

Computer Sciences Department

The Case for Fine-Grained Traffic Engineering in Data Centers

Theophilus Benson

Aditya Akella

Ming Zhang

Technical Report #1666

October 2009



The Case for Fine-Grained Traffic Engineering in Data Centers

Theophilus Benson[†], *Aditya Akella*[†] and *Ming Zhang*^{*}
[†] *University of Wisconsin-Madison*; ^{*} *Microsoft Research*

Abstract

Data center traffic characteristics are not well understood. In particular, it is not clear how the prevalent traffic patterns may impact candidate mechanisms for managing traffic inside the data centers. In this paper, we conduct a measurement study of network-level traffic patterns inside data centers. Based on our empirical insights, we design a traffic generator for creating representative workloads for traffic between TOR switches in a data center. We use this generator to evaluate several traffic engineering techniques and data center network architectures, and analyze their shortcomings. Our findings highlight the need for fine-grained traffic engineering (TE) mechanisms. We design and implement such an approach using OpenFlow and show how it can significantly improve data center TE.

1 Introduction

Data centers are being heavily employed in enterprise, consumer and university settings to run a variety of applications and cloud-based services. These range from Internet-facing “sensitive” applications, such as, Web services, instant messaging, stock updates, financial applications and gaming, to computationally intensive applications, such as, indexing Web content, data analysis, archival and scientific computing.

The performance of these applications crucially depends on the functioning of the data center network infrastructure. For example, a congested data center network, where internal traffic is routinely subjected to losses and poor throughput, could lead to search queries take longer to complete, IM message getting delayed, gaming experience deteriorating, and POP mail services and Web transactions hanging. The dissatisfied end-users and subscribers could choose alternate providers, resulting in significant loss in revenues for the data center.

Central to the well-oiled functioning of a data center is a robust network traffic engineering (TE) mechanism. Unfortunately, anecdotal evidence suggests that data center TE is in a very primitive state. Today, most operators try to tweak wide-area traffic engineering and routing mechanisms (e.g., single path routing and ECMP) to manage their data centers. This is a natural thing to do because these mechanisms come bundled with current switches and they are well-understood. However, this naive approach is effective only if data center and wide area traffic share basic similarities.

The fact that “traditional” approaches are ineffective is reinforced by recent work that has shown that data center networks could experience congestion events lasting upto a few seconds each time, during which applications experience numerous failures [16]. Thus, there is a need for data-center oriented TE mechanisms. However, designing such mechanisms is difficult today because very little is known about the traffic patterns prevalent in data center networks and how these interact with topology and routing, and their impact on the time-scales at which TE decisions need to be made.

Our paper makes four important contributions that further the state of the art in data center TE:¹ a measurement study of data center traffic characteristics; a data center workload generator to evaluate TE proposals; a comparative study of commonly used and recent proposals for traffic engineering; and, a novel fine-grained TE scheme for data centers.

★ **Measurement study:**² We empirically study the traffic patterns on data center network links with an explicit focus on TE issues. Using SNMP data collected from 19 large Internet-facing data centers, we study properties of data center physical topologies, distributions of utilizations and loss rates on data center links and their dependence on the network location of a link, and variations in link utilization and loss rate over time. Our key findings are: (1) In general, the average link utilization is low, with links within the core of a data center being more heavily utilized than other links; (2) In contrast, the loss rates are the lowest (and almost non-existent) in the core; Loss rates at other links are non-trivial in magnitude. (3) Several data center links are unused at a given time, but the set of inactive links changes constantly. Thus, losses can be avoided, but careful traffic engineering is required.

We also study low-level packet traces collected from five switches in one of the data centers. From these traces, we find evidence of ON/OFF traffic patterns, where the ON-periods, OFF-periods and inter-arrival times follow heavy-tailed distributions. These traffic patterns can help explain the utilizations and loss rates we observed from SNMP data.

★ **Workload generator:**³ Our second contribution is a

¹This paper builds on our earlier workshop paper [9]. We have highlighted key enhancements and new contributions below.

²We present a more detailed explanation of key measurement results in [9], along with some new results on traffic/loss patterns.

³This is partly a new contribution. Some of the algorithms underlying the workload generator were discussed in [9].

data center workload generator that can provide representative models for traffic between TOR switches in a data centers and help realistically evaluate various data center TE mechanisms. At the heart of the workload generator is a unique “parameter space exploration” algorithm that derives the optimal set of traffic generation parameters for TOR switches in a simulated data center such that the resulting packet stream matches empirically measured data both on microscopic (e.g. packet inter-arrivals) and macroscopic (e.g. link utilizations and loss rates) levels.

★ **Comparative TE evaluation:**⁴ The third contribution of our paper is a study of the applicability of various traffic engineering mechanisms, and of recent proposals for data center interconnection and routing that promise very high bisection bandwidth [6], toward mitigating losses inside data centers. In conducting this study, we employ our traffic generator atop a small-scale virtual data center testbed. Through our study, we find that existing techniques are unable to avoid losses arising due to the bursty nature of data center traffic. This could arise due to one of several reasons: (1) not using multipath routing, or (2) not adapting to instantaneous load, or (3) not using a global view to make traffic engineering decisions.

★ **Fine-grained TE:**⁵ The final contribution of our paper is a new, fine-grained, load-sensitive traffic engineering approach that virtually eliminates losses due to bursty traffic inside data centers. Our solution relies on a central controller that computes optimal routes after actively probing the instantaneous levels of activity within the data center network. Although a centralized, fine-grained solution seems too daunting to implement, we argue that the recent OpenFlow framework [3] could be employed effectively in realizing this approach. Furthermore, OpenFlow can be enabled several “data center-grade” switches today via a simple firmware upgrade. Using our workload generator, we find that our OpenFlow-based approach can virtually eliminate losses in data center networks.

With the growing centrality of data centers and cloud computing in everyday Internet transactions, it is crucial to understand how to run the underlying network in these settings in the most efficient fashion. Our paper sheds light on this important problem domain, identifying the pitfalls of existing mechanisms and the fundamental properties of network traffic that lead to the pitfalls. Although we present a new traffic engineering approach that addresses the drawbacks of current techniques, we believe that it is just one solution in a space of possible solutions that share some key properties. We believe that our workload generator could prove instrumental in identifying other, potentially better candidates

⁴This is a new contribution.

⁵This is a new contribution.

in this solution space. Our workload generator and the virtual data center testbed are currently available for use upon request. We are planning a public release soon.

2 Empirical Study

In this section, we present an empirical study of data center networks and data center traffic based on traces collected from data centers owned by a large corporation. Our goal is to gain insights into what constraints and requirements drive network traffic engineering mechanisms in data centers. Specifically, we aim to shed light on the following issues.

- **Topology:** What is the physical structure and size of the data center networks? How different are they from each other?
- **Macroscopic performance properties:** What are the coarse-grained characteristics of traffic observed on data center links and switches? In particular, what are the link utilizations and loss rates on the data centers? How do they vary over time and across links?
- **Microscopic traffic properties:** What are the fine-grained properties of data center network traffic? Can the traffic properties be characterized using well understood distributions?

2.1 Data Sets

To answer the above questions, we collected two sets of measurement data. The first data set comprised of SNMP data extracted from 19 corporate and enterprise data centers hosting either intranet and extranet server farms or Internet server farms. SNMP MIBs were polled every 5 minutes and data was collected over a 10-day period. These data centers support a wide range of applications such as search, video streaming, instant messaging, map-reduce, and web applications. The 19 data centers’ are spread throughout the world.

The second data set is comprised of packet traces from packet sniffers on five switches in one of the data centers (located in the US). The sniffers ran WinDump, which is able to record packets at a granularity of 10ms.

