A Throughput-Centric View of the Performance of Datacenter Topologies

Pooria Namyar  
University of Southern California

Sucha Supittayapornpong  
Vidyasirimedhi Institute of Science and Technology

Mingyang Zhang  
University of Southern California

Minlan Yu  
Harvard University

Ramesh Govindan  
University of Southern California

ABSTRACT

While prior work has explored many proposed datacenter designs, only two designs, Clos-based and expander-based, are generally considered practical because they can scale using commodity switching chips. Prior work has used two different metrics, bisection bandwidth and throughput, for evaluating these topologies at scale. Little is known, theoretically or practically, how these metrics relate to each other. Exploiting characteristics of these topologies, we prove an upper bound on their throughput, then show that this upper bound better estimates worst-case throughput than all previously proposed throughput estimators and scales better than most of them. Using this upper bound, we show that for expander-based topologies, unlike Clos, beyond a certain size of the network, no topology can have full throughput, even if it has full bisection bandwidth; in fact, even relatively small expander-based topologies fail to achieve full throughput. We conclude by showing that using throughput to evaluate datacenter performance instead of bisection can alter conclusions in prior work about datacenter cost, manageability, and reliability.

CCS CONCEPTS
• Networks → Data center networks; Network performance modeling; Network manageability; Topology analysis and generation; • General and reference → Metrics;

KEYWORDS
Data centers, Throughput, Clos topologies, Network management

ACM Reference Format:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGCOMM ’21, August 23–28, 2021, Virtual Event, USA © 2021 Association for Computing Machinery.
https://doi.org/10.1145/3452296.3472913

1 INTRODUCTION

A primary contributor to the success of cloud computing is the datacenter, a warehouse-style agglomeration of compute and storage on commodity servers. The performance of distributed applications running inside a datacenter, like search, reliable storage, and social networks, is strongly determined by the design of the datacenter network. This network consists of a topology in which switches interconnect servers. Today, datacenters routinely have tens of thousands of switches connecting hundreds of thousands of servers. Our focus, in this paper, is on the design and evaluation of topologies for such large-scale datacenters.

Datacenter topology designs. Two distinct classes of topology designs have emerged in recent years. Clos [8] based designs include Fat-tree [1], VL2 [15], Jupiter [42] and Facebook Fabric [3], and failure-resilient variants, such as F10 [36]. These hierarchical designs are bi-regular, in which a switch either connects to H servers, or none at all (Figure 1). More recent alternative designs target lower installation costs and/or incur lower management costs than Clos-based topologies. These designs employ an expander-graph to interconnect switches, and include Jellyfish [44], Xpander [47], and FatClique [52]. These topologies are uni-regular: every switch connects to H servers (Figure 1). In both classes, each server connects to exactly one switch.\footnote{\textit{1}}

Measures of topology capacity. The capacity of the data center network limits the performance of applications it hosts. Intuitively, a topology with enough capacity to permit every server to send traffic at full line rate simplifies cloud application design: operators can place application instances anywhere in the network without impacting performance, and this placement flexibility enables applications to be more cost efficient and more robust to correlated failures (e.g., of an entire rack or power domain) [15, 21, 35, 42].

Most prior work [1, 3, 15, 42, 52] has used the network’s bisection bandwidth, the smallest aggregate capacity of the links crossing the worst-case cut among all the cuts that divide the topology graph into two halves, as a measure of its capacity. A topology has full bisection bandwidth if its bisection bandwidth is at least equal to half of the total servers; for Clos-based designs, such a topology permits arbitrary application instance placement.

Other work [24, 26, 27, 50, 51] has explored an alternative measure of network capacity, throughput, defined as follows. The throughput under traffic matrix T is the highest scaling factor $\theta(T)$ such that the topology can support the traffic matrix, $T \cdot \theta(T)$,
The throughput of a topology denoted by $\theta^*$ is the worst-case throughput among all traffic matrices. A topology can support any traffic demand if and only if $\theta^*$ is at least 1 (in this case, we say the topology has full throughput). Because it can support any traffic demand, a full throughput topology also permits arbitrary application instance placement by definition.  

**How prior work uses these metrics.** These metrics can help evaluate topology design, perform cost comparisons, or assess the complexity of network expansion. As Table 1 shows a substantial body of work has used bisection bandwidth to perform these assessments on large-scale uni-regular and bi-regular topologies. (Some prior work [27, 43, 44, 47] has used throughput to perform some of these assessments, but for much smaller-scale topologies with only a few thousand servers.)

<table>
<thead>
<tr>
<th>Objective</th>
<th>Metric</th>
<th>Topology Class</th>
<th>Prior work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluate Design</td>
<td>BBW</td>
<td>uni-regular</td>
<td>[4, 42, 52]</td>
</tr>
<tr>
<td>Assess Cost</td>
<td>BBW</td>
<td>bi-regular</td>
<td>[4, 52]</td>
</tr>
<tr>
<td>Estimate Expansion</td>
<td>BBW</td>
<td>uni-regular</td>
<td>[4, 52]</td>
</tr>
</tbody>
</table>

**Table 1:** Prior work has used bisection bandwidth for large-scale evaluations.

Given this discussion, it is natural to ask: What is the difference between these metrics for uni-regular and bi-regular topologies? Should the papers listed in Table 1 have used throughput instead? How would these assessments change if they did?

To our knowledge, the literature has not explored the precise difference between these two metrics, but has explored related, but slightly different questions. Bisection bandwidth is a graph-cut based metric, and [27] has studied the relation between cut based metrics and throughput at a scale much smaller than those we consider in this paper. As well, [34] shows that the sparsest cut of any topology for a given traffic matrix is within $O(\log N)$ of its throughput for that traffic matrix. Finally, Yuan et al. [50] show that bisection bandwidth cannot predict average throughput of a topology.

In this paper, we take a first step in understanding the relationship between these metrics by making the following contributions.

**Contribution: The Difference Between Full Throughput and Full Bisection Bandwidth for Uni-regular Topologies.** We prove (§4) that for any uni-regular topology, there exists a size (in terms of the number of servers) beyond which the topology cannot have full throughput even if it has full bisection bandwidth. This is true even of small instances of uni-regular topologies with as few as 10-15K servers (§4.2). By contrast, bi-regular Clos topologies are not subject to this limit, and a full bisection bandwidth topology always has full throughput (Figure 2). This means that a topology designer cannot ensure application placement independence (more precisely, the ability to support any arbitrary traffic demand) using a full bisection bandwidth uni-regular topology (Table 1). Put differently, for uni-regular topologies, full bisection bandwidth is necessary but not sufficient to support arbitrary traffic demand; by definition, full throughput is both necessary and sufficient.

![Figure 1](image1.png)

**Figure 1:** Uni-regular and bi-regular topologies.

![Figure 2](image2.png)

**Figure 2:** Full throughput vs. Full bisection bandwidth.

**Contribution: A Throughput-Centric View.** Table 1 shows that prior work has used bisection bandwidth to evaluate uni-regular and bi-regular topologies; we show that using throughput can lead to different conclusions, impacting cost and management complexity (§5.1). It is also the more appropriate metric: as the previous contribution demonstrates, throughput better captures the capacity of both uni-regular and bi-regular topologies, while bisection bandwidth does not.

- Prior work has argued that a full bisection bandwidth Jellyfish, Xpander or FatClique uses 50% fewer switches than a full bisection bandwidth Clos [8]. We show that a full throughput Jellyfish [44], Xpander [47] or FatClique [52] uses only 25% fewer switches than a full throughput Clos. This finding is important, because the smaller cost differential may make uni-regular topologies less attractive relative to Clos (whose packaging and routing simplicity may outweigh its higher cost).

- Prior work has argued that a Jellyfish or FatClique can be expanded: (a) with minor bandwidth loss while keeping the number of servers per switch constant; (b) using a random rewiring strategy [52] simpler than that for Clos [53]. This assumes that bandwidth loss is estimated using bisection bandwidth. We show that, expanding a full throughput Jellyfish or FatClique by even a small amount, while keeping fixed the number of servers per switch, can result in a topology without full throughput. Thus, a designer wishing to maintain full throughput for uni-regular topologies after expansion may need to re-wire servers, requiring a much more complex expansion strategy than Clos.

---

2To actually achieve arbitrary instance placement, one also might need a scalable, practical routing scheme that can exploit the topology’s available capacity. For Clos-based networks, ECMP-based routing can do so. For large-scale uni-regular topologies, we believe this question is open. We don’t address this in this paper since we focus on topology properties.
Datacenter designers have traded off topology capacity for lower cost by designing over-subscribed topologies. The FatTree [1] paper defines the over-subscription ratio of a topology as the ratio of the worst-case achievable throughput between end-hosts to the aggregate bisection bandwidth. Our results suggest that, for uni-regular topologies, a more direct definition of over-subscription ratio is the throughput itself (a throughput less than 1 indicates an over-subscribed topology). We find that, for these topologies, the bisection-bandwidth based over-subscription ratio overestimates the throughput by up to 50%. Thus, a designer using that definition would build a network whose actual capacity is lower than the targeted capacity.

**Contribution:** An Efficiently-Computable, Tight, Throughput Upper Bound. The previous contributions require a way to compute the throughput of large uni-regular and bi-regular topologies. To this end, we make the following contributions.

- We prove an upper bound on the throughput of uni-regular and bi-regular topologies (§2).
- We empirically show (§3) that this upper bound is tighter and scales better than existing approaches of estimating network capacity or throughput: the throughput bound in [43], heuristics for estimating throughput in [23, 24, 51], bisection bandwidth, and sparsest cut [27].
- This scalable throughput upper bound can be used to better assess properties of datacenter topologies at larger scales than previously possible, giving a designer a greater confidence in a particular topology (§5.2). A concrete example is resilience. Prior work showed that Jellyfish [44] and Xpander [47] degrade gracefully with random link failure for up to 1K servers; we show that, for a 131K sized Jellyfish or Xpander, degradation is less than graceful (the throughput after failure for up to 1K servers; we show that, for a 131K sized Jellyfish or Xpander, degradation is less than graceful (the throughput after failure for up to 20% lower than what one might expect with graceful degradation) under random failure.

**Ethics.** This work does not raise any ethical issues.

## 2 AN UPPER BOUND ON THROUGHPUT

In this section, we prove an upper bound on throughput that applies to uni-regular and bi-regular topologies.

