The Case for Fine-Grained Traffic Engineering in Data Centers

Theophilus Benson[†]*, Ashok Anand*[†]*, Aditya Akella*[†] *and Ming Zhang* [⋆]
[†] *University of Wisconsin-Madison;* [⋆] *Microsoft Research*

Abstract

In recent years, several techniques have been suggested for routing and traffic engineering in data centers. However, not much is known about how these techniques perform relative to each other under realistic data center traffic patterns. Our preliminary study reveals that existing techniques can only achieve 80% to 85% of the ideal solution in terms of the number of bytes delivered. We find that these techniques suffer due to their inability to utilize global knowledge of the properties of traffic flows and their inability to make coordinated decision for scheduling flows at fine time-scales. Even recent traffic engineering techniques such as COPE fail in data centers despite their proven ability to adapt to dynamic variations, because they are designed to operate at longer time scales (on the order of hours, at least). In contrast, data centers, due to the bursty nature inherent to their traffic, require adaptation at much finer times scales. To this end, we define a set of requirements that a data center-oriented traffic engineering technique must possess in order to successfully mitigate congestion. In this paper, we present the design for a strawman framework that fulfills these requirements.

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's network infrastructure. For example, a congested data center network, where internal traffic is routinely subjected to losses and poor throughput, could lead search queries to take longer to complete, IM message to get delayed, gaming experience to deteriorate, and POP mail services and Web transactions to hang. The dissatisfied end-users and subscribers could choose alternate providers, resulting in a 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 techniques are in a very primitive state. Today, most operators try to tweak wide-area TE 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 the traffic of data center networks and wide area networks share basic similarities.

The fact that "traditional" approaches are ineffective is reinforced by recent work that shows that data center networks experience congestion events each lasting up to a few seconds, during which applications experience numerous failures [8]. 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 patterns interact with existing techniques.

We first seek to understand why existing common TE mechanisms (most of which are ECMP based) and recent proposals fail, what their fundamental flaws are, and how much more room there is for improvement. In order to answer these questions, we conduct simulations using real traces collected from a data center. We find that existing mechanisms achieve only 80-85% of the performance achieved by an optimal routing mechanism, indicating that there is significant room for improvement. The optimal routing mechanism uses perfect knowledge about instantaneous demands to generate routes that successfully support the current traffic with minimal congestion. We find that existing techniques perform suboptimally due to the failure to utilize multipath diversity, due to the failure to adapt to instantaneous load, or due to the failure to use a global view to make routing decisions.

While centralized routing platforms [1] can be leveraged to address the latter shortcoming, it is less clear how to design techniques that can accommodate dynamic variations in network traffic. A candidate set of techniques includes recent proposals for TE in ISPs [3, 14, 9, 13] that aim to compute routes offering robust performance under a range of traffic matrices. However, these techniques function at the granularity of hours. In contrast, measurement studies [5] of data center traffic have shown that data center traffic is bursty in nature, and unpredictable at long time-scales, making ISP-oriented techniques inapplicable.

Prior work has shown that traffic that is bursty in nature and follows a heavy-tailed distribution can be pre-

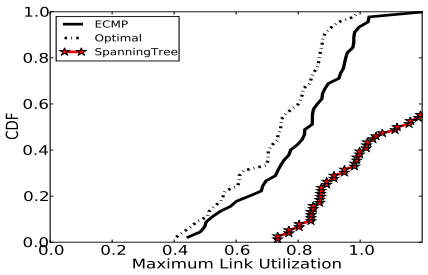

Figure 1: Distribution of the MLU for optimal, ECMP and Spanning Tree on a canonical tree topology.

dictable at short time scales of 1-5s [11]. We find from our traces that this observation is true for data center ToR-to-ToR traffic as well. Motivated by this, we seek a class of *fine-grained* techniques that leverage this short-term predictability of traffic. These *fine-grained* techniques must operate at the granularity of seconds. If designed effectively to operate atop centralized routing platforms, such techniques could potentially address the key drawbacks of existing data center TE mechanisms and offer close-to-optimal performance.