In what follows, we first briefly examine the physical topologies and sizes of the 19 data centers. Then we study the SNMP statistics to characterize link utilization and packet loss in a data center. Finally, we use the packet traces to characterize the temporal patterns of data center traffic.

2.2 Topology

Table 1 summarizes topological information in the 19 data centers. Our review of the physical topologies showed that all these data centers follow a tiered architecture, with network devices organized into *two or three* layers. Most commercial data centers are known to follow such tiered designs. The innermost and the outermost tiers are called the *core* and the *edge* layers (devices

Data-Center Name	Fraction Core Devices	Frac Aggr Devices	Frac TOR Devices	Total # Devices
DC1	0.000	0.000	1.000	5
DC2	0.667	0.000	0.333	5
DC3	0.500	0.000	0.500	6
DC4	0.500	0.000	0.500	7
DC5	0.500	0.000	0.500	7
DC6	0.222	0.000	0.778	9
DC7	0.200	0.000	0.800	13
DC8	0.200	0.000	0.800	13
DC9	0.000	0.077	0.923	26
DC10	0.000	0.043	0.957	47
DC11	0.038	0.026	0.936	78
DC12	0.024	0.072	0.904	83
DC13	0.010	0.168	0.822	210
DC14	0.031	0.018	0.951	230
DC15	0.013	0.013	0.973	302
DC16	0.005	0.089	0.906	427
DC17	0.016	0.073	0.910	562
DC18	0.007	0.075	0.918	612
DC19	0.005	0.026	0.969	763

Table 1: Statistics for the 19 data centers studied. For each data center, we present the fraction of devices in each layer.

	Core Links	Aggr Links	Edge Links
% Used	58.88%	73.7%	57.52%

Table 2: Statistics for the interfaces polled for SNMP data. The information is broken down according to the layer that the interfaces belong to. For each layer, we present the percent of interfaces that were utilized and the percent of interfaces that experienced losses.

in the edge layer are known as top-of-rack or TOR). Between the two layers, there may be an *aggregation* layer when the number of devices is large.

The first eight data centers have two layers due to their limited size (5–13 switches). The remaining eleven data centers have all the three layers. Of these, 4 data centers have few 10s of switches, while the remaining 7 have roughly an order magnitude higher number of switches on average.

2.3 Macroscopic View

Next, we examine the utilization and packet losses observed for data center links and the implications for traffic engineering. We organize our study into a set of questions that often arise in the context of TE decisions.

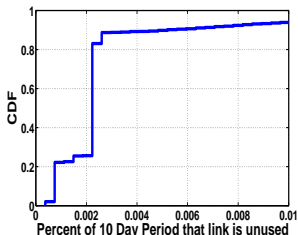


Figure 1: A CDF of the percent of times a link is unused during the 10 day interval.

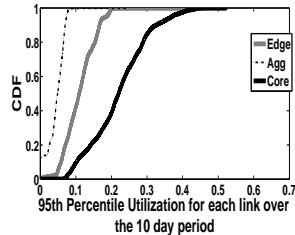


Figure 2: A CDF of the 95th link utilization at the various layers in the Data Centers studied.

Q1. Are all links heavily utilized? Table 2 provides a breakdown usage of the links across all the data centers. Roughly 60% of the core links and the edge links are actively being used during any given 5 minute interval. More surprising is the fact that 40% of the links are *not used at all* in these intervals. This observation aligns with recent observations that a significant amount of traffic in data centers is confined to within servers in a rack [16].

Figure 2 shows the CDF of the 95th percentile utilization of those used links, computed over all the 5 minute intervals where the link was utilized. On the whole, we find link utilizations to be *low* – the 95th percentile utilization is only 0-40% across all links, and under 10% and 20% for links in the aggregation and edge layers respectively.

We also note that: (1) The utilization is significantly higher in the core (median of 25%) than in the aggregation (median of 5%) and edge (median of 10%) layers. This is expected since a small number of core links multiplex traffic from a large collection of edge links (Table 2). (2) Link utilization is 4-5X lower in the aggregation layer than in the core layer, with the 95th percentile utilization not exceeding 10% for any link. Again this is expected because in our data center topologies there are nearly four times as many aggregation links as core links. Moreover, link capacities of aggregation and core links are the same in many cases. (3) Edge links have slightly higher utilization than aggregate links because they are at least 10X lower in their capacities (1Gps at the edge vs 10Gbps in the aggregation).

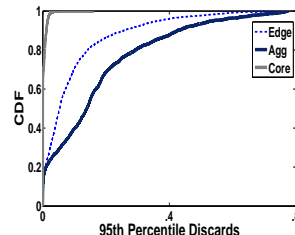


Figure 3: CDF of 95th percentile “scaled” loss rates of links at the various layers in all data center.

Q2. Does this mean there are no losses? The generally low utilization across links that we observed earlier may lead one to conclude that data centers observe no

network loss. We examine this next.

Figure 3 illustrates the CDF of 95th percentile “scaled” packet loss rates on the core, edge, and aggregation links. To compute actual link loss rates, we need the number of bytes discarded and the total number of input bytes. In contrast, SNMP counters only provide the number of packets discarded and the input bytes. Thus, we compute a “scaled” loss rate by converting discarded packets into bytes after scaling by an average packet size of 850B. We derived the scaling factor of 850B from the observations made about the size of packets observed in DC 10. Although, the real loss rates are likely to be different, comparison of loss rate distributions across the three layers is likely to be the same for real and scaled loss rates. We note immediately from Figure 3 that all layers experience some level of losses - no layer is completely loss free.

A surprising observation from Figure 3 is that, in spite of the higher utilization in the core, core links observe the least loss rates – in fact, most of the links are almost loss-free. In contrast, links near the edges of the data center observe the greatest degree of losses. A plausible explanation for this observation is that traffic traversing the data center between TOR switches is bursty in nature – the average utilization of the bursty traffic is low, but the instantaneous rates of bursts at the TORs could be high enough to lead to losses. The bursts gets smoothed at the core due to statistical multiplexing. We explore this observation in greater detail in subsequent sections.

Q3. Are the losses unavoidable? An important observation from Figure 3 is that a small fraction of the links experience much great amount of losses than the rest of the links. This observation motivates our quest for better engineering approaches as it indicates that it may be possible to route traffic on alternate paths to avoid most of the losses.

Q4. Can existing traffic engineering approaches help? Given the large number of unused links (40% are never used) and that losses are more prevalent on some links than others at a given time, an ideal traffic engineering scheme would split traffic across the over-utilized and the under-utilized links. Many existing TE schemes can perform this type of load balancing (e.g. ECMP). Next, we briefly take an empirical view into whether these approaches are applicable.

Specifically, we examine the link idleness in one of the data centers, DC 17, and observe that although a large number of links are unused, the exact set of links that are unused constantly changes. In Figure 1 we present a CDF of the fraction of the 10 day period that each unused link is found to be idle. From Figure 1, we observe that 80% of the unused links are idle for 0.002% of the 10 days or 30 minutes. Thus, although significant numbers of links are idle, the set of links that are idle in any given

5 minute interval is constantly changing.

As we will show next, the traffic in the data center can be quite bursty and traffic patterns could be very different across switches, due to which link usage is difficult to predict and existing traffic engineering schemes become less applicable.

2.4 Microscopic View

We now examine finer grained traffic characteristics.

Q5. Are there patterns in link usage? We reuse the SNMP data to examine patterns in the traffic sent or received by individual switches. If the patterns are uniform across switches (e.g. if there are diurnal patterns), then traffic engineering may be easy; On the other had, if each switch has a different send/receive pattern, then traffic engineering may not be as easy.

