Cerberus: The Power of Choices in Datacenter Topology Design*

A Throughput Perspective

CHEN GRINER, School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel
JOHANNES ZERWAS, Department of Electrical and Computer Engineering, Technical University of Munich, Germany
ANDREAS BLENK, Department of Electrical and Computer Engineering, Technical University of Munich, Germany & Faculty of Computer Science, University of Vienna, Austria
MANYA GHOBADI, Computer Science and Artificial Intelligence Laboratory, MIT, USA
STEFAN SCHMID, TU Berlin, University of Vienna & Fraunhofer SIT, Germany
CHEN AVIN, School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel

The bandwidth and latency requirements of modern datacenter applications have led researchers to propose various topology designs using static, dynamic demand-oblivious (rotor), and/or dynamic demand-aware switches. However, given the diverse nature of datacenter traffic, there is little consensus about how these designs would fare against each other. In this work, we analyze the throughput of existing topology designs under different traffic patterns and study their unique advantages and potential costs in terms of bandwidth and latency "tax". To overcome the identified inefficiencies, we propose Cerberus, a unified, two-layer leaf-spine optical datacenter design with three topology types. Cerberus systematically matches different traffic patterns with their most suitable topology type: e.g., latency-sensitive flows are transmitted via a static topology, all-to-all traffic via a rotor topology, and elephant flows via a demand-aware topology. We show analytically and in simulations that Cerberus can improve throughput significantly compared to alternative approaches and operate datacenters at higher loads while being throughput-proportional.

CCS Concepts: • **Networks → Network design principles**; **Network performance analysis**; **Data center networks**.

Additional Key Words and Phrases: throughput; network design; data centers; demand-aware; optical networks

*The first two authors contributed equally to this work.

Authors' addresses: Chen Griner, School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel, griner@post.bgu.ac.il; Johannes Zerwas, Department of Electrical and Computer Engineering, Technical University of Munich, Germany, johannes.zerwas@tum.de; Andreas Blenk, Department of Electrical and Computer Engineering, Technical University of Munich, Germany & Faculty of Computer Science, University of Vienna, Austria, andreas.blenk@tum.de; Manya Ghobadi, Computer Science and Artificial Intelligence Laboratory, MIT, Boston, USA, ghobadi@csail.mit.edu; Stefan Schmid, TU Berlin, University of Vienna & Fraunhofer SIT, Berlin, Germany, stefan.schmid@tu-berlin.de; Chen Avin, School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel, avin@cse.bgu.ac.il.

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.
© 2021 Association for Computing Machinery.
2476-1249/2021/12-ART38 $15.00
https://doi.org/10.1145/3491050

ACM Reference Format:
Chen Griner, Johannes Zerwas, Andreas Blenk, Manya Ghobadi, Stefan Schmid, and Chen Avin. 2021. Cerberus: The Power of Choices in Datacenter Topology Design: A Throughput Perspective. *Proc. ACM Meas. Anal. Comput. Syst.* 5, 3, Article 38 (December 2021), 33 pages. https://doi.org/10.1145/3491050

1 INTRODUCTION

Datacenter networks have become a critical backbone of our digital society. The traffic these networks need to serve is growing explosively, and researchers are exploring novel *optical* datacenter networks, including both *static* and *dynamic* topologies. Dynamic topologies provide additional design freedom and can help overcome the limits on the application performance [1–19].

Diversity of datacenter traffic patterns. Datacenter traffic is not only growing quickly as an aggregate, but the expanding variety of cloud applications also creates complex traffic patterns that come in many different flavors and with different performance requirements [20–22]. For instance, Hadoop-like applications exhibit an all-to-all traffic pattern [23], short query traffic has a skewed pattern [24], while all-reduce in machine learning (ML) applications exhibit a ring or tree-like traffic pattern [25–27]. Cutting-edge datacenter architecture proposals hence need to satisfy several desired requirements simultaneously. They should support high capacity and throughput, be throughput-proportional [28], minimize 99% tail latency for short flows, and maintain a low hop-count. All of these requirements must be satisfied for a range of traffic loads and applications, which may also change over time.

Diversity of datacenter topology designs. In parallel to the growing variety of traffic patterns, the architectural design choices of operators are expanding. In particular, there exist several fundamentally different optical datacenter topologies, relying on different switching technologies [29]. We can classify these topologies along the two independent dimensions, *static vs dynamic* and *demand-oblivious vs demand-aware* and we identify *three* main topology types: (i) Traditionally, datacenter networks are based on *static* and *demand-oblivious* topologies, e.g., Clos and expander graphs [30–37], (ii) More recent proposals also explore *dynamic* but *demand-oblivious* topologies, e.g., relying on rotor switches that periodically reconfigure the topology [1, 4, 5]. (iii) Furthermore, there are *dynamic* and *demand-aware* topologies that can be reconfigured according to the current traffic pattern [2, 3, 7, 10, 11, 18, 19, 38, 39]. However, there is little consensus in the networking community on how these different designs fare against each other [5, 30], in particular when it comes to *throughput* [40]. What is more, we currently lack a unified model and analytical tools to close this gap.

Matching traffic patterns to topology designs. This paper is motivated by the observation that existing datacenter network designs in many cases provide a *mismatch* between some common traffic patterns and the switching technology used in the network topology to serve them. In particular, different optical topologies provide different tradeoffs and the used topology should depend on the demand. For example, mice flows that are time-sensitive should be served on a static topology; transmitting them on dynamic topologies that require reconfiguration times may violate their latency constraints and result in high flow completion times. Elephant flows (e.g., in ML [41] or Datamining [42]) on the other hand, may benefit from dynamic demand-aware topologies. Since the reconfiguration time is relatively small compared to the transmission time of such large flows, the reconfiguration can be amortized, and throughput improved by establishing direct links between frequently communicating pairs. To give one more example, it has been shown that shuffling traffic of map-reduce applications, due to its all-to-all nature, can profit from dynamic demand-oblivious topologies providing periodic direct connectivity between *all* rack pairs [1, 4, 5].

A key concern: throughput. Our main metric of interest is the end-to-end *throughput* supported by systems in a fluid-flow model. We follow the throughput definition by Jyothi et al. [40] and focus

on the network topology (ignoring e.g., congestion control). The *throughput* of a system for a traffic demand matrix T is the highest scaling factor $\theta(T)$ for which the traffic is feasible in the system. That is, we seek the maximum scaling factor for which there exists a feasible multi-commodity *network flow* assignment that routes the traffic in the matrix $\theta(T) \cdot T$ through the network from each source to each destination. A feasible flow means that the flow rates are subject to link capacities and to flow conservation at each intermediate node. The throughput of a topology, denoted by θ^*, is defined as the worst-case throughput among all traffic matrices [40].

Potential inefficiencies: bandwidth and latency tax. To compare the throughput of different topologies, we propose to quantify their inefficiencies in terms of *taxes*. Static topologies require *multi-hop* forwarding. This can be problematic, especially for large flows: the more hops a flow must traverse, the more network capacity is consumed. As noticed in prior work [4], inefficiency arising from multi-hop routing can be seen as a "bandwidth tax" (BW-Tax). In contrast, in dynamic networks, the topology may be reconfigured to provide *direct* connectivity to elephant flows [4, 14], but at the cost of a reconfiguration time. For example, the reconfiguration time of rotor switches can be in the order of $10\mu s$ per slot [4] while reconfiguring demand-aware optical switches may take in the order of $15ms$ [43, 44]. In general, we can regard the reconfiguration time as a "latency tax" (LT-Tax). Regarding the design choice between static vs dynamic topology, we therefore observe that whereas *dynamic topologies introduce a latency tax, static topologies introduce a bandwidth tax.*

In this paper, we present a *systematic* approach to study the throughput of both static and dynamic topologies for different traffic types. Concretely, we make the following contributions.

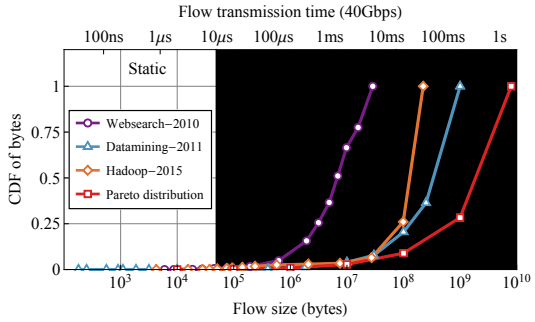
Contribution 1: Throughput analysis including bandwidth and latency tax. We first extend the throughput definition by Jyothi et al. [40] from a demand matrix T to our general traffic generation model $\mathcal{T}$ (in Section 5.1). We further extend the definition to apply also to *dynamic* network topologies rather than only static topologies [40, 45]. In turn, we present a mathematical framework that allows us to evaluate analytically the performance and trade-offs of arbitrary demands using different optical switches, considering real traffic distributions like in Figure 1 (a) [22, 42, 46]. In particular, based on our models, we formally show that all three topology *types* have a unique raison d'etre. In contrast to previous work [40, 47, 48], we propose to study the throughput via the *demand completion time* (DCT), which allows us to incorporate *both* bandwidth and latency tax into our analysis. We further show that the efficiency of different topology types depends on the *skewness* of the traffic distribution in a non-trivial manner. This enables us to provide novel insights into an abstract version of existing architectures such as *rotor-net* for RotorNet [4] and *expander-net* for Xpander and Jellyfish [30, 35], including their throughput[1] and the throughput-proportionality property [30] (see §2).

Contribution 2: A unified network model. We formalize the forte of three popular optical topologies from prior work: a static expander-based [28, 30, 35], a rotor-based [1, 4, 5, 49], and a demand-aware topology [6, 11, 12]. Based on these models, we propose a unified two-layer leaf-spine network model that simultaneously contains both *static* and *dynamic* topologies, and in particular, the three topology parts: static, rotor, and demand-aware. This generalizes the existing architectures mentioned above since the spine switches can be of *different* types (see §4.1).

Contribution 3: Matching traffic to topologies with Cerberus. Motivated by the identified inefficiencies resulting from a mismatch between traffic and topology, we describe a novel architecture, Cerberus[2], which facilitates to serve traffic on the topology part that best *matches* the traffic's characteristics. For example, latency-sensitive mice flows can be transmitted via static switches, all-to-all traffic via dynamic rotor switches, and elephant flows via demand-aware switches (§4). We

[1]Very recently, and in parallel to our work, the throughput of *expander-net* was also shown in [45].

[2]In Greek mythology, Cerberus is a dog with three heads (corresponding to our three topology parts).



(a) Flow size distributions

	expander-net	*rotor-net*	CERBERUS
BW-Tax	✓	✓	✗
LT-Tax	✗	✓	✓
$\theta(T)$	Thm 2	Thm 3	Thm 5
θ^*	0.53	0.45	Open
Datamining	0.53	0.6	0.8 (+33%)
Permutation	0.53	0.45	≈ 1 (+88%)
Case Study	0.53	0.66	0.9 (+36%)

(b) Topologies throughput

Fig. 1. (a) CDF of the number of bytes transmitted according to various flow size distributions. The top x-axis shows the ideal flow transmission time on a 40Gbps link. (b)The main topology designs we consider in this paper and their related properties (including the theorems derived in this paper) and achieved throughput (with our improvement in %).

show analytically (§5) and in simulations (§6) when CERBERUS outperforms alternative architectures, by achieving higher throughput [40]. Figure 1 (b) gives an overview of our corresponding contributions. For example, our simulations show that compared to static and dynamic demand-oblivious topologies, CERBERUS improves the throughput by 33% or more for a datamining workload, by 36% or more for a synthetic case study imitating realistic flow-size distributions, and by 88% or more for permutation matrices, i.e., sparse matrices which represent the worst-case input for oblivious designs. We further prove that CERBERUS is throughput-proportional [30], namely, that CERBERUS is able to utilize its full capacity proportionally, even when only a subset of the servers generates traffic.

As a **contribution to the research community** and to ensure reproducibility, we will make our artefacts (i.e., simulator and traces) publicly available[3].

Scope and limitations. The understanding of what is the best optical datacenter network design is a fundamental research question, and this paper focuses on a most essential aspect first, *performance*. In particular, we aim at giving explicit formal bounds on the throughput and capacity of different reconfigurable networks under general flow size distributions. At the same time, we leave additional aspects for future research. In particular, we defer the complex modeling of cost-efficiency aspects, which would require not only estimates of the purchase costs but also predictions about long-term costs; such an analysis is only relevant if it can identify potential performance gains, which is the scope of this paper. We also do not discuss the potential complexities involved in practically operating multiple topology types; however, we point out that already today's datacenters are fairly heterogeneous, combining different switch types, routers, protocols, etc. While we assume that the sizes of our three topology types are fixed, our framework supports also the study of dynamic settings (if corresponding hardware is available); our sensitivity analysis further shows that our results are fairly robust to changing traffic types. Finally, we will assume that flow sizes can be estimated [6, 50], as it is common in some of the recent related work [5, 6, 50–52].

2 THE THROUGHPUT OF *EXPANDER-NET* AND *ROTOR-NET*

We begin our investigations of the throughput provided by different datacenter networks by studying the throughput of two demand-oblivious fundamental network designs, a static one and a

[3]https://github.com/tum-lkn/cerberus

dynamic one. In particular, we consider Xpander-like [30] networks (henceforth: *expander-net*) as a representative of the former, and RotorNet-like [4] networks (henceforth: *rotor-net*) as an example of the latter. We will later show under which circumstances and by how much the throughput can be improved by our novel datacenter design whose topology matches the traffic distribution.

Our throughput analysis in this section relies on a simplified version of our more general methodology presented in more detail later and, hence, also serves as an introduction. As we will see in the following, this simplified version is already sufficient to derive a novel formal bound for the capacity of *expander-net* and *rotor-net*.

Let $T = [t_{uv}]$ be an $n \times n$ demand matrix (entries in T are traditionally given as rates [40], so we consider them as bits per second, denoted as *bps*) for a datacenter network consisting of n switches which are used to interconnect a set of racks. Each switch has k *links* of rate r connected to other switches, plus k links connected to servers in its rack (also called uni-regular topologies [45]). In the following, it suffices to focus on the rack-level network, i.e., we consider the demand matrix between racks (resp. the top-of-rack switches) rather than individual servers. We consider the *hose* model [40] and assume for now that T is *saturated*, i.e., the sum of rates in each row and column of the demand matrix is kr. Let $\hat{T}$ be the demand matrix in *bits* for *one second*, i.e., equivalent to T, but in bits and not in *bps*. Let $\text{Total}(\hat{T})$ denote the total number of bits in $\hat{T}$. Furthermore, let $\text{DCT}(\hat{T})$ denote the demand completion time (in seconds) of $\hat{T}$, in a feasible multi-commodity network flow. We then claim the following as an immediate result.