### 2.1 Complexity of Computing Throughput Bounds

A permutation matrix is one in which each row and each column has exactly one non-zero entry. A permutation matrix can indicate traffic either at the server-level (where each entry denotes traffic between two servers), or switch-level. In server-level permutation matrices, all non-zero entries are normalized to 1 while for switch-level matrices, they are the number of servers connected to the switch \( (H) \). In this section, we show that this set of switch-level permutation traffic matrices, denoted by \( \mathcal{T} \), is sufficient to characterize the throughput of uni-regular and bi-regular topologies.

**Notation.** Entry \( t_{uv} \) of the switch-level traffic matrix \( T \) describes the traffic demand from switch \( u \) to switch \( v \). Let \( \mathcal{K} \) be the set of all switches with servers, and \( H \) be the number of servers connected to each switch in \( \mathcal{K} \). To determine the throughput of the topology, we use the hose model [11]\(^3\), where every switch sends and receives traffic at no more than its maximum rate \( H \) (for simplicity, each link has unit capacity). The hose-model traffic set is the set of traffic matrices that conform to the hose model:

\[
\{ T \in \mathbb{R}_+^{[|\mathcal{K}| \times |\mathcal{K}|]} : \sum_{u \in \mathcal{K}} t_{uv} \leq H \quad \forall u \in \mathcal{K} \}.
\]

where \( \mathbb{R}_+ \) is the set of non-negative reals. This traffic set includes the commonly-used traffic matrices such as all-to-all and random permutations, and it applies not just to uni-regular topologies, but to bi-regular topologies as well. A bi-regular topology contains two types of switches: one without attached servers, and one in which each switch has \( H \) servers. Switches without servers can not source or sink any traffic, and as a result, it suffices to describe the traffic matrix only by switches with attached servers (\( \mathcal{K} \)).

Our hose model definition is consistent with [27], which bases its definition on server-level traffic matrices. Our definition uses switch-level traffic matrices, leveraging the fact that uni-regular and bi-regular topologies have \( H \) servers per switch and each server connects to exactly one switch.

**On computing the throughput of a topology.** Since the hose-model traffic set contains an infinite number of traffic matrices, computing the throughput of the topology (the minimum throughput across all traffic matrices) is intractable.

To improve the tractability, consider the following set of traffic matrices that we call the saturated hose model set, \( \mathcal{T}^* \), where each switch sends and receives traffic at exactly its maximum rate \( H \):

\[
\mathcal{T}^* = \{ T \in \mathbb{R}_+^{[|\mathcal{K}| \times |\mathcal{K}|]} : \sum_{u \in \mathcal{K}} t_{uv} = H \quad \forall u \in \mathcal{K} \}.
\]

This set dominates the hose-model traffic set, since we can always augment any hose-model traffic matrix with a non-negative value to produce a saturated hose-model traffic matrix. So, the minimum throughput across all traffic matrices in the hose model set cannot be smaller than the minimum throughput across all traffic matrices in \( \mathcal{T}^* \). However, there are still infinitely many elements in \( \mathcal{T}^* \). The following theorem shows that for uni-regular and bi-regular

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>Total number of servers</td>
</tr>
<tr>
<td>( E )</td>
<td>Total number of switch-to-switch links</td>
</tr>
<tr>
<td>( R )</td>
<td>Switch radix</td>
</tr>
<tr>
<td>( H )</td>
<td>Number of servers per switch</td>
</tr>
<tr>
<td>( S )</td>
<td>Set of switches with and without servers</td>
</tr>
<tr>
<td>( \mathcal{K} )</td>
<td>Set of switches with ( H ) servers (( \mathcal{K} \subseteq S ))</td>
</tr>
<tr>
<td>( t_{uv} )</td>
<td>Traffic demand from ( u ) to ( v ) where ( u, v \in \mathcal{K} )</td>
</tr>
<tr>
<td>( T = { t_{uv} } )</td>
<td>( [</td>
</tr>
<tr>
<td>( \mathcal{T} )</td>
<td>Saturated hose-model set</td>
</tr>
<tr>
<td>( \mathcal{T}^* )</td>
<td>Permutation traffic set</td>
</tr>
<tr>
<td>( \theta(T) )</td>
<td>Throughput under traffic matrix ( T )</td>
</tr>
<tr>
<td>( \theta^* )</td>
<td>Topology throughput (( \theta^* = \min_{T \in \mathcal{T}^*} \theta(T) ))</td>
</tr>
<tr>
<td>( L_{uv} )</td>
<td>Shortest path length from switch ( u ) to ( v )</td>
</tr>
</tbody>
</table>

\(^3\)In the hose model, the end-host traffic rate is bounded by the port speed, which means the model only permits admissible traffic patterns for the topology. Our use of the hose model is consistent with prior work [11, 27].
For a given traffic matrix \( \hat{T} \), the upper bound on the throughput of a uni-regular or a bi-regular topology is the minimum throughput across all traffic matrices in the permutation traffic set \( \hat{T} \).

**Proof Sketch.** §A contains the detailed proof, which proceeds in two steps. First, it shows that \( \hat{T} \) represents the extrema of the convex polytope formed by the traffic matrices in \( T \). Second, relying on the convexity of the set \( \hat{T} \), it shows that the minimum throughput across all traffic matrices must correspond to a permutation traffic.

Prior work [45] has used a similar convexity argument in a slightly different context, and [46] proves a similar theorem in a more limited context (for oblivious routing). Other prior work ([29], Conjecture 2.4) has stated Theorem 2.1 as a conjecture.

The size of \( \hat{T} \), while finite, grows combinatorially with the matrix dimension, so it is still infeasible to iterate over all its elements in order to compute throughput. However, in any traffic matrix in \( \hat{T} \), each switch sends traffic at full rate to exactly one other switch \( v \). We exploit this, together with the structure of uni-regular and bi-regular topologies to derive an efficiently computable upper bound on the throughput of these topologies (§2.2).

### 2.2 Throughput Upper Bound

We now use Theorem 2.1 to derive a closed-form expression for the upper bound on the throughput of a uni-regular or a bi-regular topology. Throughput is both a function of the topology and the routing algorithm used to route traffic demands; the derived upper bound is independent of the routing algorithm.

**Upper bound for uni-regular topology throughput.** The following theorem establishes a tractable closed-form expression for the throughput of a uni-regular topology. It assumes, without loss of generality, a uni-regular topology with \( H \) servers per switch, and unit link capacity.

**Theorem 2.2.** The maximum achievable throughput for a uni-regular topology, under any routing, is bounded by:

\[
\theta^* \leq \min_{T \in \hat{T}} \frac{2E}{H \sum_{(u,v) \in K^2} L_{uv}[I_{uv} > 0]}
\]

where \( E \) is the number of switch-to-switch links in the topology, \( L_{uv} \) is the shortest path length from switch \( u \) to switch \( v \), and \( [\cdot] \) is an indicator function.

**Proof Sketch.** §B contains the detailed proof, which relies on the optimal solution of the path-based multi-commodity flow problem (§H, commonly used in wide-area network traffic engineering [33]). For a given traffic matrix \( T \), path-based multi-commodity flow maximizes throughput \( \theta(T) \). Now, consider an arbitrary switch \( u \). Its total ingress traffic consists of two components: the traffic destined to its servers, which depends on \( \theta(T) \), and its transit traffic. We upper-bound the ingress traffic by the aggregate link capacity at the switch, and lower-bound it by the total transit traffic derived from the path lengths and the flow split ratios. Solving these inequalities, and applying Theorem 2.1 gives Equation 1.

### Efficiently computing the throughput bound.

The RHS of Equation 1 chooses a permutation traffic matrix that maximizes total path length. Finding this matrix is equivalent to finding worst-case traffic matrix in [27]. In that work, the authors present an intuitive form of the throughput upper bound and suggest an intuitive heuristic for constructing a “difficult” server-level traffic matrix (near-worst-case). In this paper, we formally prove the throughput upper bound and use a slightly different approach (discussed below) that constructs a switch-level traffic matrix to achieve the minimum of the RHS of Equation 1.

To find the minimum throughput, we construct a complete bipartite graph \( B \) consisting of two disjoint sets of nodes \( U \) and \( V \) from the given topology \( G \). \( U \) and \( V \) represent all the possible source and destination switches with directly connected servers in \( G \) respectively. The weight of the edge \( (u, v) \) where \( u \in U \) and \( v \in V \) is the shortest path length from switch \( U \) to switch \( V \). The permutation traffic matrix that determines the throughput bound in Equation 1 corresponds to the weighted maximum matching in \( B \). We call this the **maximal permutation matrix**.

### Extension to bi-regular topologies.

Theorem 2.2 applies to bi-regular topologies as well. Intuitively, additional switches with no servers increase capacity for transit traffic which is reflected in the numerator of Equation 1. We prove this formally in §C. The theorem also applies to uni-regular and bi-regular topologies in which each switch has a different radix \( R_u \); we have omitted the description of this extension for brevity.

Theorem 2.2 implies that throughput of a topology is proportional to total link capacity and inversely proportional to maximal total path length of the maximal permutation matrix. Prior work [43] has computed an upper-bound on the average throughput of uni-regular topologies across all uniform traffic matrices (the all-to-all and permutation matrices). In contrast, we bound the worst-case throughput, and our bound is significantly closer (§3.2) to the worst-case behavior of uni-regular topologies at all scales than the bound of [43]. Our bound is also more general: it applies to bi-regular topologies as well, and across all traffic matrices (as a consequence of Theorem 2.1).

### On server-level vs. switch-level traffic matrices.

We exploit the regularity in uni-regular and bi-regular topologies and reason about switch-level permutation traffic matrices, rather than server-level ones. This helps us efficiently compute the upper-bound even for large topologies (§3). This efficiency does not impact the throughput estimate, relative to using a server-level permutation matrix, as we now show.

If we had used the server-level TMs, the throughput upper-bound would have been the same. A switch-level maximal permutation matrix \( \hat{T} \), when converted to server-level \( \hat{T}_s \), is a solution to the corresponding server-level weighted maximum matching problem. We can prove this by contradiction. Let, for any server \( u \), \( s(u) \) be the switch connected to \( u \) and assume that \( \hat{T}_s \) can be improved by (the total path length of the permutation matrix can be increased by, see denominator of Equation 1) a set of actions on \( (u, v) \) (e.g., insertion or deletion of a flow). We can show that \( \hat{T} \) can be also improved by the same amount by a similar set of actions on \( (s(u), s(v)) \). This is because the link from the server to its directly connected
switch does not constrain throughput, so all $L_{uv,5}$ do not include it. Thus, adding/removing $(s(u), s(v))$ increases/decreases the total path length by the same amount as adding/removing $(u, v)$ does. This is a contradiction since we assumed $T$ is the maximal permutation matrix.