In general, we found that switches could differ radically in these respects, making traffic engineering rather challenging. In particular, there could be sharp differences in the amount of traffic they send vs. receive, and there may be differences in the trends of traffic volumes and losses over time. To exemplify this, in Figure 4 we show a time series of two aggregation layer switches in a 3-tier data center that were selected at random. This was derived from the SNMP data measured over a single day. We note that the number of bytes sent out from and received at the switches in (a) and (d) are almost identical. In contrast, the switch in (c) sends almost 25% more bytes than it receives and that in (b) sends nearly twice as it receives. All four switches also differ significantly in how traffic volumes vary over time.

Figures (e)–(h) show the drops (in terms of the total number of bytes drop over a 5 minute interval) observed at the corresponding switches. In the case of the fourth switch (d & h), we note a strong correlation between extent of drops and the traffic volumes. However, in all other cases, the correlation is weaker (e.g. the second switch) or even non-existent (e.g. the third switch).

We now seek to understand finer-grained characteristics of traffic. For this study, we use low level packet traces from five TOR switches in one of the data centers, DC 10, which is a 2-tier corporate data center containing intranet server farms and hosting several line-of-business applications (e.g. web services). Our observations are limited by the vantage points we have, i.e., the five TOR switches.

Q6. What are the key fine-grained properties of data center traffic? We first try to identify key patterns in the packet transmissions at the switches.

Figure 5 shows the time-series of the number of packets transmitted during several short time intervals at one of the switches. It is immediately clear that the packet arrivals exhibit an ON/OFF pattern. We observed similar ON/OFF bursty traffic patterns at the remaining four

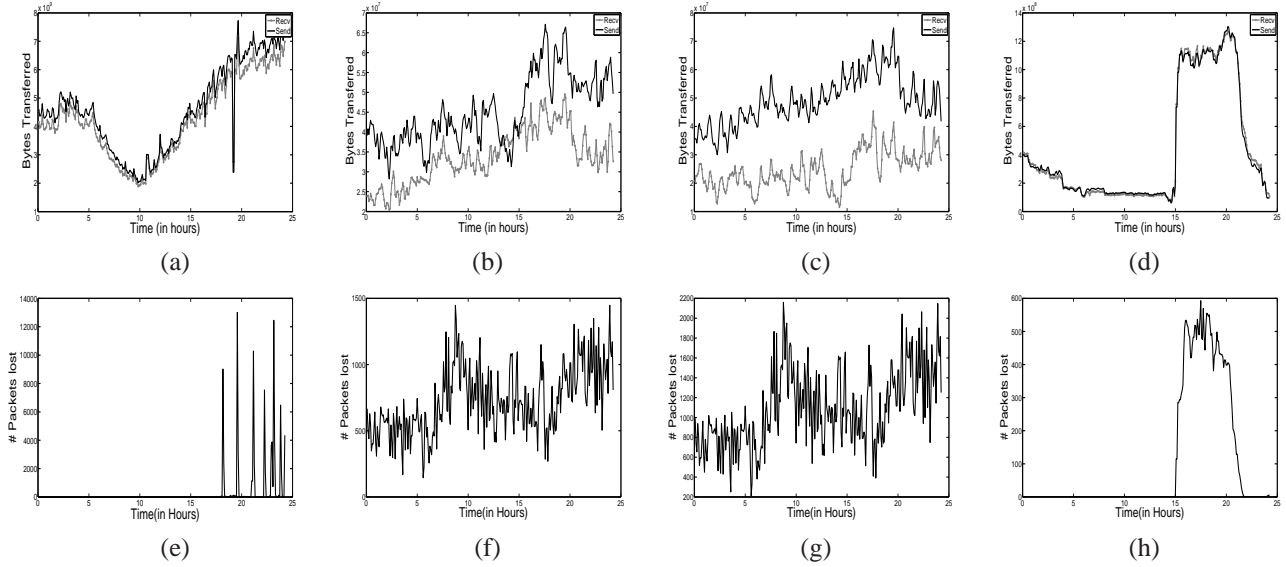


Figure 4: Time series of traffic volumes and losses from aggregation and TOR switches

switches as well.

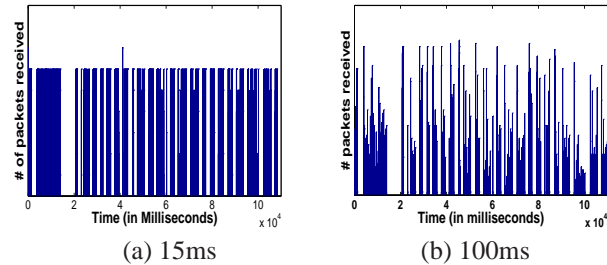


Figure 5: ON/OFF characteristics: Time series of Data Center traffic (number of packets per time) binned by two different time scales.

Next, we try to characterize the traffic using well-known statistical distributions. Such distributions have formed the bases for workload models in other context such as HTTP traffic (e.g. [8]) and we hope to leverage them in a similar way for our workload generator in § 3. To do this, we first use a packet inter-arrival time threshold to identify the ON/OFF periods in the traces. Let $arrival_{95}$ be the 95th percentile value in the inter-arrival time distribution at a particular switch. We define a $period_{on}$ as a longest continual period during which all the packet inter-arrival times are smaller than $arrival_{95}$. Accordingly, a $period_{off}$ is a period between two ON periods. To characterize this ON/OFF traffic pattern, we focus on three aspects: (i) the durations of the ON periods; (ii) the durations of the OFF periods; and (iii) the packet inter-arrival times within ON periods.

Figure 6(a) illustrates the distribution of inter-arrival times within ON periods at one of the switches. We

bin the inter-arrival times according to the clock granularity of $10\mu s$. Note that the distribution has a positive skew and a long tail. We attempted to fit several heavy-tailed distributions and found that the lognormal curve produces the best fit with the least mean error. Figure 6(b) shows the distribution of the durations of ON periods. Similar to the inter-arrival time distribution, this ON period distribution also exhibits a positive skew and fits well with a lognormal curve. The same observation can be applied to the OFF period distribution as well, as shown in Figure 6(c).

In summary, our measurement study shows that data center traffic is bursty in nature, and the unpredictability of bursty traffic means that traffic engineering in data centers could be challenging. In §4, we conduct a large scale study of a variety of traffic engineering mechanisms and show that they are unable to accommodate the aforementioned data patterns effectively. To aid in this study, we need a workload generator that reproduces data center traffic at fairly fine time-scales and with sufficient fidelity. This is the subject of the next section.

3 Data Center Workload Generator

Thorough evaluation of research ideas for data center networks is hampered by the lack of good workload models. An important contribution of this paper is a preliminary data center workload generator, constructed based on the measurement insights presented in the previous section. Our workload generator focuses on the traffic generation process driving the bytes leaving Top-Of-Rack (TOR) switches in data centers. It does not model application-level traffic characteristics nor does it model

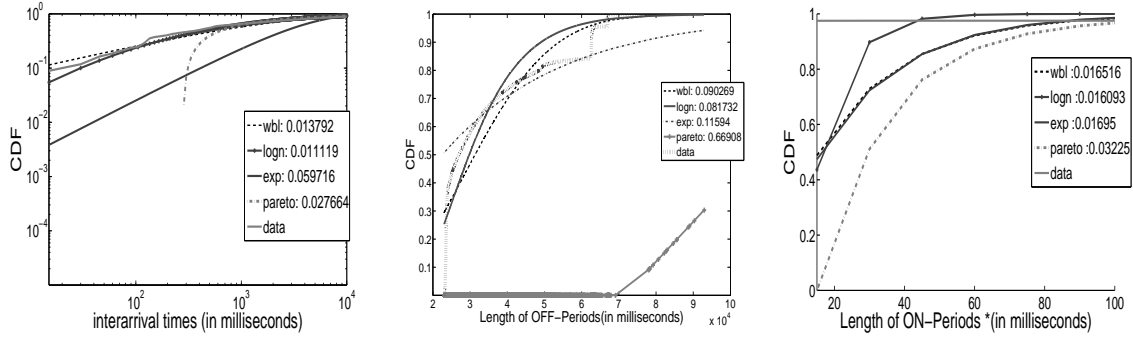


Figure 6: CDF of the distribution of the arrival times of packets at three of the switches in DC 10. The figure contains best fit curve for log-normal, Weibull, Pareto, and Exponential distributions as well as the least mean errors for each.

intra-rack traffic. Thus, our traffic generator can help researchers study the interaction between data center traffic and the underlying network interconnect and routing, but it is not useful to study the impact of application placement algorithms, for instance.