We identify key guidelines for designing such centralized fine-grained TE mechanisms. We then apply these design guidelines to develop a strawman approach called MicroTE. We present a preliminary evaluation of this approach and discuss various issues in scaling the framework and in ensuring that it can be employed in a backward compatible fashion in today's data centers.

2 Comparative Study

In this section, we evaluate the effectiveness of various traffic engineering (TE) techniques and data center network architectures at accommodating various traffic patterns. We perform an extensive study using simulations with traces from a data center running Map-Reduce style applications. This data center is comprised of 1500 servers and 75 ToR switches. We collected several days worth of network events from all the servers. From these network events, we extracted, using a mapping of servers to ToR switches, the ToR-2-ToR traffic matrix (TM). In our simulations of the various topologies and TE techniques, we feed as input into our simulator a sequence of per second ToR-2-ToR TM representing traffic over a 2 hour period. For simplicity, we do not consider the effect of change in TCP behavior due to change in the routing scheme.

Canonical tree topology: We first examine a canonical 2-Tier trie topology with two cores, similar to that used in the cloud data center from which the data was gathered. On this topology, we first examine the performance of single path static routing, and then study if ECMP can perform better.

In Figure 1, we present the cumulative distribution of the maximum link utilization (MLU) for every second for the trie topology over the 2 hour periods, when employing Spanning Tree, ECMP and optimal routing. Optimal routing is computed assuming perfect knowledge of the TM every second, where we formulate a linear program with the objective of minimizing the MLU. We observe that ECMP and Spanning Tree perform worse than the optimal algorithm. In certain cases, we notice MLU values that are greater than one for ECMP and Spanning Tree indicating loss of packets. As expected, ECMP achieves lower MLU than Spanning Tree, because ECMP leverages multiple network paths and thus a larger network bandwidth. We find that the gap between optimal routing and ECMP under heavy loads is 15-20%. This gap appears to be due to a lack of a global view in scheduling flows in ECMP.

In general, we find that ECMP is a better fit for data center networks than Spanning Tree, however, ECMP is not perfect as it still results in a substantial amount of loss. ECMP is unable to significantly reduce losses as it balances traffic across multiple paths leading to even utilization, but it fails to take into account the instantaneous load on each path which is central to controlling network-wide load and hence 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 pairs of flows to a common set of network links. Since ECMP does not re-assign based on observed load, it cannot help to overcome losses due to temporary over-subscription on the path, which may happen when both pairs of flows experience bursty transmissions at similar times.

These analyses illuminate an interesting point, namely that although TE techniques must exploit multiple-path routing in existing data center topologies in order to obtain better performance, simply striping traffic across multiple paths is insufficient.

Fat-Tree: Next, we examine a recent proposal, the Fat-Tree interconnect, that supports extremely high bisection bandwidth [2, 10]. In theory, routes can be constructed over this topology to support any traffic demand matrix. 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 TM. 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 TM demand. In [2], 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 approximate condition #2. In particular, at regular intervals (say, every second), each switch in the lower level of the topology measures the utilization of its output ports 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. In studying fat-tree we find that the local heuristic prevents this goal from being achieved. As observed by Al-Fares el al. [2], there is a 23% performance gap between Fat-Tree and optimal due to conflicts between locally optimal decisions and globally optimal decisions.

VL2: Although we have not evaluated VL2's architecture [5], we note the authors of VL2 perform a similar evaluation to ours. In evaluating VL2, Maltz et al. observed a performance gap of up to 20% with respect to the optimal, they attributed this performance gap to drawbacks in the underlying ECMP technique on which VL2 is based, namely the inability of ECMP based VL2 to track and to instantaneous adapt to demand.

To summarize, using real traces from a cloud data center, we have found that existing techniques fail to control losses in the presence of bursty traffic in data centers for one of the following reasons: (1) not using multipath routing (2) not taking instantaneous load into account and (3) not making decisions on the basis of a global view of the network.

3 Design Requirements for TE Algorithms

In summary, our evaluations show a performance gap of 15-20% between optimal routing and current routing practices. Even on recently proposed data center topologies such as Fat-Tree, the gap is 23%. The primary reason behind this gap is the lack of global knowledge. An ideal TE mechanism should configure routes dynamically by taking a global view of the future TM and computing optimal routes for the future TM at hand. One way to approximate the future TM is by extrapolating it from a series of historical TMs generated from the global view. However, this approach can only be used if the TM shows some predictability.

