CREATE: CoRrelation Enhanced trAffic maTrix Estimation in Data Center Networks

Zhiming Hu Yan Qiao Jun Luo Peng Sun Yonggang Wen
School of Computer Engineering, Nanyang Technological University, Singapore
Email: {zhu007, yqiao, junluo, sunp0003, ygwen}@ntu.edu.sg

***Abstract*—Understanding the pattern of end-to-end traffic flows in *Data Center Networks* (DCNs) is essential to many DCN designs and operations (e.g., traffic engineering and load balancing). However, little research work has been done to obtain traffic information efficiently and yet accurately. Researchers often assume the availability of traffic tracing tools (e.g., OpenFlow) when their proposals require traffic information as input, but these tools may generate high monitoring overhead and consume significant switch resources even if they are available in a DCN. Although estimating the traffic matrix between origin-destination pairs using only basic switch SNMP counters is a mature practice in IP networks, traffic flows in DCNs are notoriously more irregular and volatile, while the large number of redundant routes in a DCN further complicates the situation. To this end, we propose to utilize the service placement logs for deducing the correlations among top-of-rack switches, and to leverage the uneven traffic distribution in DCNs for reducing the number of routes potentially used by a flow. These allow us to develop an efficient CoRrelation Enhanced trAffic maTrix Estimation (CREATE) method that achieves high accuracy. We compare CREATE with two existing representative methods through both experiments and simulations; the results strongly confirm the promising performance of CREATE.**

I. Introduction

As data centers that house a huge number of inter-connected servers become increasingly central for commercial corporations, private enterprises and universities, both industrial and academic communities have started to explore how to better design and manage the *data center networks* (DCNs). The main topics under this theme include, among others, network architecture design [1]–[3], traffic engineering [4], capacity planning [5], and anomaly detection [6]. However, little is known so far about the characteristics of traffic flows within DCNs. For instance, how do traffic volumes exchanged between two servers or top-of-rack (ToR) switches vary with time? Which server communicates to other servers the most in a DCN? In fact, these real-time traffic characteristics serve as critical inputs to all above DCN operations; their absence may hamper the developments of others.

Existing proposals in need of detailed traffic flow information collect the flow traces by deploying additional modules on either ToR switches [4] or servers [7] in small scale DCNs. However, both methods require substantial deployments and high administrative costs, and they are difficult to implement thanks to the heterogeneous nature of the hardware in DCNs.[1] More specifically, the switch-based approach, on one hand, needs all the ToRs to support OpenFlow [9] and consumes a substantial amount of switch resources to maintain the flow entries.[2] On the other hand, the server-based approach, which requires instrumenting all the OS kernels of the servers or VMs to support data collection, is unavailable in most data centers [10] and is nearly impossible to be implemented peacefully and quickly while supporting a lot of cloud services in large scale DCNs.

It is natural then to ask whether we could borrow from *network tomography*, where several well-known techniques allow traffic matrices of IP networks to be inferred from link level measurements (e.g., SNMP counters) [11]–[13]. As link level measurements are ubiquitously available in all DCN components, the overhead introduced by such an approach can be very light. Unfortunately, both experiments in medium scale DCNs [10] and our simulations (see Sec. V-C) demonstrate that all existing tomographic methods perform poorly in DCNs. This attributes to the irregular behavior of end-to-end flows in DCNs and the large quantity of redundant routes between each pair of servers or ToR switches.

There are actually two major barriers to apply tomographic methods to DCNs. One is the sparsity of traffic matrix between ToR pairs. This refers to the fact that one ToR switch may only exchange flows with a few other ToRs, as demonstrated by [14]. This fact substantially violates the underlying assumption of tomographic methods including, for example, the amount of traffic a node (origin) would send to another node (destination) is proportional to the traffic volume received by the destination [11]. The other is the highly under-determined solution space. In other words, a huge number of flow solutions may potentially lead to the same SNMP byte counts. For a medium size DCN, the number of end-to-end routes is up to ten thousands [10] while the number of link constrains is only around hundreds.

In this paper, we aim at conquering the aforementioned two barriers and making *traffic matrix* (TM) estimation feasible for DCNs, by utilizing the distinctive information or features inherent to these networks. On one hand, we make use of the service placement logs (from the resource scheduler in the

*This work is supported in part by AcRF Tier 1 Grant RGC5/13.

ISBN 978-3-901882-58-6 © 2014 IFIP

[1] A DCN may contain a lot of legacy switches or servers [8].

[2] To the best of our knowledge, no existing switch with OpenFlow support is able to maintain so many entries in its flow table due to the huge number of flows generated per second in each rack.

controller node of DCNs) to derive the correlations among ToR switches, as our experiments demonstrate that racks supporting the same service tend to exchange high traffic volumes. The communication patterns between ToR pairs inferred by such an approach are far more accurate than those assumed by conventional traffic models (e.g., the gravity traffic model [11]). On the other hand, by analyzing the statistics of link counters, we find that the utilization of both core links and aggregation links is extremely uneven. In other words, there are a considerable amount of links undergoing very low utilization during the particular time interval. This observation allows us to eliminate the links whose utilization is under a certain (small) threshold and to substantially reduce the number of redundant routes. Combining the aforementioned two methods, we propose CREATE (CoRrelation Enhanced trAffic maTrix Estimation) as an efficient estimation technique to accurately infer the traffic flows between ToR switch pairs without requiring any extra measurement tools. In summary, we make the following contributions in our paper.

- We pioneer in using the service placement logs to deduce the correlations of ToR switch pairs, and we also propose a simple method to evaluate the correlation factor for each ToR pair. Our traffic model, assuming that ToR pairs with a high correlation factor may exchange higher traffic volumes, is far more accurate for DCNs than conventional models used for IP networks. This is so because different services rarely communicate with each other while servers have greater chance to exchange data if they host the same service [15].
- We innovate in leveraging the uneven link utilization in DCNs to remove potentially redundant routes. As both our experiments and those presented in [16] show that link utilization can be very uneven with a few links carrying a dominating fraction of traffic, we may consider links with very low utilization as non-existent without affecting much the accuracy of TM estimation. In fact, eliminating these lowly utilized links can effectively lessen the redundant routes in DCNs, resulting in a more determined tomography problem. Moreover, we also demonstrate that changing a low-utilization threshold has an effect of trading estimation accuracy for its complexity.
- We propose CREATE as an efficient method to infer the TM for DCNs with high accuracy. This new algorithm first calculates a prior assignment of traffic volumes for each ToR pairs using the correlation factors. Then it removes lowly utilized links and operates only on a subgraph of the DCN topology. It finally adapts a quadratic programming to determine TM under the constraints of the tomography model, the correlation-enhanced prior assignments, and the reduced DCN topology.
- We validate CREATE with both experiments on a relatively small scale testbed and extensive large scale simulations in *ns*-3. All the results strongly demonstrate that our new method outperforms two representative traffic estimation methods on both accuracy and running speed.