The central goal driving the design of our data center workload generator is that researchers using the workload generator should be confident that the traffic patterns are very close to one of the 19 data centers we have studied. The generated workloads should match both the microscopic and the macroscopic patterns.

To match the microscopic behavior, our workload generator uses the observation that the arrival processes at each TOR switch in a data center can be modeled by 3 lognormal distributions. To match the macroscopic behavior, we create appropriate packet generators whose lognormal parameters are chosen such that the loss rate and volume distributions match a given measured data center from the 19 data centers above. Our workload generator also takes as input the TOR-to-TOR traffic matrix to determine the destination for the packets generated at a TOR switch.

3.1 Workload Parameter Discovery

A crucial step in our workload generator is deriving the appropriate *realistic* distribution parameters for the packet generation processes at various TOR switches to match a target data center. While most TOR traffic is bursty, switches may differ in the relative levels of burstiness. Thus, an obvious approach of configuring packet generators directly with the exact parameters for the distributions identified in Figure 6 is likely to be incorrect. Instead, finding the right parameters requires searching through a multi-dimensional space for the best-fitting point, where each dimension in the space represents possible values of the parameters for one of the three distributions. Since each distribution is described by 2 parameters (the mean and the standard deviation), this results

in a 6-dimensional search space.

Exploring all points in the search space will result in accurate results but is clearly infeasible. In what follows, we describe a new search space exploration algorithm that is tractable and reasonably accurate. Our algorithm is iterative in nature and is similar to simulated annealing — in each iteration the algorithm explores the search space by examining the neighbors of a candidate point and moving in the direction of the neighbor with the highest “score”, where “score” measures how well the parameters corresponding to the point in question describe the coarse-grained distributions.

There are four challenges in developing such an algorithm: (1) developing an accurate scoring function for each point, (2) determining a set of terminating conditions (3) defining a heuristic to avoid getting stuck in local maxima and selecting an appropriate starting point and (4) defining the neighbors of a point and selecting the next move. In what follows, we describe how these challenges can be addressed.

We start by noting that our framework takes as input the distribution of SNMP-derived volumes ($volume_{SNMP}$), and loss rates ($lossrate_{SNMP}$) for a given link at the edge of the data center. To create a distribution of volume and loss, we aggregate several hours worth of data and assume that the target distributions remain relatively constant during this period. The approach returns as output, the parameters for the 3 distributions (on_{times} , off_{times} , $arrival_{times}$) that provide fine-grained descriptions for the traffic on the edge link.

Scoring function. An effective scoring function, for deciding the utility of a point and the appropriate direction for the search to proceed in, is not obvious. To score the parameters at a point, we utilize two techniques: first, we use a heuristic algorithm to approximate the distributions of loss and volume that the parameters corresponding to the point in question generate; we refer to these

```

DERIVEONOFFTRAFFICPARAMS( $\mu_{on}, \sigma_{on}, \mu_{off}, \sigma_{off}, \mu_{arrival}, \sigma_{arrival}$ )
  // Calculate the mean on and OFF period lengths
1   $mean_{on} \leftarrow \exp(\mu_{on}) + \sigma_{on}$ 
2   $mean_{off} \leftarrow \exp(\mu_{off}) + \sigma_{off}$ 
  // Determine the total on-time in a 5 minute interval
3   $total_{on} = 300 * (mean_{on} / (mean_{off} + mean_{on}))$ 
  // Calculate the average number of ON periods
4   $NumOnPeriods = total_{on} / mean_{on}$ .
  // Calculate the maximum number of bytes
  // that can be sent during the ON period
5   $link_{capacity} = links_{speed} * mean_{on} / 8$ .
  // Determine how much bytes can be absorbed by buffering
  // during the OFF period
6   $buf_{capacity} = \min(bitsofbuffering, links_{speed} * mean_{off}) / 8$ 
  // Iterate over ON period to calculate net volume and loss rate
  // observed over the 5 minute interval
7  for  $i = 0$  to  $NumOnPeriods$ 
    a.  $a_i \in A\{interarrival\ time\ distribution\}$ 
    b.  $vol_{on} = (mean_{on} / a_i) * pktSize$ 
    c.  $vol_{total} += \min(vol_{on}, link_{capacity} + buf_{capacity})$ 
    d.  $loss_{total} += \max(vol_{on} - link_{capacity} - buf_{capacity}, 0)$ 

```

Figure 7: Pseudocode for TOR parameter discovery.

as $volume_{generated}$ and $lossrate_{generated}$. Second, we employ a statistical test to score the parameters based on the similarity of the generated distributions to the input distributions $volume_{SNMP}$ and $lossrate_{SNMP}$.

We use a simple heuristic approach to obtain the loss rate and volume distributions $volume_{generated}$ and $lossrate_{generated}$ generated by the traffic parameters $(\mu_{on}, \sigma_{on}, \mu_{off}, \sigma_{off}, \mu_{arrival}, \sigma_{arrival})$ corresponding to a given point in the search space of parameters for a TOR switch. Our heuristic relies on the subroutine shown in Figure 7 to derive a single sample for the $volume_{generated}$ and $lossrate_{generated}$ distributions.

The subroutine determines the traffic volume and loss during a 5-minute interval $(vol_{tot}, loss_{tot})$ as the sum of loss and volume in each individual ON period in the interval. Line 1 calculates the average length of an ON period and an OFF period. The volume in an ON period is the sum of the bytes in the packets received, where packets are spaced based on the inter-arrival time distribution (calculated in Line 7.c). The loss in that ON period is the number of bytes received minus the bytes successfully transmitted during the on period and the number of bytes buffered. Throughout the calculation, we assume an average packet size of 1KB. In Line 7.b, the inter arrival time distribution is used in the generation of a_i – each time a new value, a_i , is drawn from the distribution. The distribution A in Line 7.b is a lognormal distribution with the following parameters, $(\mu_{arrival}, \sigma_{arrival})$.

We run the above subroutine several (100) times to obtain multiple samples for vol_{total} and $loss_{total}$. From these samples, we derive the distributions $volume_{generated}$ and $lossrate_{generated}$.

Next, we use the Wilcoxon similarity test [23] to compare the distribution of computed volumes $volume_{generated}$ (loss rates) against the distribution of empirical volumes $volume_{SNMP}$ (loss rates). The

Wilcoxon test is a non-parametric test that checks whether two different distributions are equally distributed around a mean – the test returns the probability that this check holds. The Wilcoxon test is used because unlike the popular t-test or chi-test, the Wilcoxon does not make any assumptions about the distribution of the underlying input data-sets. Using a distribution free statistical test allows for the underlying distribution for any of the 3 parameters to change.

We compute the score of a point as the minimum of the two Wilcoxon scores – the confidence measure for similarity – for volume distribution and loss distribution. We use the minimum for comparison instead of other functions such as average, because this forces the search algorithm to favor points with high scores for both distributions.

