Cracking Network Monitoring in DCNs with SDN

Zhiming Hu and Jun Luo
School of Computer Engineering, Nanyang Technological University, Singapore
Email: {zhu007, junluo}@ntu.edu.sg

***Abstract*—The outputs of network monitoring such as *traffic matrix* and *elephant flow identification* are essential inputs to many network operations and system designs in DCNs, but most solutions for network monitoring adopt direct measurements or inference *alone*, which may suffer from either high network overhead or low precision. Different from those approaches, we combine the direct measurements offered by *software defined network* (SDN) and inference techniques based on *network tomography* to derive a hybrid network monitoring scheme in this paper; it can strike a balance between measurement overhead and accuracy. Essentially, we use SDN to make the severely low determined network tomography (TM estimation) problem in DCNs to be a more determined one. Thus many classic network tomography algorithms in ISP networks become feasible for DCNs. By combining SDN with network tomography, we can also identify the elephant flows with high precision while occupying very little network resource. According to our experiment results, the accuracy of estimating the TM is far higher than those inferred by SNMP link counters only and the performance of identifying elephant flows is also very promising.**

I. Introduction

Network monitoring tasks such as *traffic matrix* (TM) estimation and *elephant flow detection* in *data center networks* (DCNs) are very important for both data center operators and researchers in this area. On one hand, for the data center operators, TM estimation and elephant flow detection are vital inputs for many network operations for example, traffic engineering [1], scheduling in wireless DCNs [2], [3], as well as trouble shooting [4]. On the other hand, TMs and elephant flow distributions reveal the traffic characteristics in DCNs, which help researchers design and evaluate proposals made for DCNs. For instance, without good knowledge of the traffic characteristics in DCNs, researchers may have to use the traffic characteristics in ISP networks or random TM for evaluations, which are unfair and may generate misleading results.

Most prior work adopt direct measurements for TM estimation or elephant flow detection in DCNs. These methods can be roughly divided into two groups. The first group is the switch-based approaches, which utilize sflow [5], netflow [6] or openflow [7] enabled switches to record flows that traverse a switch. They both record flows that pass through a switch and offer the input for analysis of the traffic, but consume a lot of network resource in storage and processing when traffic is high. The second group of methods are server-based [8], [9]. They commonly modify the OS or hypervisor to support data collection in each server, which generate a petabytes of uncompressed data per month as reported in [8]. And it needs extra bandwidth to collect all the data in all the servers. So they are more suitable for inspecting individual servers other than monitoring the whole DCN.

So, why not try some inference technologies such as *network tomography* to decrease the measurement overhead? Having been widely investigated in ISP networks [10]–[12], network tomography estimates the TM from the ubiquitous link counters in the network; it would be very convenient if we can adapt network tomographic methods in DCNs and apply those state-of-art algorithms. Unfortunately, due to the rich connections between the switches in DCNs, the number of end-to-end paths are far more than the number of links, which makes the network tomographic problem much more under-constrained than the case in ISP networks. Inference-based methods also find difficulty to accurately identify the elephant flows because inference results are not reliable and errors may not be well controlled.

As network tomography alone can hardly offer accurate TM estimation and elephant flow detection, and it is also too costly to instrument an entire data center for direct measurements [13], a better approach should be combining these two kinds of technologies to reach a balance between measurement overhead and accuracy [14]. Different from sflow [5] and netflow [6], *software defined network* (SDN) is an emerging paradigm that separates the control plane from the data plane in the traditional switches. Thus it introduces the centralized controls and programmability on those network devices. Due to its programmability, we can flexibly set up the SDN rules to provide more measurements for network tomography instead of monitoring every flow. With these extra measurements, network tomography algorithm may achieve promising results close to direct measurements.

In this work, we first propose to "reshape" the TM estimation problem in DCNs using SDN rules. To this end, we formulate and solve an optimization problem aiming to maximize contribution of SDN rules in improving upon the under-constrained nature of network tomography. As solving this optimization problem can be time-consuming, we also propose a practical solution to set up SDN rules. Furthermore, we apply network tomographic ideas to identifying the elephant flows, again assisted by the SDN-enabled measurements. Our main contributions in this paper are:

- We formulate a new optimization problem to maximize the identifiability [15] of a network tomography problem in DCNs given a certain number of SDN rules. As not all SDN rules may contribute to improve the identifiability, this optimization problem help to carefully design the *aggregation matrix*, which defines which part of paths to be aggregated to an SDN rule, to maximize the value of

SDN rules to a network tomographic problem.

- We propose two systematic and a fast practical solutions for constructing the SDN aggregation matrix. We first solve the problem directly with two systematic approaches. We then design a topology-aware heuristic algorithm to adaptively add SDN rules that improve the identifiability of the problem.
- We combine network tomography and SDN to identify the elephant flows. We first use SDN-enhanced network tomography to find out the "talky" *top-of-rack* (ToR) pairs, which substantially narrows the solution space. We then use network tomographic approach to locate server-to-server (potential) elephant flows in these ToR pairs.
- We validate our approach through both experiments in a data center testbed and trace driven network simulations in *ns*-3: the performance of both our TM estimation and elephant flow detection are evaluated against the state-of-art proposals.

The reminder of this paper is organized as follows. We introduce the related work in Sec. II and problem formulation in Sec. III. The design of our system was presented in Sec. IV, followed by the evaluations of our proposal in Sec. V. Finally, we conclude our work in Sec. VI.

II. Related Work

In this section, we briefly discuss the most related proposals, namely those about direct measurements in DCNs and network inference techniques in ISP networks. Some other recent works that utilize SDN for control or measurement can be found in [16]–[18].

For direct measurements in DCNs, it can be achieved by instrumenting the switches or servers. On one hand, in [1], they utilize the openflow enabled switches to record the flows in the DCNs and identify the elephant flows. It sets up a new TCAM rule for every new flow in the network. However, it may suffer from short of TCAM rules in DCNs where hundreds of thousands of new flows are generated every second [19] in each rack. On the other hand, some proposals [9], [19] monitor the network by instrumenting the servers. In [19], they install a shim layer in each server to support identification of elephant flows. While in [9], the socket lever logs are recorded and analyzed to get the traffic characteristics in the network.