Theorem 1 (DCT and Throughput). *Let T be a demand matrix (in bps). i) If $\text{DCT}(\hat{T}) \geq x$ for every feasible demand completion time of $\hat{T}$, then: $\theta(T) \leq \frac{1}{x}$. ii) If there exist a feasible network flow s.t. $\text{DCT}(\hat{T}) \leq y$, then $\theta(T) \geq \frac{1}{y}$.*

Proof. If the optimal demand completion time of T is exactly 1, then for each entry T_{uv} in the matrix T, the rate is satisfied and T is feasible also, so $\theta(T) \geq 1$. Similarly, for i) assume for the sake of contradiction that there is a network flow s.t. $\theta(T) > 1/x$. But then $\text{DCT}(\hat{T}) < x$ for that network flow. Contradiction. For ii) assume for the sake of contradiction that for the given network flow $\theta(T) < 1/y$. But then $\text{DCT}(\hat{T}) > y$. Contradiction. □

We start by bounding the throughput of *expander-net* formally defined as $G(k)$, a (random) k-regular expander. *expander-net* is an abstract representation of expander-based topologies [30, 35]. Let $\text{epl}(G(k))$ denote the *expected path length* of $G(k)$.

Theorem 2 (*expander-net* Throughput). *The throughput of* expander-net *with degree k is upper bounded by: $\theta^* \leq \frac{1}{\text{epl}(G(k))}$, where* epl *is the expected (average) path length of $G(k)$, a k-regular expander with n nodes.*

Proof sketch. We optimistically approximate the demand completion time of *expander-net*. We assume that traffic is distributed across all shortest paths with no delay due to packet loss or congestion. Hence, the only "cost" we consider is related to the path length, that is, each flow consumes bandwidth capacity proportional to its route length (i.e., the bandwidth tax). The demand completion time of T can be easily bounded as follows: $\text{DCT}(\hat{T}) \geq \text{Total}(\hat{T}) \cdot \frac{\text{epl}(G(k_s))}{k \cdot n \cdot r}$. Noticing that T is saturated and that the total demand of $\hat{T}$ is $k \cdot n \cdot r$ bits, together with Theorem 1, provides the result. □

The main ***takeaway*** from Theorem 2 is that the reduction in the throughput of *expander-net* is due to the multi-hop routing (the bandwidth tax). Networks with shorter routes may benefit from higher throughput.

To compute the throughput of dynamic topologies, we must take into account the reconfiguration time (the latency tax). The demand completion time provides us with a simple tool to address this. Next, we study *rotor-net*, an abstract topology based on rotor switches like RotorNet [4] and Sirius [1][4]. In short, these designs rely on periodic matchings (k many) between nodes emulating a full-mesh (complete graph) network by dynamically reconfiguring the circuit switches. The reconfiguration time between successive matchings is denoted as R_r, and the *slot time* δ is the circuit-hold time after each reconfiguration. The *cycle time* is the time to emulate all the (directed) links in the complete graph. To bound the performance of *rotor-net*, we define ϕ, the *traffic skewness* of T. In short, ϕ denotes the fraction of bytes in *rotor-net* which is sent through a single hop and $1-\phi$ is the fraction which is sent via two hops using Valiant routing [4]. We provide more details of ϕ later. Now, we can formally state the following about *rotor-net*.

Theorem 3 (*rotor-net* Throughput). *The throughput of* rotor-net *with reconfiguration time R_r and slot time δ is bounded by: $\theta^* \leq \frac{n}{2n-1}\frac{\delta}{R_r+\delta}$. For a given demand matrix T, we have $\theta(T) \leq \frac{1}{2-\phi(\hat{T})}\frac{\delta}{R_r+\delta}$, where $\frac{1}{n} \leq \phi(\hat{T}) \leq 1$ is the* traffic skewness *of T.*

As we discuss later, $\phi(\hat{T})$ can expressed in terms of the variation distance [53] and is equal 1 for the all-to-all demand matrix (uniform demand), so the throughput (even for saturated matrices) can significantly differ. We get as a corollary that for every demand T, $\theta^* \leq \theta(T) \leq \frac{\delta}{R_r+\delta}$ where the worst matrix is the permutation matrix and the best matrix is the uniform matrix.

Proof sketch. For now, we only prove the results for θ^*. The proof for $\theta(T)$ is presented later. By design, *rotor-net* supports only one or two hop routing [4, 5]. In the worst case, all traffic from a source is destined to a single destination (i.e., a permutation matrix), hence all routes except for a $\frac{1}{n}$ fraction, which are sent over direct links, are of length 2. Hence, the traffic skewness for this case is $\phi(\hat{T}) = \frac{1}{n}$. This leads to an average path length of $2-\frac{1}{n}$. Assuming no reconfiguration time, similarly to Theorem 2, this leads to $\mathrm{DCT}(\hat{T}) \geq 2-\frac{1}{n}$. But since the utilization of each rotor switch is $\frac{\delta}{R_r+\delta}$ (the latency tax), it increases the demand completion time by the inverse ratio so $\mathrm{DCT}(\hat{T}) \geq (2-\frac{1}{n})\frac{R_r+\delta}{\delta}$, and the result follows by Theorem 1. □

As a ***takeaway*** from Theorem 3, we notice that the throughput of *rotor-net* is determined by both the bandwidth and latency taxes. Next, we consider the throughput of *demand-aware-net*, an abstract version of a topology built only from k demand-aware switches [11, 14] (allowing to create any k-regular directed graph). Let R_d be the reconfiguration time needed for any link adjustment. We assume that individual links can be reconfigured independently, if both source and destination have an available port. In the *demand-aware-net*, for the analytical result, we consider that the demand matrix T is a union of m permutation matrices T^j each of which can have a different weight w_j. Formally, $T = \sum T^j$. In this case, we can state the following about the throughput.

Theorem 4 (*demand-aware-net* Throughput). *The throughput of* demand-aware-net *for the above demand matrix T with reconfiguration time R_d can be bounded as: $\theta(T) \geq \frac{k}{\sum_{j=1}^{m}(R_d+\frac{w_j}{r})}$.*

Proof. Again, this follows from the DCT calculation. Since T is a collection of matchings, we can optimally schedule every switch to send all the permutations matrices, k at at time. Notice that each permutation may remain active for a different time. The demand completion time of T can be then computed as $\mathrm{DCT}(\hat{T}) = \frac{1}{k}\sum_{j=1}^{m}(R_d + \frac{w_j}{r})$ and the result follows by Theorem 1. □

[4]Our approach in principle also allows us to study the throughput of Opera [5], which is slightly more complex. We leave a detailed discussion for future work.

As we can see, the reconfiguration time (latency tax) plays a critical role for the throughput. Overall, the throughput can change significantly as a function of T. The best case for *demand-aware-net* is when T is arbitrary (and unknown a priori), but a single permutation matrix. In this case, the throughput is close to one since only one reconfiguration is needed, and the total DCT of the system is $R_d + 1$, and the throughput is $\frac{1}{R_d+1}$. We further note that while a static network could potentially deal with a single permutation matrix T optimally if it is *known a priori*, *demand-aware-net* will deal with any such T in an optimal way, even when it is *unknown* a priori. The worst case for *demand-aware-net* is when T is the uniform matrix and built from a collection of n permutations matrices of low weight, i.e., each of weight kr/n. Plugging this value of w_j into Theorem 4 will give us a lower bound of $\frac{k}{nR_d+k}$, and the throughput can be close to zero. More formally, the maximal and minimal throughput can be bounded by, $\frac{k}{nR_d+k} \leq \theta(T) \leq \frac{1}{R_d+1}$.

3 MOTIVATION: DIVERSITY OF TRAFFIC PATTERNS AND TOPOLOGIES

Our main observation from the previous section is that different existing datacenter topologies can have different advantages and disadvantages depending on the scenario. This motivates us to introduce and analyze a novel datacenter network design that is tailored toward the specific setting and traffic it serves. We are mainly motivated by the hypothesis that throughput in datacenter networks can be significantly improved if the *network topology matches the demand*. In the following, before introducing our proposed design, Cerberus, we elaborate on the potential inefficiencies arising from a mismatch of traffic patterns and topologies. In particular, using an empirical example, we provide more intuition on how "taxes" can be used to quantify inefficiencies.

For the upcoming discussion, we plot in Figure 1(a) the different flow size distributions for Websearch [46], Datamining [42], and Hadoop [22] applications, as well as a synthetic Pareto distribution [1]. The figure shows that flow size distributions can vary widely across and within applications. In case of datamining traffic, about 75% of the traffic belongs to flows of sizes above 100MB. The flow size also determines the flow transmission time. The top x-axis in Figure 1 (a) shows the ideal flow transmission time on a $40Gbps$ link, which can range from microseconds for small flows up to seconds for elephant flows.

The flow transmission time and, hence, the throughput does not only depend on the flow size and the link speed, but also on the used optical switching technology. In particular, it depends on the reconfiguration time which accrues in dynamic topologies (e.g., $10\mu s$ per slot for rotor switches [4] vs $15ms$ for demand-aware switches [43, 44]). While the exact reconfiguration times will depend on the specific technology, the reconfiguration times of demand-aware topologies are likely to be higher than those of demand-oblivious topologies where reconfigurations are *periodic* (like in [4, 5, 54]).

A key observation is that whether a flow can profit from reconfigurations depends on its size. For example, an elephant flow has a large transmission time compared to the reconfiguration time. Hence, the reconfiguration, i.e., the latency tax can be amortized and may pay off in the long run.

As shown in Theorem 2, static topologies do not introduce any latency tax, however, they require multi-hop forwarding and, hence, introduce a bandwidth tax [4]: the more hops a flow has to traverse, the more network capacity is consumed.

Dynamic topologies can reduce the bandwidth tax by avoiding multi-hop forwarding, which performs particularly well for uniform all-to-all traffic patterns [4, 5]. However, such demand-oblivious dynamic architectures are not optimal for the elephant flows; as typically elephant flows carry a majority of the bytes, optimizing for elephant flows is important (see, e.g., the Datamining workload). While in principle, Valiant routing [55] can be used in combination with rotor switches to carry large flows, this again results in bandwidth tax, as demonstrated in Theorem 3.

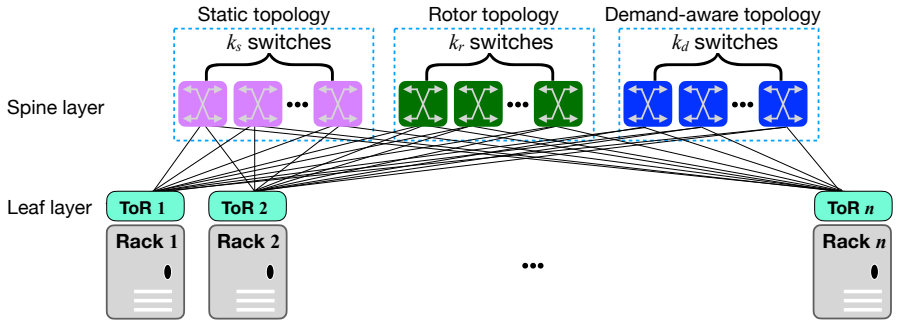


Fig. 2. The ToR-Matching-ToR (TMT) network.

In contrast, in a demand-aware topology, direct shortcuts can be set up specifically for elephant flows (Theorem 4). As a ***takeaway*** regarding the design choice between demand-oblivious vs. demand-aware topologies, we conclude that whereas demand-oblivious topologies perform well under uniform demands, demand-aware topologies perform well under skewed demands with large flows for which latency taxes can be amortized.

This paper provides a mathematical model that allows deciding on the optimal flow assignments to each topology type based on the flow size distributions. Returning to Figure 1(a), the solid vertical lines show the reconfiguration times of rotor switches ($10\mu s$) and demand-aware switches ($15ms$). The right most part (dark gray area) starts at $15ms$, which means that a flow with large size, e.g., a $500MB$ flow whose transmission time is $100ms$, can amortize 15% tax for reconfiguration. Creating a direct link for such flows will reduce the bandwidth tax to the minimum. The middle part of the figure belongs to medium-sized flows. This area starts at $10\mu s$, but if we consider $100\mu s$, which is a relatively short *slot* time with 91% utilization for rotor switches [4], a cycle through all possible links is fast. Since medium-sized flows are the vast majority of flows in the probability distribution, if sources and destinations are sampled uniformly, they result in all-to-all traffic, and can hence greatly benefit from the rotor switch's connectivity pattern. Finally, there are small flows, shown in the left white area. For these flows, we cannot afford any reconfiguration, but since their cumulative size is also small, only a minor chunk of the resources need to be allocated to them. This, for example, can be achieved with a static expander topology with a short average path length, therefore, enabling low-latency forwarding. The exact flow size thresholds for making forwarding decisions are an output of our analysis.

4 CERBERUS: NETWORK DESIGN WITH CHOICE

Motivated by our observations above, we propose and analyze a unified datacenter architecture, Cerberus, which combines all three switch and topology types. Depending on link rates and reconfiguration times as well as the given traffic mix, Cerberus computes the optimal size and composition of different topology types, to improve performance.

4.1 The ToR-Matching-ToR (TMT) Model

Cerberus relies on a two-layer leaf-spine network architecture with n leaf switches that are ToR switches and k optical spine switches that can be of different types: Static, Rotor and Demand-aware; we will denote the number of switches from each type by k_s, k_r and k_d respectively where $k = k_s + k_r + k_d$. We will discuss how to compute the value for each type later. Since each optical spine switch connects its in-out port via a *matching*, we will refer to it as the ToR-Matching-ToR (TMT)

network model. This network architecture generalizes existing architectures such as RotorNet [4] and Opera [5], by supporting multiple switch types.

More specifically, the TMT network interconnects a set N of n ToRs, $\{1, 2, \ldots, n\}$ and its two-layer leaf-spine architecture composed of leaf switches and spine switches. The n ToR packet switches are connected using k spine switches, $SW = \{sw_1, sw_2, \ldots, sw_k\}$ and each switch internally connects its in-out ports via a matching. Figure 2 illustrates a schematic view of our design. We assume that each ToR $i : 1 \leq i \leq n$ has k uplinks, where uplink $j : 1 \leq j \leq k$ connects to port i in sw_j. The directed outgoing (leaf) uplink is connected to the incoming port of the (spine) switch, and the directed incoming (leaf) uplink is connected to the outgoing port of the (spine) switch. Each spine switch has n input ports and n output ports and the connections are directed, from input to output ports. As a running example, throughout this paper we assume that the rate of each link is $r = 40Gbps$; however, our model is general and can be used for any bandwidth as we discuss later: similar results to what we report here hold also for lower rates (e.g., 10Gbps as in [4, 5, 30]) and higher rates, e.g., 100Gbps. At any point in time, each switch $sw \in SW$ provides a *matching* between its input and output ports. Depending on the switch type, this matching may be *reconfigured* at runtime to another matching. Each switch j has a set of matchings $\mathcal{M}_j$ of size $m_j = |\mathcal{M}_j|$ and m_j may be larger than one. Changing from a matching $M' \in \mathcal{M}$ to a matching $M'' \in \mathcal{M}$ takes time, which we model with a parameter R_j: the *reconfiguration time* of switch j. During reconfiguration, the links in $M' \cap M''$, i.e., the links which are not being reconfigured, can still be used for forwarding; the remaining links are blocked during the reconfiguration [56]. Depending on the technology, different switches in SW support different sets of matchings and reconfiguration times.