Neighbor selection. Each point in our search space can be described as a coordinate in the 6 dimension space. A neighbor for such a point is the closest one in the coordinate space, modulo some quantization of the real values in each dimension. For each point we evaluate 12 neighbors each of which is adjacent along one of the 6 dimensions. Once all 12 neighbors are evaluated, our algorithm chooses the point with the best accuracy, or a random neighbor if all neighbors are identical in terms of accuracy.

Termination condition. The search algorithm terminates for one of two reasons: a point with a good score ($> 90\%$) is discovered or the search fails to discover a point with a suitable score ($> 10\%$) even after a large number of iterations (1000). When a search terminates, the top score discovered during the search as well as the parameters for the corresponding point in the space are both returned.

Avoiding local optima. To avoid getting stuck in a local optimum, we run our search a predefined number of times and vary the starting points for each search. We then compare the returned values of each search and chooses the best. Two key challenges in this approach are determining the number of searches to perform and carefully determining the start point for each search to avoid repetitive searches through the same sub-space of parameters.

To solve both challenges, we partition the search space into N_{sub} regions and initiate an independent search at a randomly chosen point in each sub-region. Our algorithm performs a parallel search through each of the N_{sub} regions and selects the most accurate of the N_{sub} results to return. The choice of N_{sub} depends on the trade-off between the time consumed by the search space exploration algorithm (which deteriorates with N_{sub}) and the accuracy of the outcome (which is good for high N_{sub}). In our evaluation, we find that $N_{sub} = 64$ offers a good

trade-off between speed and accuracy.

3.2 Implementation

We implement a data center workload generator based on the parameters derived by the algorithm above. The generator takes as input a data center topology, TOR-to-TOR traffic matrix, and parameters for the distributions of traffic processes for different switches. The packet generation module for a switch is written in C++. The generated packets are randomly distributed to destination TOR based on the provided traffic matrix.

In Section 4, we describe how we employ the traffic generator within a virtualized data center testbed to compare various traffic engineering approaches. We outlined key issues we faced therein that warranted changes to our workload generator.

3.3 Coarse Grained Validation

To verify that the above approach discovers parameters for an arrival process that approximate traffic at the edge switches reasonably well, we implement the arrival process in NS2 [1] and validate the results against data from a randomly chosen data center, DC #17, which had 562 devices in all. Verification of the framework consists of three steps: (a) using the framework to generate parameters that match the SNMP data for an edge switch (b) running the NS2 simulator with the parameters as input, and (c) performing a statistical test to compare the original SNMP data to equivalent data obtain from the NS-2 simulation.

We model an edge link in NS2 to have 5 MB of input buffering, use a FIFO queuing discipline, and have a propagation delay of 5 microseconds. We evaluate the quality of a match by running the Wilcoxon test to compare data observed for the switch with the data generated by the simulation. To get the distributions of volume and loss rate from the simulator, we run the simulator several times to get a statistically significant number of data-points for either distribution. We consider a match successful if the Wilcoxon test passes with over 90% confidence.

In Figure 8, we provide a CDF of the confidence returned by the Wilcoxon test for validations run on over 200 edge links in DC #17. We note that our parameter discovery approach is reasonably accurate: it can find the appropriate parameters for arrival processes for over 90% of the devices with at least an 85% confidence according to the Wilcoxon test.

4 Comparative Study of TE Approaches

In this section, we evaluate the effectiveness of various traffic engineering techniques and/or network architectures at accommodating the traffic patterns observed in Section 2. We conduct our study along two dimensions:

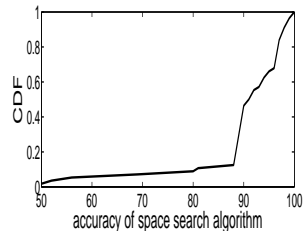


Figure 8: CDF of the validation score for each edge link simulated. Each score is a minimum of the Wilcoxon test on the volume distributions and the Wilcoxon test on the loss distributions for the switch

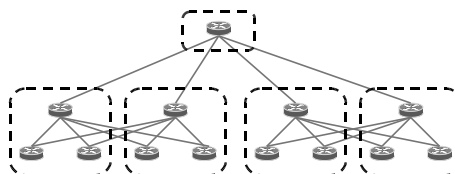


Figure 9: Tree topology with redundant links between the TOR and the aggregation layer. The edge links are 20Mbps, while the core links are 25Mbps, representing an over-subscription of 3.2.

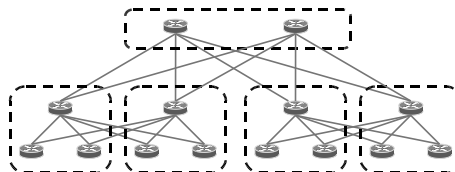


Figure 10: Tree topology with increased core capacity and path diversity. The edge links are 20Mbps, while the core links are 25Mbps, representing an over-subscription of 3.2.

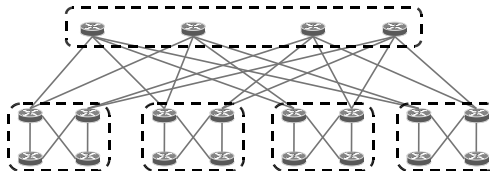


Figure 11: Fat-Tree topology ($k=4$). Each TOR has 2 end hosts [6]. Each link's capacity is 20Mbps.

- **Topology:** We examine if and to what extent a richer network topology can help alleviate prevalent congestion in a data center network. We study a simple two tier hierarchical network (Fig 9) which is similar to that used in the 3-tier data centers we studied in Section 2, the same network with more core nodes for improved bisection bandwidth (Fig 10) and the Fat-Tree topology proposed in [6] (Fig 11) that has much higher bisection bandwidth and path diversity. Note that all topologies have the same number of edge switches and the same capacities on edge links. The capacities on the other links are also shown.

- **Routing/TE:** We also study if and to what extent simple multi-path routing (i.e., ECMP) and load-sensitive multi-path routing (i.e., the flow classification heuristic in [6]) can help alleviate congestion when used in conjunction with the above topological structures. We also examine the constraints imposed by using naive, static single-path routing. We have implemented all three classes of routing approaches in our testbed described below.

We experiment using a virtual testbed of 5 physical machines. These machines have Intel Core 2 Quad CPU running at 2.66GHz, with 3072KB cache and 8GB of RAM, running Ubuntu GNU/Linux 2.6. These machines were interconnected by a 48 port Cisco Catalyst 3750G switch with 1GigE links. Using the Virtual Box package [5], we run virtual experimental topologies reflecting the ones described atop these 5 physical machines. For each topology, the traffic generators and the switches are each run within separate virtual machines. The traffic generators are user space programs. The switches are kernel modules running the OpenFlow [3] reference implementation. We use OpenFlow because it allows us to emulate a large class of routing and traffic engineering mechanisms in a relatively simple fashion without having to implement their distributed versions within our switches. The virtual links between the traffic generators and the switches are implemented in VDE [4]. To gather measurement and evaluation data from these virtual links, we modified the VDE component to enable logging and accurate rate limiting. The traffic generators are bandwidth-limited to 20Mbits/s to ensure that the physical host CPU and memory are not overwhelmed.