Network tomography [10]–[12] attracts a lot of attentions in ISP networks. In [10], it proposes a prior-based network tomography approach. They adopt gravity model to get the prior TM and formulate a least square problem to get the final estimation. A compressive sensing based methods is presented in [11], which utilizes the spatial and temporal characteristics of TM and low rank optimization to get their estimations. In [12], Kalman filter is used for modeling the traffic in continuous time slices and predicting the traffic in the next time slice. However, these proposals cannot be adapted in DCNs directly due to much more severe under-constraint situation in DCNs [8].

The most related work is iSTAMP [20], which leverages part of the TCAM rules for aggregation measurements and another part of TCAM rules for per-flow measurements. It focuses on building the observation matrix for compressive sensing network tomography and may encounter the aggregation feasibility problem in DCNs as not all the flows can be aggregated. We, on the contrary, tackle the basic under-constraint network tomography problem in DCNs with SDN, so that traditional network tomographic strategies can be applied. We take the feasibility of aggregation as a main factor to be considered when using SDN. We also propose a new practical elephant flow detection method powered by SDN and network tomography.

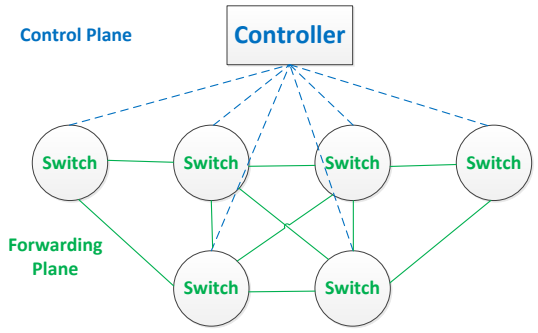


Fig. 1. SDN divides the control plane from the data plane.

III. Background and Problem Formulation

In this section, we first explain why we combine the traditional network tomography with SDN. Then we formally define our problem.

A. SDN and Network Tomography in DCNs

SDN is a fast evolving network technology and it divides the control plane from the data plane as we can see in Fig 1. As a result, new network operation or management applications can be deployed much easier in the networks without violating the previous protocols and applications.

If the traffic is not very heavy in the network, SDN is capable of setting one rule for each flow and counting its bytes. Unfortunately, this is not a practical way to measure flows in DCNs due to the huge cost it incurs. On one hand, there is not enough TCAM space in each SDN switch for now given the large (median) number of flows generated during a certain interval in DCNs [19]. On the other hand, hundreds of controllers are needed for this measurement task, as per new flow in the SDN switch will trigger a "Packet in" message and send that message to the controller of SDN [21].

Different from direct measurement offered by the SDN, network tomography [10]–[12] tries to infer the flow data through the widely available SNMP link counters. Network tomography is a mature practice in ISP networks, but it may not be directly adapted in DCNs for the dense connections between the network devices in DCNs. In other words, in DCNs such as fat-tree in Fig 2, there are much more end-to-end paths (variables) than links (available measurements). Thus the available measurements do not contain sufficient information to result in an accurate estimation. For almost the

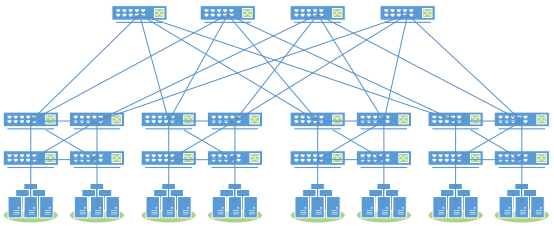


Fig. 2. The simple fat-tree topology (k=4).

same reasons, it is also infeasible for identifying the elephant flows in DCNs with network tomographic techniques alone.

Fortunately, SDN and network tomography may complement each other in solving network monitoring for DCNs. On one hand, setting up a few SDN rules to collect direct measurements can significantly improve the identifiability network tomography. On the other hand, applying network tomography to infer network traffics (rather than directly measure them) can avoid the bottleneck imposed by the network resources demanded by SDN. Therefore, combining these two techniques should definitely benefit network monitoring.

B. System Model and Problem Formulation

We consider a tree-like architecture as shown in Fig. 2, which is sufficiently representative for nowadays DCN topology. Let $\mathbf{x}(t) = \{x_1(t) \cdots x_n(t)\}$ denotes the volume of traffic on all the n paths among all the ToRs at time slot t, so the *TM* $X = \{\mathbf{x}(1) \cdots \mathbf{x}(\tau)\}$ represents the traffic in DCNs from slot 1 to τ, with rows corresponding to the indexes of pathes and columns corresponding to time slots. Here we refer to a *flow* as the volume of traffic on a certain path between two ToRs during time slot t.

In network tomography, X is what we want and the available measurements are link counts $\mathbf{y}(t) = \{y_1(t) \cdots y_m(t)\}$ on m links, respectively. The SNMP link counts can be polled from the link counters in the switches through SNMP protocols. Then we take $Y = \{\mathbf{y}(1) \cdots \mathbf{y}(\tau)\}$ to denote the *link count matrix*. The relationship between the unknown flow data X and available measurements Y in network tomography can be reflected as follows:

$$AX = Y, \tag{1}$$

where $A = \{a_{ij} | i = 1 \cdots m, j = 1 \cdots n\}$ denotes the *routing matrix*, which indicates whether a path traverses a link in the topology. For instance, $a_{ij} = 1$ if the j-th path contains i-th link, and $a_{ij} = 0$ otherwise.

Though network tomography often entails solving an under-constrained linear equation system, the under-constrained situation gets severely exacerbated in DCNs: the number of paths is always much larger than the number of links in different scales of DCNs. While the deteriorated situation hurts the identifiability of TM estimation, we propose to use SDN rules to improve the identifiability. By setting the wildcard rules in DCNs, openflow can record the volume of aggregate traffic on multiple paths. Let $B_{r \times n}$ denote the *aggregation matrix* indicating if the traffic through a certain path is measured by a TCAM rule. More specifically, $b_{ij} = 1$, means that the volume of traffic on j-th path during a certain interval will be aggregated on i-th TCAM rule, and $b_{ij} = 0$ otherwise.

