ATME: Accurate Traffic Matrix Estimation in both Public and Private Datacenter Networks

Zhiming Hu, *Student Member, IEEE,* Yan Qiao and Jun Luo, *Member, IEEE*

Abstract—Understanding the pattern of end-to-end traffic flows in *datacenter 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 have high monitoring overhead and consume significant switch resources even if they are available in a DCN. Although estimating the *traffic matrix* (TM) between origin-destination pairs using only basic switch SNMP counters is a mature practice in IP networks, traffic flows in DCNs show totally different characteristics, while the large number of redundant routes in a DCN further complicates the situation. To this end, we propose to utilize resource provisioning information in public cloud datacenters and the service placement information in private datacenters 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 ATME as an efficient TM estimation scheme that achieves high accuracy for both public and private DCNs. We compare our two algorithms with two existing representative methods through both experiments and simulations; the results strongly confirm the promising performance of our algorithms.

Index Terms—Measurements, Traffic Matrix, Datacenter Networks, Cloud Computing.

1 Introduction

As datacenters that house a huge number of interconnected 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 *datacenter networks* (DCNs). The main topics under this theme include, among others, network architecture design [1], [12], [13], traffic engineering [2], scheduling in wireless DCNs [9], [16], capacity planning [21], and anomaly detection [11]. 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, which are normally expressed in the form of *traffic matrix* (TM for short), serve as critical inputs to all the above DCN operations.

Existing proposals in need of detailed traffic flow information collect the flow traces by deploying additional modules on either switches [2] or servers [10] in small scale DCNs. However, both methods require substantial deployments and high administrative costs, and they are difficult to be implemented thanks to the heterogeneous nature of the hardware in DCNs [28]. More specifically, the switch-based approaches, on one hand, need all the ToRs to support flow tracing tools such as OpenFlow [26], and consume a substantial number of switch resources to maintain the flow entries.[1] On the other hand, the server-based approaches, which require instrumenting all the servers or VMs to support data collection, are unavailable in most datacenters [22] and are 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* (TMs) of IP networks to be inferred from link level measurements (e.g., SNMP counters) [29], [34], [35]. 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 [22] and our simulations (see Sec. 7) demonstrate that 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 TM among ToR Pairs. This refers to the fact that one ToR switch may only exchange flows with a few other ToRs, as demonstrated in [15], [22], [30]. 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 [34]. The other barrier is the highly under-determined solution space. In other words,

- *Zhiming Hu and Jun Luo are with the School of Computer Engineering, Nanyang Technological University, Singapore. E-mail: {zhu007, junluo}@ntu.edu.sg.*
- *Yan Qiao is with School of Information and Computer, Anhui Agricultural University, China. The work was done when she was a post-doctoral researcher at NTU. E-mail: qiaoyan101@gmail.com.*
- *Preliminary results were presented in Proceedings of the 13th IFIP Networking, 2014 [19].*

1. 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.

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 [22] while the number of link constrains is only around hundreds.

As TMs are sparse in general, correctly identifying the zero entries in them may serve as crucial priors. In both public and private DCNs, if two VMs/servers are occupied by different users, which can be derived from *resource provisioning information*, we can be rather sure that these VMs/servers would not communicate with each other in most cases. Moreover, in private DCNs[2], we may further take advantage of having the *service placement information*. This allows us to deduce that two VMs/servers belonging to same user would probably not communicate with each other if they host different services, because different services in DCNs rarely exchange information [8].

In this paper, we aim at conquering the aforementioned two barriers and making TM estimation feasible for DCNs, by utilizing the distinctive information or features inherent to these networks. First, we make use of the resource provisioning information in a public cloud and the service placement information in a private datacenter (both can be obtained from the controller node of DCNs) to derive the correlations among ToR switches. The communication patterns among ToR pairs inferred by such approaches are far more accurate than those assumed by conventional traffic models (e.g., the gravity traffic model [34]). Second, by analyzing the statistics of link counters, we find that the utilizations of both core links and aggregation links are extremely uneven. In other words, there are a considerable amount of links undergoing very low utilization during a 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 ATME (Accurate TM Estimation) as an efficient estimation scheme to accurately infer the traffic flows among ToR switch pairs without requiring any extra measurement tools. In summary, we make the following contributions in our paper.