In adapting the workload generator from Section 3 to our virtualized environment, we encountered a few challenges. The first involved the kernel timer resolution: we found the family of sleep functions provided by the kernel to be imprecise. This imprecision arose due to interaction between the timers and clocks of the virtual and guest operating systems. To overcome this constraint and to be able to generate packets that closely match the said distributions, each traffic generator simply performs a “busy wait” until the target time, at which point a packet gets sent. This is accurate but imposes a slightly higher

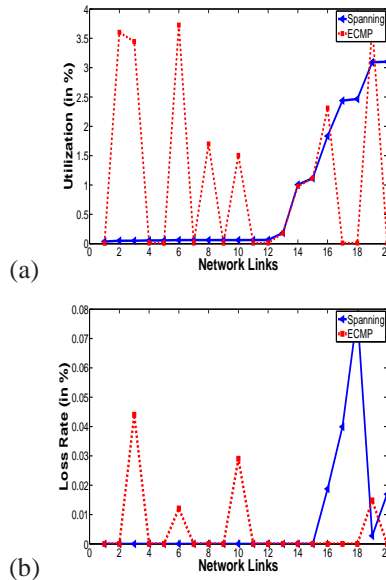


Figure 12: Utilization (a) and loss rates (b) of links in the constrained topology.

CPU load.

The second involved the speed of packet generation – we observed that the virtualized end hosts couldn’t generate packets faster than 30Mbits due to limitations in the virtual box implementation. To resolve this issue, we scaling down the arrival process by a constant factor c , where c is the link rate of the link being simulated divided by the link rate of the testbed link.

In our evaluation, we use two TOR-to-TOR matrices: (1) uniformly random and (2) gravity-model based. We only present results for the uniform traffic matrix. Observations for the uniformly random traffic matrix are qualitatively similar.

When comparing different approaches, we ensure that the exact same traffic patterns are used by fixing the random seed used in our traffic generators. This enables a direct and accurate comparison of the approaches.

4.1 Constrained Topology

We first examine the topology in Figure 9 where there is a single core node and limited path diversity (atmost two equal length paths between any pair of nodes, with both the paths passing through the single core node). We first examine the performance of single path static routing, and then study if ECMP (where traffic is split across two equal cost paths) can perform better.

In Figure 12(a), we present the utilization and loss rates of the various links connecting both tiers under both ECMP and single-path routing. As expected, we see that ECMP spreads traffic across links in the network: ECMP leaves 8 links unused, while single path routing

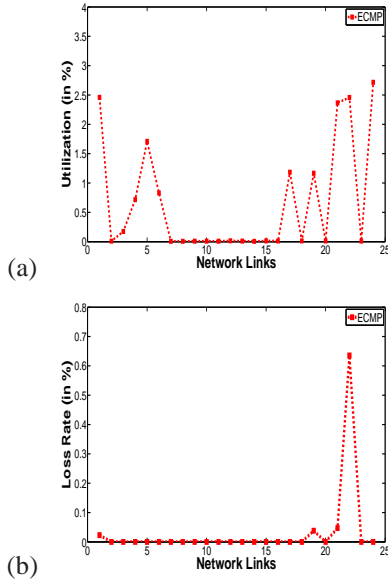


Figure 13: Utilization (a) and loss rates (b) of links in the topology with higher bisection bandwidth.

leaves 11 links unused. In Figure 12(b), we show the per-link losses for ECMP and single-path routing, where the links are ordered according to their utilization under single-path routing. We find that ECMP, on account of spreading traffic more evenly, does a better job of reducing loss rates: both techniques have 4 links with losses but the losses in ECMP are lower. In general, we find that ECMP is a better fit for data center networks, however, it is not perfect as it still results in a substantial amount of loss. ECMP is unable to reduce losses significantly as it balances traffic across multiple paths leading to even utilization, but it does not take into account the instantaneous load on each path which is central to controlling losses. Consider two source-destination pairs whose traffic is highly bursty, but the average load due to either pair is low. Nothing stops ECMP from assigning the two sets of flows to a common set of network links. Since ECMP does not re-assign based on observed load, it cannot help overcome losses due to temporary over-subscription on path, which may happen when both sets of flows experience bursty transmission at similar times.

These experiments point to the fact that shows that, traffic engineering techniques must exploit multiple-path routing in existing data center topologies in order to obtain better performance. But, simply striping traffic across multiple paths is insufficient.

4.2 Topology With Higher Bisection Bandwidth

A key issue with the above topology is its poor bisection bandwidth. In this section, we examine to what extent increasing the bisection bandwidth, by doubling the

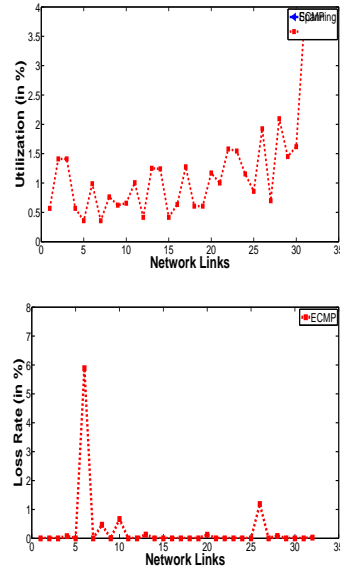


Figure 14: Utilization(top) and loss rates (bottom) of links in the topology with higher bisection bandwidth in the Fat-tree topology

capacity in the core can help. As the previous section showed, using multipath routing is crucial, in this section, we simply study the impact of using ECMP on the richer topology in Figure 10. The higher path diversity in Figure 10 allows us to use ECMP with up to four candidate paths per source destination pair (as opposed to just two in the previous section).

The results are shown in Figure 13. Comparing Figure 12(a) and Figure 13(a), we see that the average per-link utilization is lower in the topology with greater bisection bandwidth and path diversity. Similarly, from Figure 12(b) and Figure 13(b), we note that the average loss rate is lower for the richer topology. However, the maximum loss rate has increased significantly (0.08% to 0.8%).

This suggested that the fundamental drawback of ECMP, namely its load-insensitive nature, cannot be fully masked even by a higher degree of path diversity. Bursty traffic accentuates this drawback of ECMP.

4.3 Fat-Tree

Finally, we examine a recent proposal, the Fat-Tree interconnect, that supports extremely high bisection bandwidth [6, 19]. This topology is shown in Figure 11. In theory, routes can be constructed over this topology to support any traffic demand matrix in a completely loss-free manner. However, this is true only as long as: (1) the traffic demands do not overflow link capacities of servers, (2) routes computed are optimal for the current traffic matrix. In practice, condition #1 would likely

always hold, but condition #2 is harder to ensure as it requires constant recomputation of routes matching the current traffic demand matrix. In [6], the authors leverage a fixed number of shortest path routes between each source-destination pair, and use a *local* heuristic to balance load across the shortest paths in order to meet condition #2 to an approximate extent. In particular, at regular intervals (say, every second), each switch in the lower level of the topology measures the utilization of its output ports (averaged over some period of time, say 10s) and reassigns a minimal number of flows if the utilizations of the output ports are mis-matched. Ideally, the fat-tree topology should be able to ensure *zero losses* on all links. We now study how the local heuristic may prevent this goal from being achieved using our workload generator.

The results are shown in Figure 14. Comparing Figure 14(a) and Figure 13(a), we see that the average per-link utilization is *higher* in the Fat-tree topology than in the topology considered in the previous section (i.e. Figure 10). However, this is because all links in the Fat-tree topology are of the same low capacity, while links connecting directly to the core in Figure 10 have higher capacity than other links.

In Figure 14(b), we show the loss rates for the Fat-Tree topology. Surprisingly, losses are not any better than any of the previous settings! We argue that this occurs because of the local heuristic employed for balancing load on network paths. When each switch makes a local decision to rebalance traffic, it does not take into account how the rebalancing interacts with traffic further downstream. The downside of taking a purely local view of re-balancing is exacerbated in the presence of bursty traffic, especially when multiple bursty sources cause sudden over-subscription on remote links on paths to their destinations.

To summarize, using our workload generator on a virtual testbed, we have found that existing techniques fail to control losses in the presence of bursty traffic in data centers for one of several reasons: (1) Not using multi-path routing (2) Not taking instantaneous load into account and (3) Not making decisions on the basis of a global view of the network.