However, the actual throughput of the topology under switch-level maximal permutation matrix is always less than or equal to the server-level one. If the server-level maximal permutation matrix, when converted to switch-level, is not a permutation matrix, a similar line of proof as Theorem 2.1 can show that the corresponding switch-level traffic matrix is a convex combination of some switch-level permutation traffic matrices. So, at least one of the switch-level permutation matrices has lower throughput than this TM. Hence, considering switch-level matrices not only improves the scalability of our throughput bound but also better captures the minimum throughput of the topology.

3 EVALUATING THE THROUGHPUT UPPER BOUND

In this section, we show that throughput upper bound (abbreviated TUB) (a) accurately estimates the worst case throughput and (b) all previously proposed throughput estimators [23, 24, 43, 51] produce worse estimates for uni-regular topologies and servers per switch ($H$).

3.1 Throughput Gap

In this section, we compute the throughput gap between the throughput upper bound (abbreviated TUB) and the throughput achieved by routing a “worst-case” traffic matrix, and show that this gap is small.

**Methodology.** Prior work [27] has shown that maximal permutation matrix can achieve worst-case throughput. We have independently verified this. For small topologies, we exhaustively compared the throughput of every TM under KSP-MCF, and the maximal permutation matrix achieves the lowest throughput. For large topologies, we compared the throughput of the maximal permutation matrix with 20 random permutations, and observed that the throughput of maximal permutation matrix is constantly lower, and the gap between these two increases with scale.

To demonstrate that the throughput gap is small, we need to select a routing scheme. We have found that it suffices to solve a path-based multi-commodity flow [33] over K-shortest paths (KSP-MCF, see §H). To compute the throughput gap, we sweep values of $K$ until increasing $K$ does not increase throughput$^5$; in most cases, $K = 100$ suffices to match TUB. As an aside, we do not mean to suggest that KSP-MCF is practical for large networks; especially for uni-regular topologies, finding a scalable routing scheme that can achieve high throughput is an open question left to future work.

**Other details.** For all results in the paper, we use METIS [28] to (over) estimate bisection bandwidth, Gurobi [18] to solve linear programs for MCF, the networkx [19] implementations of K-shortest paths [49] and the igraph [9] implementation of maximum bipartite matching [32, 40]. FatClique deviates slightly from our definition of uni-regular topologies: in a FatClique topology, $H$ is loose in the range 3K – 15K because (a) the proof of Theorem 2.2 uses the observation that throughput is highest when

---

$^4$Our code is available at https://github.com/USC-NSL/TUB

$^5$§J shows the results for different values of $K$
all paths between each source-destination pair are shortest paths and (b) topologies in this size range have fewer shortest paths, so KSP-MCF routes traffic over non-shortest paths. (Figure 4(a) plots the distribution of the fraction of flows over shortest and non-shortest paths for different topology sizes).

Interestingly, topologies with 100K – 180K servers have a smaller fraction of shortest paths (Figure 4(b)), so we expect TUB to be loose in that range (we cannot confirm this because KSP-MCF does not scale to those sizes), but expect the throughput gap to be small beyond that range because the fraction of shortest paths increases. However, in §E, we show that the maximum possible throughput gap approaches zero asymptotically. Future work can explore better throughput bounds that exploit diversity in non-shortest paths.

**Xpander and FatClique.** Figure 3 shows the throughput gap for Xpander and FatClique, for different values of $H$. Like Jellyfish at $H = 8$, the gap is significant at small scales between 5K – 15K for these topologies and the gap is close to zero for larger instances.

**Bi-regular Topologies.** For Clos-based bi-regular topologies, ECMP is able to achieve (close to) full throughput (modulo differences in flow sizes [15]). We find that TUB’s estimate is also 1 for different Clos topologies, showing that the gap is zero for them as well (Table A.1).

### 3.2 Comparison with other throughput metrics

Prior work has proposed other ways of estimating throughput. For uni-regular topologies, we expect TUB to be (a) faster and (b) more accurate than these other methods, because it leverages properties of uni-regular topologies. In this section, we validate this intuition.

**Efficiently computing TUB.** Before doing so, we briefly discuss some empirical results for the speed of computing TUB. The bottleneck in this computation is the weighted maximum matching in a complete bipartite graph. Several network analysis tools such as networkx [19] and igraph [9], have an efficient implementation of weighted maximum matching. Furthermore, our computation scales well because we abstract the server-level traffic into a switch-level traffic matrix, so that the number of nodes in the constructed bipartite graph reduces significantly. On a machine with 64GB of RAM, we were able to find the throughput upper bound for topologies with up to 180K servers with $H = 8$ within 20 minutes. For calibration, on the same platform, computing the throughput for routing a permutation traffic matrix using KSP-MCF does not scale beyond 50K servers, and using full-blown MCF does not scale beyond 8K servers.

**Comparison alternatives.** Prior work [27] has compared throughput (i.e., the solution to MCF) with cut-based metrics, such as sparsest-cut (using an eigenvector based optimization in [26]) and bisection bandwidth, and [43] computes an upper bound on average throughput of uni-regular topologies across uniform traffic matrices. In addition to these, we compare our method to two other throughput estimators developed for general graphs. Hoefler’s method [51] divides a flow into sub-flows on each path between source and destination, and splits the capacity of a link equally across all flows traversing it. Jain’s method [24] incrementally routes flows on each path; at each step it allocates residual capacity on a link to all new flows added to the link at this step and iterates until no paths remain.

**Results.** Figure 5 compares TUB against these alternatives, for Jellyfish topologies with 8 servers per switch. Results for other topologies are similar (omitted for brevity).

**Small to medium scale.** Figure 5(a) shows the throughput gap (determined using the methodology described in §3.1) for topologies with up to 25K servers. TUB has the smallest throughput gap across all alternatives. In the range 15K – 25K, TUB’s throughput gap is zero, that of others is higher than 0.2, and sometimes as high as 0.4. To illustrate why it is important to have a small throughput gap, consider a scenario in which a network operator wishes to design a full throughput topology; if she uses a loose throughput estimator, the resulting topology may not actually have full throughput.

Moreover, TUB is among the most efficient of the alternatives (Figure 5(b)).

It is both more accurate, and faster than Jain’s method (JM) and Hoefler’s method (HM). These have large throughput gaps at larger topology sizes (Figure 5(a)). JM and HM exploit edges of each available path, but their estimates are loose because they assume all the sub-flows going through each edge get a fair share of the edge’s capacity. This assumption may not maximize the throughput of a traffic matrix; to do this, flows that currently have lower throughput should get more share of the available capacity. JM and HM are a few orders of magnitude slower than TUB (Figure 5(b)) because they exploit more of the topological structure.

Bisection bandwidth and [43] scale better than TUB, but their estimates have large error. Bisection bandwidth is a loose cut-based estimate of throughput as shown by [27] at small scales, and proven by us in §4. Figure 5(a) empirically verifies this at much larger scales than [27]. Computing exact bisection bandwidth for general networks is intractable [4], so we use a fast heuristic [28] that approximates the bisection bandwidth. Furthermore, the bound in [43] relies on average distance among all the pairs of switches, based on the fact that every switch splits its traffic equally and sends to all the other switches in the average case. Our bound, however, considers structural properties (e.g., distance between individual pairs) to maximize the congestion by routing the traffic between pairs with the largest distance. Therefore, the gap for TUB is smaller than that for [43], but TUB is slower since it considers more details about the topology.

**Large scale.** Figure 5(c) plots the bisection bandwidth, and the throughput estimated by TUB, and by [43], for topologies for up to 300K servers. At these scales, we cannot compute KSP-MCF to estimate the throughput, so we depict the absolute throughput values. [43]’s throughput estimate is consistently and considerably higher across the entire range compared to TUB’s. The latter’s computational complexity is comparable to that of [43], except for the range 200K – 280K where TUB exhibits a non-monotonic behavior. TUB attempts to choose disjoint pairs of switches with large distances from each other to construct the maximal permutation matrix, but in topologies of this size range, there are fewer of these pairs with longest possible distance (i.e., diameter), so it takes longer for the
algorithm to search for these disjoint pairs. We expect to significantly reduce the search by parallelizing the weighted maximum matching implementation; we have left this to future work.

**Summary.** TUB’s throughput gap is smaller than those of prior estimators and scales to up to 300K servers. This enables us to revisit whether prior evaluations of large-scale topologies using bisection bandwidth would yield different conclusions if throughput were used instead (§5).

## 4 LIMITS ON THE THROUGHPUT OF UNI-REGULAR TOPOLOGIES

In this section, using Theorem 2.2 we establish asymptotic limits on the size of full-throughput uni-regular topologies. Then, exploiting TUB’s scalability and tightness (§3), we establish practical limits on the size of full-throughput uni-regular topologies for different values of $H$.

### 4.1 Asymptotic Limits

**A throughput upper bound for all uni-regular topologies.** Theorem 2.2 determines an upper-bound on the throughput for a given uni-regular or bi-regular topology, independent of routing. The following theorem, which applies only to uni-regular topologies, establishes an upper-bound on the throughput across all uni-regular topologies, independent of routing.

**Theorem 4.1.** The maximum achievable throughput of any uni-regular topology with $N$ servers, switch radix $R$ and $H$ servers per switch under any routing is:

$$\theta^* \leq \frac{N(R - H)}{H^2D}$$  \tag{2}$$

where:

$$D = d\left(\frac{N}{H} - 1\right) - \frac{R - H}{R - H - 2}\left(\frac{(R - H - 1)d - 1}{R - H - 2} - d\right)$$

and $d$ is the minimum diameter required to accommodate $N/H$ switches computed using Moore bound [39].

**Proof Sketch.** §D contains the detailed proof. We observe from Equation 1 that throughput is lowest for switch pairs $(u, v)$ for whom the shortest path length $L_{uv}$ is high. Our constructive proof first bounds the number of switches whose distance is at least $m$ from a given switch (Lemma 8.1 in §D). Then, we construct (Algorithm 1 in the Appendix) the maximal permutation traffic matrix in which each switch exchanges traffic with other switches that are furthest from it (Lemma 8.2 in §D). This construction maximizes $L_{uv}$, and from this construction and using Lemma 8.1, we can bound the number of communicating switch pairs whose distances are at least $m$ hops of each other. The bound applies to the denominator of the RHS of Theorem 2.2, resulting in a throughput upper bound independent of the traffic matrix (Lemma 8.3 in §D). □

This theorem formalizes the intuition captured in Figure 6. Fundamentally, a uni-regular topology is constrained by the fact that every switch has to have $H$ servers. The figure shows topologies in which 3-port switches have at most $H = 1$ servers. The leftmost 4-switch topology has full throughput. However, the addition of a single switch (the middle topology) drops throughput significantly. To recover full throughput in this setting, we need to add four more switches with no servers; these provide additional transit capacity. Figure 7 shows the worst-case TM for the middle topology along with the optimal routing of the TM. It also presents the throughput of the same TM on the bi-regular topology with 4 additional switches.