The new network tomography problem using both SNMP link counts and openflow wildcard rule counters can be formulated as follows:

$$\begin{pmatrix} A \\ B \end{pmatrix} * X = D * X = \begin{pmatrix} Y \\ Q \end{pmatrix}, \tag{2}$$

where $Q_{r \times \tau}$ can be obtained through the SDN counters.

As we can see in Eqn.(2), maximizing the rank of the *measurement matrix* D given A implies that B delivers new measurements linearly independent of the set of available measurements, so it would potentially lead to higher identifiability and estimation accuracy. So how to determine the aggregation matrix B in order to to maximize the rank of D is the **first problem** we address in this paper. To determine the optimal aggregation matrix B given a routing matrix A, we formulate the problem as follows:

$$\begin{aligned}
\text{maximize} \quad & Rank(D) && (3)\\
\text{s.t.} \quad & d_{ij} = a_{ij} \quad \forall i = 1 \cdots m, j = 1 \cdots n && (4)\\
& \sum_{i=m+1}^{m+r} d_{ij} \leq 1 \quad \forall j = 1 \cdots n && (5)\\
& \lambda_k \in \{0, 1\} \quad \forall k = 1 \cdots n_a && (6)\\
& \mathbf{d}_i = \lambda_k \times \mathbf{c}_k \ \forall i = m+1 \cdots m+r, \\
& \qquad \forall k = 1 \cdots n_a && (7)\\
& \sum_{k=1}^{n_a} \lambda_k \leq r && (8)
\end{aligned}$$

The objective function is to maximize the rank of measurement matrix D, thus maximize the identifiability of network tomography problem in DCNs. The first constraint (4) indicates that the first m rows of D is the same with the rows of A. The number of times each flow can be aggregated by all the TCAM rules is refined in (5). Given the current practice of TCAM, each flow can be mapped into only one TCAM entry [20]. We introduce binary variables λ in (6) to denote whether to select the rows in $C_{n_a \times n}$ as part of D or not, where C is a binary matrix whose rows indicate all feasible aggregations of paths in the DCNs. If $\lambda_k = 1$, then the k-th row of C is attached to D, and $\lambda_k = 0$ otherwise as shown in (7). Finally, the number of TCAM rules allocated for TM estimation is constrained in (8) by an upper bound r.

Our **second problem** is to identify the *elephant flows* with limited SDN rules in DCNs, where an elephant flow is the flow that occupies more than 10% of the link capacity during a certain interval. Here we refer to a *flow* as the total volume of traffic from one server to another. In particular, if the number of SDN rules for elephant flow detection is n_r and the number of elephant flow is n_e, we often have $n_r \leq n_e$. Therefore, we aim to assign SDN rules to the flows with higher possibilities to be an elephant flow.

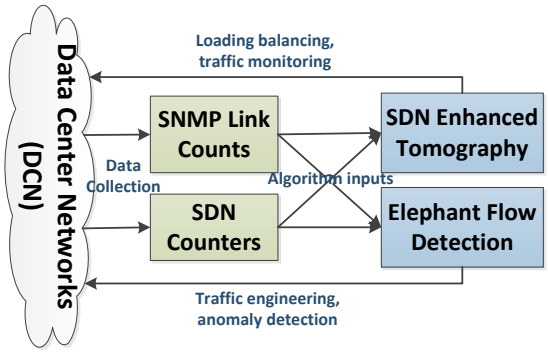


Fig. 3. System Design.

IV. SDN Enhanced Network Tomography and Elephant Flow Detection

In this section, we first present the system overview of our traffic monitoring system in DCNs. We also explain how we get the optimal measurement matrix D to improve the performance of network tomography. We then discuss how to effectively locate the elephant flows in DCNs.

A. System Overview

Our system is shown in Fig. 3. We collect the SNMP link counts, which are widely available in the network. We also set up the SDN rules properly and collect the SDN counters. After that we can utilize the data for two sorts of network measurement tasks. Firstly, we propose to combine SNMP link counts and SDN counters to maximize the performance of network tomography. We formulate an optimization problem and propose two systematic algorithms to get the measurement matrix D, so as to improve the identifiability of our network tomography problem. We also propose a topology-aware heuristic algorithm to efficiently compute the measurement matrix D in practice. Secondly, we use SDN counters and SNMP link counts to identify the elephant flows among servers. At the first stage, the total volume of traffic among ToRs (through SDN-enhanced network tomography) help us to find out the "talky" ToR pairs (the ToR pairs with more interchange traffic). At the second stage, we can further take use of SNMP link counts and network tomographic strategies to identify the elephant flows among servers.

B. Network Tomography with SDN

1) Systematic Measurement Matrix Design: It is well known that maximum rank matrix completion is a fairly complicated optimization problem [22], so we choose to handle the rank maximization problem through the widely used norm minimization approach in compressive sensing [23]. The idea is to minimize the sum of the squared inner products of columns in D. More specifically, we seek to minimize $\|D^TD - I\|_F^2$ to generate more orthogonal columns [23], thus can increase the rank of D with high probability and in turn potentially improve the performance of network tomography.

