' curve in Fig. 12(a). (Thus, by design, the throughput for random permutations is expected, and verified, to be 1.) In addition to the random permutation, we test the following other traffic matrices: (a) All-to-all: where each server communicates with every other server; and (b) *x*% Chunky: where each of *x*% of the network's ToRs sends all of its traffic to *one* other ToR in this set (*i.e.*, a ToR-level permutation), while the remaining $(100-x)$% ToRs are engaged in a server-level random permutation workload among themselves.

Our experiments showed that using the network to interconnect the same number of servers as in our earlier tests with random permutation traffic, full throughput is still achieved for all but the chunky traffic pattern. In Fig. 12(b), we present results for 5 chunky patterns. Except when a majority of the network is engaged in the chunky pattern, throughput is within a few percent of full throughput. We note that 100% Chunky is a hard to route traffic pattern which is easy to avoid. Even assigning applications to servers randomly will ensure that the probability of such a pattern is near-zero.

Even so, we repeat the experiment from Fig. 12(a) where we had measured the number of servers our modified topology supports at full throughput under random permutations. In this instance, we require our topology to support full throughput under the 100% Chunky traffic pattern. The results in Fig. 12(c) show that the gains are smaller, but still significant, 22% at the largest size, and increasing with size. It is also noteworthy that all-to-all traffic is easier to route than both the other workloads.

8.2 From Flows to Packets

Following the method used by Jellyfish [23], we use Multipath TCP (MPTCP [28]) in a packet level simulation to test if the throughput of our modified VL2-like

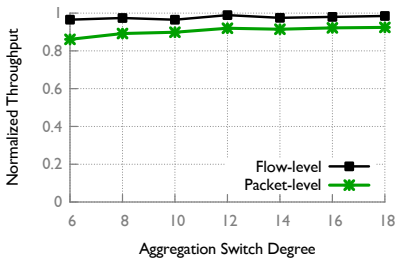


Figure 13: *Packet level simulations of random permutation traffic over our topology show that throughput within a few percent of the optimal flow-level throughput can be achieved using MPTCP over the shortest paths.*

topology is similar to what flow simulations yield. We use MPTCP with the shortest paths, using as many as 8 MPTCP subflows. The results in Fig. 13 show that throughput within a few percent (6% gap at the largest size) of the flow-level simulations is achievable. Note that we deliberately oversubscribed the topologies so that the flow value was close to, but less than 1. This makes sure that we measure the gap between the flow and packet models accurately — if the topology is overprovisioned, then even inefficient routing and congestion control may possibly yield close to full throughput.

9 Discussion

Why these traffic matrices? In line with the design objective of hosting arbitrary applications at high throughput, the approach we have taken is to study *difficult* traffic matrices, rather than TMs specific to particular environments. We show in [16] that an all-to-all workload bounds performance under any workload within a factor of 2. As such, testing this TM is more useful than any other specific, arbitrary choice. In addition, we evaluate other traffic matrices which are even harder to route (Fig. 12(c)). Further, our code is available [24], and is easy to augment with arbitrary traffic patterns to test.

What about latency? We include a rigorous analysis of latency in terms of path length (Fig. 1(b), 2(b)), showing that average shortest path lengths are close to optimal in random graphs. Further, Jellyfish [23] showed that even worst-case path length (diameter) in random graphs is smaller than or similar to that in fat-trees. Beyond path length, latency depends on the transport protocol's ability to keep queues small. In this regard, we note that techniques being developed for low latency transport (such as DCTCP [3], HULL [4], pFabric [5]) are topology agnostic.

But randomness?! 'Random' $\not\Rightarrow$ 'inconsistent performance': the standard deviations in throughput are $\sim$1% of the mean (and even smaller for path length). Also, by 'maximizing the minimum flow' to measure throughput, we impose a strict definition of fairness, eliminating the possibility of randomness skewing results across flows. Further, Jellyfish [23] showed that random graphs achieve flow-fairness comparable to fat-trees under a practical routing scheme. Simple and effective physical cabling methods were also shown in [23].

Limitations: While we have presented here foundational results on the design of both homogeneous and heterogeneous topologies, many interesting problems remain unresolved, including: (a) a non-trivial upper bound on the throughput of heterogeneous networks; (b) theoretical support for our §5.1 result on server distribution; and (c) generalizing our results to arbitrarily diverse networks with multiple switch types.

Lastly, we note that this work does not incorporate functional constraints such as those imposed by middleboxes, for instance, in its treatment of topology design.