**Relationship between bisection bandwidth and throughput.** Using Theorem 4.1, we can derive a necessary condition for any full throughput uni-regular topology:

$$D \leq \frac{N(R - H)}{H^2}$$  \tag{3}$$

Unlike bi-regular topologies where Clos topologies have full bisection bandwidth and full throughput (see below), uni-regular
topologies can have full bisection bandwidth, but not full throughput (as illustrated in Figure 2). Table 3 shows the maximum number of servers each topology family can support without violating Equation 3 (switch radix $R$ is 32). It shows that the largest full throughput uni-regular topology with 8 servers per switch can only support 111K servers, while the largest full bisection bandwidth Jellyfish, Xpander, or FatClique topologies can support over 20M servers! (In Table 3, for all uni-regular topologies, we were unable to estimate the bisection bandwidth for topologies larger than 20M servers because of computational limits.)

Scaling limits on uni-regular topologies. Another way of stating the results in Table 3 is that no uni-regular topology with $H = 8$ and more than 111K servers can have full throughput. This implies that there is a bound on the number of servers that a full-throughput uni-regular topology can have. Corollary 1 formalizes this; we prove it in §G.

**Corollary 1.** For a given switch radix $R$ and servers per switch $H$, there exists a $N^*(R,H)$ such that for $N \geq N^*(R,H)$, no full throughput uni-regular topology exists with $N$ servers, switch radix $R$ and $H$ servers per switch.

Every Clos-based topology always has full throughput. In contrast to these scaling limits for uni-regular topologies, a fully-deployed Clos-based topology always has full throughput. In §2.1, we observed that Theorem 2.1 applies to Clos-based topologies. Prior work has shown that a multi-stage Clos can (re-)arrangeably support every permutation traffic matrix [25, 41]. Since Clos is a bi-regular topology, it must have a throughput of 1 because, by Theorem 2.1, it suffices to consider only permutation traffic matrices to compute the throughput, and Clos can support all permutation traffic matrices (i.e., for each matrix in $\mathcal{T}$, Clos has a throughput of 1). Thus, bi-regular topologies like VL2 [15] and FatTree [1], being Clos topologies, have full throughput. We conjecture that F10 [36] also has full throughput (F10 uses a different striping than Clos), but have left it to future work to prove that.

### 4.2 The Full-Throughput Frontier

Table 3 shows the largest possible number of servers any uni-regular topology can support at full throughput. However, this bound is loose in part because it applies generically to all uni-regular topologies. In this section, for each topology family, we characterize, as a function of $H$, the largest size beyond which no topology has full-throughput

<table>
<thead>
<tr>
<th>Topology</th>
<th>Condition</th>
<th>$H = 8$</th>
<th>$H = 7$</th>
<th>$H = 6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uni-regular</td>
<td>Equation 3</td>
<td>111K</td>
<td>256K</td>
<td>3.97M</td>
</tr>
<tr>
<td>Jellyfish</td>
<td>Full-BBW</td>
<td>&gt;20M</td>
<td>&gt;20M</td>
<td>&gt;20M</td>
</tr>
<tr>
<td>Xpander</td>
<td>Full-BBW</td>
<td>&gt;20M</td>
<td>&gt;20M</td>
<td>&gt;20M</td>
</tr>
<tr>
<td>FatClique</td>
<td>Full-BBW</td>
<td>&gt;20M</td>
<td>&gt;20M</td>
<td>&gt;20M</td>
</tr>
</tbody>
</table>

Table 3: Maximum number of servers, each topology setup can support without violating the condition.

The uni-regular topology can support the given worst-case permutation traffic matrix with throughput of 2, while the bi-regular topology with 4 additional switches can support the TM at full throughput. In the uni-regular topology setup, the optimal routing is the following: $\frac{1}{2}$ of each flow is routed through the shortest path while $\frac{1}{2}$ of each flow is routed through the non-shortest path. 

**Results.** Figure 8 shows the results of these experiments for Jellyfish, Xpander, and FatClique.

**Jellyfish and Xpander.** Figure 8(a) shows the full-throughput and full-bisection bandwidth frontier curves for Jellyfish, and Figure 8(b) for Xpander. For both Jellyfish and Xpander, there is a large gap between these curves; there are many topologies that have full bisection bandwidth, but do not have full throughput. In some configurations (specifically $H = 7$ and 8), these topologies cannot achieve full throughput even with 10K-15K servers. At $H = 9$, these topologies can support a few hundred servers with full throughput. For $H = 6$, Jellyfish and Xpander can support full throughput up to 225K servers (off-scale in Figure 8(a), Figure 8(b)).

How does throughput degrade beyond the frontier? At 7 servers per switch, a Jellyfish with 13K servers has a $\text{tub}$ of 1, with 15K servers a $\text{tub}$ of 0.94, and with 17K servers a $\text{tub}$ of 0.89. Similar results hold for Xpander. This appears to suggest that the throughput of these topologies degrade gracefully beyond the frontier, but we have left a more detailed analysis to future work.

**FatClique.** Because FatClique instances can be non-monotonic with respect to throughput, the full-throughput frontier curve is approximately the boundary separating the blue (Throughput) points from the red (BBW) points in Figure 8(c). Like Jellyfish and

---

Some topologies smaller than this size may also not have full throughput because $\text{tub}$ is an upper bound.
Xpander, there are no FatClique topologies above 10K which have full throughput by the TUB for \( H \) values of 7 and 8 (at these values, above 10K, all instances are labeled BBW).

**Takeaways.** While uni-regular topologies have elegant designs (Jellyfish and Xpander) and useful manageability properties (FatClique), their throughput scaling is fundamentally limited (§4), and many of their topology instances do not have full-throughput even at scales far smaller than modern data centers (e.g., Amazon AWS with more than 50K servers [2], Google Jupiter with more than 30K servers [42]). At these larger scales, these topologies can use smaller values of \( H \), but this can negate the cost advantages of uni-regular topologies, as we show next.

## 5 A THROUGHPUT-CENTRIC VIEW OF TOPOLOGY EVALUATIONS

In this section, we revisit prior work on topology evaluation from a throughput-centric perspective.

### 5.1 Throughput vs. Bisection Bandwidth

§4.1 shows that, for uni-regular topologies, throughput and bisection bandwidth are different, and that, by definition, throughput accurately captures the capacity of the network. Here we explore whether conclusions from prior work that has used bisection bandwidth to evaluate uni-regular topologies would change if throughput were used instead. Table 4 summarizes our findings.

**Topology Cost.** Datacenter designers seek highly cost-effective designs [35]. FatClique [52] and Jellyfish [44] have compared the cost of their designs against Clos-based topologies by generating full bisection bandwidth instances of their topology using the minimum number of switches, and then comparing that number against a Clos with the same number of servers. Figure 9 shows what would happen if they had, instead, generated full throughput instances, for topologies with different sizes and switch radices.

Figure 9(a) and Figure 9(b) show that the full throughput Jellyfish and Xpander built from 32-port switches use about 33% more switches than the full bisection bandwidth topology at the scale of 32K and 131K servers (because, to achieve full throughput at larger sizes, uni-regular topologies must use a smaller \( H \)). This increase in the number of switches for FatClique is approximately 27%. This affects the comparison with Clos\(^7\); Clos uses 1.8x more switches compared to uni-regular topologies to achieve full bisection bandwidth\(^8\) but only 1.3x more relative to full throughput uni-regular topologies.

Figure 9(c) demonstrates that, at higher switch radices, the impact of the choice of metric is more severe for uni-regular topologies. To do this experiment, we needed to normalize the scale of the topology relative to the radix of a switch. A natural way to normalize this is to design a uni-regular topology with as many servers as a 1/8th 4-layer Clos. (Percentages are Full-TUB/Full-BBW - 1.)

---

1 In this and subsequent evaluations, for Clos topologies the number of servers per switch for leaf switches is always equal to \( \frac{R}{2} \), where \( R \) is the switch radix, while the rest of the switches have no servers.

2 Results for bisection bandwidth are consistent with findings of [44, 52]
topologies with the same number of servers as a 1/8th Clos for the corresponding switch radix. At a radix of 64, a 1/8th Clos has 263K servers. Figure 9(c) shows the percentage increase in the number of switches required to support Full Throughput over those required to support Full BBW. This fraction increases with switch radix; with 64-port switches, Full BBW requires almost 50% more switches.

This difference can change a topology designer’s tradeoff analysis. Clos and uni-regular topologies differ in one other important way: the former has demonstrated, through wide deployment, a simple and practical routing scheme (ECMP) that can achieve high throughput, but proposed routing for uni-regular topologies rely on routing schemes such as MPTCP [48] over K-shortest paths [49], ECMP-VLB hybrid [29] or FatPaths [7]. The deployment and operational cost of these schemes is not known, so, if the relative switch cost advantage of uni-regular topologies is low, a designer might find them less attractive when other costs, such as routing, are taken into account.

Fabric Expansion. As recent work has shown [52, 53], datacenter fabrics are rarely deployed at full scale initially. Rather, for a Clos-based topology like Jupiter [42], a designer starts by determining a target number of servers in the datacenter and the number of layers needed in the Clos topology to achieve that scale. Then, they can incrementally deploy the topology, often in units of superblocks [53].

One attractive aspect of some uni-regular topologies like Jellyfish over Clos is that, at least conceptually, their expansion is simpler and requires no advance planning [44, 47, 52]. For example, it is possible to add one switch and its servers to Jellyfish by randomly removing links and connecting the opened ports to the new switch. It is easy to see, from Figure 8(a), that this expansion likely preserves full bandwidth. For example, if one starts with a 5K Jellyfish topology with \( H = 8 \), and augments it to 10K servers, the resulting topology is still under the BBW line, so has full bisection bandwidth.

However, this expansion strategy may not always preserve full throughput. In the same example, at 10K servers with \( H = 8 \), the topology is above the Throughput line: in other words, while the topology before expansion has full throughput, the final topology does not.