4.2 Cerberus' Topology Components

Our network model can be instantiated with different switches, accounting for the different and specific switch characteristics. Thereby different typology types or components can be created. In this paper, we consider three fundamental topology components: a static part, a rotor-only part, and a demand-aware part. These components may either be realized by different switch technologies, or by a single switch technology that can support multiple modes of operation. The former may be more cost effective (e.g., static topologies are cheaper), while the latter is more flexible.

Abstractly, the three topologies can be described in a unified form using a collection of spine switches. Each spine switch type, in turn is defined by a 4-tuple, $sw = (m, \mathcal{M}, S, R)$ where m is the number of matchings the switch can support; $\mathcal{M}$ is the specific set or sequence of m matchings the switch can realize; S is the minimal circuit-hold time a switch needs to remain in a matching that contains a specific link before switching to the next matching which does not contain this link; R is the reconfiguration time of changing between matchings. Using our notation, the three topologies can be formalized as follows:

- **Demand-aware topology (Demand-aware):** We create a demand-aware topology using a collection of k_d Demand-aware reconfigurable switches. Each such switch is described by the tuple $sw = (n!, \mathcal{M}, S, R_d)$. Demand-aware switches have the freedom to flexibly reconfigure to *any* of the $m = n!$ possible directed (perfect) matchings; i.e., $\mathcal{M} = S_n$ where S_n is the symmetric group of $[1...n]$ and each matching is represented as a permutation. The demand-aware switch can be implemented using off-the-shelf 3D MEMS technology with reconfiguration time in the order of tens of ms. In this paper, we assume $R_d = 15ms$ which is the typical reconfiguration time of a 3D MEMS switch [44]. The circuit-hold time S can change during the operation of the Demand-aware switch. While in this case the minimal time is zero, as a rule of thumb $S \gg R_d$ for the reconfiguration to be worthwhile. Otherwise, the utilization of the link would be small as most of the time is spent on reconfiguration.

- **Rotor-based topology (ROTOR):** A rotor-based topology consists of the union of k_r ROTOR switches. Each switch is described by the tuple $sw = (n-1, \mathcal{M}, \delta, R_r)$: a ROTOR switch cycles through $n-1$ matchings specified by $\mathcal{M}$, emulating a fully-connected network (i.e., complete graph) and, hence, providing high bandwidth to all-to-all traffic. Our ROTOR switch is a slight generalization of the original ROTOR switches [4] used in RotorNet since our model uses $n-1$ matchings and not n/k matchings as proposed originally. (We explain why this generalization improves the performance of RotorNet in Section B in the Appendix.) The reconfiguration time for the ROTOR switch is in the order of few μs. We assume $R_r = 10\mu s$ as in [4, 5]. The circuit-hold time of a ROTOR switch is called the *slot* time and is denoted as δ. The slot time is tunable and depends on the reconfiguration time, where a reasonable setup is at least $\delta = 9R_r = 90\mu s$ to reach 90% amortization of the reconfiguration time. This setup was used in the original work [4, 5], but our bounds hold for any settings.
- **Static topology (STATIC):** We describe the static topology as a union of k_s matchings, where each matching can be implemented, e.g., using an optical patch panel (our analysis also applies to electrical static topologies). In the case of a static component, the 4-tuple switch specification can be represented by: $sw = (1, \mathcal{M}, \infty, 0)$ where $\mathcal{M} = \{M\}$ has a single (i.e., $m = 1$) predefined matching that does not change over time ($S = \infty, R = 0$). The static switches are cost-effective components to create regular graphs, such as expander graphs, providing low latency for short flows using multi-hop routing. Prior work has shown that good expanders can be obtained by taking the union of a few matchings [57].

Let (k_s, k_r, k_d) denote a TMT network consisting of k_s STATIC switches, k_r ROTOR switches, and k_d DEMAND-AWARE switches. We will refer to a network consisting only of k STATIC switches, i.e., a network with $(k, 0, 0)$, as *static-net*; as we will see, the static topology component of CERBERUS specifically relies on expander graphs, and we will hence refer to this network as *expander-net*. We will further refer to the network consisting of only ROTOR switches, i.e., networks with $(0, k, 0)$, as *rotor-net*, and to topology components consisting only of DEMAND-AWARE switches, i.e., a network with $(0, 0, k)$, as *demand-aware-net*. On the contrary, CERBERUS uses a mix of switch types, (k_s, k_r, k_d), where $k = k_s + k_r + k_d$.

We also note that the TMT network can be used to model many existing systems. For example, RotorNet [4], Opera [5] and Sirius [1] rely on periodic matchings and can be modelled as a *rotor-net*. Networks like ProjecToR [11], Eclipse [14], or Helios [6] rely on demand-aware matching. To be more specific, while for example the demand-aware links of ProjecToR are based on free space optics, conceptually it can still be modelled as a *demand-aware-net*; ProjecToR additionally uses a static electric network, which in our conceptual model can also be described using an *expander-net*. Our model and analysis also apply to Xpander [30], which can be modelled as an *expander-net* as well (even though it is based on electrical switches).

4.3 Cerberus' Flow Assignment Algorithm

Given a (k_s, k_r, k_d) network, we next describe the high-level flow assignment algorithm. The analysis of the exact parameters will appear in the next section. CERBERUS operates by dividing flows into three categories, based on three sizes: *small* (s), *medium* (m) and *large* (ℓ) flows. The size *thresholds* to assign flows to these categories are denoted by t_m and t_ℓ. Namely, *small* flows are of size less than t_m, *medium* flows are of size more than t_m and less than t_ℓ, and *large* flows are of size larger than t_ℓ. We first determine the size threshold for medium flows, t_m, as the number of bit transmitted in a slot time in ROTOR switches, i.e., $t_m = \delta r$. The goal is to ensure a low delay for small flows and a high link utilization for medium flows. We then set the large flow size threshold, t_ℓ, as the minimum flow size that will have shorter completion time on DEMAND-AWARE switches, see

Algorithm 1 CERBERUS flow assignment

```
1:  Switch depending on flow size
2:    Case small flow:                          ▷ latency-sensitive flow
3:      send to static expander                 ▷ multi hop
4:    Case medium flow:
5:      send to rotor-based topology            ▷ using 1 or 2 hops
6:    Case large flow:
7:      If a direct link is available to reconfigure:
8:        send to demand-aware topology         ▷ single hop
9:      Else                                    ▷ Under-provisioned demand-aware
10:       send to rotor-based topology          ▷ using 1 or 2 hops
```

Eq. (7) in the next section. In summary, the threshold for large flows depends on the reconfiguration times of the ROTOR and DEMAND-AWARE switches and can be computed based on Theorem 5, Eq. (7), which we present later.

Algorithm 1 describes how CERBERUS distributes the traffic classes among the three switch types: small, latency-sensitive flows are forwarded via a static expander built from k_s STATIC switches; large flows are transmitted via the k_d many DEMAND-AWARE switches in the system; and the remaining (medium) flows describing e.g., all-to-all traffic which is not latency-sensitive, are routed via the k_r ROTOR switches. CERBERUS manages the large flows using an approach which can be seen as a distributed link cache: when a new demand-aware connection needs to be established, an existing link must be replaced or "evicted". While this introduces interesting optimization opportunities, in the following, we will focus on a simple strategy: when a large flow should be sent to the DEMAND-AWARE switches, but there are no available ports to serve it (the related source/destination ports are already serving other flows), we greedily transmit the large flow via the ROTOR switches. When this happens continuously, we say that the demand-aware switches are *under provisioned*. In the next section, we derive k_d^*, the optimal number of DEMAND-AWARE switches (under a given traffic assumption) to maximize the throughput.

5 CERBERUS: THROUGHPUT ANALYSIS

This section describes how CERBERUS' logic calculates (*i*) the topology parameters for each type of topology; (*ii*) the flow assignment strategy for each topology; and (*iii*) the throughput and demand completion times for the system as a whole. To do this, we first describe the general traffic generation model that our analytical results hold for.

5.1 Traffic Generation Model and Metrics

Inspired by prior work [1, 5, 30], we consider the following fundamental traffic generation model: flows (commodities) arrive over time, according to a sequence $\sigma = (f_1, f_2, \ldots)$ where the f_is are individual flows. Each flow (commodity) $f_i = (t_i, s_i, d_i, \Delta_i)$ has an arrival time t_i, a source rack $s_i \in N$, a destination rack $d_i \in N$, and a size f_i (in bytes)[5].

More formally, for our analysis, let $\mathcal{T}(x, L, \mathcal{D})$ be a traffic generation model where $x \in [0, 1]$ is the fraction of active ToRs, $L \in [0, 1]$ is the load of each active ToR, and $\mathcal{D}$ is a flow size distribution. $\mathcal{T}$ generates traffic as follows: First, we select uniformly at random x fraction of the ToRs as active. Next, each active ToR generates traffic at an average rate of $L \cdot k \cdot r$, where r is the line rate (in *bps*) and k is the number of uplinks (and spine switches). Flow arrival times follow a Poisson

[5]Flows are originally generated by servers, but our focus here is on the flows from a rack-level perspective.

process[6] (where inter arrival times follow an exponential distributions) and flow sizes are sampled independently from the flow size distribution $\mathcal{D}$. Only active ToRs are sources and for each source the destination ToR is chosen uniformly at random from the set of active ToRs. The expected total amount of traffic (in bits) generated per second is *(active-ToRs)×(traffic-per-ToR)* $= (xn)(L \cdot k \cdot r)$, where n is the number of ToRs.

Our main metric of interest is the end-to-end *throughput* supported by systems in a fluid-flow model. As before, we follow [40] in defining network throughput, and focus on the network topology (and ignoring e.g., congestion control). We extend the definition of [40] from a demand matrix to our general traffic generation model, $\mathcal{T}(x, L, \mathcal{D})$. Furthermore, we study *dynamic* network topologies rather than static topologies.

The *throughput* of a system for a traffic generation model $\mathcal{T}(x, L, \mathcal{D})$ and a given x (fraction of active nodes) is the *maximum* value L for which the traffic is feasible in the system. That is, we seek the maximum L for which there exists a feasible multi-commodity network flow assignment that routes the flows generated by $\mathcal{T}$ through the network from each source v to each destination w, subject to link capacities and flow conservation. Following Theorem 1, our approach is to study the throughput by computing analytically the *demand completion time* (DCT): the expected time it takes to serve the accumulated demand matrix $\hat{T}$, built from the flows generated by $\mathcal{T}$ in one second. For a given x and L, if the completion time is $\text{DCT}(\hat{T}) \leq 1$, then the network has throughput at least L for this x.

We consider two basic cases for the throughput: i) $x = 1$ where all ToRs are active, and ii) $x \ll 1$ and the traffic is *skewed*. For the latter case, we are interested whether the network is *throughput-proportional*. Following the definitions and observation in [30] (see also Figure 2 in [30]), a network is *throughput-proportional* when it is "able to distribute its capacity evenly across the set of servers with actual traffic demands" [30]. Namely, the *non-active* ToRs can share their capacity and help the x fraction of *active* ToRs. Hence, L will be proportional to total network capacity, and not only to capacity of the active ToRs.

5.2 Main Result: Analysis of Cerberus' Throughput

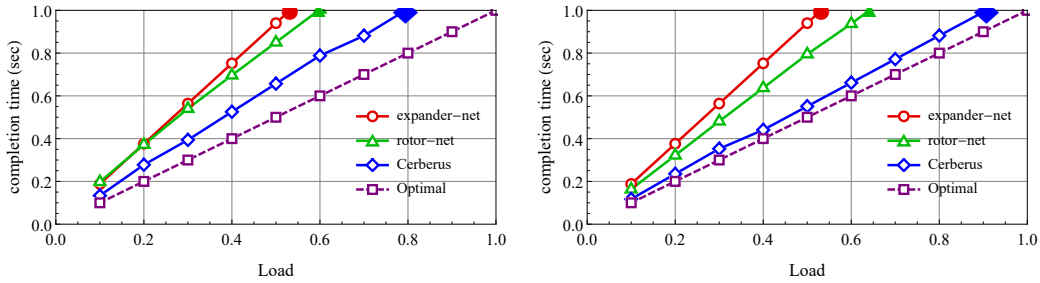
In this section, we present our theoretical results on the throughput of *expander-net*, *rotor-net* and Cerberus, for the $\mathcal{T}(x, L, \mathcal{D})$ traffic model. Due to space constrains, the detailed analysis will appear in the Appendix A. Simulations and numerical evaluations will follow in Section 6.

We start by considering the case $x = 1$ when all ToRs are active with the same load L. We show that for all considered systems, *expander-net*, *rotor-net*, and Cerberus, the demand completion time grows (almost) linearly with load. Moreover, for a wide range of parameters (as we will show later), Cerberus consistently achieves the highest throughput and has the lowest demand completion times.

Informally, an example of the main theoretical result for this case is demonstrated in Figure 3, which compares the demand completion times for $x = 1$ of Cerberus vs. *expander-net* and *rotor-net*, both for the Datamining [42] and the Case Study flow distributions (workloads), for $k = 32$ and $n = 256$. The Case Study distribution is a simple flow distribution for illustrative proposes, which we discuss in Section 6.1.

The figure presents the expected demand completion times, $\text{DCT}(\hat{T}(1, L, \text{Datamining}))$ and $\text{DCT}(\hat{T}(1, L, \text{CaseStudy}))$, as the load L increases. We can see that all systems follow an almost linear trend while Cerberus is able to be closer to the optimal completion time, which is L seconds for a load L, and sustain lower completion times throughout. The *throughput* of each system and flow size distribution is the maximum load for which $\text{DCT}(\hat{T}) \leq 1s$. Recall that the maximum throughput

[6]The exact rate of the Poisson process is a function of L, r and $\mathcal{D}$.