The rest of the paper is organized as follows. We first survey the related work in Sec. II. Then we formally describe our problem in Sec. III. Sec. IV analyzes the traffic data in real DCNs and reveals the two observations on ToR correlations; we also present our CREATE method for TM estimation in the same section. We evaluate CREATE using both real testbed and different scales of simulations in Sec. V, before concluding our paper in Sec. VI.

II. Related Work

As data center networking has recently emerged as a hot topic for both academia and industry, numerous studies have been conducted to improve its performance [1]–[6]. However, little work has been devoted to the traffic measurement, although the awareness of traffic flow pattern is a critical input to all above network designs or operations. Most proposals, when in need of traffic matrices, rely on either switch-based or server-based method to obtain them.

The switch-based method (e.g., [4]) adopts programmable ToR switches (e.g., OpenFlow [9] switch) to record flow data. However, this method may not be feasible for three reasons. First, it incurs a high switch resource consumption to maintain the flow entries. For example, if there are 30 servers per rack, the default lifetime of a flow entry is 60 seconds, and on average 20 flows are generated per host per second [17], then the ToR switch should be able to maintain $30 \times 60 \times 20 = 36,000$ entries, while the commodity switches with OpenFlow support such as HP ProCurve 5400zl can only support up to 1.7k OpenFlow entries per linecard [7]. Secondly, hundreds of controllers are needed to handle the huge number of flow setup requests. In the above example, the number of control packets can be as many as 20M per second. And a NOX controller can only process 30,000 packets per second [17]; thus it needs about 667 controllers to handle the flow setups. Finally, not all the ToR switches are programmable in DCNs with legacy equipments, while the data center owners may not be willing to pay for upgrading the switches.

The server-based method requires a special module to be inserted into the OS kernel on each server to support flow data collection [7], [18]. Also, the heterogeneity of data center servers may also complicate the problem: dedicated modules may need to be prepared for different servers and their OSs. Moreover, adding this module does cost server resources to perform flow monitoring. Finally, similar to the switch-based approach, the willingness of the data center owner to upgrade all servers may yet be another obstacle.

Network tomography has long been an important and efficient approach to obtain traffic information in IP networks. For example, tomogravity [11] adapts the gravity model to get the prior TM and SRMF [12] is shown to perform better than others when the TM is low rank. One study that has partially motivated our work is [10]: it investigates the nature of DCN traffic on a single MapReduce data center and poses the question whether traffic matrices can be inferred from link counters by tomographic methods. In a way, the answer given in [10] is negative due to the fundamental differences between

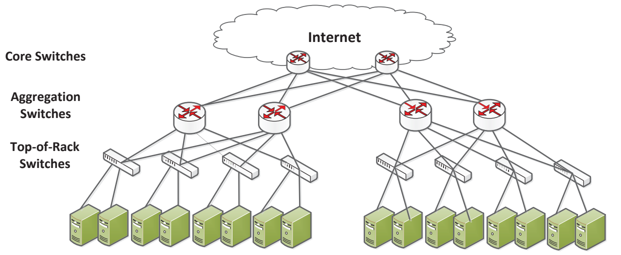


Fig. 1. An example of conventional data center network architecture, suggested by Cisco [21].

DCNs and IP networks, which invalidate the assumptions made by conventional tomographic methods [11], [12]; we explain these in Sec. I as two obstacles. We have proposed methods to get the coarse-grained TM in [19], [20], but we hereby aim to overcome these obstacles and hence make a fine-grained TM estimation viable in DCNs.

III. Definitions and Problem Formulation

We consider a typical DCN as shown in Fig. 1. It consists of N Top-of-Rack (ToR) switches, aggregation switches, and core switches connecting to the Internet. There are R services running in this DCN. Note that our method is not confined to this commonly used DCN topology; it accommodates other more advanced topologies also, e.g., VL2 [2], fat-tree [1], as will be shown in our simulations.

We denote by $X_{i \rightharpoonup j}$ the traffic sent from the i-th ToR to the j-th ToR and by $X_{i \leftrightarrow j}$ the volume of traffic exchanged between the two switches. Given the volatility of DCN traffic, we further introduce $X_{i \rightharpoonup j}(t)$ and $X_{i \leftrightarrow j}(t)$ to represent values of these two variables at discrete time t.[3] Note that although these variables would form the TM for conventional IP networks, we actually need more detailed information for the DCN traffic pattern: the routing path(s) taken by each traffic flow. Therefore, we split $X_{i \leftrightarrow j}(t)$ on all possible routes between the i-th and j-th ToRs. Let $\mathbf{X}(t) = [X_1(t), X_2(t), \cdots, X_P(t)]$ represents the volumes of traffic on all possible routes between ToR pairs, where P is the total number of the routes. Consequently, the *traffic matrix* $\mathbb{X} = [\mathbf{X}(1), \mathbf{X}(2), \cdots, \mathbf{X}(T)]$, where T is the total number of time periods, is the one we need to estimate.

The observations that we utilize to make the estimation are the *SNMP counters* on each port of switches. Basically, we poll the SNMP MIBs for bytes-in and bytes-out of each port every 5 minutes. The SNMP data obtained from a port can be interpreted as load of the link with that port as one end; it equals to the total volume of the flows that traverse the corresponding link. In particular, we denote by ToR_i^{in} and ToR_i^{out} the total "in" and "out" bytes at the i-th ToR. We represent links in the network as $\mathbf{L} = \{L_1, L_2, \cdots, L_M\}$, where M is the number of links in the network. Let $\mathbf{B} = \{B_1, B_2, \cdots, B_M\}$ denote the bandwidth of the links, and $\mathbf{Y}(t) = \{Y_1(t), Y_2(t), \cdots, Y_M(t)\}$ denote the traffic loads of the links at discrete time t, and $\mathbb{Y} = [\mathbf{Y}(1), \mathbf{Y}(2), \cdots, \mathbf{Y}(T)]$ becomes the *load matrix*. [4]