As $\|D\|_F^2 = \mathrm{tr}(D^TD)$, it is straightforward to see that minimizing $\|D^TD - \zeta\|_F^2$ is equivalent to minimizing $\mathrm{tr}[(D^TD - \zeta)^2]$, where we use ζ because we do not normalize the binary matrix D as in [23]. It can then be expanded and simplified in a way similar to [20]:[1]

$$\begin{aligned}\underset{D}{\text{minimize}} \quad & \sum_{i=1}^{m+r}\sum_{j=1}^{n}(1-2\zeta_{jj})d_{ij} \\ & + \sum_{i=m+1}^{m+r}\sum_{j=1}^{n}\sum_{p=1,p\neq j}^{n} d_{ij}d_{ip} \\ & + \sum_{i=1}^{m+r}\sum_{j=1}^{n}\sum_{p=1,p\neq i}^{m+r} d_{ij}d_{pj} \\ & + \sum_{i=1}^{m+r}\sum_{j=1,j\neq i}^{m+r}\sum_{p=1}^{n}\sum_{q=1,q\neq p}^{n} d_{ip}d_{iq}d_{jp}d_{jq} \\ & + \sum_{j=1}^{n}\zeta_{jj}^2 \qquad (9) \\ \text{s.t.} \quad & (4)-(8)\end{aligned}$$

As the above program is not linear, we can further transfer it to be an *integer linear program* (ILP) by introducing new variables. Given binary variables x, y, z, $z = xy$ implies the following three constraints: $z \le x, z \le y, z \ge (x + y - 1)$. Hence we take $\mathbf{u}$ to denote the elements in $D(1 : m+r, :) = \mathbf{u} = [u_1, \cdots, u_{(m+r)n}]$, $\mathbf{s} = [s_1, \cdots, s_{n_s}]$ for the multiplication of two elements, $\mathbf{t} = [t_1, \cdots, t_{n_t}]$ for multiplication of four elements, where $n_s = rn(n-1) + (m+r)n(m+r-1)$ and $n_t = (m+r)(m+r-1)n(n-1)$. So we have:

$$\begin{aligned}\underset{\mathbf{u},\mathbf{s},\mathbf{t}}{\text{minimize}} \quad & \sum_{i=1}^{m+r}\sum_{j=1}^{n}(1-2\zeta_{jj})u_{((i-1)n+j)} \\ & + \sum_{j=1}^{n}\zeta_{jj}^2 + \sum_{i=1}^{n_s}s_i + \sum_{i=1}^{n_t}t_i \qquad (10) \\ \text{s.t.} \quad & s_i \le u_j, s_i \le u_k, \\ & s_i \ge 0, s_i \ge u_j + u_k - 1 \qquad (11) \\ & t_i \le u_j, \\ & t_i \le u_k, t_i \le u_p, t_i \le u_q, \\ & t_i \ge 0, \\ & t_i \ge u_j + u_k + u_p + u_q - 3 \qquad (12) \\ & \sum_{i=m+1}^{m+r} u_{(i-1)*n+j} \le 1 \quad \forall j = 1 \cdots n \qquad (13) \\ & u_{(i-1)*n+j} = a_{ij} \quad \forall i = 1 \cdots m \ \forall j = 1 \cdots n \\ & u_{(i-1)*n+j} = \lambda_k \times c_{kj} \ \forall i = m+1 \cdots m+r, \\ & \qquad \forall j = 1 \cdots n \ \forall k = 1 \cdots n_a \qquad (14) \\ & (6) \textit{ and } (8)\end{aligned}$$

After we transform the non-linear problem in (9) into an ILP in (10) by setting $s_i = u_ju_k$ and $t_i = u_ju_ku_pu_q$ and introducing the constraints in (11) and (12), respectively. We

[1]Whereas we aim to optimize the aggregation matrix B given the routing matrix A, B is optimized independently in [20].

can now solve it with CPLEX [24]. The computational complexity of inputting this problem to a solver is $\mathcal{O}((m+r)^2n^2)$, but that of solving it can be exponential [25].

2) Iterative Rank Maximization: As the objective function in the last section is to minimize $\|D^TD - \zeta\|_F^2$ instead of directly maximizing the rank of D directly, it may not obtain an optimal solution in some cases. Therefore, we also propose an iterative method to construct the matrix D with guaranteed maximum rank, based on the fast rank computation algorithm [26]. We gradually pick up a row in C, which specifies the paths that can be aggregated by a SDN rule, and add it to D, then we use the algorithm in [26] to quickly calculate the rank of new matrix and test whether it increases the rank or not. If it does, and it satisfies other constraints in the formulation, we then add that row to D. Otherwise, we continue to try other rows in C until we tried all the rows in C or the increased rows exceed r. The fast rank computation algorithm [26] has a time complexity of $\mathcal{O}(mn(\log min\{m,n\})^2)$ and our method invokes this algorithm at most n_a times.

3) Feasibility of Aggregations: Eqn.(7) shows that the last r rows of D must be selected from C that stores all the feasible aggregations of paths. In other words, the SDN rules need to be able to measure the aggregation of paths. If we aggregate the paths randomly, then there is high possibility that some aggregations of paths cannot be measured by the current standard of SDN. For the current standard and practice of SDN (e.g., Openflow), it can only update the counters when the header of the packet matches the "flow match fields" (srcIP, destIP, srcPort, destPort, etc.). In this setting, measuring the volume of traffic on each path may not be feasible unless we measure every flow in the network and record the route of the flows, which is too costly. So how to use the SDN counters to record the aggregations of paths (i.e., how to obtain the row of C) is very important.

Fortunately, the ip address of the servers in a certain rack are commonly block based, which means we can set up TCAM wildcard rules to record the total volume of flows between two racks. And these flows traverse multiple paths between these two racks for the multi-path routing strategies such as ECMP [27]. Then we can utilize the relationship between the ip address and the location of the ToR to set up the TCAM wildcard rules and record the traffic on the paths between two racks. For instance, in the SDN wildcard rule, if we set the source ip address to be the ip block of the first rack, and the destination ip address to be the ip block of the third rack, then the value of the SDN counter for this rule should be the volume of traffic on all the paths between the first ToR and third ToR. In this way, we can use the TCAM rules to record the volume of total traffic of aggregated paths in DCNs. And the aggregations of paths between any two racks would make up a SDN rule, which is also one row in C.