10 Conclusion

Our result on the near-optimality of random graphs for homogeneous network design implies that homogeneous topology design may be reaching its limits, particularly when uniformly high throughput is desirable. The research community should perhaps focus its efforts on other aspects of the problem, such as joint optimization with cabling, or topology design for specific traffic patterns (or bringing to practice research proposals on the use of wireless and/or optics for flexible networks that adjust to traffic patterns), or improvements to heterogeneous network design beyond ours.

Our work also presents the first systematic approach to the design of heterogeneous networks, allowing us to improve upon a deployed data center topology by as much as 43% even at the scale of just a few thousand servers, with this improvement increasing with size. In addition, we further the understanding of network throughput by showing how cut-size, path length, and utilization affect throughput.

While significant work remains in the space of designing and analyzing topologies, this work takes the first steps away from the myriad point solutions and towards a theoretically grounded approach to the problem.

Acknowledgments

We would like to thank our shepherd Walter Willinger and the anonymous reviewers for their valuable suggestions. We gratefully acknowledge the support of Cisco Research Council Grant 573665. Ankit Singla was supported by a Google PhD Fellowship.

References

[1] CPLEX Linear Program Solver. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM*, 2008.

[3] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. DCTCP: Efficient packet transport for the commoditized data center. In *SIGCOMM*, 2010.

[4] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda. Less is More: Trading a little Bandwidth for Ultra-Low Latency in the Data Center. *NSDI*, 2012.

[5] M. Alizadeh, S. Yang, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. Deconstructing datacenter packet transport. *HotNets*, 2012.

[6] B. Bollobás and W. F. de la Vega. The diameter of random regular graphs. In *Combinatorica 2*, 1981.

[7] V. G. Cerf, D. D. Cowan, R. C. Mullin, and R. G. Stanton. A lower bound on the average shortest path length in regular graphs. *Networks*, 1974.

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

[9] F. Comellas and C. Delorme. The (degree, diameter) problem for graphs. http://maite71.upc.es/grup_de_grafs/table_g.html/.

[10] A. R. Curtis, T. Carpenter, M. Elsheikh, A. Lopez-Ortiz, and S. Keshav. Rewire: An optimization-based framework for unstructured data center network design. In *INFOCOM*, 2012.

[11] A. R. Curtis, S. Keshav, and A. Lopez-Ortiz. LEGUP: using heterogeneity to reduce the cost of data center network upgrades. In *CoNEXT*, 2010.

[12] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: A hybrid electrical/optical switch architecture for modular data centers. In *SIGCOMM*, 2010.

[13] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: A scalable and flexible data center network. In *SIGCOMM*, 2009.

[14] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *SIGCOMM*, 2009.

[15] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. Dcell: A scalable and fault-tolerant network structure for data centers. In *SIGCOMM*, 2008.

[16] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla. Measuring and Understanding Throughput of Network Topologies. Technical report, 2014. http://arxiv.org/abs/1402.2531.

[17] F. T. Leighton. Introduction to parallel algorithms and architectures: Arrays, trees, hypercubes. 1991.

[18] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 1999.

[19] M. Miller and J. Siran. Moore graphs and beyond: A survey of the degree/diameter problem. *ELECTRONIC JOURNAL OF COMBINATORICS*, 2005.

[20] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. Portland: A scalable fault-tolerant layer 2 data center network fabric. In *SIGCOMM*, 2009.

[21] J.-Y. Shin, B. Wong, and E. G. Sirer. Small-world datacenters. *ACM SOCC*, 2011.

[22] A. Singla, P. B. Godfrey, and A. Kolla. High Throughput Data Center Topology Design. Technical report, 2013. http://arxiv.org/abs/1309.7066.

[23] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Network Data Centers Randomly. In *NSDI*, 2012.

[24] A. Singla, S. A. Jyothi, C.-Y. Hong, L. Popa, P. B. Godfrey, and A. Kolla. TopoBench: A network topology benchmarking tool. https://github.com/ankitsingla/topobench, 2014.

[25] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang. Proteus: a topology malleable data center network. In *HotNets*, 2010.

[26] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan. c-through: Part-time optics in data centers. In *SIGCOMM*, 2010.

[27] C. Wiki. The Degree-Diameter Problem for General Graphs. http://goo.gl/iFRJS.

[28] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, 2011.

[29] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang. Mdcube: A high performance network structure for modular data center interconnection. In *CoNext*, 2009.

[30] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng. Mirror mirror on the ceiling: flexible wireless links for data centers. In *SIGCOMM*, 2012.