- We creatively use resource provisioning information in public datacenters for deriving the prior TM among ToRs. We group all the VMs into several clusters with respect to different users, resulting in the effect that communications only happen within the same cluster and the potential traffic patterns are epitomized among all VMs in turn.
- We pioneer in using the service placement information in private datacenters 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.
- We innovate in leveraging the uneven link utilization in DCNs to remove potentially redundant routes. Essentially, we may consider links with very low utilization as non-existent without affecting much the accuracy of TM estimation, while they effectively lessens 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 ATME as an efficient scheme to infer the TM for DCN ToRs with high accuracy in both public and private DCNs. ATME first calculates a prior assignment of traffic volumes for each ToR pairs using aggregated traffic of VM pairs (in public DCNs) or the correlation factors (in private DCNs). Then it removes lowly utilized links and thus operates only on a sub-graph of the DCN topology. It finally adapts a quadratic programming to determine the TM under the constraints of the tomography model, the enhanced prior assignments, and the reduced DCN topology.
- We validate ATME with both experiments on a relatively small scale datacenter 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. 2. Then we present system model and formally describe our problem in Sec. 3. In Sec. 4, we reveal some traffic characteristics in DCNs and propose the architecture of our system design motivated by those traffic characteristics. After that, we present the way we compute the prior TM among ToRs and the link utilization aware network tomography in Sec. 5 and Sec. 6, respectively. We evaluate ATME using both real testbed and different scales of simulations in Sec. 7, before concluding our paper in Sec. 8.

2 Related Work

As datacenter networking has recently emerged as a hot topic for both academia and industry, numerous studies have been conducted to improve its performance [1], [2], [5], [11]–[13], [21]. 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 TMs, rely on either switch-based or server-based methods.

The switch-based methods (e.g., [2]) normally adopt programmable ToR switches (e.g., OpenFlow [26] switch) to record flow data, then utilize those flow data for higher layer applications or measurements [25], [32], [33]. However, these methods may not be feasible for three reasons. First, they incur high switch resource consumptions 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 [31], 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 [10]. Second, hundreds of controllers are needed to handle the

2. For private DCNs, the owner knows everything about what services are deployed and where the services are hosted in the datacenter.

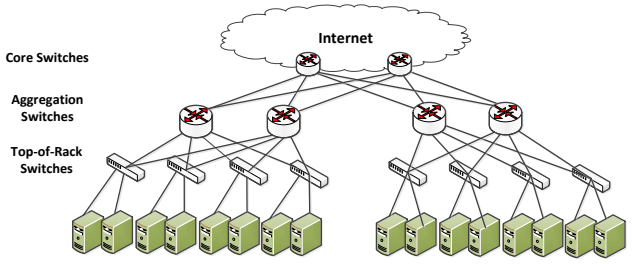


Fig. 1. An example of conventional DCN architecture, suggested by Cisco [20].

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 [31]; 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 datacenter owners may not be willing to pay for upgrading the switches.

The server-based methods require to instrument all the servers to support flow data collection [7], [10]. In an operating datacenter, it is very difficult to instrument all the servers while supporting a lot of ongoing cloud services. Also, the heterogeneity of servers may also complicate the problem: dedicated softwares may need to be prepared for different servers and their OSs. Moreover, it does cost server resources to perform flow monitoring. Finally, similar to the switch-based approaches, the willingness of datacenter owners 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 [34] adapts the gravity model to get the prior TM, and SRMF [35] is shown to perform better than others when the TM is lowly ranked. One study that has partially motivated our work is [22]: it investigates the nature of DCN traffic on a single MapReduce datacenter and poses the question that whether TMs can be inferred from link counters by tomographic methods. In a way, the answer given in [22] is negative due to the fundamental differences between DCNs and IP networks, which invalidate the assumptions made by conventional tomographic methods [34], [35]; we explained these in Sec. 1 as two obstacles. We have proposed methods to get the coarse-grained TM in [27], but we hereby aim to overcome these obstacles and hence make a fine-grained TM estimation viable in DCNs.

3 Definitions and Problem Formulation

We consider a typical DCN as shown in Fig. 1. It consists of n ToR switches, aggregation switches, and core switches connecting to the Internet. Note that our method is not confined to this commonly used DCN topology; it accommodates other more advanced topologies also, e.g., VL2 [12], fat-tree [1], as will be shown in our simulations.

We let $x'_{i\rightarrow j}$ denote the estimated volume of traffic sent from the i-th ToR to the j-th ToR and $x'_{i\leftrightarrow j}$ denote the estimated volume of traffic exchanged between the two switches. Given the volatility of DCN traffic, we further introduce $x'_{i\rightarrow j}(t)$ and $x'_{i\leftrightarrow j}(t)$ to represent values of these two variables at discrete time t, where $t \in [1, \Gamma]$.[3] Note that although these variables would form the TM for conventional IP networks, we actually need more detailed information of 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 among ToR Pairs, where p is the total number of the routes. Consequently, the *traffic matrix* $X = [\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(\Gamma)]$, where Γ is the total number of time periods, is the one we need to estimate. Our commonly used notions are listed in Table 1, where we drop time indices for brevity.

The observations that we utilize to make the estimation are the *SNMP counters* on each port of the 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 the 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 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 $Y = [\mathbf{y}(1), \mathbf{y}(2), \cdots, \mathbf{y}(\Gamma)]$ becomes the *load matrix*. [4]

Based on the network tomography, the correlation between traffic assignment $\mathbf{x}(t)$ and link load assignment $\mathbf{y}(t)$ can be formulated as

$$\mathbf{y}(t) = A\mathbf{x}(t) \qquad t = 1, \cdots, \Gamma, \tag{1}$$

where A denotes the routing matrix, with rows corresponding to links and columns indicating routes among 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 X using the load matrix Y derived from the easy-collected SNMP data.