TABLE I
Commonly used notations

Notation	Description
N	The number of ToR switches in the DCN
M	The number of links in the DCN
P	The number of routes in the DCN
R	The number of services running in the DCN
T	The number of time periods
$\mathbb{A}$	Routing matrix
$\mathbf{L}$	$\mathbf{L} = [L_i]_{i=1,\cdots,M}$, where L_i is the i-th link
$\mathbf{B}$	$\mathbf{B} = [B_i]_{i=1,\cdots,M}$, where B_i is the bandwidth of L_i
$\mathbf{Y}$	$\mathbf{Y} = [Y_i]_{i=1,\cdots,M}$, where Y_i is the load of L_i
K_i	The number of server belonging to the i-th rack
$X_{i \rightharpoonup j}$	The traffic send from the i-th ToR to the j-th ToR
$X_{i \leftrightarrow j}$	The traffic exchanged between the i-th and j-th ToRs
$\mathbf{X}$	$\mathbf{X} = [X_r]_{r=1,\cdots,P}$, where X_r is the traffic on the r-th routing path
$\bar{X}_r$	The prior estimation of the traffic on the r-th routing path
ToR_i^{in}	The total "in" bytes of the i-th ToR
ToR_i^{out}	The total "out" bytes of the i-th ToR
$\mathbb{S}$	$\mathbb{S} = [S_{ij}]_{i=1,\cdots,R;j=1,\cdots,N}$, where S_{ij} is the number of servers under the j-th ToR that run the i-th service
$Corr_{ij}$	The correlation coefficient between the i-th and j-th ToR.
θ	Link utilization threshold

An extra piece of information that we require is the *service placement logs* recorded by controllers of a DCN. We analyze the service placement logs in the controller nodes in a DCN, and get the *service placement matrix* $\mathbb{S} = [S_{ij}]$ with rows corresponding to services and columns representing the ToR switches. In particular, $S_{ij} = k$ means that there are k servers under the j-th ToR running the i-th service in the DCN. We also denote by K_j the number of servers belonging to the j-th rack. These quantities will be used to compute the correlation coefficient between different ToRs in Sec. IV.

As there are a lot of available routes between any of two ToR switches, the correlation between traffic assignment $\mathbf{X}(t)$ and link load assignment $\mathbf{Y}(t)$ can be formulated as

$$\mathbf{Y}(t) = \mathbb{A}\mathbf{X}(t) \qquad t = 1, \cdots, T, \tag{1}$$

where $\mathbb{A}$ denotes the routing matrix, with rows corresponding to links and columns indicating routes between ToR switches. $A_{k\ell} = 1$ if the ℓ-th route traverses the k-th link; $A_{k\ell} = 0$ otherwise. In this paper, we aim to efficiently estimate the TM $\mathbb{X}$ using the load matrix $\mathbb{Y}$ derived from the easy-collected SNMP data. Our commonly used notions are listed in Table I, where we drop time indices for brevity.

Although (1) is a typical system of linear equations, it is impractical to solve it directly. On one hand, the traffic pattern in DCNs is practically sparse and skew [14]. Fig. 2, adopted from [14], plots the traffic normalized volumes between ToR switches in a DCN with 75 ToRs. The sparse and skew nature

[3] Involving time as another dimension of the TM was proposed earlier in [12], [13].

[4] We only consider intra-DCN traffic in this paper. However, our CREATE method can easily take care of DCN-Internet traffic by considering the Internet as a "special rack".

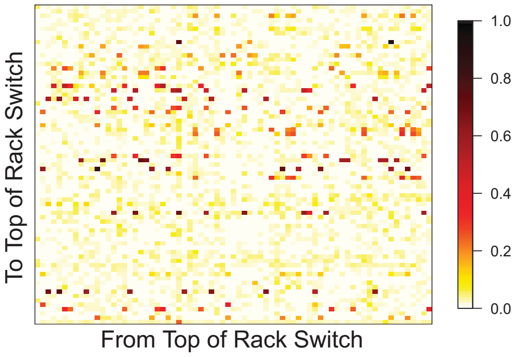


Fig. 2. The TM across ToR switches reported in [14].

of TM in DCNs can be immediately seen from the figure: only a few ToRs are hot and most of their traffic goes to a few other ToRs. On the other hand, as the number of unknown variables is much more than the number of observations, the problem is highly under-determined. For example in Fig. 1, the network consists of 8 ToR switches, 4 aggregation switches and 2 core switches. The number of possible routes in $\mathbb{A}$ is more than 100, while the number of link load observations is only 24. Even worse, the difference between these two numbers grows exponentially with the number of switches (i.e., the DCN scale). Consequently, directly applying tomographic methods to solve (1) would not work, and we need to derive a new method to handle TM estimation in DCNs.

IV. Design of CREATE

As directly applying network tomography to DCNs is infeasible due to the rich connections between different ToRs, we propose two procedures to pre-process the input data such that the tomographic methods can be applied. We shall first introduce the rationale of the pre-processing procedures, then we present our CREATE method that combines these two procedures with a fine-tuned tomographic algorithm.

A. Traffic Characteristics of DCNs

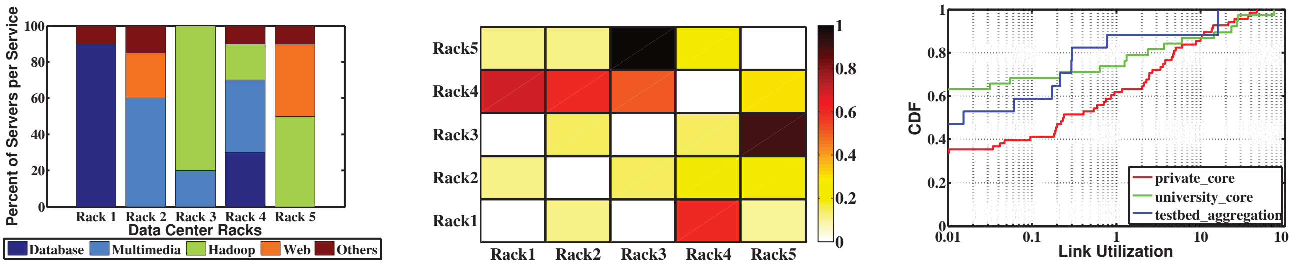
To motivate our design principles of CREATE, we focus on analyzing the traffic characteristics of real DCNs. As mentioned earlier, several proposals including [14], [22] have indicated that the TM of DCN ToRs is very sparse. More specifically, for each ToR in a DCN, it only exchanges data flows with a few other ToRs rather than most of them. For instance, in Fig. 2, we can see that each ToR is exchanging major flows with no more than 10 out of 74 other ToRs; the remaining ToR pairs either share very minor flows or not at all. If we could figure out the cause of this sparsity, we would be able to adjust the prior estimation of a TM to make tomographic algorithms work.