(a) Datamining [42] workload (b) Case Study workload

Fig. 3. The expected demand completion time as the load L increases, for the Datamining and Case Study workloads.

is one. For the Datamining workload, Cerberus has a throughput of 0.8, while *expander-net* and *rotor-net* have throughout of at most 0.53 and 0.6, respectively. For our Case Study, Cerberus has a throughput of 0.9, while *expander-net* and *rotor-net* can have throughout of at most 0.53 and 0.66, respectively. These values are shown also in Figure 1(b).

Formally, let $\hat{T}(L)$ be an accumulated demand matrix built from the flows generated by $\mathcal{T}(1, L, \mathcal{D})$ in one second for n ToRs each with k uplinks (and $\mathcal{D}$ is clear from the context). We denote by $\text{DCT}(sys, \hat{T}(L), k)$ the demand completion time of a system $sys \in \{expander\text{-}net, rotor\text{-}net, Cerberus\}$. Recall that $expander\text{-}net = (k, 0, 0)$, $rotor\text{-}net = (0, k, 0)$ and $Cerberus = (k_s, k_r, k_d)$. Recall that we categorize flows to three sizes and let τ denote the size type of a flow where $\tau \in \{s, m, \ell\}$ and let $\hat{T}(L, \tau)$ denote the expected number of bytes in flows of type τ in $\hat{T}(L)$. Note that $\hat{T}(1, s) + \hat{T}(1, m) + \hat{T}(1, \ell) = nkr$. Let $\phi = \phi(\hat{T})$ denote the expected *traffic skewness* of $\hat{T}$ resulting from the flow size distribution $\mathcal{D}$. Formally, ϕ denotes the fraction of packets (or bytes) sent in *rotor-net* via a single hop, and $1 - \phi$ is the fraction of packets that are sent using Valiant routing [4], using two hops, when serving $\hat{T}$. We approximate ϕ for a given distribution and the load (or empirical distribution) as one minus the variation distance [53] from the uniform distribution. This means that links (or source-destination pairs) that are active above the average load can send packets via links that are below the average load (using two hops). We discuss ϕ in more detail in Section 6.4. We can now state the following formal results about the throughput of Cerberus:

Theorem 5 (Cerberus Throughput). *Let $\hat{T}(L)$ be the accumulated demand matrix of flows that were generated by $\mathcal{T}(1, L, \mathcal{D})$ in one second for n ToRs each with k uplinks of rate r. The expected throughput, i.e., the maximum L for which T is feasible, of* Cerberus *is:*

$$\theta(T) = \frac{\hat{T}(1, \ell)}{nk_d^*} \left(R_d \mathbb{E}\left[\frac{1}{|f|}\right] + \frac{1}{r} \right) \tag{1}$$

where $|f|$ is the size of flow f.The expected size of the reciprocal flow sizes, is taken only over large flows in $\mathcal{D}$, k_d^* is the optimal number of Demand-aware switches and R_d is their reconfiguration time.

To find the throughput of Cerberus using Eq. (1), we first need to compute two important parameters: i) the threshold of large flows, and ii) the optimal division of Demand-aware switches and Rotor switches. The threshold for large flows, t_ℓ can be computed by:

$$t_\ell \geq \frac{R_d \cdot t_m \cdot r}{(2 - \phi) \cdot r \cdot (R_r + \delta) - t_m} \tag{2}$$

The ratio between the optimal number of DEMAND-AWARE switches, k_d^*, to the optimal number of ROTOR switches, k_r^* is given by:

$$\frac{k_d^*}{k_r^*} = \frac{\hat{T}(1,\ell)}{\hat{T}(1,m)/t_m} \cdot \frac{R_d \mathbb{E}\left[\frac{1}{|f|}\right] + \frac{1}{r}}{(2-\phi_m)(R_r+\delta)} \tag{3}$$

where ϕ_m is the traffic skewness of the medium size flows. If we assume k_s is set to some constant value, as we do later, $k_d^* + k_r^*$ will be known, and we can compute k_d^* from Eq. (3).

Analysis overview: The result is obtained using Theorems 3 and 4. First, we compute the completion times when sending flows over a pure demand-aware topology. This setting comes with a (large) latency tax, based on the reconfiguration time R_d. Then, the key observation in computing the throughput of CERBERUS, is that the optimal throughput is achieved when the completion time of CERBERUS's different subsystems is *balanced*. Formally,

$$\mathrm{DCT}(Cerberus, \hat{T}(L), k) = \max \begin{cases} \mathrm{DCT}_{\mathcal{S}}(\hat{T}(L,s), k_s) \\ \mathrm{DCT}_{\mathcal{R}}(\hat{T}(L,m), k_r) \\ \mathrm{DCT}_{C}(\hat{T}(L,\ell), k_d) \end{cases} \tag{4}$$

Where $\mathrm{DCT}_{\omega}(\hat{T}, k')$ denotes the *demand completion time* to serve the demand $\hat{T}$ using these k' switches of type ω and $\mathcal{S}, \mathcal{R}, C$ denote the STATIC, ROTOR and DEMAND-AWARE switches respectively. Using the previous calculation for *rotor-net* and *demand-aware-net* we can find the optimal partition of switch types k_d^* and k_r^*. As mentioned the full details appear in Appendix A.

As we show in the next section, Theorem 2, Theorem 3, and Theorem 5 enable us to numerically estimate the throughput of the different systems given the input parameters (i.e., reconfiguration times, rates, etc.). Regarding the skewed traffic (where $x < 1$), a generalized result, but with a slightly more complex analysis, can be obtained. See Appendix A.2 for details. Following the extension of Theorem 5, for the case $x \ll 1$, we can derive another important result which was previously known only for expanders:

COROLLARY 1. *All three systems:* expander-net, rotor-net *and* CERBERUS *are throughput-proportional.*

6 SIMULATIONS & NUMERICAL RESULTS

In order to complement our analytical results, we conduct simulations and perform a numerical evaluation of our system at scale.

6.1 Methodology

We use a custom event-based flow-level simulator written in Python. In contrast to a packet-based simulator, a flow-level simulator assigns rates to flows in an end-to-end fashion. No packet buffering or switch processing latency occurs.

The considered network setup consists of $n = 64$ ToRs, $k = 16$ spine switches, $R_r = 10\mu s, R_d = 15ms, \delta = 100\mu s$ and a rate $r = 40\,Gbps$. We generate traffic corresponding to 16 hosts per ToR (1024 hosts in total) with $40\,Gbps$ uplinks. The total host uplink capacity of the topology is $40.96\,Tbps$, which is at a comparable scale as other studies, e.g., Sirius [54] ($51.2\,Tbps$), Opera [5] ($51.84\,Tbps$), or MegaSwitch [8] ($63.36\,Tbps$). MegaSwitch's and Sirius' simulations operate in a similar range as we target; however, their implementations are not publicly available. In addition, the number of flows per simulation (200K with a mean flow size of 100KB) indicates that only very short periods, $< 1\,s$, were evaluated for Sirius. The main evaluation of Opera focuses on a scenario with much

(a) Case Study distribution

Flow size	5KB	0.5MB	1GB
Type	small (s)	medium (m)	large (ℓ)
PDF (flow)	4.95%	95%	0.05%
PDF (byte)	$\approx$ 1%	$\approx$ 49%	$\approx$ 51%
Trans. Time	$1\mu s$	$100\mu s$	$200ms$
Served by	static	rotor	demand aware

(b) Parameters for different distributions

Demand (flow distribution)	Average flow size	Threshold $\lvert I\rvert_{opt}$	Bytes in large flows	Skew ϕ	Rotor k_r^*	Demand-aware k_d^*
WebSearch	$1.87MB$	$407MB$	$\approx$ 0 %	0.92	32	0
Case Study	$0.97MB$	$131MB$	0.51 %	0.57	16	16
Hadoop	$4.86MB$	$169MB$	0.74%	0.69	10	22
Datamining	$7.86MB$	$107MB$	0.79%	0.45	9	23
Pareto	$11.16MB$	$73MB$	0.97%	0.17	1	31

Table 1. (a) The PDF of flows and bytes for our Case Study. (b) Empirical values for parameters of different flow size distributions (for $x = 1$ and $L = 0.5$ as an example).

smaller topologies (only 6.48 $Tbps$) and lower load levels than ours. For larger scales [5, Figure 12], the details regarding flow generation are missing so that it cannot be judged.

The complexity of the *centralized rotor-net* implementation prohibits scaling to larger networks. The allocation of indirect traffic (2-hop routing) requires to evaluate each source-destination rack pair. The scale of this operation is quadratic in the number of ToRs and executed for every slot. An optimized, centralized *rotor-net* algorithm is left for future work. For brevity, only the algorithms used in the simulation are described in the following.

6.1.1 Flow Assignment Algorithms. Three systems are compared: *expander-net*, *rotor-net* and Cerberus:

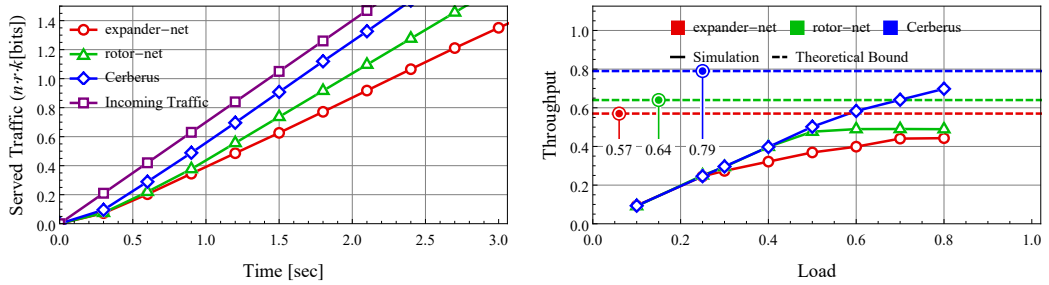
expander-net. We use greedy single path routing on the expander graph built from the union of k matchings. First, all flows in the system are grouped by their ToR source-destination pair. For each pair of ToRs, the flows are sorted by their size in increasing order. The algorithm then iterates over the pairs and tries to greedily allocate as many flows as possible.

rotor-net. In contrast to the original, distributed *rotor-net* algorithm implementation, the flow-based algorithm implementation uses a centralized control plane. It performs flow allocations on a ToR-by-ToR basis with a fixed, deterministic order. More advanced approaches are left for future work.

Cerberus. Generally, Cerberus makes use of the algorithms *rotor-net* and *expander-net* for each dedicated topology part. Following Algorithm 1, the algorithm first greedily searches for available circuits on the demand-aware topology for the large flows. In case, no path can be found, the algorithm tries to allocate large flows on the *rotor-net* part. Otherwise, the flows have to wait until capacity becomes available, in which case the algorithm tries again to (1) allocate flows on the demand-aware or (2) on the *rotor-net* part.

We acknowledge that these algorithms do not obtain optimal solutions, e.g., in contrast to solving a multi-commodity flow problem [45] or our derived upper bounds. Rather they trade-off tractability of simulations at the desired scale and performance.

6.1.2 Traffic and Flow Size Distributions. We consider traffic generated with the traffic model described in Section 5.1 using six flow size distributions. The first set of distributions are the ones shown in Figure 1(a) and include three empirical distributions, from Websearch [46], Datamining [42], and Hadoop [22] applications, as well as a synthetic Pareto distribution [1]. Another distribution we consider is what we denote as the *Case Study* distribution and it is used to provide more intuition and highlight the benefit of Cerberus. The Case Study flow distribution has flow sizes of three categories: small (s), medium (m), and large (ℓ). Recalling the CDF of flow sizes from real-world traffic shown in Figure 1(a), we observe that more than 50% of the bytes are transmitted by elephant flows. Therefore, the probability distribution function (PDF) of flows sizes of the Case Study are as follows (see Table 1(a)). Small flows (defined to be of size $5KB$) constitute 4.95% of the total number of flows. These flows are considered to be latency-sensitive, but capture a tiny



(a) Accumulated traffic processed at the load $L = 0.7$ (b) Poisson arrivals - throughput

Fig. 4. Throughput from flow-level simulations for Datamining [42].

fraction of bytes. About 50% of the bytes are transmitted in large (elephant) flows of size $1GB$. Large flows, despite constituting a large share of the traffic in terms of number of bytes, represent just a tiny fraction of 0.05% of all flows. Medium flows are of size $0.5MB$; they form the majority of the flows, namely 95%, and represent all-to-all traffic (among active ToRs) since the source and the destination are chosen uniformly.

6.2 Simulation Results

The simulation is used to compare the different systems for an online traffic scenario that is based on the Datamining [42] flow distribution. In this scenario, the inter-arrival times of the flows follow an exponential distribution with mean values depending on the loads. The smaller the inter arrival times the higher the load. Note that the algorithms do not know when flows will arrive, but have perfect knowledge of the flow sizes.

Given the parameters setup of the simulation, the (maximum) throughput of each system as computed from Theorem 5 is: for Cerberus it is 0.79, for *rotor-net* 0.64, and for *expander-net* 0.57, respectively.

To validate this result, Figure 4(a) presents the *accumulated* traffic (in bits) that was served by each system as a function of the simulation time for traffic generated for load $L = 0.7$. From Theorem 5, Cerberus should be able to support this rate while *rotor-net* and *expander-net* do not. We can observe that initially the systems are unstable as flows start arriving, but around 0.3 seconds, they become more stable and are able to process the incoming traffic at a constant rate. This processing rate, or the slope of the accumulated served traffic, shows the throughput of the system for the given traffic load. We additionally plot the total incoming (or generated) traffic which is at rate 0.7 as expected (purple line). We can see that Cerberus is able to support this load and maintain the same rate (slope), which means that the *unserved* traffic volume stays constant and the throughput of Cerberus is *above* 0.7 as we analytically calculated. In contrast, both *rotor-net* and *expander-net* rates (slopes) are lower than 0.7 which means that the *unserved* traffic volume grows infinitely with time and their throughput must be lower than 0.7, as we analytically showed.