4) Topology-aware Efficient Approach in Practice: As the optimal solutions may be time consuming, in this section, we propose a topology-aware heuristic, based on two propositions indicating the cases when we cannot increase the rank of the measurement matrix given the link counts and previously installed SDN rules, to avoid setting these SDN rules to increase the rank of network tomography problem. Here are the two propositions:

Proposition 1. *If we set up the SDN rules for the ToR pairs within every pod in fat-tree architecture[2] first and then the ToR pairs across different pods both in an lexicographical order, then the SDN rules for the following ToR pairs would not increase the rank of the network tomography problem:*

- *For the ToR pairs within each pod in lexicographical order, the last pair in each pod.*
- *The pairs between all the ToRs (except the last ToR) and the last ToR.*
- *The pairs between the last ToR in the penultimate pod and the ToRs in the last pod (except the last ToR, which is included in the last case).*
- *The pair between the last ToR in the last third pod and the last ToR in the penultimate pod.*

Proof: For the first case, that is because in the topology of tree-like topology, we can get the total volume of traffic that stays in each pod (cluster) by the SNMP link counts. So the last pair in each pod is not necessary. Because we can simply compute the volume of traffic for the last pair by using the total volume stayed in each pod to minus the sum of the volumes of other internal pairs, which are already known.

The second case happens in the ToR pairs that across different pods. So for all the ToRs except the last ToR, the last ToR in the last pod is the last ToR that they may make up a ToR pair with. And because we have already known the total volume of traffic between one ToR and all the other ToRs (through link counts) and the volume of traffic from this ToR to all the other ToRs except the last ToR. So we can easily calculate the volume between these ToRs and the last ToR. So this case cannot bring any new information and thus cannot increase the rank. The pairs between these ToRs and the last ToR are also the *last pair* for these ToRs.

Similar with the previous case, the pairs between the last ToR in the penultimate pod and the ToRs in the last pod are the *last pairs* for the ToRs in the last pod (recall the orders in our setting). So for the ToRs in the last pod, they can compute the volume of these pairs by using the total volume (through link counts) to minus the total volume of traffic between them and other ToRs.

In the last case, we can consider this ToR pair as the *last pair* for the last ToR in the penultimate pod. So this case would not increase the rank also. ■

Proposition 2. *For a k-ary fat-tree topology[3], SDN can increase the rank by at most* $k^4/8 - 3 * k^2/4 - k$.

Proof: The total number of ToRs is $k^2/2$. The number

[2]For tree architectures, pod is equal to the cluster that consists of two aggregation switches and four ToR switches. This proposition also applies in tree architecture.

[3]The computing process is the same under tree architecture, so we omit the details.

of all the possible ToR pairs is $k^2/2 * (k^2/2 - 1) * 1/2 = k^4/8 - k^2/4$. Based on **Proposition 1**, the number of ToR pairs that could not increase the rank is $k^2/2 + k$. So SDN rules can increase the rank by at most $k^4/8 - 3 * k^2/4 - k$. ■

According to our experiment results in Sec. V-B, all these $k^4/8 - 3 * k^2/4 - k$ rules can be used to increase the rank of D in practice; this number is about twice of the number of link counts available in the fat-tree topology. Therefore, as long as we follow the rules stated in the proposition to set up the SDN rules, we can guarantee maximizing the rank of measurement matrix D, thus potentially improving the identifiability of network tomography problem. The result in Sec. V-B also shows this topology-aware heuristic approach can achieve optimal solutions.

C. Identify Elephant Flows

There are mainly two ways of identifying elephant flows. The first way is to observe all the flows and pick the elephant flows, which is the expensive way. The second way is to only keep an eye on the most possible elephant flows. Our approach belongs to the second way.

We use the following two steps to identify the elephant flows. Firstly, we divide the total number of SDN rules allocated for elephant flow detections among ToR pairs based on the volume of traffic from one ToR to another. This is possible because we have obtained the total volume of traffic among ToRs from the SDN-enhanced network tomography. We also take as a prior that the ToR pairs exchanging more traffic have greater chance to contain elephant flows. We then distribute the SDN rules proportionally to the volumes of traffic from one ToR to another. Secondly, we need to locate the server pairs under ToR pairs for setting up the SDN rules to pinpoint the elephant flows. Our currently strategy is to use the SNMP link counters for calculating (through a low-level network tomography) the weight of each server pairs first, which in turn allow us to set up SDN rules in a greedy manner: rules are set up for server pairs with higher weight (where we have greater chance to hit elephant flows) first until we used up all the SDN rules for this ToR pair.

The algorithm details are shown in **Algorithm 1**, where n_{tor} and n_{ser} are the number of ToRs and servers. We first divide the total SDN rules proportionally with the total volume of traffic from one ToR to another in lines 1–4. In line 6, we calculate the weight for each server pair based on the throughput of the servers. After that, we sort the server pairs by weights in a descending order and allocate SDN rules for the server pairs greedily in lines 7–8 for monitoring. The algorithm adaptively sets up the SDN rules in each interval. The time complexity of our algorithm is $\mathcal{O}(n_{tor}^2)$.

Algorithm 1: Elephant Flow Detection Algorithm

Input: The volume of traffic from ToR_i to ToR_j, $\{x_{ij} | i = 1, \cdots, n_{tor}, j = 1, \cdots, n_{tor}\}$.
The total in/out bytes of servers $\{server_k^{out}, server_k^{in} | k = 1, \cdots, n_{ser}\}$.
The number of available SDN rules for elephant flow detection n_r.

Output: The set of server pairs $\mathcal{P}$ that may have elephant flows.

1 - The sum of the volume of traffic among the ToR pairs $s_x = \sum_{i,j=1, i \neq j}^{n_{tor}} x_{ij}$.
2 **for** $i = 1$ **to** n_{tor} **do**
3 **for** $j = 1$ **to** n_{tor} **do**
4 - Let $r_{ij} = \lfloor x_{ij}/s_x \times n_r \rfloor$ denotes the number of rules for server pairs from rack i to j.
5 **if** $(i \neg j) \wedge (r_{ij} \geq 1)$ **then**
6 - For all the server pairs $(server_g, server_h)$ where $server_g$ is under rank i and $server_h$ is under rack j, we calculate the multiplication of $server_g^{out}$ and $server_h^{in}$ for each pair.
7 - We sort these server pairs by the multiplication results in a descending order.
8 - The first r_{ij} server pairs will be added to $\mathcal{P}$ and assigned SDN rule for monitoring.
9 **return** $\mathcal{P}$

V. Evaluation

In this section, we first present the experiment settings and metrics. We then show the performance of rank maximization of measurement matrix. Furthermore, we discuss about the performance of TM estimation and elephant flow detection in our monitoring system.