5 Fine-grained Traffic Engineering

Our study in the previous section shows that an ideal traffic engineering solution should: (1) work with a global view of data center network traffic and accommodate network-wide objectives for managing traffic, (2) react very quickly and accurately to changes in network traffic patterns and (3) should rely on multi-path routing.

In this section, we propose a centralized fine-grained traffic engineering heuristic that satisfies the above properties. Our heuristic relies on a logically central con-

troller and monitoring agents on switches. The monitoring agents update the controller with instantaneous traffic demands and the traffic matrix on a periodic basis. The central controller computes routes that optimize a network-wide objective, e.g. minimize the average loss across all links, and installs the routes. We first describe the LP-based heuristic that the controller can employ to derive the routes (§5.1). We then describe how our traffic engineering approach can be implemented in OpenFlow (§5.2). Finally we conclude with an evaluation of our OpenFlow-based implementation (§5.3).

5.1 Linear Program Formulation

We formulate a linear program for deriving network-wide routes based on the instantaneous traffic demands and the traffic matrix. We note that our LP formulation is simply a heuristic to solve the problem of allocating instantaneous traffic demands on network paths so that the network-wide loss rates (or some function of them) are minimized. The LP is certainly not the most efficient approach to solve this problem — we believe that it is possible to develop much simpler and faster heuristics for the above optimization problem. We leave this issue for future work.

To formulate our linear program, we represent the data center by a graph $G = (V, E)$. Each vertex $u \in V$ is either an application end point, or an <IP address, port> pair, or a switch. Each edge is a physical link between a server and a switch or between two switches.

Let $f_{u,v,e}$ denote the fraction of traffic between the pair (u, v) on edge e after considering loss at the buffer on e . For each end-point pair (u, v) , and edge e , let $loss_{u,v,e}$ denote the fraction of traffic between u and v that is lost at the output port for edge e . For both f and $loss$, the variables are defined for only those edges which lie on one of the top k shortest paths between u and v . Let $T_{u,v}$ be the net traffic demand (in bytes) between u and v . Let $w^+(n)$ indicate the outgoing edges for node n , and $w^-(n)$ indicate the incoming edges.

We set up flow conservation constraints. For each end-point pair u, v , for a given source u , we have:

$$\sum_{e \in w^+(u) \text{ and } SP(u,v)} (f_{u,v,e}) = 1$$

For intermediate node n :

$$\sum_{e \in w^-(n) \text{ and } SP(u,v)} f_{u,v,e} =$$

$$\sum_{e \in w^+(n) \text{ and } SP(u,v)} (f_{u,v,e} + loss_{u,v,e})$$

For destination v :

$$\sum_{e \in w^-(v) \text{ and } SP(u,v)} f_{u,v,e} = 1 - \sum_{e \in SP(u,v)} loss_{u,v,e}$$

For each edge e ,

$$\sum_{e \in SP(u,v)} f_{u,v,e} T_{u,v} \leq Cap_e$$

Objective: The objective of the formulation then is to minimize some function of the loss rates observed on various links. In particular, we consider minimizing the aggregate loss on all links:

$$\text{Minimize: } \sum_{(u,v)} \sum_{e \in SP(u,v)} loss_{u,v,e}.$$

Other weighted functions of per-link loss can also be considered.

Integer solution: The output of the above LP is the set of variables $f_{u,v,e}$ which indicate the fraction of traffic between u and v that is routed on edge e . In our formulation, we force these variables to be 0 or 1, so that all traffic between u and v is routed along the same physical path. This ensures that all bytes in an application flow arrive in order (of course, route changes could introduce reordering).

Number of variables: The number of variables and constraints in the LP determines its run time. As stated earlier, for each pair u, v , we consider f variables for only those edges e that belong to shortest paths between u and v . In the data centers we studied, the path length is at most 6, and if we choose at most the top-4 shortest paths, it results in 24 variables per pair. For a data center with A application end-points, the total number of variables is $A^2 \times 24$. A typical value for A in a large data center is 10,000. Assuming there is skew in the instantaneous traffic demand, considering the top 1000 or so pairs would suffice. In this case, the number of variables is 1000×24 .

5.2 Implementation Using OpenFlow

We implemented the above fine-grained traffic engineering approach in OpenFlow. In OpenFlow, an external, logically-central “NOX” controller [2] written in software can add and delete forwarding entries at fine time-scales. We can achieve low-level programmatic control over routing and forwarding by defining the appropriate NOX policies that determine how flows are treated by the network, using global network-wide information.

In a generic implementation of OpenFlow, when an edge switch does not have forwarding entries for a flow, e.g., when the first packet arrives for a flow never seen before, it contacts the NOX controller, which then computes and installs entries for the flow at edge switches and other appropriate switches. Each entry maps a 10 tuple for the flow (this includes the input port, source and destination IP address and ports, source and destination Ethernet addresses, protocol, VLAN ID, Ethertype) to a next hop; this is in contrast to destination-based forwarding tables in typical switches. Once routes are installed, each switch also tracks per-flow statistics using a simple

counter. Most major switch vendors are starting to support, or already support, OpenFlow. On existing switches it can be enabled using a simple firmware upgrade.

In our data center traffic engineering framework, we implement the central controller in NOX. As stated earlier, the first packet of a flow is punted to the controller, that computes a default route for it. The controller polls the switches at regular intervals of δ_{poll} seconds to track the *instantaneous* traffic demands $T_{u,v}$ between *active* application end-points u and v . Note that such active end points already have routing state set up in the switches. The controller uses this information to compute smoothed averaged traffic demand for various source-destination pairs (using an EWMA). Using these as input, the controller solves the linear program using an off-the-shelf LP solver (we use CVX [12]). If the routes have changed for the active end-points, the controller adds (and deletes) forwarding entries at the appropriate switches.

We employ two optimizations to reduce how often the controller computes and installs routes: (1) If the improvement in the objective function for the new traffic demands is not significant (we use 5%), then the controller ignores the new solution and carries on with the current route. (2) If the smoothed traffic demands for various source destination pairs have not changed substantially (we again use 5%), then the controller avoids solving the LP altogether.

We note that our implementation satisfies the three key requirements outlined at the beginning of this section: (1) using multipath routing; In fact, our approach uses the top- k shortest paths, some of which may be longer than the other, to exploit path diversity to the fullest, (2) working on fine time-scales, as determined by the parameters δ_{poll} and the EWMA weights, and (3) working with a global view by using a central controller to reason about network wide routes.

We also note that, on account of satisfying the three requirements, our approach, should in theory result in *loss-free* routing when applied atop the Fat-Tree topology. On other regular topologies, our approach may not offer completely loss-free routing, but it should result in significantly lower losses. We investigate this in the next section and find, in fact, that our approach is able to ensure loss-free routing over all three topologies.

5.3 Evaluation

Microbenchmarks. We ran our LP for the test topologies in the previous section. Note that the topologies have similar sizes as the first 10 data centers we studied in Section 2. For these data center topologies, the LP completed in just a few microseconds each even when using a commodity desktop with a 2GHz CPU and 1GB RAM. We also created LPs with toy traffic patterns for

the larger data centers and found that the LP runs in a few ms. With greater parallelism and processing, the run time for the LP can be made negligible.

As in prior work [10], we estimated the number of new flow requests that the NOX controller can handle on a commodity desktop. Note that in our setting, this would be required for the first packet of a new application flow. We found that 15000 flows/s can be handled easily without any additional CPU overhead. With greater parallelism, processing and memory at the controller, we believe that this can be made 1-2 orders of magnitude faster. This would suffice for the flow arrival rates reported in recent study on data center application-level traffic patterns [16] where the median arrival rate was measured to be 10^5 flows per second.