Recent measurement studies [8, 4] have examined the predictability of traffic in data centers; these studies show that data center traffic is bursty and that the TM lacks predictability at times scales of 150 seconds or longer. Furthermore, Benson et al. [4], show that the arrival processes for data center traffic follows an ON-OFF pattern whose parameters can be explained by heavy tailed distributions, which provides further evidence that traffic is bursty and unpredictable at long time-scales.

Given these results, initially it appears as though there is no predictability. In what follows, we present preliminary results to show that predictability exists at short time-scales. Such observations have been used in recent studies to augment data centers with "flyways" [7, 12].

We examine the change in traffic exchanged between each pair of ToR switch and find that across different time periods, approximately 35% or 0.35 of the *total* traffic exchanged between pairs of ToR remains predictable. This is shown more precisely in Figure 2, where we present the distribution of the fraction of total traffic demand contributed by those ToR pairs which had no significant change in traffic over a 1 second time period; we use $>$ 20% change as a threshold for significant change. From Figure 2, we observe that for 80% of the 1s intervals, more than 35% of the total traffic demand in the 1s interval remains predictable (i.e., does not change significantly) for at least 1 second into the future. This result provides us the proof of a reasonable amount of short-term stability in the traffic demands of data centers.

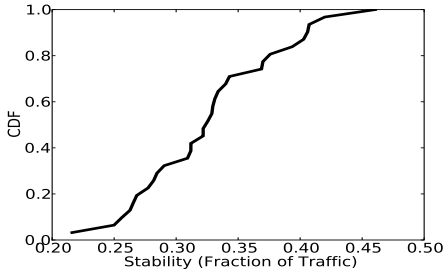

Figure 2: Distribution of the fraction of total traffic demand, contributed by ToR pairs who had insignificant change in next second.

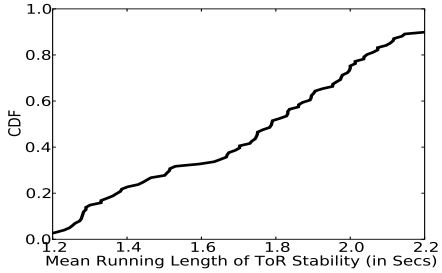

Figure 3: Distribution of the mean run-length for top 100 ToR pairs, that contributed to 80% of total traffic within a 1s interval

Next, we attempt to determine the duration of the observed stability more accurately. To do this, we examine the run-length of the sequence of seconds where change in traffic for each ToR pair remains insignificant, i.e., less than 20% compared to the demand at the beginning of the sequence. In Figure 3, we present the distribution of the mean run-lengths for each ToR-pair. From this figure, we observe that 60% of the pairs remain predictable for between 1.6 and 2.2 seconds on average – this proves, that for the predictable traffic, we should be able to use routes based on historical TM for the last 1 second to effectively route them.

3.1 Design Requirements

Pulling all of our observations from our study of TE mechanisms in data centers and from our study on the predictability of data center traffic, we have established a

set of three design principles that a TE mechanism must adhere to in order to effectively alleviate loss in a data center:

(1) Multipath Routing: An ideal approach must take advantage of the path diversity available in the data center's topology. Failure to do this will limit the available network capacity and increase the likelihood of congestion.

(2) Coordinated Scheduling Using a Global View of Traffic: An ideal approach must coordinate the scheduling of traffic across the available network paths. This requires using information from the global view of the network. Failure to do this will lead to suboptimal scheduling of flows by network elements - network devices may choose locally optimal paths which could inadvertently create globally sub-optimal paths that lead to congestion.

(3) Exploiting Short-Term Predictability for Adaptation: An ideal approach must take advantage of short term predictability where applicable but must also adapt quickly to variations in the underlying traffic patterns, while generating routes that closely approximate the performance of the optimal TE. In the worst case, such an algorithm should perform no worse than existing approaches. If the approach performs poorly, then it provides operators with no incentive to adopt it.