Thus, when planning a datacenter topology, a designer must carefully consider future target expansion sizes and choose \( H \) accordingly. If the target size is 10K, the topology designer needs to plan in advance (as in Clos) and start with a \( H = 7 \) instance in order to preserve throughput after expansion. (The alternative is to re-wire servers, which can significantly increase the cost of expansion).

Over-subscription. The Fat-Tree work [1] defined a topology’s over-subscription ratio as the ratio between the actual bisection bandwidth and full bisection bandwidth. This definition can be misleading when applied to uni-regular topologies. For these topologies, the throughput itself is a measure of over-subscription. A throughput of \( f \) indicates that each server can send traffic at a fraction \( f \) of its line rate, corresponding to an over-subscription ratio of \( 1/f \).

Table 5 illustrates the difference between these two definitions of over-subscription ratio for uni-regular topologies. For all uni-regular topologies we have measured, the over-subscription ratio defined using throughput is lower than bisection bandwidth-based over-subscription ratio. For Clos, these two values are identical.

This suggests that, for uni-regular topologies, throughput is a more conservative measure of over-subscription. It is also more accurate, since it measures the upper bound of the actual achievable throughput.

5.2 Scaling Throughput Evaluations

§3 shows that TUB better estimates worst-case throughput and scales better than most of the previous throughput estimators. Here we revisit the conclusions from prior work that has evaluated topology properties at smaller-scales using other ways to estimate throughput. Table 6 summarizes our findings; we describe these below.

Cost and Expansion. Singla et al. [44] have estimated throughput using ideal routing on a few random permutations and show that Jellyfish can support 27% more servers at full throughput than a Fat-Tree [1] using the same number of switches. They conjecture that this advantage improves by using a higher radix switch. In §K, we show that: (1) the cost advantage at the largest considered size in [44] is only 8% when TUB is used to estimate throughput, and (2) the cost advantage does not improve by using a higher radix switch. Similarly, Xpander has used ideal routing on all-to-all traffic matrices to estimate the throughput, and has shown that their topology is more cost efficient than Fat-tree, and allows incremental expandability up to any size with minor throughput loss. In §L, we show that throughput of Xpander can drop significantly when

\[ \text{Table 4: Throughput vs. Bisection Bandwidth. Conclusions can change significantly.} \]

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Topology} & \textbf{N} & \textbf{H} & \textbf{BBW} & \textbf{Throughput} \\
\hline
Jellyfish & 32K & 10 & 3:4 & 1/2 \\
Xpander & 32K & 10 & 3:4 & 1/2 \\
FatClique & 32K & 8.6 & 3:4 & 2/3 \\
Clos & 32K & 32 & 1:2 & 1/2 \\
\hline
\end{tabular}
\end{table}

\[ \text{Table 5: Throughput-based vs BBW-based over-subscription ratio. Numbers in one row are computed on the same topology.} \]

\[ \text{\textsuperscript{2}The instance of FatClique we chose for this experiment uses a different } H \text{ than the instances of Jellyfish and Xpander, which is why it has a different throughput.} \]
Jellyfish supports 27% more servers at full throughput than a (same-equipment) Fat-tree (at ~900 servers) and this advantage improves by using higher port switches.

Xpander uses 80% – 95% switches to support the same number of servers as Fat-tree at the scale of ~4K servers. At the largest considered size in [47], Xpander uses more than 95% switches. However, at larger scale (~40K servers), Xpander uses 80% switches (matching the number reported in [47]).

Xpander using random rewiring can be incrementally expanded to any size while preserving high performance.

At some scales, Xpander can be as much as 20% less resilient compared to optimal resiliency using 32-port switches.

Failure Resiliency. Prior work has explored the resiliency of Jellyfish [44] and Xpander [47] to random link failures for relatively small topologies (at the scale of a few thousand servers). To do this, they compute the throughput achieved by ideal routing (using multi-commodity flow, which limits scaling) for a few randomly chosen permutation matrices. The showed that, at these scales, these topologies degrade gracefully, defined as follows. If \( \Theta \) is the throughput of a topology without failure, and a randomly chosen fraction \( f \) of all links fail, then the nominal throughput under failure is \( (1-f) \Theta \). The minimal permutation matrix (Figure 4(b)) has used a similar definition to assess failure resilience in WAN switches. We say a topology degrades gracefully if the actual throughput (in our experiments, the throughput upper bound) under failure closely matches the nominal throughput under failure.

This relationship between deviation from the nominal, and the number of shortest paths, is more evident when comparing Figure 10(c) with Figure 4(b). The former plots the root mean square deviation from the nominal as a function of topology size. In the latter, the number of shortest paths decreases steadily from 24K to 131K; in Figure 10(c), the deviation increases correspondingly. Xpander exhibits same behavior as Jellyfish under random link failures.

Takeaway. This example illustrates how \( \text{TUB} \) can reveal previously unobserved properties of a topology at larger scales. Using our bound, we are able to measure the resiliency of uni-regular topologies for up to 131K. Using the throughput estimators in [44, 47] (full-blown MCF), we are unable to scale beyond 8K servers on our platform.

6 PRACTICAL CONSIDERATIONS

The importance of worst-case bounds. Focusing on worst-case bounds can result in pessimistic designs and evaluations. In many situations, it may be appropriate to focus on average case performance. However, datacenter topologies, once deployed, are used for several years [42]; in this time, traffic demands can grow significantly. Because it is hard to predict demand over longer time-frames, datacenter designers have focused on worst-case measures (like bisection bandwidth) as a design aid to maximize the lifetime of their designs. \( \text{TUB} \) follows this line of thinking: this paper shows that \( \text{TUB} \) is a better measure of worst-case performance for uni-regular topologies than bisection bandwidth.

Clos-based deployments. Most deployed datacenter designs today are Clos-based. However, designers are actively exploring other lower-cost designs, one of which is the spine-free design [22], in which the spine or topmost layer of switch blocks is replaced by direct connections between the intermediate-layer (or aggregation layer) pods [1]. Pods may carry transit traffic between other pods.

Practical Workloads. In this paper, we have compared full-bisection bandwidth topologies with full throughput topologies.
Deployed topologies are often over-subscribed; a deployed Clos might have less than full bisection bandwidth. These deployments work well because operators carefully manage datacenter workloads to ensure that they don’t exceed fabric capacity. They also leave spare capacity for management operations such as expansion and upgrade [42, 53]. For Clos, the bisection bandwidth of the oversubscribed topology is a good measure of the capacity. For uni-regular topologies, $t_{ub}$ is a better measure of capacity for an oversubscribed network (§5.1).

**Benchmarking routing designs.** Aside from topology, routing design also determines whether the datacenter is able effectively utilize its capacity in serving workloads. For uni-regular topologies, or variants thereof, $t_{ub}$ can be used to understand how well a proposed routing design can utilize capacity.

7 RELATED WORK

**Datacenter Designs.** Prior work has investigated a large body of topology designs focusing on high bisection bandwidth, cost-effective topologies with low diameter [8, 30, 44, 47, 52]. Our paper addresses the performance of many of these topology designs. We do not evaluate topologies such as SlimFly [6] and Dragonfly [30]. These focus on reducing latency, but, to scale to today’s datacenters, they generally need switches with much higher port counts than available with merchant silicon. For instance, with a 64-port switch, a SlimFly can support 32K servers, but a 4-stage Clos can accommodate 2.1M. We emphasize that $t_{ub}$ applies to these two topologies as well as they are uni-regular. Prior work has described server-centric topologies such as DCell [17] and BCube [16] which equip servers with multiple ports and route packets through servers. Server-based forwarding can be highly unreliable [42], so deployed datacenters have not adopted these designs, and we have not considered these in this paper. Future work can explore throughput bounds for this class of topologies.

A more recent direction focuses on reconfigurable topology designs [13, 14, 20, 37, 38, 54] that adapt the topology in response to the observed traffic. Most reconfigurable topology designs adapt instantaneously to shifts in traffic demand, and attempt to minimize flow completion times. To the extent that each adapted topology is uni-regular or bi-regular, Theorem 2.2 will apply to the topology. However, we have left it to future work to understand how topology throughput relates to the objective of minimizing flow completion times, the focus of topology reconfiguration.

**Throughput.** As discussed earlier, significant prior work exists on throughput in datacenters. Some work [50] has explored the application-level throughput under different traffic conditions. Prior work has developed a theoretical understanding of throughput [12, 27, 43]. Of these, [12] compares performance of 3 throughput-approximating algorithms (Jain [24], Hoefler [23, 51], and an LP-based approximation), and show that Jain method is a more accurate approximation model compared to the other two in capturing the average throughput over all the flows. More recently, [43] focuses on approximating average throughput under uniform traffic, and [27] studies the relationship between traffic-dependent sparsest-cut and throughput at the scale of few thousand servers. Inspired especially by the latter two papers, we derive a tight throughput upper bound across all traffic matrices and explore it to understand practical scaling limits for uni-regular topologies, and the utility of a throughput-centric view in evaluating properties of datacenter topologies. We also compare $t_{ub}$ against many of these prior approaches.

**Practical Routing.** In practice, throughput highly depends on the routing algorithm and the underlying topology. ECMP is optimal for the Clos family [1, 15, 42]. For Jellyfish, Xpander, and FatClique, routing strategies like an ECMP-VLB hybrid [29] and FatPaths [7] have shown promising throughput performance. We have left it to future work to understand the gap between achievable throughput using these more practical routing strategies and $t_{ub}$.

8 CONCLUSIONS AND FUTURE WORK

This paper broadens our understanding of the throughput metric for datacenter topology performance, and its relationship to bisection bandwidth. We derive a closed-form expression for the upper bound of the throughput ($t_{ub}$) of a given topology that is independent of routing. This bound applies to most proposed datacenter topologies. For a sub-class of these designs, uni-regular topologies, we are able to derive an upper-bound on throughput that applies to any instance in this sub-class, using which we show that uni-regular topologies are fundamentally limited: beyond a certain scale, they cannot have full throughput even if they have full bisection bandwidth. In practice, many instances of uni-regular topologies with 10-15K servers cannot have full throughput. Finally, we demonstrate that $t_{ub}$ to evaluate properties of a topology can result in different conclusions compared to using other metrics. Future work can explore the throughput gap between $t_{ub}$ and the throughput achievable using practical routing algorithms, explore the throughput of Clos-variants like [36], scale $t_{ub}$ to even larger topologies, and improve its tightness.