According to the literature, as well as our experience with our own data center, the sparse nature of TM in DCNs may originate from the correlation between traffic and service. In other words, racks running the same services have higher chances to exchange traffic flows, and the volumes of the flows may be inferred by the number of instances of the shared services. Bodík *et al.* [15] have analyzed a medium scale DCN and claimed that only 2% of distinct service pairs communicates with each other. Moreover, several proposals such as [23], [24] allocate almost all virtual machines of the same service under one aggregation switch to prevent traffic from going through oversubscribed network elements. Consequently, as each service may only be allocated to a few racks and the racks hosting the same services have a higher chance to communicate with each other, it naturally leads to sparse traffic matrices between DCN ToRs. To better illustrate this phenomenon in our DCN, we collect the socket level logs in each server to form the ground truth of the TM. We show the placement of services in 5 racks using the percentage of servers occupied by individual services in each rack in Fig. 3(a), and we depict the traffic volumes exchanged between these 5 racks in Fig. 3(b). Clearly, the racks that host more common services tend to exchange greater volumes of traffic (e.g., racks 3 and 5 whose more than 50% of the traffic flows are generated by the "Hadoop" service), whereas those do not share any common services rarely communicate (e.g., racks 1 and 3). So our first observation is the following:

> **Observation 1**: The TM between DCN ToRs is very sparse, but, fortunately, the pattern can be inferred by the service placements, thanks to the correlation between traffic and service.

Although using service placements can infer the skewness in the TM, the existence of multiple paths between every ToR pair still persists. Interestingly, literature does suggest that some of these routing paths can be removed to simplify the DCN topology by making use of link statistics. According to Benson *et al.* [16], the link utilizations in DCNs are rather low in general. They collect the link counts from 10 DCNs ranging from private DCNs, university DCNs to Cloud DCNs and reveal that about 60% of aggregation links and more than 40% of core links have low utilizations (e.g. in the level of 0.01%). To give more concrete examples, we retrieve the data sets publicized along with [16], as well as the statistics obtained from our DCN, then we draw the CDF of core/aggregation link utilizations in three DCNs for one representative interval selected from several hundred 5-minutes intervals in Fig. 3(c). As shown in the figure, more than 30% of the core links in a private DCN, 60% of core links in an university DCN and more than 45% of aggregation links in our testbed DCN only have the utilizations less than 0.01%.

Due to the low utilization of certain links, eliminating them will not affect much the estimation accuracy but will greatly reduce the possible routes between two racks. For instance, in an conventional DCN shown in Fig. 1, eliminating a core link will reduce 12.5% of the routes between any two ToRs, while cutting an aggregation link halves the outgoing paths from any ToR below it. Therefore, we may significantly reduce the number of potential routes between any two ToRs by eliminating the lowly utilized links. Although this may come at a cost of slightly losing actual flow counts, the



(a) Percentages of servers per service in our DCN. Only services in 5 racks are shown.

(b) The traffic volume from one rack (row) to another (column) with the service placement in (a).

(c) Link utilizations of three DCNs, with "private" and "university" from [16] and "testbed" being our own DCN.

Fig. 3. Characterizations of DCN traffics. (a) and (b) indicate service-traffic correlations, while (c) demonstrates low utilization of a large fraction of links.

overall estimation accuracy should be improved thanks to the eliminating of the ambiguity in the actual routing path taken by the major flows. So another of our observations is:

> **Observation 2**: Eliminating the lowly utilized links can greatly mitigate the under-determinism of our tomography problems in DCNs; it thus has the potential to increase the overall accuracy of the TM estimation.

B. *CREATE Architecture*

Based on these two observations, we design CREATE as a novel TM estimation method for DCNs. In a nutshell, we periodically compute the service correlations between different ToRs and eliminate lowly utilized links. This allows us to perform network tomography under a more accurate prior TM and a more determined system (with fewer routes). To the best of our knowledge, CREATE is the first practical algorithm for accurate traffic inference in DCN. As shown in Fig. 4, it takes three main steps to estimate the TM for DCN ToRs. First of all, CREATE calculates the correlation coefficient between different ToRs based on the service placement logs. Secondly, it eliminates the lowly utilized links to reduce redundant routes and narrow the space of potential TM suggested by the load vector $\mathbf{Y}(t)$. Finally, it takes the SNMP counters and the correlation coefficients as input to estimate the TM between different ToRs. We shall first give more details about these steps, and then present a working example.

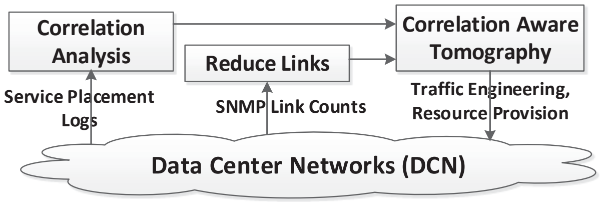

Correlation Analysis
Reduce Links
Correlation Aware Tomography
Service Placement Logs
SNMP Link Counts
Traffic Engineering, Resource Provision
Data Center Networks (DCN)

Fig. 4. The CREATE architecture.

1) Building Blocks of CREATE: The first step stems from **Observation 1**: we design a novel way to evaluate the correlation coefficient between two ToRs, leveraging on the easily obtained service placement logs. We use $Corr_{ij}$ to quantity the correlation between the i-th and j-th ToRs, and we calculate it as follows:

$$Corr_{ij} = \sum_{k=1}^{R} [(S_{ki} \times S_{kj})/(K_i \times K_j)] \quad i,j = 1,\cdots,N, \quad (2)$$

where the concerning quantities are derived from the service placement logs, as defined in Sec. III.

The second step is then motivated by **Observation 2**. We collect the SNMP link counts and compute the link utilization for each link. If the link utilization of a link is below a certain threshold θ, we consider the flow volumes of the routes that pass the link as zero, which effectively removes this link from the DCN topology. As a result, the number of variables in the equation system (1) can be substantially reduced, resulting in a more determined tomography problem. On one hand, this thresholding sets non-zero link counts to zero, possibly resulting in estimation errors. On the other hand, it removes redundant routes and mitigates the under-determinism of the tomography problem, potentially improving the estimation accuracy. In our experiments, we shall try different values of the threshold to see the trade-off between these two sides.