In addition to the above principles, for a traffic engineering approach to be adopted today, it must meet certain other requirements such as: (1) requiring minimal changes to data center network, (2) scaling to large topologies, and (3) performing in a manner that is agnostic to the properties of the data center's underlying topology as data centers may differ in their network design.

Next, we discuss our TE proposal keeping these design goals in mind.

4 MicroTE: our design proposal

In this section, we present the design of a strawman framework, called MicroTE, that satisfies the design requirements laid out in Section 3.

4.1 MicroTE: Architecture

MicroTE is a TE mechanism with the ability to adapt to variations in the underlying network traffic patterns at a *microscopic* (per-second) level. We believe that this can be implemented within the OpenFlow [1] framework, thus allowing data centers to leverage our framework by simply applying a firmware upgrade to the switches and designating a (logically-central) server as the controller.

In Figure 4, we show the key components of our framework and show how they would interact. We discuss each of these components in detail and examine the trade-offs that can be made in implementing them below:

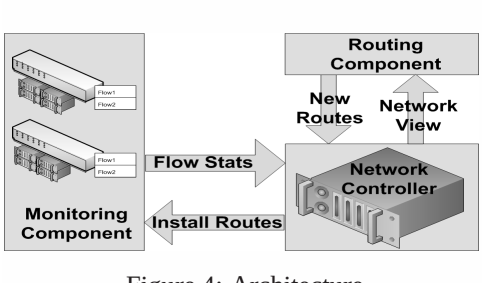

Figure 4: Architecture

Network Controller: This component retrieves the traffic/flow statistics from the monitoring component and uses this data to create a global view of the network, i.e., a view of the network topology together with the instantaneous traffic demands between different ToR switches. It then passes this information to the routing component. Also, the network controller relays routes received from the routing component to the switches, i.e., it populates the flow tables in individual switches using the OpenFlow API.

Monitoring Component: The monitoring component could reside on an OpenFlow-enabled ToR switch, or on endhosts residing in the rack to which the switch is attached. Its goal is to monitor demands, or flow statistics, between the rack in question and other network locations. In terms of obtaining the flow statistics, there are two design options we can adopt:

(1) The network controller polls switches at periodic intervals of δ_{poll} seconds. The switches respond to the network controller with byte counts for flow entries for inter-ToR traffic, where the counts are aggregated over the δ_{poll} seconds between successive polls. The Openflow framework already provides a mechanism for the controller to poll switches in this fashion. Since we are interested in exploiting short-term predictability, δ_{poll} should be on the order of a few seconds (1 to 5s).

(2) End-hosts in a rack can perform measurements of their traffic sending patterns to other parts of the data center, and inform the controller of the demands, either at regular intervals or in a triggered fashion.

Choosing the appropriate option among the above two involves understanding the trade off between scale and implementation complexity. The first option is simple to implement and fits perfectly within the OpenFlow framework. However, it fails to scale for several reasons: (1) If the network controller needs to constantly poll all switches on nearly a per-second granularity then this creates a significant amount of control traffic on the network, and (2) If all switches respond regardless of significance of the change in traffic experienced then both the network and the switch (in particular, the switch's CPU) may be overwhelmed by the transmission of flow statistics. This particularly impacts switches that have a multitude of flow entries. In the case of the second ap-

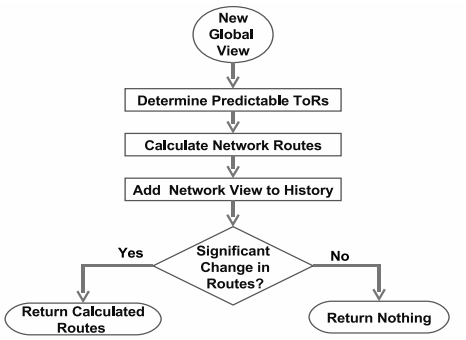

Figure 5: Flow chart of the logic within the routing component.

proach, additional implementation complexity must be introduced for modifying endhosts to perform monitoring. However, the generation of flow statistics will no longer be bottlenecked by the ToR switch's processor, and the network will not be overwhelmed by unnecessary polling control messages from the network controller. Furthermore, there will be no need to report flow statistics when there is no significant change in traffic patterns. This is because an end-host based framework allows triggered updates of traffic demands to the controller (when traffic demands change significantly), while a purely switch based approach, at least in the current implementation of OpenFlow, only supports polling by the controller, which is far less flexible.