Finally, we conducted a limited number of tests to measure the overhead of polling a commercial OpenFlow enabled switch (from NEC) at 10ms intervals for the instantaneous traffic demands. We also measured the overhead of installing and deleting routing table entries at the same time scales (10ms), which reflects the worst case load on the switch when our technique is employed for TE. In both cases, we found that the switch’s forwarding performance was not hurt by the frequent polling and flow table operations.

TE Performance. Based on our microbenchmarks above, we set a small polling interval of $\delta_{poll} = 10\text{ms}$. We set weights on the EWMA such that the currently polled instantaneous values of traffic demands receive much higher weight than historical values.

We tried our fine-grained traffic engineering approach on all three topologies in Section 4 using the exact same traffic models as before. For the Fat-Tree topology (Figure 11), we found that our approach was able to ensure that there were *no losses* at any switch at all. The promise of the Fat-Tree interconnect was that it should be possible to find routes that ensure zero losses and our approach is indeed able to find such routes *by making accurate network-wide decisions at fine enough time-scales*.

On the other two topologies (Figures 9 and 10), we found that our fine grained traffic engineering approach was again able to eliminate all losses. This was surprising, especially since this also applied to the constrained topology in Figure 9, where there was very little path diversity. Upon digging deeper, we found that the bursty traffic patterns at the edge switches meant that *in any given small time window*, such as our polling interval of 10ms, *a significant number of links remain unused*. By operating at fine time-scales, within each polling window, our approach is *easily* able to find non-overlapping paths to accommodate the bursts from all active nodes.

6 Related Work

Measurements of data centers. Very little is known

about the nature of data center traffic. Aside from our paper, the only other work to have considered this issue is [16], where the authors study application level logs collected on a cluster of servers at a single data center. The data center studied is not virtualized and runs a single application. In contrast, we study 19 data centers that run a variety of applications. The study examines application level traffic matrices, flow arrival and departure patterns, failures at the application level due to (presumed) network events, and flow duration. Our focus on link and switch level statistics complements and reinforces the application-level observations in [16]. A key observation in [16] is that congestion does happen in data center networks and congestion events could last up to a few seconds each time. Also, during these events, applications face numerous failures in reading from and writing data to the network. These observations further bolster the need for effective TE in data centers.

Traffic Engineering. Traditional TE techniques, e.g., those applied to WAN traffic in ISP networks, work with predicted traffic matrices and operate on coarse time-scales of several hours [7, 24, 18]. These are inapplicable to data centers. A recent proposal, TEXCP [15], tries to improve responsiveness to real time traffic variations by monitoring available bandwidth at fine time-scales using active probes between ingress-egress node pairs. TEXCP still operates on much coarse time-scales than what is needed for data centers. Also, it relies on local rerouting decisions while our work argues that a global view of network traffic and centrally computing routes are most desirable for data centers.

Data center TE has not received direct attention. Recent approaches [6, 13, 19] have proposed better interconnects that can, in theory, support arbitrary traffic matrices as long as the the traffic generated or received by a server in the data center does not overwhelm its network link. Our work shows that for these approaches to effectively support such traffic matrices and meet the theoretical guarantees in practice they need to know the instantaneous traffic demands and make fine-grained routing decisions. Our fine-grained TE approach can apply to these interconnection approaches. It can even apply to other approaches that propose using data center servers as waypoints [14].

On a related note, we observe that our approach can be installed in current data centers via a simple firmware upgrade. In contrast, [6, 13, 19] require upgrades/forklift changes to the hardware in data centers.

Concurrent work on “incast” [11, 22, 21] propose that for certain traffic matrices, the solution lies not in the network but at the the endhost. The author propose certain changes to the networking stack which can eliminate incast congestion collapse. This solution only works for certain traffic matrices. Our solution generalizes to a

much larger range of traffic matrices. Furthermore, our traffic engineering approach is complementary to and can coexist with end-host based approaches.

Workload generators. Numerous studies [17, 20] have been performed on modeling wide-area and Ethernet traffic. Such models have informed various approaches for traffic engineering, anomaly detection, provisioning and synthetic workload generation. Our traffic generator makes a similar contribution to research in data center networking.

7 Conclusion

In this paper, we have presented a measurement study of network-level traffic patterns inside 19 commercial data centers. Our measurements reveal that link utilizations are low and loss rates are significant on data center links, and that the bursty nature of data center traffic is responsible for this. We have designed a data center workload generator based on these insights that can be used to produce workloads that closely match any one of the 19 data centers that we have studied. Using the workload generator, we were able to identify three key drawbacks in existing approaches for traffic engineering in data centers with respect to their ability to control losses on data center links: (1) lack of multi-path routing, (2) lack of load-sensitivity and (3) lack of a global view in making traffic engineering decisions. Our insights have motivated the design of a centralized fine-grained traffic engineering mechanism that can alleviate losses. We have implemented this approach using OpenFlow and have found it to be very effective.

This paper significantly advances the state of the art in data center traffic engineering in terms of providing empirical insights, providing deeper understanding of existing traffic engineering techniques in the context of data centers, and providing a new and effective technique for data center traffic engineering.

The workload generator and the virtual data center testbed we used in our study are currently available for use upon request. We are planning a public release soon.

References

- [1] The network simulator - ns-2. <http://www.isi.edu/nsnam/ns/>. <http://www.isi.edu/nsnam/ns/>.
- [2] NOX: An OpenFlow Controller. <http://noxrepo.org/wp/>.
- [3] The OpenFlow Switch Consortium. <http://www.openflowswitch.org/>.
- [4] VDE: Virtual Distributed Ethernet. <http://vde.sourceforge.net/>.
- [5] VirtualBox. <http://www.virtualbox.org>.
- [6] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM*, pages 63–74, 2008.
- [7] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Infocom*, 2000.
- [8] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. pages 151–160, 1998.
- [9] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding Data Center Traffic Characteristics. In *Proceedings of Sigcomm Workshop: Research on Enterprise Networks*, 2009.
- [10] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: taking control of the enterprise. In *SIGCOMM*, 2007.
- [11] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph. Understanding tcp incast throughput collapse in datacenter networks. In *WREN '09: Proceedings of the 1st ACM workshop on Research on enterprise networking*, pages 73–82, New York, NY, USA, 2009. ACM.
- [12] M. Grant, S. Boyd, and Y. Ye. CVX: Matlab software for disciplined convex programming, ver. 1.1. Available at www.stanford.edu/~boyd/cvx/, Nov. 2007.
- [13] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VI2: a scalable and flexible data center network. In *SIGCOMM*, 2009.
- [14] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: a high performance, server-centric network architecture for modular data centers. In *SIGCOMM*, 2009.
- [15] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: responsive yet stable traffic engineering. In *SIGCOMM*, 2005.
- [16] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The Nature of Data Center Traffic: Measurements and Analysis. In *IMC*, 2009.
- [17] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. Netw.*, 2(1), 1994.
- [18] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: existing techniques and new directions. In *SIGCOMM '02*, 2002.
- [19] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. Portland: a scalable fault-tolerant layer 2 data center network fabric. In *SIGCOMM*, 2009.
- [20] V. Paxson and S. Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Trans. Netw.*, 3(3):226–244, 1995.
- [21] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan. Measurement and analysis of tcp throughput collapse in cluster-based storage systems. In *FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies*, pages 1–14, Berkeley, CA, USA, 2008. USENIX Association.
- [22] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective fine-grained tcp retransmissions for datacenter communication. In *SIGCOMM*, 2009.
- [23] F. Wilcoxon. Biometrics bulletin. 1:80–83, 1945.
- [24] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. Traffic Engineering with Estimated Traffic Matrices. Miami, FL, Oct. 2003.