**Acknowledgements.** We thank our shepherd Michael Schapira, and the anonymous reviewers for their feedback on the paper. This material is based upon work supported by the U.S. National Science Foundation under grants No. CNS-1901523, CNS-1705086, and CNS-1955422.


APPENDIX

Appendices are supporting material that have not been peer-reviewed.

A Proof of Theorem 2.1

Proof. In a doubly-stochastic matrix, each row and each column contain non-negative values that add up to 1. The Birkhoff-von Neumann theorem states that the $n \times n$ permutation matrices form the vertices of the convex polytope containing the set of $n \times n$ doubly-stochastic matrices. We observe that $\mathcal{T}$ contains all doubly-stochastic matrices scaled by $H$. From the Birkhoff-von Neumann theorem, it follows that the vertices of the convex polytope containing $\mathcal{T}$ is the set of traffic matrices in $\hat{T}$. It remains to show that the minimum throughput across $\hat{T}$ is always equal to that across $\mathcal{T}$.

To prove that $\min_{T \in \mathcal{T}} \theta(T) = \min_{T \in \hat{T}} \theta(T)$, let $\theta^* = \theta(T^*)$ be the minimum of the LHS achieved at traffic matrix $T^* \in \mathcal{T}$. We will show by contradiction that at least one permutation traffic $T \in \hat{T}$ leads to this $\theta^*$. Specifically, let $\theta^* = \min_{T \in \hat{T}} \theta(T)$. Suppose there is no such permutation traffic matrix. Let $\hat{\theta} > \theta^*$ and $\bar{\theta} = \min_{T \in \hat{T}} \theta(T)$ be the minimum achieved by some permutation traffic matrix in $\hat{T}$. Carathéodory’s theorem [5] implies that there exists at most $|\mathcal{K}|^2 + 1$ permutation traffic matrices ($T_x$) in $\hat{T}$ such that

$$T^* = \sum_{x=1}^{|\mathcal{K}|^2+1} \alpha_x T_x, \quad \sum_{x=1}^{|\mathcal{K}|^2+1} \alpha_x = 1, \quad \alpha_x \in [0,1] \quad \forall x.$$

Given this, we can use a convex combination of permutation traffic matrices ($T_x$) and ($\bar{\alpha}_x$) to construct traffic matrix $T^*$ and a solution to the multi-commodity flow problem under $T^*$. The throughput of this solution cannot be less than $\hat{\theta}$, since all permutation traffic matrices have a throughput of at least $\bar{\theta}$. This leads to a contradiction, because we have assumed that $\theta^* < \bar{\theta}$. Thus, there must exist a permutation traffic matrix $T_x \in \hat{T}$ such that $\theta^* = \theta(T_x)$.

\[\Box\]

B Proof of Throughput Bound for uni-regular Topology

Proof. Let $\mathcal{K}$ denote the set of all switches with $H$ servers. Fix a permutation traffic matrix $T$ from $\hat{T}$. We solve a path-based multi-commodity flow problem (§H, commonly used in wide-area network traffic engineering [33]) that maximizes throughput $\theta(T)$ under this traffic matrix $T$. At each switch $u$, the ingress traffic consists of 1) traffic destined to servers attached to $u$ and 2) transit traffic $X_u(T)$. This ingress traffic is bounded by the capacity of network-facing ports, and we have $X_u(T) + \theta(T) \sum_{u \in \mathcal{K} \setminus \{u\}} t_{uv} \leq R_u - H_u$ for every $u \in \mathcal{K}$, where $R_u$ is the number of used ports in switch $u$. (This models the fact that, for many uni-regular topologies, some ports are left unused on switches.) Summing over $u \in \mathcal{K}$ gives

$$\sum_{u \in \mathcal{K}} X_u(T) \leq \sum_{u \in \mathcal{K}} (R_u - H_u) - \theta(T) \sum_{u \in \mathcal{K} \setminus \{u\}} t_{uv}. \quad (4)$$

The LHS of the above inequality is equal to the total transit traffic in the network caused by traffic matrix $T$. Alternatively, we can compute the total transit traffic based on the set of paths $P_{uv}$ and split ratios for those paths $\beta_p(T)$ as

$$\sum_{u \in \mathcal{K}} X_u(T) = \theta(T) \sum_{u \in \mathcal{K} \setminus \{u\}} \sum_{p \in P_{uv}} t_{uv} \sum_{p \in P_{uv}} \beta_p(T)(|\text{len}(p)| - 1). \quad (5)$$

Since all the paths in $P_{uv}$ are at least the shortest path and $\sum_{p \in P_{uv}} \beta_p(T) = 1$ for all $u, v \in \mathcal{K}$, we can rewrite the above equation as an inequality:

$$\sum_{u \in \mathcal{K}} X_u(T) \geq \theta(T) \sum_{u \in \mathcal{K} \setminus \{u\}} t_{uv}(L_{uv} - 1). \quad (6)$$

From Equation 4 and Equation 6, we have

$$\theta(T) \leq \frac{\sum_{u \in \mathcal{K}} (R_u - H_u)}{\sum_{u \in \mathcal{K}} \sum_{v \in \mathcal{K} \setminus \{u\}} t_{uv} L_{uv}}.$$

This throughput holds under every traffic matrix $T$ for every $T \in \hat{T}$. Taking the minimum over the set yields

$$\theta^* = \min_{T \in \mathcal{T}} \theta(T) \leq \min_{T \in \hat{T}} \theta(T) \leq \frac{\sum_{u \in \mathcal{K}} (R_u - H_u)}{\sum_{u \in \mathcal{K}} \sum_{v \in \mathcal{K} \setminus \{u\}} t_{uv} L_{uv}}.$$

Finally, using the facts that (a) $\sum_{u \in \mathcal{K}} (R_u - H_u) = 2E$, (b) every traffic matrix is a permutation traffic, and (c) the length of the shortest path from a switch to itself is equal to 0, we have the throughput upper bound in Equation 1.

\[\Box\]

C Proof of Throughput Bound for bi-regular Topology

Proof. Let $S$ and $\mathcal{K}$ denote the set of all switches and switches with $H$ servers respectively. Fix a permutation traffic matrix $T$ from $\mathcal{T}$. We solve a path-based multi-commodity flow problem that maximizes throughput $\theta(T)$ under this traffic matrix $T$. At each switch $u$, the ingress traffic consists of 1) traffic destined to servers attached to $u$ and 2) transit traffic $X_u(T)$. This ingress traffic is bounded by the capacity of network-facing ports, and we have $X_u(T) + \theta(T) \sum_{u \in \mathcal{K} \setminus \{u\}} t_{uv} \leq R_u - H_u$ for every $u \in \mathcal{K}$. Summing over $u \in \mathcal{K}$ gives

$$\sum_{u \in \mathcal{K}} X_u(T) \leq \sum_{u \in \mathcal{K}} (R_u - H_u) - \theta(T) \sum_{u \in \mathcal{K} \setminus \{u\}} t_{uv}. \quad (7)$$

Similarly, at every switch $u$ with no directly connected server, the ingress traffic only consists of transit traffic $X_u(T)$, and we have $X_u(T) \leq R_u - H_u$ for every $u \in S \setminus \mathcal{K}$. Summing over $u \in S \setminus \mathcal{K}$ gives

$$\sum_{u \in S \setminus \mathcal{K}} X_u(T) \leq \sum_{u \in S \setminus \mathcal{K}} (R_u - H_u). \quad (8)$$

From Equation 7 and Equation 8, we have

$$\sum_{u \in S} X_u(T) \leq \sum_{u \in S \setminus \mathcal{K}} (R_u - H_u) - \theta(T) \sum_{u \in \mathcal{K} \setminus \{u\}} t_{uv}. \quad (9)$$

The rest of the proof is similar to Theorem 2.2 (§B).

\[\Box\]
**Algorithm 1: Construction of traffic matrix**

**Input:** Topology $\mathcal{G} = (\mathcal{K}, \mathcal{E})$, Server per switch $H$

**Output:** Traffic matrix $T$

1. $Q \leftarrow \emptyset$
2. $T \leftarrow 0 \in \mathbb{R}^{\mathcal{K} \times \mathcal{K}}$
3. for $u \in \mathcal{K} \setminus Q$ do
   4. $v \leftarrow \arg\max_{v' \in \mathcal{K} \setminus Q} L_{uv'}$
   5. $(t_{uv}, t_{vu}) \leftarrow (H, H)$
   6. $Q \leftarrow Q \cup \{u, v\}$

$\Box$

## D Proof of Theorem 4.1.

**Lemma 8.1.** Given a uni-regular topology with total servers $N$ and $H$ servers per switch, for every switch $u$, the number of switches with at least $m$ hops away from the switch is at least

$$W_m = \frac{N}{H} - 1 - (R - H) \frac{(R - H - 1)^{m-1} - 1}{R - H - 2}, \quad m \in \{1, \ldots, d\}$$

where $d$ is the minimum diameter computed using Moore bound [39].

Proof. Fix switch $u$. Let $y_i$ be the number of switches with distance $i$ from switch $u$. Since every switch has $R - H$ switch-to-switch ports, the number of switches with distance $i$ from $u$ is bounded by $y_i \leq N - H$. The number of switches with distance $i$ hops away from switch $u$ can be recursively bounded by $y_i \leq (R - H - 1)y_{i-1} = (R - H - 1)^i(R - H)$, as each $i$-th switch has one port connecting to $(i - 1)$-th switch. Since there are total $N/H$ switches, the number of switches with at least $m$ hops away from switch $u$ is

$$\frac{N}{H} - 1 - \sum_{i=1}^{m-1} y_i \quad \text{and at least}$$

$$\frac{N}{H} - 1 - \sum_{i=1}^{m-1} y_i \geq \frac{N}{H} - 1 - (R - H) \frac{(R - H - 1)^{m-1} - 1}{R - H - 2} \quad \text{where} \quad d$$

Algorithm 1 generates a traffic matrix with high pair-wise shortest path length. In each iteration (Line 3-7), from unpicked switches, it arbitrarily picks a switch $u$ and then a switch $v$ which maximizes the shortest path length from $u$ (Line 4). Then, it updates entries $t_{uv}$ and $t_{vu}$ of the traffic matrix $T$ with $H$.

**Lemma 8.2.** Given a uni-regular topology with total servers $N$ and $H$ servers per switch, Algorithm 1 constructs a traffic matrix with at least $W_m$ non-zero entries whose shortest path lengths are at least $m$, for $m \in \{1, \ldots, d\}$.