Thus, we adopt an end-host based approach for monitoring. One server per rack is designated to collect traffic demands for the entire rack and report to the controller every δ_{poll} seconds by default. A full report can be avoided if no significant change is observed. To control the overhead, reports can be compressed. Also, the designated server may generate a triggered update before the δ_{poll} interval lapses to report sudden changes in traffic. This would require servers to constantly monitor their traffic and report to the designated server at timescales smaller than δ_{poll}.

Routing Component: This component would compute the network paths using the global view provided by the network controller.

Figure 5 shows a flowchart of the actions performed by the routing component. The routing component will be invoked by the network controller once it has assembled a new global view of traffic within the network. Routes are computed using a routing algorithm that can take advantage of the path diversity available in the data center topologies. Thus, by taking into account the global view of traffic for generating routes that utilize all network paths, our first and second design requirements are fulfilled.

Furthermore, the routing component tracks the predictable and unpredictable ToR-to-ToR entries in the TM based on the available historical measurements of the entries. For the unpredictable ToR-to-ToR demands, the component assigns routes based on ECMP. By doing this, the routing component avoids the risk of performing worse than the existing ECMP based approaches, thus satisfying the third design requirements. For predictable ToR-to-ToR demands, the routing component uses a specific routing algorithm that optimizes routes for these entries, given that the rest of the entries are routed using ECMP. The routing component returns these paths to the network controller only if the routing component finds that the newly created paths are different than the paths currently being used. The routing component is tasked with performing this check as opposed to relegating this functionality to ToR switches or end-hosts, because while individual switches and end host can check for variations within their portion of the network's TM, they are unable to determine if these variations require a significant change in the network paths.

In choosing the routing algorithm to implement, we must examine several alternatives ranging from LP-based formulations to heuristics, such as greedy routing algorithms. Ideally, the algorithm chosen will provide the best trade off between speed and accuracy.

For now we ignore the tussle between speed and accuracy, and we implement an LP-based solution and use it to examine the data center traces. In formulating our LP, we set the objective function to minimize the maximum link utilization and we set flow conservation constraints as usual (details omitted for brevity). We consider ToR-to-ToR demands as predictable if demands change by less than 20% across a 1 second interval. In doing this preliminary evaluation, we are able to determine if we can leverage such short-term predictable ToR-to-ToR demands for obtaining better routes that further reduce the MLU compared to the techniques evaluated in Section 2 bringing it closer to optimal.

We present our results in Figure 6. We use a canonical tree topology for these experiments, the same as that used for Figure 1. Our findings show that in 80% of the 1s intervals, the difference between the MLU for the optimal approach and the MLU for MicroTE is small (less than 6% difference) and the difference is significantly smaller than that between the optimal and ECMP (Figure 1), indicating leveraging short-term predictability in some part of TM can indeed result in more adaptive routing that is able to contain congestion better. We do note that there are situations ($<$3%) where the MLU is greater than 1. This arises because the traffic is not perfectly predictable for the ToR pairs in question. In future work, we plan to address this issue by monitoring traffic patterns from each rack to examine if there are exceptions to the assumption regarding predictability, and changing the route computation to simply using ECMP whenever

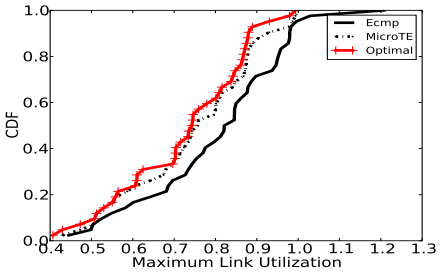

Figure 6: Comparison of MicroTE, Optimal and ECMP

the assumption fails.

5 Other Related Work

Traffic Engineering. Traditional TE techniques, e.g., those applied to WAN traffic in ISP networks, work with predicted TMs and operate on coarse time-scales of several hours [3, 14, 9]. These are inapplicable to data centers because a data center's traffic patterns are predictable at much finer time scales. Other more responsive TE proposals for ISP, such as TEXCP [6], rely on local load balancing decisions which our work shows to be suboptimal in the context of data centers.