A. Experiment Settings and Metrics

The topologies of DCNs used in experiments and simulations are specified as follows. Our testbed that is shown in Fig. 4 consists of 10 ToRs, 3 aggregation switches and 1 core switch, which are organized in a tree architecture. For the simulations, we adopt both tree architecture and fat-tree architecture. In tree architecture, we use the topology with 32 ToRs, 16 aggregation switches and 3 core switches. Each 2 aggregation switches are connected with 4 ToRs and all the core switches. In the fat-tree architecture, we use the k=8 fat-tree with 8 pods. So it also has 32 ToRs, but has 32 aggregation switches and 16 core switches.

Fig. 4. Inside view of our testbed.

For evaluating the performance of rank maximization of measurement matrix, we first compare the the number of ranks that can be improved by the two systematic algorithms and one heuristic algorithm when the number of SDN rules is increased one by one. We also compare the max number of ranks that can be achieved in the SDN-enabled network tomography problem with the rank of routing matrix in different architectures.

TM estimation is an important application of our monitoring system. For the performance of TM estimation, we choose two state-of-art network tomography algorithms as our

baselines. They are *Sparsity Regularized Matrix Factorization (SRMF)* [11] and *TomoGravity* [10], which are widely used in the network tomography problem. We use the default S and T as suggested in [11] for *SRMF* and directly apply gravity model to get the prior TM for *TomoGravity* (more accurate prior TM design in DCNs can be found in [28]). After that we compare the performance of the two state-of-art algorithms with our proposals SRMF_SDN and Tomo_SDN by both real data center traffic and our trace-driven network simulation traffic.

The first metric we used for TM estimation is *Relative Error* (RE), which is for measuring the gap between ground truth and estimated value as shown in the following equation:

$$\mathrm{RE}_i = |x_i - \hat{x_i}|/x_i, \tag{15}$$

where x_i is the ground truth and $\hat{x_i}$ is the estimated value.

We also adopt *Root Mean Square Relative Error* (RMSRE) for measuring the performance of estimating big flows on the paths among ToRs. We use γ to denote the threshold of the size of flows that we may concern. Here is the definition:

$$\mathrm{RMSRE}(\gamma) = \sqrt{\frac{1}{n_\gamma} \sum_{i=1, x_i > \gamma}^{n_x} \left(\frac{x_i - \hat{x_i}}{x_i} \right)^2}, \tag{16}$$

where n_γ is the number flows $x_i > \gamma$ and n_x is the number of flows in the ground truth.

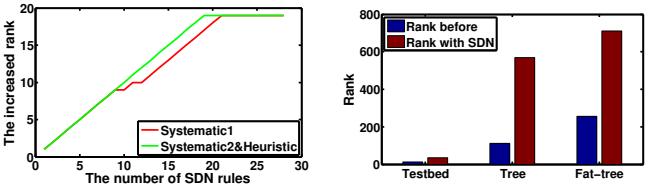
We also investigate the performance of identifying the elephant flows among servers. Here a flow means the traffic from one server to another. When its size exceeds a certain threshold η, which is 10% of the link capacity in our paper, it is considered as an elephant flow. We use the possibility of accurate identification p_{true} in Eqn.(17) to evaluate the accuracy of elephant flow detection. And we will see how it changes with the number of SDN rules that are allocated for this measurement task.

$$p_{true} = P(\hat{x_i} \geq \eta | x_i \geq \eta). \tag{17}$$

B. Rank Maximization of Measurement Matrix

In this section, let us first look at the performance of maximizing the rank of the measurement matrix D to maximize the identifiability of network tomography problem with a certain number of SDN rules. For the time consuming process of solving ILP, we design a small tree architecture with eight ToRs, two aggregation switches and one core switch. The results of these three algorithms (two systematic and one heuristic algorithm) is shown in Fig. 5(a). In this figure, we can see that both the systematic2 (iterative rank maximization) and topology-ware heuristic approach achieve the optimal solution. In other words, they can add one rank to the measurement matrix with each row (SDN rule) because they both ignore the cases that cannot increase the rank. While systematic1 (the first systematic approach) gets a near optimal solution, it needs two more SDN rules to reach max rank. That is because the objective function is to reduce the coherence of the columns of the matrix, which would increase the rank of the matrix, but cannot ensure the max rank.

In Fig. 5(b), we can see the max number of ranks that we can add to the original routing matrix given the feasibility of aggregations in SDN. The details of topologies used in this experiment are presented in Sec. V-A. The blue (left) pillar represents the rank of routing matrix and the red (right) pillar denotes the rank of D after we add all the possible constraints with SDN rules. The rank is about *three* times to the one of original routing matrix in the testbed and fat-tree. And it is about *five* times to the rank of routing matrix in the tree architecture case that is because the routing matrix in tree architecture is more sparse. So as we can see, we can increase the rank of the network tomography problem in DCNs greatly and improve the estimation accuracy thereafter, as we will see in the next section.



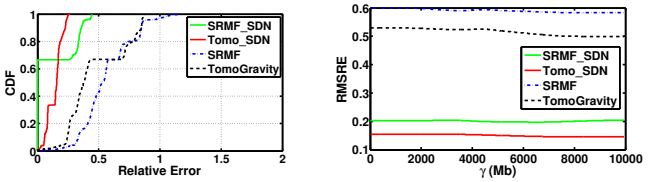
(a) Increasing the rank with SDN rules. (b) The number of ranks before and after setting up SDN rules.

Fig. 5. The performance of adding the ranks in DCNs.

C. Testbed Evaluation

In this section, we show the TM estimation results with the data collected from our private operational DCN.