Proof. We will show that there are at least $W_m$ non-zero entries whose shortest path lengths are at least $m$ at the end of $k_m$-th iteration of Algorithm 1 for every $m$. Fix $m$ and $W_m$ from Lemma 8.1. Let $Q_k$ be the set of switches already picked after $k$-th iteration and $Q_0 = \emptyset$. In the $k$-th iteration, switches $u$ and $v$ are picked from unpicked switches in $\mathcal{K} \setminus Q_{k-1}$ such that $v$ maximizes the shortest path length from $u$. Let $\mathcal{V}_m^u$ denote the set of switches with distance of at least $m$ hops from switch $u$. We observe that (a) if $|\mathcal{V}_m^u \setminus Q_{k-1}|$ is non-empty, $v$ will be picked from $\mathcal{V}_m^u \setminus Q_{k-1}$; (b) if $W_m - 2(k - 1) > 0$, then $\mathcal{V}_m^u \setminus Q_{k-1}$ is non-empty because $|\mathcal{V}_m^u \setminus Q_{k-1}| \geq |\mathcal{V}_m^u| - |Q_{k-1}| \geq W_m - 2(k - 1) > 0$. (We use Lemma 8.1 that $|\mathcal{V}_m^u| \geq W_m$ and the fact that $|Q_{k-1}| = 2(k - 1)$.) Then, we choose $k_m = \lfloor (W_m + 1)/2 \rfloor$, which always exists because $W_m$ is monotonically decreasing and at the highest $W_1 = |\mathcal{K}|-1$, the chosen $k_1 = \lfloor |\mathcal{K}|/2 \rfloor$ is feasible. Therefore, in the $k_m$-th iteration, we have $W_m - 2(k_m - 1) > 0$ (satisfying (b)), so $|\mathcal{V}_m^u \setminus Q_{k-1}|$ is non-empty (satisfying (a)), and $v$ is picked from $\mathcal{V}_m^u$. Thus, at the end of the iteration, there are $2k_m$ pairs and all of them have shortest path lengths at least $m$ since they are selected from $\bigcup_{u \in \mathcal{K}} \mathcal{V}_m^u$. Further, their number is at least $W_m$ because $2k_m = 2\lceil (W_m + 1)/2 \rceil \geq W_m$. $\Box$

**Lemma 8.3.** Given a uni-regular topology with total servers $N$ and $H$ servers per switches, a traffic matrix $T$ constructed from Algorithm 1 has the following property:

$$\max_{T \in T_{\{u,v\} \subseteq \mathcal{K}}} \sum_{(u,v) \in \mathcal{K}^2} L_{uv} \left| t_{uv} - 0 \right| \geq \sum_{m=1}^{d} W_m,$$

where $W_m$ for $m \in \{1, \ldots, d\}$ is defined in Lemma 8.1 and $d$ is the minimum diameter from Moore bound [39].

Proof. Since the traffic matrix $T$ constructed from Algorithm 1 is a permutation traffic matrix, it follows that

$$\max_{T \in T_{\{u,v\} \subseteq \mathcal{K}^2}} \sum_{(u,v) \in \mathcal{K}^2} L_{uv} \left| t_{uv} - 0 \right| \geq \sum_{(u,v) \in \mathcal{K}^2} L_{uv} \left| t_{uv} - 0 \right|.$$

It remains to show that $\sum_{(u,v) \in \mathcal{K}^2} L_{uv} \left| t_{uv} - 0 \right| \geq \sum_{m=1}^{d} W_m$. In the traffic matrix $T$, let $\mathcal{V}_m$ be the set of switch pairs whose shortest path lengths are at least $m$ hops. From the definition, we know that $\mathcal{V}_2 \subseteq \mathcal{V}_{d-1} \subseteq \ldots \subseteq \mathcal{V}_1$, and $\mathcal{V}_m \setminus \mathcal{V}_{m+1}$ only contains switch pairs with exactly $m$ hops for $m \in \{1, \ldots, d\}$. It follows that

$$\sum_{(u,v) \in \mathcal{K}^2} L_{uv} \left| t_{uv} - 0 \right| \geq \sum_{m=1}^{d} m |\mathcal{V}_m \setminus \mathcal{V}_{m+1}|$$

Applying the fact that $|\mathcal{V}_m| \geq W_m$ for every $m \in \{1, \ldots, d\}$ from Lemma 8.1, we have

$$\sum_{(u,v) \in \mathcal{K}^2} L_{uv} \left| t_{uv} - 0 \right| \geq \sum_{m=1}^{d} W_m.$$
From Equation 12 and using the fact that in uni-regular topologies, $2E = H (R - H)$, we have the upper bound in Equation 2. □

E Asymptotic behavior of throughput gap

In §3.1, we pointed out that the throughput gap for Jellyfish might be expected to be non-zero in the range 100K – 180K servers, but could not confirm this because our KSP-MCF implementation does not scale to these sizes. To be able to quantify the throughput gap for topologies larger than our computational limit for KSP-MCF, we expect our throughput bound to match the actual gap approaches zero asymptotically. In other words, for very large observation in Corollary 2 showing that the theoretical throughput gap is a parameter to the lower bound calculation) in Theorem 8.4. Define the theoretical throughput gap to be the difference between the upper and lower bounds (for a given $M$). Intuitively, the theoretical throughput gap shows the maximum possible gap one can expect when using our bound in Theorem 2.2. Figure A.1 shows that the magnitude of the theoretical gap as a function of the topology size. (we use $M = 1$; at this setting, each topology has at least 300 distinct paths between each source-destination pair across the entire range of topology sizes we have considered, which is sufficient for our path-based MCF computation §H).

Figure A.1 shows that the maximum possible gap at these scales is going to be smaller than that of $3K - 15K$. Moreover, the theoretical gap decreases as the size of the topology grows. We prove this observation in Corollary 2 showing that the theoretical throughput gap approaches zero asymptotically. In other words, for very large topologies, we expect our throughput bound to match the actual topology throughput.

![Figure A.1: Theoretical throughput gap.](image)

We first start by stating the following assumption that always holds in all of our experiments.

**Assumption 1.** Given a traffic matrix $T$ and a corresponding solution of our path-based MCF, the ingressing capacity of network-facing ports is saturated by traffic at every switch:

$$X_u(T) + \theta(T) \sum_{v \in K \setminus \{u\}} t_{uv} = R_u - H_u \quad \text{for every} \quad u \in K$$

$$X_u(T) = R_u \quad \text{for every} \quad u \in S \setminus K,$$

where $X_u(T)$ is the amount of transit traffic on switch $u$ as a result of routing the traffic matrix $T$. Note that $H_u = 0$ for every switch with no servers, and it is omitted in the second equality.

Intuitively, the assumption holds in practice because datacenter topologies are designed such that all the link capacities can be fully utilized, as are the ingress capacities. We use this assumption to prove a bound on throughput gap. Let $M$ denote the additive path length such that every path length is bounded by

$$\text{len}(p) \leq L_{uv} + M \quad \text{for every} \quad p \in P_{uv}, \quad \text{and every} \quad (u, v) \in K^2.$$  

**Theorem 8.4.** Under a permutation traffic matrix $T \in \mathcal{T}$, when Assumption 1 holds with the additive path length $M_T$ (depending on $T$), the maximum achievable throughput of a topology (either uni-regular or bi-regular) is at least:

$$\theta(T) \geq \frac{2E}{NM_T + H \sum_{(u,v) \in K^2} L_{uv} \sum t_{uv} > 0}.$$  

**Proof.** Let $S$ denote the set of all switches. From Assumption 1, we sum the transit traffic $X_u(T)$ over all switches and have the following equality

$$\sum_{u \in S} X_u(T) = \sum_{u \in S} (R_u - H_u) - \theta(T) \sum_{u \in S} \sum_{v \in K \setminus \{u\}} t_{uv}. \quad (14)$$

Note that Assumption 1 changes the inequality in Equation 9 to equality due to all ingress capacity is fully utilized.

Alternatively, we can compute the total transit traffic ($\sum_{u \in S} X_u(T)$) based on Equation 5;

$$\sum_{u \in S} X_u(T) = \theta(T) \sum_{u \in S} \sum_{v \in K \setminus \{u\}} t_{uv} \sum_{p \in P_{uv}} \beta_p(T)(\text{len}(p) - 1).$$

Since length of all the paths in $P_{uv}$ is at most $L_{uv} + M_T$ from the definition of the additive path length, we have;

$$\sum_{u \in S} X_u(T) \leq \theta(T) \sum_{u \in S} \sum_{v \in K \setminus \{u\}} t_{uv}(L_{uv} + M_T - 1). \quad (15)$$

From Equation 14 and Equation 15, we have

$$\theta(T) \geq \frac{\sum_{u \in S} (R_u - H_u)}{\sum_{u \in S} \sum_{v \in K \setminus \{u\}} t_{uv}(L_{uv} + M_T)}.$$  

Finally, using the fact that a) $\sum_{u \in S} (R_u - H_u) = 2E$, b) $T$ is a permutation traffic matrix, c) $L_{uv} = 0$ for every switch $u$ and d) the sum of all the entries except the diagonals of the traffic matrix $T$ is at most $N_i$; we can derive the throughput lower bound in Equation 13. □

The above theorem states the lower bound of throughput with respect to the additive path length $M_T$ depending on a given permutation traffic matrix $T$. Our path-based MCF computation shows that $M_T = 1$ is sufficient to provide enough path diversity to make Assumption 1 valid for all Jellyfish, Xpander and FatClique. Using Theorem 8.4, we show that the gap between the upper bound and the lower bound can be arbitrarily small when the network size is sufficiently large and when a mild assumption holds.

**Assumption 2.** The additive path length for the maximal permutation traffic matrix $\hat{T}$ does not increase with a topology size such that $M_T = O(1)$.
COROLLARY 2. When Assumptions 1 and 2 hold, for any positive value \( \epsilon > 0 \), any uni-regular topology with \( N \) servers has \( N_\epsilon^u \) such that for every \( N \geq N_\epsilon^u \):

\[
\theta^* - \theta_{ib} \leq \epsilon
\]

where \( \theta^* \) is the throughput upper bound from Theorem 2.2 and \( \theta_{ib} = \min_{T \in T^\ell} \theta_{ib}(T) \) is the minimum of throughput lower bound \( \theta_{ib}(T) \) from Theorem 8.4.