In the last step, we take the correlation coefficients and the reduced DCN topology as input to estimate the TM through a prior based tomography method. More specifically, we first compute $X_{i \leftrightarrow j}$ as the volume of traffic between ToR_i and ToR_j based on the correlations by the following procedure.

$$\begin{aligned} X_{i \rightharpoonup j} &= ToR_i^{out} \times \frac{Corr_{ij}}{\sum_{k=1}^{N} Corr_{ik}} & i,j = 1,\cdots,N, \\ X_{i \leftrightarrow j} &= X_{i \rightharpoonup j} + X_{j \rightharpoonup i} & i,j = 1,\cdots,N. \end{aligned}$$

Due to symmetry, $X_{i \rightharpoonup j}$ can also be computed through ToR_j^{in}. In order to compute the prior TM, we estimate the traffic volumes on each route by dividing the total number of bytes between two ToRs equally on every route connecting them. The reason for this equal share is the widely used ECMP [25] in DCNs; it by default selects routing paths between two switches with equal probability on each. The computed prior TM will give us a good start in solving a quadratic programming problem to determine the final estimation. We describe the detailed algorithm in Sec. IV-C.

As our TM estimation takes the time dimension into account (to cope with the volatile DCN traffics), one may wonder whether the correlation coefficients $[Corr_{ij}]$ have to be computed for each discrete time t. In fact, as it often takes a substantial amount of time for servers to accommodate new services, the service placements will not change frequently [15]. Therefore, once $[Corr_{ij}]$ are computed, they can be used for a certain period of time. Recomputing these coefficients are needed only when a new service is deployed or an existing service is quit. Even under those circumstances, we only need to re-compute the coefficients between the ToRs that are affected by service changes.

2) A Working Example: Fig. 5(a) presents an example to illustrate how CREATE works. The three colors represent three services deployed in the data center as follows:

- service_1: $\text{server}_2(\text{rack}_1), \text{server}_{12}(\text{rack}_6)$,
- service_2: $\text{server}_4(\text{rack}_2), \text{server}_6(\text{rack}_3), \text{server}_{13,14}(\text{rack}_7)$,
- service_3: $\text{server}_8(\text{rack}_4), \text{server}_{10}(\text{rack}_5)$.

The correlation coefficients between the ToR pairs are shown in Table II. Fig. 5(b) is the result of reducing lowly utilized links through thresholding, hence we can estimate the traffic volumes on the remaining paths from one ToR to another. More specifically, ToR_2 is related to ToR_3 and ToR_7 under a coefficient 0.25 and 0.5, respectively. So if ToR_2 totally sends out 10000 bytes during the 5 minutes interval, the traffic sent to ToR_3 and ToR_7 should be $10000 * 0.25/(0.25+0.5) = 3334$ and $10000 * 0.5/(0.25+0.5) = 6667$, respectively. After eliminating the lowly utilized links, there is only one route from ToR_2 to ToR_7. So the prior estimation of the traffic volume on that route is indeed 6667, the estimated traffic sent from ToR_2 to ToR_7. A similar situation applies to ToR_2 and ToR_3. The estimated prior TM is then fed to the final estimation, as discuss later in Sect. IV-C.

TABLE II
CORRELATION COEFFICIENTS OF THE WORKING EXAMPLE

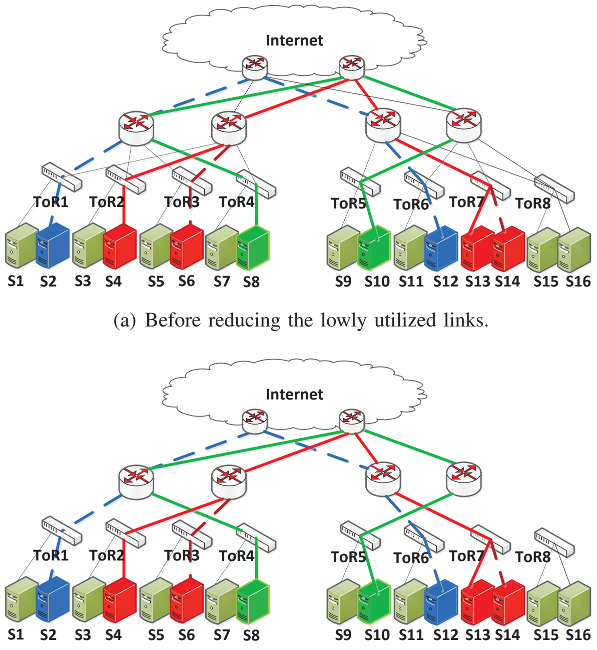
ToR Pairs	1:2-5	1:6	1:7,8	2:3	2:4-6	2:7	2:8	3:7	4:5
Corr. Coef.	0	0.25	0	0.25	0	0.5	0	0.5	0.25

C. The Algorithm Details

We use pseudocode to present our CREATE method in **Algorithm 1**. The algorithm takes the routing matrix $\mathbb{A}$, bandwidth vector $\mathbf{B}$, load vector $\mathbf{Y}$, service placement matrix $\mathbb{S}$, the ToR SNMP counts, and the link utilization threshold θ as the main inputs, and it returns the traffic vector $\mathbf{X}$. As the algorithm runs for every time t, we drop the time indices.

After computing the correlation coefficients in line 1, we remove the lowly utilized links and the related routes. In particular, we check every link if its utilization is below θ (lines 3) and we update $\mathcal{P}_{ij}$ (the set of routes between the i-th and j-th ToRs) by removing the routes that contain low utilized links (line 6); we also update $\mathbb{A}$, $\mathbf{X}$ and $\mathbf{Y}$ by removing the corresponding rows and components. Then we compute the prior traffic vector $\bar{\mathbf{X}}$ using ToR_i^{out}, ToR_j^{out} and the correlation coefficients. Lines 9–11 compute the volume of traffic exchanged between the i-th and j-th ToRs, and line 12 assigns the traffic to each routes between the two ToRs equally. Finally, the algorithm applies a quadratic programming to refine $\bar{\mathbf{X}}$ to obtain $\mathbf{X}$ subject to the constraints posed by $\mathbf{Y}$ and $\mathbb{A}$ (line 13).

Here we provide more details on the computation involved in QuadProgram. Basically, we want to obtain $\mathbf{X}$ that is closest to $\bar{\mathbf{X}}$ but satisfies the tomographic conditions. This



(a) Before reducing the lowly utilized links.