Figure 4(b) extends 4(a) and shows the normalized throughput of each system (i.e., the processing rate of incoming traffic) for *different* traffic loads (solids lines) . Cerberus achieves higher throughput for higher loads. Interestingly, for loads from 0.1 to 0.4, there is no significant difference between Cerberus and *rotor-net* since both systems can support the incoming load. Only for loads higher than 0.5, a Cerberus achieves higher throughput than *rotor-net*: for instance, for the 0.6 load case Cerberus throughput is higher by 0.1, which translates to 4 $Tbps$ higher throughput (i.e., $0.1 \cdot 16 \cdot 40\,Gbps \cdot 64$). Additionally, we plot the analytical bounds for the (maximum) throughput of each system as computed from Theorem 5 (dashed lines). Cerberus, *rotor-net* and *expander-net*

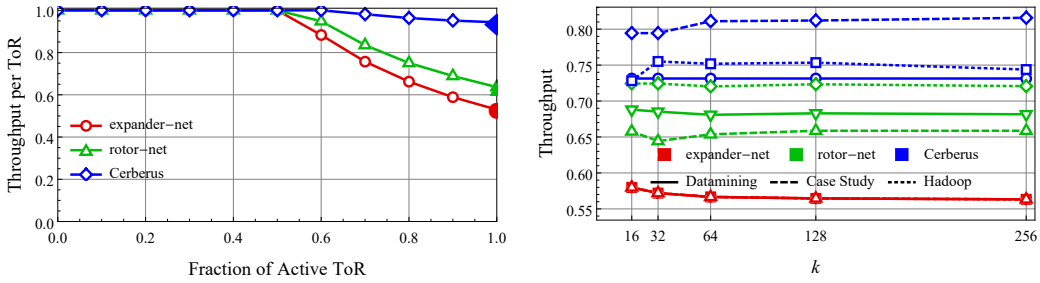


Fig. 5. (a) Datamining [42] - throughput-proportionality. (b) Throughput - different distributions and scale.

have throughput of 0.79, 0.64 and 0.57, respectively. It is important to note that these bounds are computed for the accumulated demand matrix, so they present an upper bound for the on-line traffic generation process. Nevertheless, all systems approach the theoretical bound, preserving the rank between them and the relative improvements.

The simulation results also confirm the benefit of having demand-aware components. This benefit becomes visible especially at higher loads, which are also an interesting operational region for datacenter operators aiming to efficiently utilize their infrastructure resources.

6.3 Large-Scale Numerical Evaluation

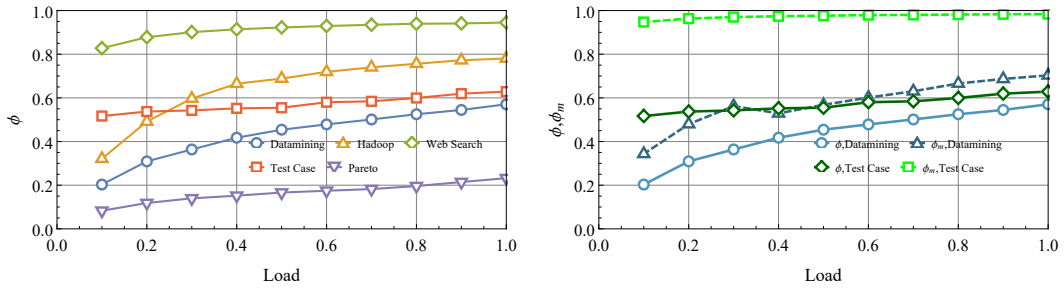
In this section, we consider a more realistic scenario where $n = 256$ ToRs and $k = 32$ spine switches and the other parameters default values are $R_r = 10\mu s, R_d = 15ms, \delta = 100\mu S$ and $r = 40Gbs$. Using Theorem 2,3 and 5, we can calculate all the necessary parameters for any given flow size distributions. For example, the results for the distributions of Figure 1(a) are presented in Table 1(b) for $x = 1$ and load $L = 0.5$.

Our numerical results based on Theorems 2,3 and 5 are also shown in Figure 3 which compares the demand completion times for $x = 1$, of Cerberus vs *expander-net* and *rotor-net*, for the Datamining and Case Study flow distributions. Figure 5(a) is based on Theorem 7 (see the Appendix for details) and compares the throughput for the skewed traffic ($x < 1$) of Cerberus vs *expander-net* and *rotor-net* for the Datamining flow distribution. The figure presents a *throughput-proportional* [30] experiment when the traffic is generated using the skewed traffic generation scheme $x \leq 1$, as the fraction x of active ToRs increases on the x-axis. Interestingly, our results show that both *rotor-net* and Cerberus are (theoretically) throughput-proportional similar to *expander-net*. Both Demand-aware and Rotor switches play a role in this; while Demand-aware switches distribute as much traffic as possible directly, Rotor switches are used to spread flows to non-active ToRs to use their help. This is shown formally in Theorem 7.

The benefits of Cerberus are evident from both Figures 3 and 5(a): Cerberus is both more efficient than the other systems for $x = 1$, achieves lower demand completion times, and has the same or higher throughput as the other systems for skewed traffic when $x \leq 1$. Figure 5(b) presents the throughput of Cerberus, *rotor-net* and *expander-net* for larger networks size with $n = 4k$ ToRs and different flow size distributions. The higher throughput of Cerberus compared to *rotor-net* and *expander-net* is kept at larger scale. Similar results (not shown) hold for $n = 8k$ and $n = 16k$.

6.4 Evaluation of the Skewness Parameter ϕ

An important parameter in our analysis is the *traffic skewness* value $0 \leq \phi \leq 1$ which approximates the fraction of bytes that are transmitted using one hop in *rotor topology*. When traffic is close to



(a) Traffic skewness ϕ for several distributions

(b) ϕ_m & ϕ for Datamining and Case Study

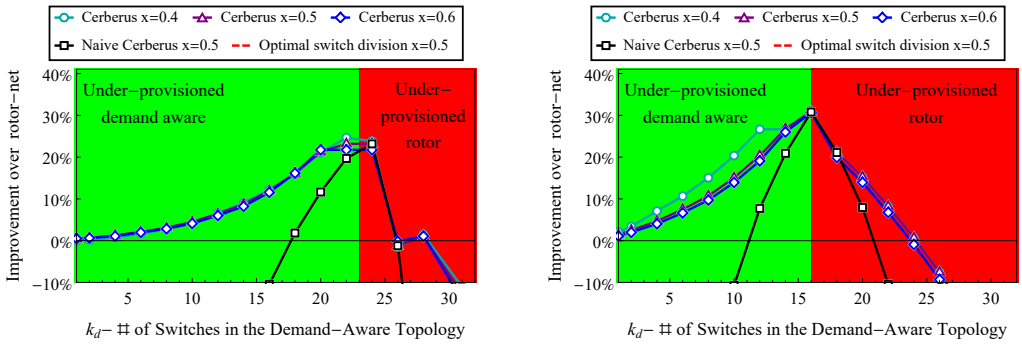
Fig. 6. (a) The behavior of ϕ on several different distributions. (b) The behavior of ϕ and ϕ_m for both Datamining [42] and the Case Study distributions.

uniform among destinations for a ToR, then ϕ will be close to 1: there are no available other ToRs to help the current ToR. When the destinations for a ToR are skewed, e.g., one large flow toward a single destination, ϕ will be close to 0 and most of the traffic will be transmitted via two hops (using Valiant routing [4]); this takes advantage of destinations that are free to help, but comes at the cost that flows (or packets) that use two hops take additional capacity from the network (i.e., "bandwidth tax").

In the following, we hence examine this parameter in more details. To approximate ϕ for a given distribution (or an empirical distribution), we define it as one minus the variation distance [53] from the uniform distribution. For a finite state PDF $P = \{p_1, p_2, \ldots, p_n\}$, the variation distance from the uniform distribution is defined as: $\Delta(P) = \frac{1}{2}\sum_{i=1}^{n}|p_i - \frac{1}{n}|$. $\Delta(P)$ measures the probability mass above the average. If P represents the distribution of traffic toward destinations for a ToR, then $\Delta(P)$ will capture the fraction of bytes that can benefit from sending by two hops (using the help of destinations that have load below the average); and $\phi = 1 - \Delta(P)$ is the fraction of packets that will use a single hop. Therefore, the average number of hops that a packet takes is: $1 \cdot \phi + 2(1 - \phi) = 2 - \phi$. Since ToRs are symmetric in our traffic model, a ToR that sends $(1 - \phi)$ of its traffic via two hops, asking the help of other ToRs, is expected to receive similar requests to help other ToRs. This means that the expected total traffic to be transmitted by all ToRs will be $\hat{T}(L)(2 - \phi)$. We can optimistically assume it will now be divided uniformly among destinations (otherwise the DCT of *rotor topology* will only be larger).

Figure 6(a) presents how ϕ behaves for several different distributions on varying degrees of load between 0.1 and 1. As a general rule, we see that the value of ϕ increases monotonically for all distributions. However, the rate at which it increases slows down as we approach higher loads. This is expected since the higher the load, the less skewed the empirical distribution is going to be. When the load approaches ∞ (above the line rate), ϕ will tend towards 1 as all source-destination pairs are sampled uniformly (regardless of the flow size). But the value of ϕ and its behavior for different loads can still vary significantly. For example, we can see that for Web Search, ϕ starts very high at around 0.8, and increases up to almost 1, while looking at the Pareto distribution, ϕ starts low, at around 0.1, and increases up to more than 0.2. This can be explained when looking at the two distributions, Web Search has few large flows, and as a result it is more uniform, while Pareto has some very large flows. There is also variance in the a degree of change for ϕ, for example, Hadoop increases from 0.25 to 0.8 while our Case Study changes by only approximately 0.1.

In Figure 6(b), we see how the sub-division of the traffic into large and medium flows affects the values of ϕ. In both the Datamining and our Case Study distributions, we observe that ϕ is



(a) Datamining [42] (b) Case Study

Fig. 7. The ratio between the demand completion time of Cerberus and *RotorNet* for non optimal switch divisions on the Datamining [42] distribution.

always lower than ϕ_m (which is the skewness of the traffic that is transmitted to the *rotor topology* of Cerberus). This is a natural outcome of our system's flow assignment algorithm: since there are much more medium flows than large flows, their resulting distribution is more uniform than for the relatively few large flows. This result is desirable, because, as discussed earlier, *rotor topology* is more efficient when ϕ is higher. We can also observe that among these two traffic distributions our Case Study shows higher values of ϕ_m, approaching 1; this means that all bytes are sent across to their destination in a single hop. Datamining has a more skewed flow size distribution (not only a single medium flow size), and as a result its medium flows yield a less uniform destinations distribution, giving a lower ϕ_m.

7 SENSITIVITY ANALYSIS

7.1 Sensitivity to Optimal Switch Division

We so far assumed that flows are always assigned "optimally" to their appropriate subsystem, according to Eq. (3). We will refer to this assignment of flows to switches as a "naive implementation" of Cerberus, which is optimal as long as the flow distribution remains unchanged. We will now explore how robust Cerberus is to non-optimal switch divisions, due to e.g., changes in the flow size distribution. While our vision is that future switches will be able to adapt their mode of operation and adjust to the demand, we will show that Cerberus is fairly robust even without such adjustments. To this end, we analyze two scenarios. First, we consider the case that the Demand-aware switches are *under-provisioned*: there are too many large flows for the demand-aware topology to handle and complete at the same time as the Rotor switches. In the second scenario, the Rotor topology is under-provisioned and there are too many medium flows for Rotor switches to handle and finish at same time as the Demand-aware topology. In Figure 7, (a) & (b) we see the expected ratio between the DCT of *rotor-net* and Cerberus for the naive implementation, and for Cerberus implemented according to Algorithm 1. The figure shows the behaviors of these two algorithms for the Datamining and Case Study distributions. The results are shown for both implementations for a load of $L = 0.5$. For Cerberus, we also show the results for the loads $L = 0.4$ and $L = 0.6$. We can see that in all cases the best ratio is attained when the system is given an optimal switch division (marked with the red dashed line); this represents an improvement of about 30% for the Case Study, and about 20% improvement for Datamining. When the switch division strays from the optimum, the ratio decreases. On the left (under-provisioned

demand-aware topology), when more and more large flows are sent to Rotor switches, we see that on all load levels Cerberus tends towards having the same DCT as *rotor-net*. However, lower loads tend to have a slower decrease in performance. This could be due to large flows being more sparse at lower loads, allowing the Rotor switches to spread the extra load more evenly. Overall, a similar phenomenon can be seen on the right of the optimal division (under-provisioned rotor topology). We see that Cerberus performs less favorably, as more and more medium size flows are sent to Demand-aware switches. Demand-aware switches have a high reconfiguration time, making them ill-advised to use the in the same manner for medium flows, as we have used them for large flows. This motivates the following improvement to Algorithm 1: we can use the "extra" Demand-aware switches (the residual switches past the optimal) to create a static expander topology, avoiding the high latency tax for medium flows.

7.2 Sensitivity to Parameters Setup

How will our conclusions change in the future, when, e.g., Cerberus has to serve rates of 100Gbps instead of 40Gbps, reconfiguration times will become faster, or flow sizes further increase? And what about the sensitivity of our results to our parameters setup? We can show that as long as certain proportions remain the same, similar benefits for Cerberus will arise. For example, assume that the rate of network cards increases by a factor λ, i.e., $r' = \lambda r$, and let us consider two scenarios. In a first scenario, assume that the probability distribution of the demand, $\mathcal{D}$, is scaled by a factor of λ as well. In this case the thresholds, t_m and t_ℓ are also increased by λ, and the proportion between medium and large flows remains the same, hence the overall throughput and DCT will also remain unchanged: see Theorem 6 and Eqs. (5), (8) and (9). Let us now consider a second scenario where the reconfiguration times R_d, R_r and the slot time δ are all reduced by a factor of λ. Also in this case the equations show that all demand completion times remain the same. These are just two examples: in fact, for any λ there are infinitely many combinations that will keep the demand completion times unchanged.

Generally, we can indeed expect that rates will increase further in the future, while the reconfiguration times will decrease. For example, newly proposed fully integrated, high port-count silicon photonic switches are capable to support sub-microsecond reconfiguration time across hundreds of ports [58]. As flow sizes are likely to further increase as well [41, 42], we believe that the observed benefits will stay in the future.

8 RELATED WORK

Our paper builds upon several innovative approaches that were recently developed to improve datacenter network performance. We will organize our review of related works according to the classification considered in this paper: static and dynamic, where the latter is further subdivided into demand-oblivious and demand-aware.

The Clos topologies and multi-rooted fat-trees are the most widely deployed static datacenter networks, and come in different flavors [31, 36, 37]. Recently, also modular hypercubic network designs [33, 34] as well as expander-based topologies [30, 35] have been proposed and analyzed.

Dynamic demand-oblivious topologies were introduced by Mellette et al. In their first work on RotorNet [4], a scalable optical datacenter network design (circuit-based), the authors show that very high bandwidth can be provided by actually emulating a full-mesh network, dynamically reconfiguring the circuit switches constituting the datacenter. RotorNet is a hybrid design and serves low-latency traffic over a static network. In a follow up work, Opera [5], the authors improved upon RotorNet by presenting a deterministic reconfiguration scheme which ensures that at any moment in time, the network implements an expander graph, while over time, bandwidth-efficient single-hop paths are provided between all racks. However, in contrast to a network combining a static

expander with an *independent* demand-oblivious rotor-based switch (as used also in Cerberus), such dynamic expander graphs introduce challenging routing and synchronization constraints. Recently, Ballani et al. [1] showed that nanosecond-granularity reconfiguration is possible in an optically-switched network connecting thousands of nodes.