Proof. From Assumption 1, it holds for every permutation matrix \( T \in T^\ell \) that

\[
\theta^* - \theta_{ib}(T) \leq \theta^* - \min_{T \in T^\ell} \theta_{ib}(T) \leq \frac{2E}{(NM_F + H \sum_{(u,v) \in K^2} L_{uv}[L_{uv} > 0])}
\]

Let \( \hat{T} = [t_{uv}] \) be the maximal traffic matrix that minimizes the right side of Equation 1. We observe that it also minimizes the last term above, and we have

\[
\theta^* - \theta_{ib}(T) \leq \frac{2ENM_F}{(NM_F + H \sum_{(u,v) \in K^2} L_{uv}[L_{uv} > 0])}
\]

Using Lemma 8.3 and Lemma 8.1, we have;

\[
\sum_{(u,v) \in K^2} L_{uv}[L_{uv} > 0] \geq \frac{d(N}{H - 1) - \frac{R - H}{R - H - 2} \left( \frac{(R - H - 1)^{d - 1}}{R - H - 2} - d \right) = D.
\]

Equation 16 and Equation 17 lead to

\[
\theta^* - \theta_{ib}(T) \leq \frac{2ENM_F}{(NM_F + HD)(HD)}
\]

Since the above inequality holds for every \( T \in T^\ell \), it holds at the worst-case gap

\[
\theta_{ub} - \min_{T \in T^\ell} \theta_{ib}(T) \leq \frac{2ENM_F}{(NM_F + HD)(HD)}.
\]

Similar to Corollary 1, we can prove that above inequality goes to 0 as \( N \) increases because every \( MF \) is bounded by a constant independent of \( N \) under Assumption 2. \( \square \)

F Throughtput of bi-regular Clos topologies under TUB

TUB is tight for bi-regular Clos topologies as well, giving throughput equal to 1 for different topology sizes (Table A.1).

<table>
<thead>
<tr>
<th>N</th>
<th>#Layers</th>
<th>#SWs</th>
<th>TUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>8192</td>
<td>3</td>
<td>1280</td>
<td>1.00</td>
</tr>
<tr>
<td>32768</td>
<td>4</td>
<td>7168</td>
<td>1.00</td>
</tr>
<tr>
<td>131072</td>
<td>4</td>
<td>28672</td>
<td>1.00</td>
</tr>
</tbody>
</table>

**Table A.1:** Clos: TUB is always 1.

G Proof of Corollary 1

Proof. This follows directly from Equation 2 in Theorem 4.1. We can show that, in \( D \), the term containing \( Nd \) dominates the other terms for large enough \( N \). This is a direct consequence of defining \( d \) as the minimum diameter that required to accommodate \( N/H \) switches (Moore bound [39]). As a result, in the RHS of the Equation 2, the numerator grows as \( N \) and the denominator grows as \( Nd \). Therefore, \( \theta^* \) approaches zero with increasing \( N \), so there must always exist a \( N^* \) at which \( \theta^* \) falls below 1. \( \square \)

H Path-based Multi-commodity Flow LP formulation

In this section, we briefly introduce the path-based MCF formulation (common in WAN traffic engineering [33]) used throughout the paper. Given a traffic matrix \( T = [t_{uv}] \) and set of paths between every pair of switches with servers \( (P_{uv}) \), the throughput of the optimal traffic matrix is the solution to the following LP formula in which \( f_p \) denotes the amount of flow on path \( p \):

maximize \( \theta \)

subject to

\[
\sum_{p \in P_{uv}} f_p \geq t_{uv} \quad \forall (u, v) \in K^2
\]

\[
\sum_{(u,v) \in K^2} \sum_{p \in P_{uv}} f_p \leq |v| \quad \forall e \in E
\]

\[
f_p \geq 0 \quad \forall (u,v) \in K^2, \forall p \in P_{uv},
\]

where \( E \) is the set of directional links with unit capacity.

I Metric Adjustments for FatClique

In a FatClique, the number of servers attached to each switch can differ by at most 1. To generalize the maximal permutation traffic matrix generation to accommodate this case, we changed weight assignment of edges in the complete bipartite graph from \( w_{u,v} = L_{uv} \) to \( w_{u,v} = L_{uv} \min(H_u, H_v) \). The latter weight assignment takes into account the maximum amount of flow between each \( u,v \) pair along with their distance. More precisely, if in a permutation traffic matrix \( t_{uv} \) is non zero, it should be the minimum of \( H_u \) and \( H_v \) since it should conform to the hose-model traffic constraints §2. So, Equation 1 can be re-written as;

\[
\theta^* \leq \min_{T \in T} \frac{2E}{\sum_{(u,v) \in K^2} L_{uv} \min(H_u, H_v) t_{uv} [t_{uv} > 0]}
\]

Equation 18 is exactly same as Equation 1 when all the switches have exactly the same \( H \). To find the maximal permutation traffic matrix, we need to find the traffic matrix that minimizes the LHS of Equation 18. This is equivalent to solving the maximum weight matching in a bipartite graph (§2), with the revised weight assignment.

This approach does not yield the global minimum of the throughput bound since Theorem 2.1 does not hold when \( H \) differs across the switches. A linear programming (LP) formulation can compute the global minimum [31]. However, we use our matching method to infer the maximal permutation traffic matrix for FatClique, for three reasons. First, in FatClique, the number of servers connected to each switch can differ only by 1, so the difference between global minimum and throughput bound computed using this approach is negligible. Second, algorithms for solving maximal weight matching are more efficient than solving an LP. Third, the permutation traffic
matrix generated using our approach is harder to route compared to an LP generated traffic matrix.

J Throughput Gap for different values of $K$
Figure A.5 illustrates the absolute difference between path-based multi-commodity flow over $K$-shortest paths and our throughput bound for different values of $K$ (i.e., throughput gap). The results for $K = 60, 100, 200$ are very similar to each other; a gap of non-zero for small size topologies, followed by a close-to-zero gap for larger instances. The only exception is some instances of FatClique exhibit large throughput gaps in the 5K – 15K compared to Jellyfish and Xpander because FatClique cannot fully utilize available capacity with $K = 60, 100$ for KSP-MCF. However, after increasing $K$ to 200, the throughput gap decreases for Jellyfish.

For $K = 20$, the gap remains significant even at large topologies since 20-shortest paths do not provide enough diversity to completely exploit the network capacity, and some of the capacity remains unused.

K Scaling of Throughput-based Cost Comparison
Other than bisection bandwidth, Jellyfish [44] and Xpander [47] use full throughput of random permutations and all-to-all traffic matrices under MCF to assess the cost advantage of their topologies. However, throughput under random permutations and all-to-all traffic matrices can be significantly larger than (worst-case) throughput [27]. Moreover, as discussed in §3.1, MCF and KSP-MCF cannot scale to the size of current datacenters. In this section, we show how conclusions can change when using our bound to perform cost comparisons at larger scale.

Jellyfish. Singla et al. [44] have shown that at the scale of <900 servers Jellyfish can support 27% more servers than a Fat-tree [1] built with same equipment, and conjecture that this cost advantage increases by using a higher radix switch. Figure A.2 shows the relative difference of the maximum servers between Jellyfish and Fat-tree for different switch radices. Using $\text{TUB}$, at the scale of 686 servers ($R = 14$, which is the largest scale considered in [44]), Jellyfish can support only 8% more servers than a (same equipment) Fat-tree (the leftmost point in Figure A.2), dropping the cost advantage of Jellyfish by 3x. Moreover, using a higher radix switch does not result in higher cost advantage of Jellyfish over Fat-tree. In fact, using a higher radix switch might result in drop in the cost advantage. For example, using 98-port switches instead of 64-port causes the cost advantage to drop slightly from 25% to 22%.

Xpander. Valadarsky et al. [47] have shown that at the scale of <4K servers, Xpander can support the same number of servers as Fat-Tree [1] at full throughput using 80% – 85% of the switches. As Figure A.3 shows at the maximum considered scale in [47] (3.5K servers, the left most point), Xpander should use more than 95% switches compared to the same size Fat-tree. However, as the scale grows, the cost advantage of Xpander over Fat-tree increases, matching the numbers reported in [47].

L Throughput of uni-regular topologies under expansion
Jellyfish [44] and Xpander [47] have shown that using a very simple expansion algorithm (random rewiring), their design can be expanded to any size with minor throughput loss while preserving the number of servers per switch $H$. Jellyfish uses bisection bandwidth as their throughput metric while Xpander assesses the throughput by solving MCF on all-to-all traffic matrix.

Jellyfish. In §5.1, we show that Jellyfish requires advanced planning in order to preserve full throughput, otherwise, even very small expansion can turn Jellyfish into a topology with less than full throughput. To better understand the amount of throughput degradation, Figure A.4 shows the throughput (computing using $\text{TUB}$, normalized by the topologies initial throughput (before expansion). At each step, we expand the topology by 20% of the initial size until its size reaches the 2.6x of the initial topology. For 10K servers, Figure A.4 shows that throughput drops by more than 20% when expanding the topology by only 0.6x. On the other hand, when the initial topology size is 32K, throughput drop is negligible (<1%). We emphasize that these results are consistent with §4.2: Jellyfish with $H=6$ and initial size 8K has full throughput even after expanding by 2.6x. However, it faces the throughput drop as well.

This suggests that operators should be cautious when expanding uni-regular topologies depending on the topology’s initial and target size as they might face significant throughput drops. $\text{TUB}$, therefore, helps topology designers to identify and understand these scenarios before deploying and expanding their desired topology.

Xpander. Using $\text{TUB}$ to assess the Xpander’s performance under expansion results in similar conclusions as expanding Jellyfish does. Similar to Jellyfish, operators who adopt Xpander should have the
target size in mind and choose $H$ accordingly. Otherwise, they either end up having a topology with less than full throughput or have to rewire the servers, bearing a significant cost. The throughput degradation is also very similar to Jellyfish (Figure A.4); at some scales (e.g., 10K), expanding the Xpander even by a very small ratio degrades the throughput by as much as 25%.

**Figure A.4:** Throughput of uni-regular topologies under expansion.
Table A.5: Throughput bound vs K-shortest paths Multi-commodity flow for different values of K (20, 60, 100, 200).

- (a) Jellyfish, K=20
- (b) Jellyfish, K=60
- (c) Jellyfish, K=100
- (d) Jellyfish, K=200
- (e) Xpander, K=20
- (f) Xpander, K=60
- (g) Xpander, K=100
- (h) Xpander, K=200
- (i) FatClique, K=20
- (j) FatClique, K=60
- (k) FatClique, K=100
- (l) FatClique, K=200