(b) After reducing the lowly utilized links

Fig. 5. Four different line styles represent four flows and three different colors represent three services.

Algorithm 1: CREATE Algorithm

```
   Input: $\mathbb{A}$, $\mathbf{B}$, $\mathbf{Y}$, $\mathbb{S}$, $\{ToR_i^{out} | i = 1, \cdots, N\}$, $\theta$
   Output: $\mathbf{X}$
1  $[Corr_{ij}] \leftarrow$ Correlation($\mathbb{S}$)
2  for $k = 1$ to $M$ do
3      if $Y_k/B_k \leq \theta$ then
4          forall the $r \in P_{ij}$ do
5              if $r$ contains $L_k$ then
6                  $\mathcal{P}_{ij} \leftarrow \mathcal{P}_{ij} - \{r\}$;  Adjust $\mathbb{A}$, $\mathbf{X}$ and $\mathbf{Y}$
7  for $i = 1$ to $N$ do
8      for $j = i + 1$ to $N$ do
9          $X_{i \rightharpoonup j} \leftarrow ToR_i^{out} * Corr_{ij}/(\sum_{1 \leq k \leq N} Corr_{ik})$
10         $X_{j \rightharpoonup i} \leftarrow ToR_j^{out} * Corr_{ij}/(\sum_{1 \leq k \leq N} Corr_{kj})$
11         $X_{i \leftrightarrow j} \leftarrow X_{i \rightharpoonup j} + X_{j \rightharpoonup i}$
12         forall the $r \in \mathcal{P}_{ij}$ do  $\bar{X}_r \leftarrow X_{i \leftrightarrow j}/|\mathcal{P}_{ij}|$ ;
13 $\mathbf{X} \leftarrow$ QuadProgram($\mathbb{A}, \bar{\mathbf{X}}, \mathbf{Y}$)
14 return $\mathbf{X}$
```

problem can be formulated as follows:

$$\begin{aligned}\text{Minimize} \qquad & \|\mathbf{X}-\bar{\mathbf{X}}\| + \|\mathbb{A}\mathbf{X}-\mathbf{Y}\| \\ \text{s.t.} \qquad & \|\mathbb{A}\mathbf{X}-\mathbf{Y}\| \geq 0\end{aligned}$$

where $\|\cdot\|$ is L_2-norm of a vector. To tackle this problem, we first compute the deviation $\tilde{\mathbf{Y}} = \mathbf{Y} - \mathbb{A}\bar{\mathbf{X}}$, then we solve the following constrained least square problem to obtain the $\tilde{\mathbf{X}}$ as the adjustments to $\bar{\mathbf{X}}$ for offsetting the deviation $\tilde{\mathbf{Y}}$.

$$\begin{aligned}\text{Minimize} \qquad & \|\mathbb{A}\tilde{\mathbf{X}} - \tilde{\mathbf{Y}}\| \\ \text{s.t.} \qquad & \mu\tilde{\mathbf{X}} \geq -\bar{\mathbf{X}}\end{aligned} \tag{3}$$

We use a tunable parameter μ, $0 \leq \mu \leq 1$ to make the tradeoff between the similarity to the prior solution and the precise fit to the link loads. The constraint is meant to guarantee a non-negative final estimation $\hat{\mathbf{X}}$. Finally, $\hat{\mathbf{X}}$ is obtained by making a tradeoff between the prior and the tomographic constraint as $\hat{\mathbf{X}} = \bar{\mathbf{X}} + \mu\tilde{\mathbf{X}}$. According to our experience, we take $\mu = 0.8$ to give a slightly more bias towards the prior.

Obviously, The dominant running time of the CREATE algorithm is spent on $\mathsf{QuadProgram}(\mathbb{A}, \bar{\mathbf{X}}, \mathbf{Y})$, whose main component (3) is equivalent to a *non-negative least squares* (NNLS) problem. The complexity of solving this NNLS is $\mathcal{O}(M^2 + P^2)$, but can be reduced to $\mathcal{O}(P \log M)$ though parallel computing in a multi-core system [26].

V. Evaluation

We evaluate our CREATE both in a testbed and by simulations in this section.

A. *Experiment Settings*

We implement CREATE together with two representative TM inference algorithms:

- Tomogravity [11] is known as a classical TM estimation algorithm that performs well in IP networks. In contrast to CREATE, it assumes traffic flows in the networks follow the gravity traffic model, that traffic exchanged by two ends is proportional to the total traffic on the two ends.
- Sparsity Regularized Matrix Factorization (SRMF for short) [12] is a state-of-art traffic estimation algorithm. It leverages the spatio-temporal structure of traffic flows, and utilizes the compressive sensing method to infer TM by rank minimization.

These algorithms serve as benchmarks to evaluate the performance of CREATE under different network settings.

We quantify the performance of the three algorithms using four metrics: *Relative Error* (RE), *Root Mean Squared Error* (RMSE),*Root Mean Squared Relative Error* (RMSRE) and the computing time. RE is defined for individual elements as:

$$\mathrm{RE}_i = |X_i - \hat{X}_i| / X_i, \tag{4}$$

where X_i denotes the true TM element and $\hat{X}_i$ is the corresponding estimated value. RMSE and RMSRE are metrics to evaluate the overall estimation errors:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_x}\sum_{i=1}^{N_x}\left(X_i - \hat{X}_i\right)^2}, \tag{5}$$

$$\mathrm{RMSRE}(\tau) = \sqrt{\frac{1}{N_\tau}\sum_{i=1, X_i > \tau}^{N_x}\left(\frac{X_i - \hat{X}_i}{X_i}\right)^2}. \tag{6}$$

Similar to [11], we use a τ to pick up the relative large traffic flows since small ones may not be important for engineering DCNs. N_x is the number of elements in the ground truth X and N_τ is the number of elements $X_i > \tau$.

B. *Testbed Evaluation*

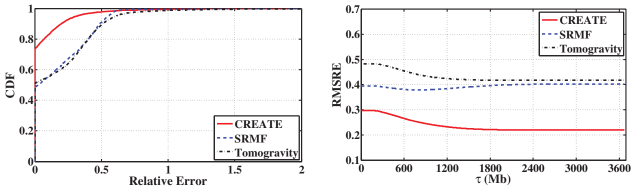
1) Testbed Setup: We use a testbed with 10 switches and about 300 servers as shown in Fig. 6 for our experiments. And the architecture for this testbed DCN is a conventional tree similar to Fig. 1. The testbed hosts a variety of services and part of which has been shown in Fig. 3(a). We gather the service placement logs and SNMP link counts for all switches. We also record the flows exchanged between servers by setting linux iptable rules in each server (not a scalable approach) to form the ground truth of TM between ToRs. The data are all collected every 5 minutes.