The networks discussed above have in common that their topology does not depend on the current demand. In contrast, the goal of demand-aware networks is to exploit specific spatial and temporal structure in the workload [20]: several measurement studies show that traffic matrices are known to be sparse and skewed [20–22, 59, 60], and that traffic can be bursty over time [61, 62].

Existing demand-aware networks can be classified according to the granularity of reconfigurations. Solutions such as Proteus [16], OSA [17] or DANs [38], among other, are more coarse-granular and e.g., rely on a (predicted) traffic matrix. Solutions such as ProjecToR [11, 63], MegaSwitch [8], Eclipse [14], Helios [6], Mordia [64], C-Through [12], ReNet [65] or SplayNets [13] are more fine-granular and support per-flow reconfiguration and decentralized reconfigurations. Reconfigurable demand-aware networks may rely on expander graphs, e.g., Flexspander [39], and are currently also considered as a promising solution to speed up data transfers in supercomputers [18, 19]. Demand-aware networks raise novel optimization problems related to switch scheduling [66], and recently interesting first insights have been obtained both for offline [14] and for online scheduling [15, 63, 67, 68].

Due to the increased reconfiguration time experienced in demand-aware networks, many of these solutions additionally rely on a fixed network. For example, ProjecToR always maintains a "base mesh" of connected links that can handle low-latency traffic while it opportunistically reconfigures free-space links in response to changes in traffic patterns. To give another example, OSA allows to reserve some circuit-switch ports specifically to ensure connectivity for low-latency traffic; MegaSwitch could similarly support low-latency traffic. However, we are not aware of any work analyzing the capacity of different reconfigurable networks under general flow size distributions in a formal and explicit way as we do. Our paper is also the first to provide a unified methodology which enables us to study the consequences of mismatching traffic to specific network types.

9 CONCLUSION AND DISCUSSION OF FUTURE WORK

This paper uncovered a potential of serving datacenter traffic with the switch technology that best matches its structure. By tapping into this potential, we developed a solution, Cerberus, which we showed to significantly improve throughput. To this end, we also presented an analytical framework which allowed us deriving explicit formal performance bounds for general flow size distributions, and also reported on empirical results for the most widely studied traffic patterns. Our results reveal a non-trivial dependency of the performance on the reconfiguration times provided by the specific technology and the skewness of the traffic pattern. This sheds an interesting new light on the performance tradeoffs in existing architectures, to which our analysis applies.

We believe that our work opens several interesting avenues for future research. For example, we find it intriguing that the simple algorithms we presented already provide good results, which motivates us to study more complex algorithms in the future. In general, we understand our work on the performance aspects of reconfigurable networks as a first step, and it remains to explore several additional aspects, such as cost-efficiency or robustness.

In particular, when implementing Cerberus based on the, mainly theoretical, model presented here, we expect some challenges. One important issue is the classification of the flows into small, medium, or large. In the paper, we assume that this is known before the flow is sent on either system, and we also assume that it is easy to route flows individually, according to their size. Any implementation of Cerberus will have to find a practical solution for these issues, which we hope

to explore in future work. For example, promoting a flow from smaller size to larger size might be possible if it stays longer in the system.

Our analysis also assumes that every large flow triggers a reconfiguration of the Demand-aware switches. A more practical version might optimize this by sending two large flows (with the same source and destination address) on the link which only had to be reconfigured once. This will allow Cerberus to better utilize the available bandwidth. Cerberus can tolerate some deviation from the expected switch division as we have seen in Section 7.1. In order to reach optimal results, however, Cerberus requires either the flow distribution to be static or switches to change roles from time to time (from Static switches to Rotor switches, etc.). This might be realized by a flexible switch which can turn into either switch type. Finally, we note that implementing *expander-net* via demand-aware switches is relatively easy, but does not utilize the full flexibility of such switches.

ACKNOWLEDGEMENTS

We thank our shepherd Ran Ben Basat and the anonymous reviewers. The comments we received along the cycle of the submission process, helped to significantly improve the quality of the paper. This work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 864228 - AdjustNet). The work was also funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 438892507. The authors alone are responsible for the content of the paper. This work is part of the PhD thesis of the first two authors.

REFERENCES

[1] H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen, *et al.*, "Sirius: A flat datacenter network with nanosecond optical switching," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pp. 782–797, 2020.

[2] 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," *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, vol. 42, no. 4, pp. 443–454, 2012.

[3] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," in *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2009.

[4] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, "Rotornet: A scalable, low-complexity, optical datacenter network," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 267–280, ACM, 2017.

[5] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, "Expanding across time to deliver bandwidth efficiency and low latency," in *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pp. 1–18, 2020.

[6] 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," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 339–350, 2011.

[7] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "Firefly: A reconfigurable wireless data center fabric using free-space optics," in *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 319–330, ACM, 2014.

[8] L. Chen, K. Chen, Z. Zhu, M. Yu, G. Porter, C. Qiao, and S. Zhong, "Enabling wide-spread communications on optical fabric with megaswitch," in *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, (USA), pp. 577–593, USENIX Association, 2017.

[9] Y. J. Liu, P. X. Gao, B. Wong, and S. Keshav, "Quartz: A new design element for low-latency dcns," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 283–294, Aug. 2014.

[10] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "Osa: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 2, pp. 498–511, 2014.

[11] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "Projector: Agile reconfigurable data center interconnect," in *Proceedings of the 2016 ACM SIGCOMM*

Conference, pp. 216–229, ACM, 2016.
[12] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 327–338, 2011.
[13] S. Schmid, C. Avin, C. Scheideler, M. Borokhovich, B. Haeupler, and Z. Lotker, "Splaynet: Towards locally self-adjusting networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 24, no. 3, pp. 1421–1433, 2016.
[14] S. B. Venkatakrishnan, M. Alizadeh, and P. Viswanath, "Costly circuits, submodular schedules and approximate carathéodory theorems," *Queueing Systems*, vol. 88, no. 3-4, pp. 311–347, 2018.
[15] R. Schwartz, M. Singh, and S. Yazdanbod, "Online and offline greedy algorithms for routing with switching costs," *arXiv preprint arXiv:1905.02800*, 2019.
[16] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, "Proteus: a topology malleable data center network," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 8, ACM, 2010.
[17] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "Osa: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Transactions on Networking*, vol. 22, pp. 498–511, April 2014.
[18] M. Hampson, "Reconfigurable optical networks will move supercomputerdata 100x faster," in *IEEE Spectrum*, 2021.
[19] F. Douglis, S. Robertson, E. Van den Berg, J. Micallef, M. Pucci, A. Aiken, M. Hattink, M. Seok, and K. Bergman, "Fleet—fast lanes for expedited execution at 10 terabits: Program overview," *IEEE Internet Computing*, 2021.
[20] C. Avin, M. Ghobadi, C. Griner, and S. Schmid, "On the complexity of traffic traces and implications," in *Proc. ACM SIGMETRICS*, 2020.
[21] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 267–280, ACM, 2010.
[22] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 123–137, ACM, 2015.
[23] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, vol. 45, pp. 123–137, ACM, 2015.
[24] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *SIGCOMM*, 2010.
[25] A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *CoRR*, vol. abs/1802.05799, 2018.
[26] A. Faraj, P. Patarasuk, and X. Yuan, "A study of process arrival patterns for mpi collective operations," *International Journal of Parallel Programming*, vol. 36, no. 6, pp. 543–570, 2008.
[27] C. Yang, "Tree-based allreduce communication on mxnet," 2018.
[28] A. Singla, "Fat-free topologies," in *Proc. 15th ACM Workshop on Hot Topics in Networks (HotNets)*, pp. 64–70, 2016.
[29] M. N. Hall, K.-T. Foerster, S. Schmid, and R. Durairajan, "A survey of reconfigurable optical networks," in *Optical Switching and Networking (OSN), Elsevier*, 2021.
[30] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 281–294, ACM, 2017.
[31] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 63–74, ACM, 2008.
[32] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *2008 International Symposium on Computer Architecture*, pp. 77–88, IEEE, 2008.
[33] 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," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.
[34] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "Mdcube: a high performance network structure for modular data center interconnection," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pp. 25–36, ACM, 2009.
[35] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers, randomly.," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, vol. 12, pp. 17–17, 2012.
[36] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *ACM SIGCOMM computer communication review*, vol. 45, no. 4, pp. 183–197, 2015.
[37] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A fault-tolerant engineered network," in *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pp. 399–412, 2013.
[38] C. Avin, K. Mondal, and S. Schmid, "Demand-aware network designs of bounded degree," in *Proc. International Symposium on Distributed Computing (DISC)*, 2017.

[39] M. Y. Teh, Z. Wu, and K. Bergman, "Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 12, no. 4, pp. B44–B54, 2020.
[40] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla, "Measuring and understanding throughput of network topologies," in *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 761–772, IEEE, 2016.
[41] M. Khani, M. Ghobadi, M. Alizadeh, Z. Zhu, M. Glick, K. Bergman, A. Vahdat, B. Klenk, and E. Ebrahimi, "SiP-ML: High-Bandwidth Optical Network Interconnects for Machine Learning Training," SIGCOMM, 2021.
[42] 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 *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pp. 51–62, 2009.
[43] MEMS-Optical-Switches. http://www.diconfiber.com/products/mems_matrix_optical_switches.php.
[44] "Edge 64 Optical Circuit Switch." https://www.calient.net/products/edge640-optical-circuit-switch/.
[45] P. Namyar, S. Supittayapornpong, M. Zhang, M. Yu, and R. Govindan, "A throughput-centric view of the performance of datacenter topologies," in *To appear in Proceedings of the ACM SIGCOMM 2021 conference*, 2021.
[46] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *Proceedings of the ACM SIGCOMM 2010 conference*, pp. 63–74, 2010.
[47] A. Singla, P. B. Godfrey, and A. Kolla, "High throughput data center topology design.," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 29–41, 2014.
[48] N. Jain, A. Bhatele, X. Ni, N. J. Wright, and L. V. Kale, "Maximizing throughput on a dragonfly network," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 336–347, IEEE, 2014.
[49] Y. S. Fainman, J. Ford, W. M. Mellette, S. M. G. Porter, A. C. Snoeren, G. Papen, S. Saeedi, J. Cunningham, A. Krishnamoorthy, M. Gehl, C. T. DeRose, P. S. Davids, D. C. Trotter, A. L. Starbuck, C. M. Dallo, D. Hood, A. Pomerene, and A. Lentine, "Leed: A lightwave energy-efficient datacenter," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, pp. 1–3, 2019.
[50] Y. Ben-Itzhak, C. Caba, L. Schour, and S. Vargaftik, "C-share: Optical circuits sharing for software-defined data-centers," *arXiv preprint arXiv:1609.04521*, 2016.
[51] L. Chen, J. Lingys, K. Chen, and F. Liu, "Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 191–205, 2018.
[52] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-agnostic flow scheduling for commodity data centers," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pp. 455–468, 2015.
[53] J. S. Rosenthal, "Convergence rates for markov chains," *Siam Review*, vol. 37, no. 3, pp. 387–405, 1995.
[54] H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen, *et al.*, "Sirius: A flat datacenter network with nanosecond optical switching," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pp. 782–797, 2020.
[55] L. G. Valiant, "A scheme for fast parallel communication," *SIAM journal on computing*, vol. 11, no. 2, pp. 350–361, 1982.
[56] X. S. Huang, X. S. Sun, and T. E. Ng, "Sunflow: Efficient optical circuit scheduling for coflows," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pp. 297–311, 2016.
[57] O. Goldreich, "Basic facts about expander graphs," in *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pp. 451–464, Springer, 2011.
[58] T. J. Seok, N. Quack, S. Han, R. S. Muller, and M. C. Wu, "Large-scale broadband digital silicon photonic switches with vertical adiabatic couplers," *Optica*, vol. 3, pp. 64–70, Jan 2016.
[59] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proc. 9th ACM SIGCOMM conference on Internet measurement*, pp. 202–208, 2009.
[60] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 5, pp. 44–48, 2012.
[61] S. Zou, X. Wen, K. Chen, S. Huang, Y. Chen, Y. Liu, Y. Xia, and C. Hu, "Virtualknotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter," *Computer Networks*, vol. 67, pp. 141–153, 2014.
[62] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy, "High-resolution measurement of data center microbursts," in *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, (New York, NY, USA), pp. 78–85, ACM, 2017.
[63] J. Kulkarni, S. Schmid, and P. Schmidt, "Scheduling opportunistic links in two-tiered reconfigurable datacenters," in *33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2021.

[64] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 447–458, Association for Computing Machinery, 2013.
[65] C. Avin and S. Schmid, "Renets: Statically-optimal demand-aware networks," in *Proc. SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, 2021.
[66] N. McKeown, "The islip scheduling algorithm for input-queued switches," *IEEE/ACM transactions on networking*, vol. 7, no. 2, pp. 188–201, 1999.
[67] M. Dinitz and B. Moseley, "Scheduling for weighted flow and completion times in reconfigurable networks," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1043–1052, 2020.
[68] M. Bienkowski, D. Fuchssteiner, J. Marcinkowski, and S. Schmid, "Online dynamic b-matching with applications to reconfigurable datacenter networks," in *Proc. 38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (PERFORMANCE)*, 2020.
[69] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bulletin of the American Mathematical Society*, vol. 43, no. 4, pp. 439–561, 2006.

APPENDICES

A THROUGHPUT AND DCT ANALYSIS

In this section we present analytical results for the performance of Cerberus. We analyze the throughput and the demand completion time: the *total* time it takes to serve a demand matrix from the traffic generation model that arrives in one second. Theorem 6 and 7 in turn enable us to derive our main result, in Theorem 5, about the throughput.

A.1 Analysis of the Case $x = 1$ (All Active)

We first prove the following result.

Theorem 6. *Let $\hat{T}(L)$ be the accumulated demand matrix of flows that were generated by $\mathcal{T}(1, L, \mathcal{D})$ in one second for n ToRs with k uplinks of rate r. Let ϕ be the skewness of $\hat{T}(L)$. The expected demand completion times of the systems* Cerberus, expander-net *and* rotor-net *are as follows.*
For Cerberus *(upper bound):*

$$\mathrm{DCT}(Cerberus, \hat{T}(L), k) = L \cdot \alpha \tag{5}$$

where $\alpha = \frac{\hat{T}(1,\ell)}{k_d^*}\left(\mathbb{E}\left[\frac{R_d}{|f|}\right] + \frac{1}{r}\right)$. The expected size (of the reciprocal flow sizes) is taken only over large flows. k_d^* is the optimal number of Demand-aware switches, computed from the ratio between the optimal number of Demand-aware switches to Rotor switches denoted by k_r^*:

$$\frac{k_d^*}{k_r^*} = \frac{\hat{T}(1,\ell)}{\hat{T}(1,m)/t_m} \cdot \frac{\mathbb{E}\left[\frac{R_d}{|f|}\right] + \frac{1}{r}}{(2-\phi_m)(R_r+\delta)} \tag{6}$$

where ϕ_m is the traffic skewness of the medium size flows.
The threshold for large flows, t_ℓ can be computed by:

$$t_\ell \geq \frac{R_d \cdot t_m \cdot r}{(2-\phi)\cdot r \cdot (R_r+\delta) - t_m} \tag{7}$$

For* rotor-net:*

$$\mathrm{DCT}(\text{rotor-net}, \hat{T}(L), k) = L \cdot \beta \tag{8}$$

where $\beta = (2-\phi)\frac{R_r+\delta}{\delta}$.