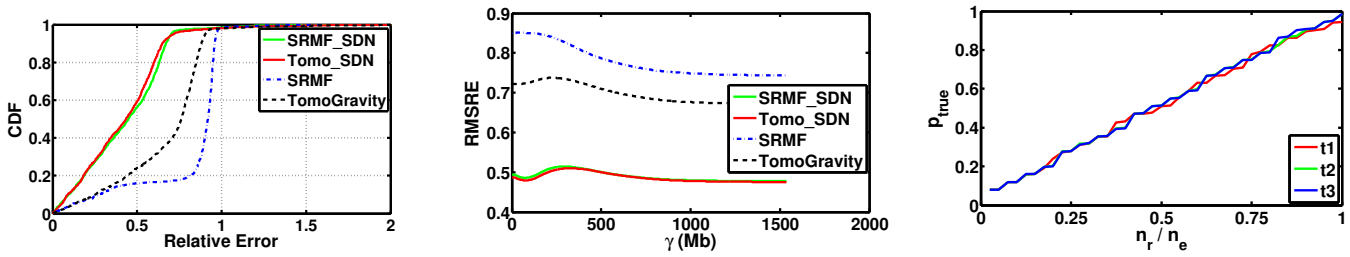
1) Testbed Setup: Our data center testbed hosts a lot of applications and services such as web services and multimedia applications. We collect the SNMP link counts and routing information as the inputs of network tomography. We also collect the SDN counters in the openflow-enabled ToR switches to get the volume of traffic among racks to add the constraints for the network tomography problem. We use the *linux iptable* (not a scalable approach) to collect all flows as ground truth.



(a) The CDF of RE (b) The RMSRE under different γ

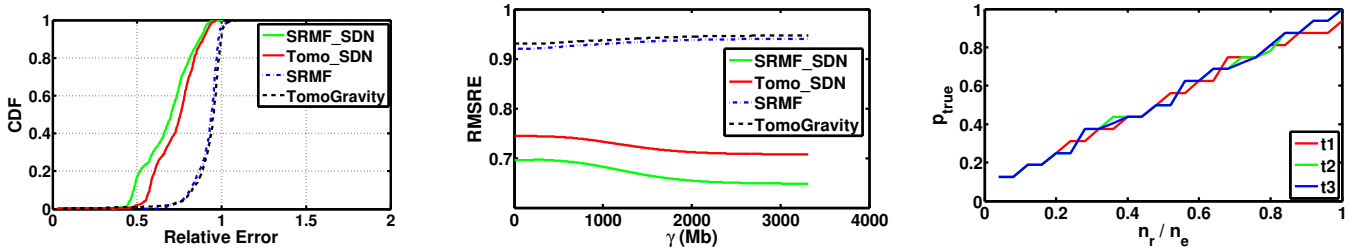
Fig. 6. The CDF of RE (a), the RMSRE (b) for estimating TM.

2) Testbed Results: In Fig. 6(a), we draw the CDF of RE for the two state-of-art algorithms both before and after adding the constraints with SDN rules. As we see in this figure, both our proposals outperforms TomoGravity and SRMF in estimation accuracy. Especially for SRMF_SDN, after adding the constraints, it can accurately estimate more 60% of flows in this case, and most of those entries are zero entries. An explanation for this is that the SRMF_SDN knows whether there is traffic between two ToRs through SDN counters. If the volume of traffic between two ToRs is zero, the volumes



(a) The CDF of RE (b) The RMSRE under different γ (c) The possibility of true positive.

Fig. 7. The CDF of RE (a), the RMSRE (b) for estimating TM and the performance of elephant flow detection (c) under tree architecture.



(a) The CDF of RE (b) The RMSRE under different γ (c) The possibility of true positive.

Fig. 8. The CDF of RE (a), the RMSRE (b) for estimating TM and the performance of elephant flow detection (c) under fat-tree architecture.

on those paths between the two ToRs are naturally zero. While for Tomo_SDN, it cannot identify those zero entries efficiently for the way it compute the prior TM, but we can still tell the improvements with more constraints available.

As the RE shows the estimation errors of separate flows, we also show the RMSRE of the algorithms, which reflects overall estimation accuracy changing with the threshold of big flows, in Fig 6(b). We can easily see that both of our proposals have much lower RMSRE after adding the constraints and Tomo_SDN has lower RMSRE than SRMF_SDN, which are both consistent with the results in Fig. 6(a). We can also see that the RMSREs of our two SDN enhanced algorithms are non-increasing with increasing γ, which means that the two algorithms are capable of estimating the big flows with even higher accuracy also.

D. Simulation Evaluation

As the scale of data center testbed is still limited, we conduct extensive simulations with *ns*-3 to validate our solutions. We evaluate both TM estimation and elephant flow detection in the simulation.

1) Simulation Setup: We both simulate the tree architecture and fat-tree architecture with the topologies described in Sec. V-A. We did not use BCube [29] in this work, which is server-based and different from the rack-based topology. It would be interesting to adapt our approach in BCube in our future work. All the links are 1Gbps.

Similar with *Hedera* [1], we use both random and stride pattern in generating the traffic with *ns*-3. The distributions of flow sizes follow the characteristics reported in [8], [13]. For instance, 80% of flows are smaller than 10KB, and most of the bytes in the network lies in the top 10% of flows. More specifically, we install both on-off and bulk send applications in the servers with random and stride while following the distributions of flow sizes. The packet size is set to be 1400 Bytes. And we use TCP flows for the application layer protocol and ECMP [27] for routing.

In each test case, we record the size and route of each flow as the ground truth. We also collect the aggregated link counts of each link for network tomography. For the purpose of elephant flow detection, we also record the total in/out bytes of each server. After we collect all the traffic data, then the data will be fed to the network tomography algorithms and elephant flow detection algorithm implemented in Matlab.

2) Simulation Results: The simulation results under tree architecture is shown in Fig. 7. In Fig. 7(a), we depict the CDF of RE of the four algorithms. As we can see in this figure, about 90% of REs in SRMF_SDN and Tomo_SDN is lower than 0.65, which is much better than the case with SNMP only network tomography (SRMF and TomoGravity) for the extra measurements powered by SDN.