(a) The outside view of our DCN.

(b) The inside view of our DCN.

Fig. 6. Hardware testbed with 10 racks and more than 300 servers.

2) Testbed Results: Fig. 7(a) plots the CDF of REs of the three algorithms. Clearly, CREATE performs significantly better than another two: it can accurately estimate the volumes of more than 78% traffic flows. As the TM of our DCN may not be of low rank, SRMF performs similarly to tomogravity.

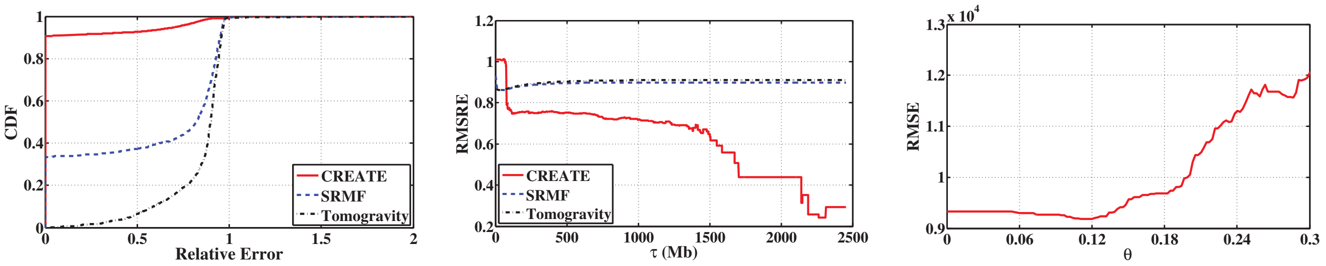


(a) The CDF of RE.

(b) The RMSRE under different τ

Fig. 7. The CDF of RE and RMSRE of the three algorithms on testbed.

We then study these algorithms with respect to the RMSREs in Fig. 7(b). It is natural to see the RMSREs of all three algorithms are non-increasing in τ, because estimation algorithms are all subject to noise for the light traffic flows, but they normally performs better for heavy traffic flows. However, CREATE still achieves the lowest RMSRE for all values of τ among the three. As our experiments with real DCN traffic are confined by the scale of our testbed, we will conduct more experiments with larger DCNs through simulations in *ns*-3.

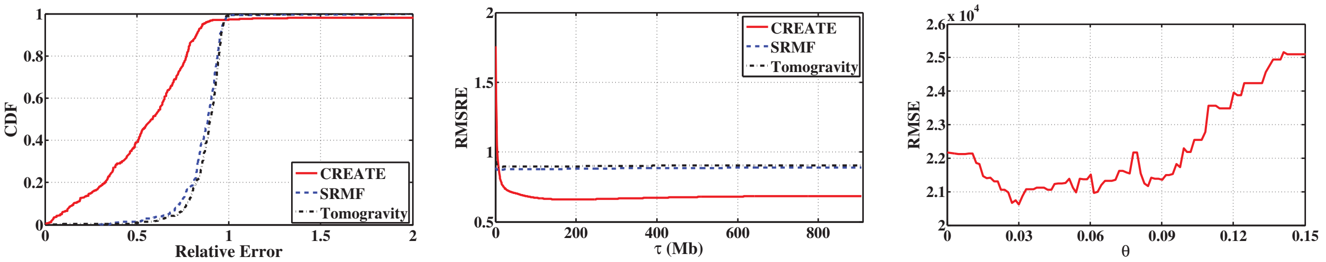


(a) The CDF of RE

(b) The RMSRE under different τ

(c) The RMSE under different θ.

Fig. 8. The CDF of RE (a), the RMSRE (b), and the RMSE (c) of the three algorithms for estimating TM under conventional tree architecture.



(a) The CDF of RE

(b) The RMSRE under different τ

(c) The RMSE under different θ.

Fig. 9. The CDF of RE (a), the RMSRE (b), and the RMSE (c) of the three algorithms for estimating TM under fat-tree architecture.

C. Simulation Evaluations

1) Simulation Setup: We adopt both the conventional data center architecture [21] and fat-tree architecture [1] as our experimental topologies. For the conventional tree, there are 32 ToR switches with 20 servers in per rack, 16 aggregation switches, and 3 core switches. And for fat-tree, we use $k = 8$ levels with the same number of ToR switches as the conventional tree, but with 32 aggregation switches, 16 core switches and 4 servers per rack. The link capacities are all set to be 1Gbps. We could not conduct simulations on BCube [3] because it does not arrange servers into racks. It would be an interesting problem to study how to extend our proposal for estimating the TM for servers in BCube.

We install both on-off and bulk-send applications in *ns*-3. We randomly deploy services in a DCN. The packet size is set to be 1400 bytes, and the flow sizes are randomly generated but following the characteristics of real DCNs [6], [10], [16]. For instance, 10% of the flows contributes to about 90% of the total traffic in DCN [2], [4]. We use TCP flows in our simulations [27], and apply the widely used ECMP [25] as the routing protocol.

We record the total number of bytes and packets that enter and leave every port of each switch in the network every 5 minutes. We also record the total bytes and packets of flows on each route in the corresponding time periods as the ground truth. For every setting we run simulations for 10 times.

To evaluate the computing time, we measure the time period starting from when we input the topologies and link counts to the algorithm until the time when all TM elements are returned. All three algorithms are implemented by Matlab (R2012b) on 6-core Intel Xeon CPU @3.20GHz, with 16GB of memory and the Windows 7 64-bit OS.

2) Simulation Results: Fig. 8(a) compares the CDF of REs of the three algorithms under conventional tree architecture and we set $\theta = 0.001$. The advantage of CREATE over the other two algorithms stems from the fact that CREATE can clearly find out the ToR pairs that do not communicate with each other. Tomogravity has the worst performance because it gives each ToR pair a communication traffic whenever one of them has “out” traffic and the other has “in” traffic, thus introducing non-existing positive TM entries. SRMF obtains the TM by rank minimization, so it performs better than tomogravity when our random traffic does lead to low rank TM. The worse performance of SRMF (compared with CREATE) may be its over-fitting of the sparsity in eigenvalues, according to [10].