***For* expander-net** *(lower bound):*

$$\mathrm{DCT}(\text{expander-net}, \hat{T}(L), k) \geq L \cdot \gamma \tag{9}$$

where $\gamma = \text{epl}(G(k))$ *is the expected path length of the* k*-regular expander with* n *nodes.*

Recall that τ denotes the type of a flow where $\tau \in \{s, m, \ell\}$ is the flow type and let t_m and t_ℓ denote the *thresholds* sizes to decide the flows type. Namely, *small* flows are of size less than t_m, *medium* flows are of size more than t_m and less than t_ℓ and, *large* flows are of size larger than t_ℓ.

We set the size of the medium threshold to be exactly $t_m = \delta r$ the flow size that can be transmitted in one slot.

Consider k switches all of them of the same type $\omega \in \{\mathcal{S}, \mathcal{R}, \mathcal{C}\}$. We denote by $\text{DCT}_\omega(T, k)$ the *demand completion time* to serve the demand T using these k switches. Further, let $\text{DCT}(sys, T, k)$ denote the demand completion time of system $sys \in \{\textit{expander-net}, \textit{rotor-net}, Cerberus\}$ for demand T using k switches. Note that for *expander-net* and *rotor-net* we have $\text{DCT}(\textit{expander-net}, T, k) = \text{DCT}_\mathcal{S}(T, k)$ and $\text{DCT}(\textit{rotor-net}, T, k) = \text{DCT}_\mathcal{R}(T, k)$ respectively, since these systems are built from a single switch type. But this is not the case for Cerberus which is built form a combination of all three switches types. We present the analysis from the easy to the hard case, first *expander-net*, then *rotor-net* and finally Cerberus.

A.1.1 Analysis of expander-net. We start by (optimistically) approximating the demand completion time of *expander-net*. We assume that traffic is distributed along all shortest paths with no delay due to packet loss or congestion. Hence, the only "cost" we consider is related to the path length, that is, each flow consumes bandwidth capacity proportional to the route length (i.e. "bandwidth tax"). For example, if the route length of all flows is two and all ToRs are working uniformly, the maximum achievable load is 50%; otherwise the total traffic would exceed the network capacity: the number of ToRs times the number of switches times the rate, i.e., $n \cdot k \cdot r$. Now let $G(k_s)$ denote a (random) k_s-regular expander built from k_s (random) matchings and let $\text{epl}(G(k_s))$ denote the *expected path length* of $G(k_s)$. The demand completion time of traffic $\hat{T}(L)$ can be bounded as follows:

$$\text{DCT}_\mathcal{S}(\hat{T}(L), k_s) \geq \hat{T}(L) \cdot \frac{\text{epl}(G(k_s))}{k_s \cdot r \cdot n} \tag{10}$$

From this we can compute the bound for an expander made from k switches and for our traffic model with load L:

$$\text{DCT}(\textit{expander-net}, \hat{T}(L), k) = \text{DCT}_\mathcal{S}(\hat{T}(L), k) = \frac{L \cdot nkr \cdot \text{epl}(G(k))}{nkr} = L \cdot \text{epl}(G(k)) \tag{11}$$

Therefore the completion time is linear in L. Using our test parameters we have found that a (random) 32-regular expander with 256 nodes has an expected path length that is about 1.85.

A.1.2 Analysis of rotor-net. Next we consider *rotor-net*. First we consider a completely uniform demand between all possible pairs denoted as $\overline{U}$ (arriving at time zero). In this all-to-all case, *rotor-net* will be almost optimal by serving requests in each slot according to the current matching of each switch; all ports will be continuously operating at 100% throughput, sending flows directly (in a single hop) from source to destination. The only inefficiency will be due to the reconfiguration time (the "latency tax"), to reconfigure between slots. In our setting this overhead is equal to $R_r/(R_r + \delta)$, about 9% if we use our parameters. So the demand completion time of k Rotor switches and such uniform demand $\overline{U}$ is for a single ToR (and for the system): (number of slots in $\overline{U}$) / (number switches) $\times$ (time for a slot). Formally,

$$\text{DCT}_\mathcal{R}(\overline{U}, k) = \frac{\overline{U}}{k \cdot r} \cdot \frac{R_r + \delta}{\delta} = \frac{\overline{U}}{\delta \cdot r} \cdot \frac{R_r + \delta}{k} = \frac{\overline{U}}{t_m} \cdot \frac{R_r + \delta}{k}$$

Next we consider the case of traffic $\hat{T}(L)$. In this case all ToRs sample flow sizes from the same flow distribution $\mathcal{D}$, but flows can have different sizes so we cannot assume the traffic is uniform

among all pairs or all-to-all. Dealing with non-uniform flows is more complex, since the number of larger flows could be relativity small, potentially leaving many links in each slot inactive.

A *rotor-net* overcomes this problem by using Valiant routing (load balancing) [4] where flows and packets can be sent via two hops and not directly. Flows (or packets) that use two hops may take additional capacity from the network (i.e. "bandwidth tax"). We model this situation with a *traffic skewness* parameter $0 \leq \phi \leq 1$ which approximates the fraction of bytes that a ToR in *rotor-net* sends using one hop. When traffic is close to uniform across destinations for a ToR, then ϕ will be close to 1 since there are no available other ToRs to help the current ToR. When the destinations for a ToR are skewed. e.g., one large flow toward a single destination, ϕ will be close to 0 and most of the traffic will be sent via two hops, taking advantage of destinations that are free to help. To approximate ϕ for a give distribution (or empirical distribution) we define it as one minus the variation distance [53] from the uniform distribution. For a finite state PDF $P = \{p_1, p_2, \ldots, p_n\}$ the variation distance from the uniform distribution is defined as:

$$\Delta(P) = \frac{1}{2}\sum_{i=1}^{n}|p_i - \frac{1}{n}| \tag{12}$$

$\Delta(P)$ measures the probability mass above the average. If, for a ToR, P represents the distribution of the destinations' load, $\Delta(P)$ will capture the fraction of packets that can benefit from being sent along two hops (using the help of destinations that have load below the average); $\phi = 1 - \Delta(P)$ is the fraction of packets that will use a single hop. Therefore the average number of hops that a packet takes is: $1 \cdot \phi + 2(1-\phi) = 2 - \phi$. Since ToRs are symmetric in this traffic model (i.e., $\hat{T}(L)$) a ToR that sends $(1-\phi)$ of its traffic via two hops, asking for the help of other ToRs, would expect similar requests to help other ToRs. This means that the expected total traffic which needs to be sent will be $\hat{T}(L)(2-\phi)$. We can optimistically assume that it will now be divided uniformly among destinations (otherwise the DCT will only be larger) and we can formally generalize the DCT lower bound to:

$$\mathrm{DCT}_{\mathcal{R}}(\hat{T}(L), k) = \frac{\hat{T}(L)}{t_m}(2-\phi)\frac{R_r + \delta}{nk} = \frac{\hat{T}(L)}{nkr}(2-\phi)\frac{R_r + \delta}{\delta} \tag{13}$$

Since *rotor-net* is composed of k Rotor switches we can now bound the demand completion time of load x for *rotor-net* by:

$$\mathrm{DCT}(\textit{rotor-net}, \hat{T}(L), k) = \mathrm{DCT}_{\mathcal{R}}(\hat{T}(L), k) = \frac{Lrk}{t_m}(2-\phi)\frac{R_r + \delta}{k} = L(2-\phi)\frac{R_r + \delta}{\delta} \tag{14}$$

If ϕ is a constant then this is a linear function. In practice ϕ can vary, but nevertheless the function can still be approximated well by a linear function. This is demonstrated in Figure 3. The slope of this linear function for *rotor-net*, in our setting and the flow distribution of our case study is about 1.53, while for the Datamining distribution, is 1.88.

A.1.3 Analysis of Cerberus. We now turn to discuss Cerberus whose analysis is a bit more complex. Let's start with large flows which, according to Algorithm 1, are transmitted via the Demand-aware switches. The Demand-aware component operates by reconfiguring a direct link from the source of a flow to its destination. The flow is then transmitted along a single hop. Assuming the reconfiguration time of a single Demand-aware switch is R_d and the transmission time of a single large flow f of size $|f|$ is $|f|/r$, the flow completion time for a single flow on a single switch is $R_d + \frac{|f|}{r}$.

Finding the threshold for large flows. To determine the threshold we would like to find a value for a flow size $|f|$ for which sending the flow f on a k-Rotor-switch network is slower than

transmitting the same flow on a k-Demand-aware-switch network. Following Eq. (13) we note that the transmission time is a function of the global traffic skewness ϕ:

$$\begin{aligned}\frac{R_d + \frac{|f|}{r}}{k} &\leq \frac{|f|}{t_m} \cdot (2-\phi) \cdot \frac{R_d + \delta}{k} \\ R_d &\leq \frac{|f|}{t_m} \cdot (2-\phi) \cdot (R_r + \delta) - \frac{|f|}{r} \\ \frac{R_d \cdot t_m \cdot r}{(2-\phi) \cdot r \cdot (R_r + \delta) - t_m} &\leq |f| \end{aligned} \tag{15}$$

When we use our parameters we have that for $\phi = 0$ (all packets of the flow are sent via two hops in *rotor-net*), $|f| = 15MB$, and for $\phi = 1$ (all packets of the flow are sent via one hops in *rotor-net*), $|f| = 187.5MB$. The threshold we use in our evaluation is $125MB$.

The demand completion time. For a given partition of the k switches to the three types of switches: k_s Static switches, k_r Rotor switches and k_d demand-aware switches, and a uniform traffic model, $\hat{T}(L)$ with load x, the demand completion time of Cerberus is the maximal completion time among the three sub-components. Formally:

$$\mathrm{DCT}(Cerberus, \hat{T}(L), k) = \max \begin{cases} \mathrm{DCT}_{\mathcal{S}}(\hat{T}(L,s), k_s) \\ \mathrm{DCT}_{\mathcal{R}}(\hat{T}(L,m), k_r) \\ \mathrm{DCT}_{\mathcal{C}}(\hat{T}(L,\ell), k_d) \end{cases} \tag{16}$$

Assuming the sources and destinations are distributed uniformly and there are k_d demand-aware switches, then the demand completion time of a single ToR (and by symmetry all ToRs) is approximated by

$$\mathrm{DCT}_{\mathcal{C}}(\hat{T}(L,\ell), k_d) = \frac{1}{k_d} \sum_{f \in \hat{T}(L,\ell)} (R_d + \frac{|f|}{r}) \tag{17}$$

In the worst case we have $\hat{T}(L,\ell)/t_\ell$ large flows, each of size t_ℓ: as the flows get larger, the reconfiguration time R_d is better amortized by the transmission time of the flow. The upper bound of the demand completion time is then

$$\mathrm{DCT}_{\mathcal{C}}(\hat{T}(L,\ell), k_d) \leq \frac{\hat{T}(L,\ell)}{t_\ell} \cdot \frac{R_d + \frac{t_\ell}{r}}{nk_d} = \frac{\hat{T}(L,\ell)}{nk_d} \left(\frac{R_d}{t_\ell} + \frac{1}{r} \right) \tag{18}$$

When the threshold t_ℓ is far from the average large flow it is better to take the expected completion time.

$$\begin{aligned}\mathrm{DCT}_{\mathcal{C}}(\hat{T}(L,\ell), k_d) &= \sum_{f \in \hat{T}(L,\ell)} \left(\frac{\Pr(f)\hat{T}(L,\ell)}{|f|} \frac{R_d + \frac{|f|}{r}}{nk_d} \right) = \sum_{f \in \hat{T}(L,\ell)} \left(\frac{\Pr(f)\hat{T}(L,\ell)}{nk_d} \left(\frac{R_d}{|f|} + \frac{1}{r} \right) \right) \\ &= \frac{\hat{T}(L,\ell)}{nk_d} \left(\mathbb{E}\left[\frac{R_d}{|f|} \right] + \frac{1}{r} \right) = L \frac{\hat{T}(1,\ell)}{nk_d} \left(\mathbb{E}\left[\frac{R_d}{|f|} \right] + \frac{1}{r} \right) \end{aligned} \tag{19}$$

Next we discuss how to find an optimal partition of the switches. For example, assuming $k_s = 5$ and that the completion time for the expander component is negligible (due to low traffic volume), the optimal division of the switches, denoted as k_r^* and k_d^*, is such that the completion time of the corresponding components will be identical. This follows from the fact that the completion time

of each sub-component, as shown above, is monotonically decreasing in the number of switches. Equalizing the two components allows us to compute the optimal number of switches, as follows (where ϕ_m denotes the skewness of the medium size traffic):

$$\begin{aligned} \mathrm{DCT}_C(\hat{T}(L,\ell), k_d^*) = \mathrm{DCT}_{\mathcal{R}}(\hat{T}(L,m), k_r^*) \Rightarrow \\ \frac{\hat{T}(L,\ell)}{nk_d^*}\left(\mathbb{E}\left[\frac{R_d}{|f|}\right]+\frac{1}{r}\right) = \frac{\hat{T}(L,m)}{t_m}\cdot(2-\phi_m)\cdot\frac{R_r+\delta}{nk_r^*} \Rightarrow \\ \frac{k_d^*}{k_r^*} = \frac{\hat{T}(L,\ell)}{\hat{T}(L,m)/t_m}\cdot\frac{\mathbb{E}\left[\frac{R_d}{|f|}\right]+\frac{1}{r}}{(2-\phi_m)(R_r+\delta)} \end{aligned} \tag{20}$$

For example, for the Case Study distribution and traffic load $L = 0.5$ we calculated ϕ_m and by plugging in our defult values, we get that $k_d^* = 16$ and $k_r^* = 16$ (using rounding and recalling that $k_r^* + k_d^* = 32$).