In Fig. 7(b), we draw the figure about how RMSRE changes with the threshold of flow size γ. First of all, the SDN enabled approaches (SRMF_SDN and Tomo_SDN) perform much better than the SRMF and TomoGravity due to the more determined solution space. Secondly, all the lines show the decreasing trends when γ is bigger than 200MB, which shows higher accuracy for estimating bigger flows.

We show the possibility of accurately identifying the elephant flows (among servers) in Fig. 7(c). n_r means the number of SDN rules for elephant flow detection and n_e is the number of elephant flows. We draw the data for three different time slices. The step of n_r/n_e is 0.025. As we can see in this figure, p_{true} is almost equals to the percent of available SDN rules n_r/n_e and it reaches almost 100% when $n_r/n_e = 1$, which means that the performance of our approach is pretty close

to direct measurements since it can use the SDN counters to narrow the searching space first, and then adopt tomography strategies for identifying elephant flows.

We also conduct the simulations under fat-tree architecture as shown in Fig. 8. More specifically, in Fig. 8(a), we show the CDF of RE of the four algorithms. We can see that both SRMF_SDN and Tomo_SDN achieve better results than the SRMF and TomoGravity. But we can also see that all the four algorithms in this case perform worse than the case under tree architecture, which is because the gap between available measurements and unknown variables is much bigger due to the rich connections among switches in fat-tree architecture. In such cases, the extra measurements are very necessary.

In Fig. 8(b), both SRMF_SDN and Tomo_SDN show apparent decreasing trends, while SRMF and TomoGravity show increasing trends with the increasing of γ, which again shows the necessity of the constraints powered by SDN especially in the case where the number of unknown variables is far more than available measurements such as in fat-tree.

We show the performance of elephant flow detection in Fig. 8(c). Similar to the result under tree architecture, the performance is also very well under fat-tree architecture. The step of n_r/n_e is 0.04. The p_{true} also improves gradually with more SDN rules. And as long as the number of SDN rules is enough, we can accurately identify almost all the elephant flows, which is comparable to direct measurements for elephant flow detection.

VI. Conclusion and Future Work

In this paper, we have proposed a monitoring system combining SDN and network tomography to estimate the traffic matrix (TM) and to identify the elephant flows effectively in DCNs. We have introduced both systematic and heuristic algorithms to maximize the identifiability of network tomography problem with SDN rules, under a given routing matrix; they are shown to be greatly improving the performance of network tomography in DCNs. We have also designed an algorithm applying tomographic ideas with SDN counters to accurately identify the elephant flows in DCNs. Finally, we have conducted extensive performance evaluation on our approach through both experiments on a data center testbed and simulations with *ns*-3; the results have strongly confirmed the effectiveness and practicality of our monitoring system.

We are on the way to design more applications such as traffic engineering and anomaly detection on top of our monitoring system. We expect to evaluate the performance of applications based on our monitoring system against on direct measurements or network tomography alone in DCNs.

References

[1] 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.

[2] Y. Cui, H. Wang, X. Cheng, D. Li, and A. Yla-Jaaski, "Dynamic Scheduling for Wireless Data Center Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 12, pp. 2365–2374, 2013.

[3] K. Han, Z. Hu, J. Luo, and L. Xiang, "RUSH: RoUting and Scheduling for Hybrid Data Center Networks," in *Proc. of IEEE INFOCOM*, 2015.

[4] 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.

[5] "Sflow." [Online]. Available: http://www.sflow.org/

[6] B. Claise, "Cisco systems NetFlow services export version 9," 2004.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] Q. Zhao, Z. Ge, J. Wang, and J. Xu, "Robust Traffic Matrix Estimation with Imperfect Information: Making Use of Multiple Data Sources," in *Proc. of ACM SIGMETRICS/Performance*, 2006, pp. 133–144.

[15] Y. Vardi, "Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data," *Journal of the American Statistical Association*, vol. 91, no. 433, pp. 365–377, 1996.

[16] Y. Cui, S. Xiao, C. Liao, I. Stojmenovic, and M. Li, "Data Centers as Software Defined Networks: Traffic Redundancy Elimination with Wireless Cards at Routers," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 31, no. 12, pp. 2658–2672, 2013.

[17] M. Yu, L. Jose, and R. Miao, "Software Defined Traffic Measurement with OpenSketch," in *Proc. of USENIX NSDI*, 2013, pp. 29–42.

[18] N. van Adrichem, C. Doerr, and F. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *Proc. of IEEE/IFIP NOMS*, May 2014, pp. 1–8.

[19] 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.

[20] M. Malboubi, L. Wang, C.-N. Chuah, and P. Sharma, "Intelligent SDN based Traffic (de)Aggregation and Measurement Paradigm (iSTAMP)," in *Proc. of IEEE INFOCOM*, 2014, pp. 934–942.

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

[22] N. J. Harvey, D. R. Karger, and S. Yekhanin, "The Complexity of Matrix Completion," in *Proc. of 7th annual ACM-SIAM symposium on Discrete algorithm*. ACM, 2006, pp. 1103–1111.

[23] L. Zelnik-Manor, K. Rosenblum, and Y. C. Eldar, "Sensing Matrix Optimization for Block-Sparse Decoding," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4300–4312, 2011.

[24] "IBM ILOG CPLEX Optimizer." [Online]. Available: http://www.ibm.com/software/integration/optimization/cplex-optimizer/

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*. WH Freeman & Co., 1979.

[26] H. Y. Cheung, T. C. Kwok, and L. C. Lau, "Fast Matrix Rank Algorithms and Applications," *Journal of the ACM (JACM)*, vol. 60, no. 5, p. 31, 2013.

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

[28] Z. Hu, Y. Qiao, J. Luo, P. Sun, and Y. Wen, "CREATE: CoRrelation Enhanced trAffic maTrix Estimation in Data Center Networks," in *Proc. of IFIP Networking*, 2014, pp. 1–9.

[29] 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.