We study the RMSREs of the three algorithms under different τ in Fig. 8(b). Again, CREATE exhibits the lowest RMSRE and a (expectable) reducing trend with the increasing of the τ, while the other two remain almost constant in τ. In Fig. 8(c), we then study how the RMSE changes with the threshold θ of link utilizations, which is an important parameter to fine-tune the performance of CREATE. As we can see in this figure, when we gradually increase the threshold, RMSE does slightly decrease until the sweet point $\theta = 0.12$. While the improvement on accuracy may be minor, the computing time can be substantially reduced as we will show later.

Fig. 9 evaluates the same quantities as Fig. 8 but under fat-tree architecture, which has even more redundant routes. We set $\theta = 0.001$. Since TM in fat-tree DCNs is far more

TABLE III
THE COMPUTING TIME (SECONDS) OF THE THREE ALGORITHMS UNDER DIFFERENT SCALES OF DCNS

Switches	Links	Routes	Computing Time			
			CREATE		Tomo-gravity	SRMF
			θ =0.001	θ =0.01		
51	256	7360	0.54	0.51	2.54	1168.22
102	320	46272	8.12	7.81	73.59	-
204	1024	381312	813.23	614.67	1654.46	-

sparse, the errors are evaluated only against the non-zero elements in TM. In general, CREATE retains its superiority over others in both RE and RMSRE. The effect of θ becomes more interesting in Fig. 9(c) (compared with Fig. 8(c)); it clearly shows a "valley" in the curve and a sweet point around $\theta = 0.03$. This is indeed the trade-off effect of θ mentioned in Sec. IV-B1: it trades the estimation accuracy of light flows for that of heavy flows.

Tab. III lists the computing time of the three algorithms under conventional tree architecture. Obviously, CREATE performs much faster than both tomogravity and SRMF. While both CREATE and tomogravity have their computing time grow quadratically with the scale of the DCNs, SRMF often cannot deliver a result within a reasonable time scale. In fact, if we slightly increase θ, we may further reduce the computing time, as shown in Tab. III. In summary, our algorithm both has a higher accuracy and faster running speed compared to the two state-of-art algorithms.

VI. CONCLUSION

To meet the increasing demands for detailed traffic characteristics in DCNs, we make the first step towards estimating the traffic matrix (TM) between all ToR switches in a DCN, relying on only the easily accessible SNMP counters and the service placement logs. We pioneer in applying tomographic methods to DCNs by overcoming the barriers of solving the ill-posed linear system in DCN for TM estimation. We first obtain two major observations on the rich statistics of DCNs traffic. The first observation reveals that the TMs between ToRs of DCNs are extremely sparse, and the traffic patterns can be roughly inferred from service placement logs. The other observation argues that eliminating a part of links with low utilization can greatly increase both overall accuracy and the efficiency of TM estimation. Based on these two observations, we develop a new TM estimation method CREATE which is applicable to most prevailing DCN architectures without any additional infrastructure support. We validate CREATE with both hardware testbed and simulations, and the results show that CREATE outperforms the two well-known TM estimation methods on both accuracy and efficiency.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proc. of ACM SIGCOMM*, 2008, pp. 63–74.

[2] 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 *Proc. of ACM SIGCOMM*, 2009, pp. 51–62.

[3] 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 *Proc. of ACM SIGCOMM*, 2009, pp. 63–74.

[4] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks," in *Proc. of USENIX NSDI*, 2010.

[5] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM Placement and Routing for Data Center Traffic Engineering," in *Proc. of IEEE INFOCOM*, 2012, pp. 2876–2880.

[6] P. Gill, N. Jain, and N. Nagappan, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications," in *Proc. of ACM SIGCOMM*, 2011, pp. 350–361.

[7] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead Datacenter Traffic Management Using End-host-based Elephant Detection," in *Proc. of IEEE INFOCOM*, 2011, pp. 1629–1637.

[8] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis," in *Proc. of ACM SoCC*, 2012, pp. 7:1–7:13.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 69–74, 2008.

[10] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The Nature of Data Center Traffic: Measurements & Analysis," in *Proc. of ACM IMC*, 2009, pp. 202–208.

[11] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast Accurate Computation of Large-scale IP Traffic Matrices from Link Loads," in *Proc. of ACM SIGMETRICS*, 2003, pp. 206–217.

[12] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spatio-temporal Compressive Sensing and Internet Traffic Matrices," in *Proc. of ACM SIGCOMM*, 2009, pp. 267–278.

[13] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot, "Traffic Matrices: Balancing Measurements, Inference and Modeling," in *Proc. of ACM SIGMETRICS*, 2005, pp. 362–373.

[14] K. Srikanth, P. Jitendra, and B. Paramvir, "Flyways To De-Congest Data Center Networks," in *Proc. of ACM HotNets*, 2009.

[15] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, "Surviving Failures in Bandwidth-Constrained Datacenters," in *Proc. of ACM SIGCOMM*, 2012, pp. 431–442.

[16] T. Benson, A. Akella, and D. A. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *Proc. of ACM IMC*, 2010, pp. 267–280.

[17] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying NOX to the Datacenter," in *Proc. of HotNets*, 2009.

[18] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine Grained Traffic Engineering for Data Centers," in *Proc. of ACM CoNEXT*, 2011, pp. 8:1–8:12.

[19] Y. Qiao, Z. Hu, and J. Luo, "Efficient Traffic Matrix Estimation for Data Center Networks," in *Proc. of IFIP Networing*, 2013, pp. 1–9.

[20] Z. Hu, Y. Qiao, and J. Luo, "Coarse-Grained Traffic Matrix Estimation for Data Center Networks," *Computer Communications*, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2014.02.016

[21] C. D. C. Infrastructure, "2.5 Design Guide," 2007. [Online]. Available: http://goo.gl/kBpzgh

[22] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall, "Augmenting Data Center Networks with Multi-Gigabit Wireless Links," in *Proc. of ACM SIGCOMM*, 2011, pp. 38–49.

[23] H. Ballani, P. Costa, T. Karagiannis, and A. I. Rowstron, "Towards Predictable Datacenter Networks." in *Proc. of ACM SIGCOMM*, 2011, pp. 242–253.

[24] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees," in *Proc. of ACM Co-NEXT*. ACM, 2010, pp. 15:1–15:12.

[25] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," United States, 2000.

[26] Y. Luo and R. Duraiswami, "Efficient Paraller Non-Negative Least Square on Multi-core Architectures," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2848–2863, 2011.

[27] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center Tcp (DCTCP)," in *Proc. of ACM SIGCOMM*, 2010, pp. 63–74.