Following Eq. (19) we can now approximate the demand completion time of Cerberus as the completion time of the k_d^* Demand-aware switches (recall that it is equal to the completion time of the k_r^* Rotor switches).

$$\mathrm{DCT}(Cerberus, \hat{T}(L), k) \le L\cdot\frac{\hat{T}(1,\ell)}{nk_d^*}\left(\mathbb{E}\left[\frac{R_d}{|f|}\right]+\frac{1}{r}\right) \tag{21}$$

Interestingly, this is again a linear function (for the same switch partition and threshold), and considering our default parameters and the distribution of our case study, we obtain a slope of 1.09, while for the data mining distribution the slope is 1.25. In both cases the slope is smaller than those of *expander-net* and *rotor-net*.

Figure 3 presents the demand completion times for *expander-net*, *rotor-net* and Cerberus for $\hat{T}(L)$, both for the Case Study and for Datamining, with optimal switches partitioned according to Table 1. We can clearly see that Cerberus outperforms the other systems for every load. For the Datamining use case, Cerberus can operate without saturating the network with load up to of 80%, while *rotor-net* and *expander-net* can work up to 60% and 53% respectively.

A.2 Analysis of the Case $x < 1$ (Skewed)

Next we analyze the skewed traffic model $\mathcal{T}(x, L, \mathcal{D})$, where only a fraction x of the ToRs are active and operating at load L. Let $T_x(L)$ be the accumulated demand matrix of flows that were generated by $\mathcal{T}(x, L, \mathcal{D})$ in one second for the n ToRs with k uplinks of rate r.

Our goal in this scenario is to design a network which allows for a maximal possible throughput L. Let $L^*(x) = \arg\max_L \mathrm{DCT}(T_x(L)) \le 1$ and $L^* \le 1$ is highest throughput that the network can support (given x).

The assumption of skewed traffic is more realistic than uniform traffic, but it is also harder to analyze. The main challenge arises from the fact that under the skewed traffic model, only a fraction x of the ToRs are active, but in principle, it may be possible to utilize the $1-x$ fraction of non-active ToRs to help the active ones. The question is if this can be done, and what is the maximum achievable throughput $L^*(x)$. We formally analyze this scenario by computing the demand completion time for $T_x(1)$ (with ports working at full throughput r). If the completion time is less than one second, we say that the throughput is one; otherwise we adjust the throughput, L to find the largest throughput for which the completion time is less than one second. Let $T_x(L, \tau)$, as before, denote the expected number of bytes per second in flows of type τ generated by $T_x(L)$.

We start again with the *expander-net*. We optimistically assume the *expander-net* has all the good properties expanders should have [69], i.e., large expansion, multiple disjoint paths, small mixing times, etc. Basically these properties guarantee that even for a small set of communicating ToRs, the traffic will efficiently spread across the entire network, utilizing the non-active ToRs as much as possible. Therefore, as before, we assume only the capacity restriction and consider the demand completion time as:

$$\mathrm{DCT}(\textit{expander-net}, T_x(L), k) = Lx \cdot \mathrm{epl}(G(k)) \tag{22}$$

We can find the throughput by solving for L.

$$Lx \cdot \mathrm{epl}(G(k)) = 1 \tag{23}$$

From this we can find $L^*(x)$ as:

$$L^*(x) = \min\left(\frac{1}{x \cdot \mathrm{epl}(G(k)}, 1\right) \tag{24}$$

Following the definitions and observations in [30] and in particular Figure 2 within, the above Eq. (24) shows that a static expander will be a *throughput-proportional network*, namely it will be "able to distribute its capacity evenly across only the set of servers with traffic demands" [30].

What about a rotor-based network or a network based on Cerberus? Achieving throughput-proportionality seems to be non-trivial. For example if the load is 50%, how can the network exploit the capacity of the 50% non-active ToRs to help the active ones work at 100% throughput? In [30], it was shown that the fat-tree is not throughput-proportional.

Interestingly, we can show that Rotor switches actually work very well in this scenario and that *rotor-net* is also a throughput-proportional network. This is a novel result and was not discussed in previous work [4, 5, 30]. The result is due to the Valiant routing property of *rotor-net*: if every flow is transmitted using this mechanism, flows will be forwarded using the non-active ToR, keeping *all* switch ports sending at full rate. Moreover, if a fraction x of the network is active and generates a uniform traffic (within the active fraction), this means that a fraction x of the *time*, switch ports can directly communicate to their destinations, and only a $1-x$ fraction of the time flows will need to use two hops. If the traffic within the x fraction of active ToRs is non-uniform we can again use the skewness parameter ϕ to approximate the fraction of traffic that needs one hop and the fraction that needs two hops. The average number of hops will then be $x(\phi + 2(1-\phi)) + 2(1-x) = 2 - \phi x$. The total amount of traffic to be sent in the whole network will now be $T_x(1)(2-\phi x) = xnkr(2-\phi x)$, but since we are using two hops, it will be uniformly divided among ToRs and destinations, and we can approximate the demand completion time of a single ToR as:

$$\mathrm{DCT}(\textit{rotor-net}, T_x(1), k) = \mathrm{DCT}_{\mathcal{R}}(T_x(1), k) = \frac{T_x(1)}{t_m}(2-\phi x) \cdot \frac{R_r + \delta}{nk} = x(2-\phi x) \cdot \frac{R_r + \delta}{\delta} \tag{25}$$

And find the throughput by solving for L.

$$\begin{aligned} \mathrm{DCT}(\textit{rotor-net}, T_x(L), k) = \mathrm{DCT}_{\mathcal{R}}(T_x(L), k) = 1 \Rightarrow \\ 1 = Lx(2-\phi x) \cdot \frac{R_r + \delta}{\delta} \end{aligned} \tag{26}$$

and

$$L^*(x) = \min\left(\frac{1}{x(2-\phi x) \cdot \frac{R_r+\delta}{\delta}}, 1\right) \tag{27}$$

The above equation implies that *rotor-net* is throughput-proportional. For the setting of our default parameters and our case study distribution with $\phi = 0.49$ it supports up to 50% of the ToRs working at full rate, the same as for the expander.

Observation 1. rotor-net *is throughput-proportional.*

We next discuss Cerberus. We already know that the Rotor switches are throughput-proportional, so what about the demand-aware switches? Due to reconfiguration time, the demand-aware switches may not be able to serve all large flows generated by the active ToRs. Furthermore, our current design of demand-aware switches does not support 2-hops routing. The "latency tax" due to reconfigurations in our numerical example is at most 15%, $\frac{R_d}{t_\ell/r}$, so when an active ToR is working at a full rate, it cannot send all of its large flows to the demand-aware switches.

Let z denote the expected fraction of large flows that k_d demand-aware switches can send in a second for a given ToR. We can find it using Eq. (19)

$$\begin{aligned} \mathrm{DCT}_C(\hat{T}(z,\ell),k_d) = \frac{\hat{T}(z,\ell)}{nk_d}\left(\mathbb{E}\left[\frac{R_d}{|f|}\right]+\frac{1}{r}\right) = 1 \Rightarrow \\ 1 = z\frac{\hat{T}(1,\ell)}{nk_d}\left(\mathbb{E}\left[\frac{R_d}{|f|}\right]+\frac{1}{r}\right) \Rightarrow \\ z = \frac{nk_d}{\hat{T}(1,\ell)\left(\mathbb{E}\left[\frac{R_d}{|f|}\right]+\frac{1}{r}\right)} \end{aligned} \tag{28}$$

Using z and following Algorithm 1, we can approximate the expected fraction of large flows, denoted as x^* per active ToR which cannot fit the Demand-aware switches and which will be sent to the Rotor switches when working at rate L (in $T_x(L)$), by:

$$x^* = \max(\frac{L-z}{L}, 0) \tag{29}$$

The expected amount of such traffic per active ToR will be $x^*L\hat{T}(1,\ell)/n = x^*\hat{T}(L,\ell)/n$.

Assuming that small flows are transmitted via static switches in Cerberus with negligible completion time, we can compute the demand completion time of Cerberus as the demand completion time of the k_r^* Rotor switches (since the Demand-aware switches are set up to have demand such that they finish at 1s). Following Eq. (25), we have:

$$\begin{aligned} \mathrm{DCT}(Cerberus, T_x(1), k) &= \mathrm{DCT}_{\mathcal{R}}(T_x(1,m) \cup xnx^*\hat{T}(1,\ell), k_r^*) \\ &= \frac{T_x(1,m)+xnx^*\hat{T}(1,\ell)}{t_m}(2-\phi x)\frac{R_r+\delta}{nk_r^*} \\ &= x\frac{\hat{T}(1,m)+x^*\hat{T}(1,\ell)}{t_m}(2-\phi x)\frac{R_r+\delta}{k_r^*} \end{aligned} \tag{30}$$

where ϕ is the skewness parameter that fit the traffic generated by $T_x(1,m) \cup x^*\hat{T}(1,\ell)$.
To find $L^*(x)$ we again need to solve for L s.t.

$$1 = \mathrm{DCT}(Cerberus, T_x(L), k) \Rightarrow$$
$$1 = x\frac{\hat{T}(L,m) + x^*\hat{T}(L,\ell)}{t_m}(2-\phi x)\frac{R_r+\delta}{k_r^*} \Rightarrow$$
$$1 = xL\frac{\hat{T}(1,m) + x^*\hat{T}(1,\ell)}{t_m}(2-\phi x)\frac{R_r+\delta}{k_r^*} \tag{31}$$

and $L^*(x)$ can be found numerically.

Figure 5(a) shows the throughput of *expander-net*, *rotor-net* and Cerberus following Eq. (24), (25) and (30), for the Datamining. All systems are throughput-proportional with Cerberus preforming better at higher loads (> 70%). We note that Cerberus is not optimized in the sense that, for example at 50% load, 50% of the links of the demand-aware switches are not used: this corresponds to about 25% of network capacity which is unused. We believe performance could hence be improved further by making these links static and random, and support the expander links.

We note that Figures 3(a) and 5(a) are related: for each system, the maximum supported throughput L^* under $x = 1$, i.e., $\mathrm{DCT}(T_1(L)) = 1$ in Figure 5 (b) exactly corresponds to the point on Figure 3 where the load is L and the demand completion time is $1s$, i.e., $\mathrm{DCT}(\hat{T}(1,L)) = 1$.

Thus we have derived the following theorem.

Theorem 7 (Skewed Traffic). *Let $T_x(L)$ be the accumulated demand matrix of flows that were generated by $\mathcal{T}(x, L, \mathcal{D})$ in one second for n ToRs with k uplinks of rate r. Let ϕ be the skewness of $T_x(L)$. The expected throughput L of the active ToR for the systems* Cerberus, expander-net *and* rotor-net *can be computed by solving the following equations for L:*
For Cerberus*:*

$$\mathrm{DCT}(Cerberus, T_x(L), k) = 1 \Rightarrow$$
$$xL\frac{\hat{T}(1,m) + x^*\hat{T}(1,\ell)}{t_m}(2-\phi x)\frac{R_r+\delta}{k_r^*} = 1 \tag{32}$$

For* rotor-net:*

$$\mathrm{DCT}(\text{rotor-net}, T_x(L), k) = 1 \Rightarrow$$
$$\min\left(\frac{1}{x(2-\phi x)\cdot\frac{R_r+\delta}{\delta}}, 1\right) = L^*(x) \tag{33}$$

For* expander-net:*

$$\mathrm{DCT}(\text{expander-net}, T_x(L), k) = 1 \Rightarrow$$
$$\min\left(\frac{1}{x\cdot \mathrm{epl}(G(k)}, 1\right) = L^*(x) \tag{34}$$

where $\mathrm{epl}(G(k))$ *is the expected path length of a random k-regular expander with n nodes.*

B THE GENERALIZED ROTOR SWITCH

In the original RotorNet [4] paper it is proposed to distribute the $n-1$ ($N_R - 1$ in their terminology) matchings of the complete graph among the k (N_{sw} there) switches, such that each switch is assigned $\lceil \frac{n-1}{k} \rceil$ or $\lfloor \frac{n-1}{k} \rfloor$ matchings and the cycle time is $\lceil \frac{n-1}{k} \rceil$ times the *slice* time, where the slice time is the *slot* time plus the reconfiguration time. See Figure 8.

There are two problem with this approach:

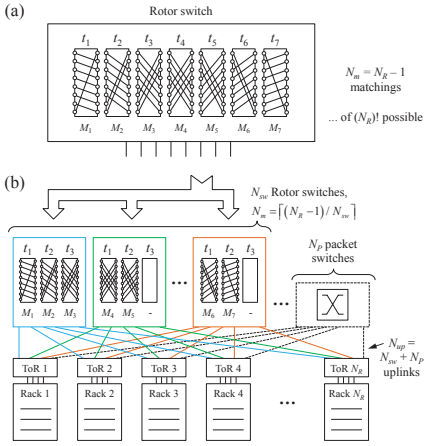


Figure 4: (a) A Rotor switch cycles through $(N_R - 1)$ matchings to provide full connectivity between racks. (b) Physically, these matchings are distributed among N_{sw} Rotor switches which, taken together, provide full connectivity.

Fig. 8. The original figure showing the RotorNet matching distribution between switches [4]

(1) When a new switch is added to the system a new distribution of the matching between the switches is needed. Similarly when switch is temporarily failing, some matching are not available.
(2) Synchronization issues can arise and raise questions such as: when some switches have $\lceil \frac{n-1}{k} \rceil$ matchings and some $\lfloor \frac{n-1}{k} \rfloor$ matchings, what should be the cycle time? And what is the slot time on the switch with less matchings? What can we say about the average time that a particular link will appear next in a matching?

We propose a slightly different approach (which should not require a significant technological change) which solves the above problems. Our proposal is that *every* switch will rotate among the $n-1$ matchings, keeping the same slot time for each matching as before. The only difference between the switches is a *time shift* in rotation between matching. Consider the example of Figure 8. There are seven matchings: $M_1, M_2, \ldots M_7$ where sw_1 rotates between $\{M_1, M_2, M_3\}$, sw_2 rotates between $\{M_4, M_5\}$ and sw_3 rotates between $\{M_6, M_7\}$. We propose that all switches rotate between the seven matchings, but at time 0, sw_1 will start with M_1, sw_2 will start with M_4 and sw_3 will start with M_6. At time 4, for example, the configuration will be sw_1 with M_4, sw_2 with M_7 and sw_3 with M_2. And so on. Each switch cycles via all matchings in a round robin manner. We then define the cycle time as the average time until all links of the complete graph appeared. This will be exactly $\frac{n-1}{k} \times$ (slice time). Moreover when a new switch is added to the system (e.g., to improve performance), then we do not need to redistribute the matchings between the switches; all we need to do it to update the shift between them. The new cycle time will become $\frac{n-1}{k+1} \times$ (slice time). Also when a switch fails, all needed matchings are still cycling with the active switches.

Received August 2021; revised October 2021; accepted November 2021