New Architectures. Contrary to our approach of reducing congestion by reconfiguring the routing within the network, others [5, 2] have argued for a forklift upgrade of the data center's network. They argue for the replacement of existing networks with a new network substrate that can support a larger bisection bandwidth. We argue that such techniques will face slow adoption for two reasons: (1) a forklift upgrade will require significant capital investment thus delaying deployment and (2) these techniques use ECMP style TE which we showed to be sub-optimal in section 2.

Augmenting Data Center Networks. Complementary to our approach, is the use of techniques such as flyways([7, 12]) that argue for adding extra links as a means of tackling hot spots. These extra links provide additional capacities, where and whenever needed. Like us, they also find the predictability of traffic demands at short time scales, allowing flyways to keep up with the changing demand. In contrast, our approach argues for fine grained traffic engineering with existing links while leveraging short term predictability of traffic demands. Our approach is more general and applicable to any data center topology, including data center topologies with flyways.

6 Conclusion and Future Work

In this paper, we studied the effectiveness of various TE techniques and found that many of these techniques are inadequate for today's data centers for at least one of these three reasons; (1) lack of multipath routing, (2) lack of load-sensitivity and (3) lack of a global view in making TE decisions. Given the drawbacks of existing techniques, we set off to determine the requirements for an ideal TE mechanism for data centers by examining the traffic patterns within a cloud data center. In studying the traffic patterns, we observe that data centers contain short-term predictability that last on order of several seconds. Pulling together our observations, we developed a set of design requirements for an ideal TE mechanism; (1) must utilize multi-path routing, (2) must coordinate scheduling of traffic, and (3) must adapt to short term predictability.

To this end, we developed MicroTE, a strawman framework, that satisfies our design goals. We propose the use of the OpenFlow framework as a means of aggregating and creating a global view of the network. Also, the use of the OpenFlow framework allows MicroTE to coordinate scheduling of traffic within the network. We discuss various options for routing and polling algorithms that allow MicroTE to adapt to short term predictability as well as to perform multipath routing.

Acknowledgements. This work is supported in part by an NSF FIND grant (CNS-0626889), an NSF CAREER Award (CNS-0746531), an NSF NetSE grant (CNS-0905134), and by grants from the UW-Madison Graduate School. Theophilus Benson is supported by an IBM PhD Fellowship.

References

[1] The OpenFlow Switch Consortium. `http://www.openflowswitch.org/`.
[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM*, pages 63–74, 2008.
[3] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Infocom*, 2000.
[4] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding Data Center Traffic Characteristics. In *Proceedings of Sigcomm Workshop: Research on Enterprise Networks*, 2009.
[5] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: a scalable and flexible data center network. In *SIGCOMM*, 2009.
[6] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: responsive yet stable traffic engineering. In *SIGCOMM*, 2005.
[7] S. Kandula, J. Padhye, and P. Bahl. Flyways to de-congest data center networks. In *Proc. ACM Hotnets-VIII*, New York City, NY. USA., Oct. 2009.
[8] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The Nature of Data Center Traffic: Measurements and Analysis. In *IMC*, 2009.
[9] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: existing techniques and new directions. In *SIGCOMM '02*, 2002.
[10] 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.
[11] K. Park and T. Tuan. Performance evaluation of multiple time scale tcp under self-similar traffic conditions. *ACM Trans. Model. Comput. Simul.*, 10(2):152–177, 2000.
[12] G. Wang, D. G. Andersen, M. Kaminsky, M. Kozuch, T. S. E. Ng, K. Papagiannaki, M. Glick, and L. Mummert. Your data center is a router: The case for reconfigurable optical circuit switched paths. In *Proc. ACM Hotnets-VIII*, New York City, NY. USA., Oct. 2009.
[13] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. Cope: traffic engineering in dynamic networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):99–110, 2006.
[14] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. Traffic Engineering with Estimated Traffic Matrices. Miami, FL, Oct. 2003.