Although Eqn. (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 skewed [30]. As shown in Fig. 2, the sparse and skew nature 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 in Eqn. (1), 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 the architecture 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 Eqn. (1)

3. Involving time as another dimension of the TM was proposed earlier in [29], [35].

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

TABLE 1
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
Γ	The number of time periods
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
λ_i	The number of servers belonging to the i-th rack
$x'_{i \rightarrow j}$	The estimated volume of traffic send from the i-th ToR to the j-th ToR
$x'_{i \leftrightarrow j}$	The estimated volume of traffic exchanged between the i-th and j-th ToRs
$\mathbf{x}$	$\mathbf{x} = [x_i]_{i=1,\cdots,p}$, where x_i is the traffic on the r-th routing path
$\bar{x}_i$	The prior estimation of the traffic on the i-th routing path.
ToR_i^{in}	The total "in" bytes of the i-th ToR during a certain interval
ToR_i^{out}	The total "out" bytes of the i-th ToR during a certain interval
S	$S = [s_{ij}]_{i=1,\cdots,r;j=1,\cdots,n}$, 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	The threshold of link utilization
$\mathcal{T}$	The set of tuples for (userId, serverId, rackId)
$\mathcal{T}_u$	The set of VMs owned by the u-th user
$\mathcal{T}^i$	The set of VMs in i-th rack.
v_i^{in}	The total "in" bytes of i-th VM during a certain interval.
v_i^{out}	The total "out" bytes of i-th VM during a certain interval.
e_{ab}	The volume of traffic from a-th VM to b-th VM.
$\mathcal{U}$	The set of all users.
q	The total number of VMs in the datacenter.

would not work, and we need to derive a new method to handle TM estimation in DCNs.

4 Overview

As directly applying network tomography to DCNs is infeasible for several challenges, we first reveal some observations about the traffic characteristics in DCNs. Then we present the system architecture of ATME that applies these observations to conquer the challenges.

4.1 Traffic Characteristics of DCNs

As mentioned earlier, several proposals including [15], [22], [30] have indicated that the TM among 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. Fig. 2, adopted from [30], plots the traffic normalized volumes among ToR switches in a DCN with 75 ToRs. 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 share either very minor flows or nothing. Therefore our first observation is the following:

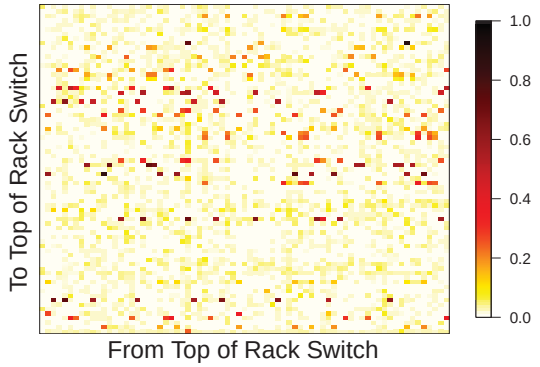


Fig. 2. The TM across ToR switches reported in [30].

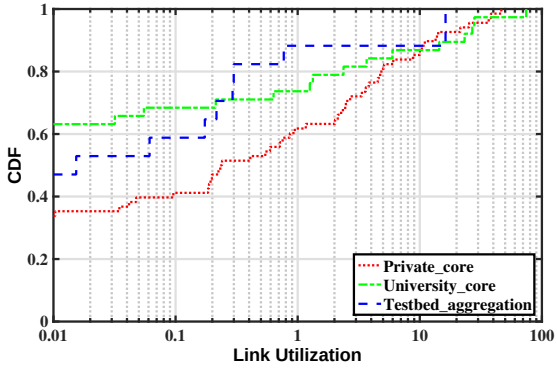


Fig. 3. Link utilizations of three DCNs, with "private" and "university" from [6] and "testbed" being our own DCN.

> **Observation 1**: TMs among ToRs are very sparse, so prior TMs among ToRs should also be sparse with similar sparse patterns to gain enough accuracy for the final estimation.

Although we may infer the skewness in the TM in some way (more details can be found in the following sections), the existence of multiple routes 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.* [6], 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 [6], 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-minute intervals in Fig. 3. 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 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 number of 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 the ToR below it. Therefore, we may significantly reduce the number of potential routes between any two ToRs by eliminating the lowly utilized links. Though this comes at a cost of slightly losing actual flow counts, the overall estimation accuracy or the running speed should be improved, thanks to the elimination of the ambiguity in the actual routing path taken by the major flows. 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 and the speed of the TM estimation.

4.2 ATME Architecture

Based on these two observations, we design ATME as a novel prior-based TM estimation method for DCNs. In a nutshell, we periodically compute the prior TM among 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, ATME is the first practical system for accurate TM estimation in both public and private DCNs.

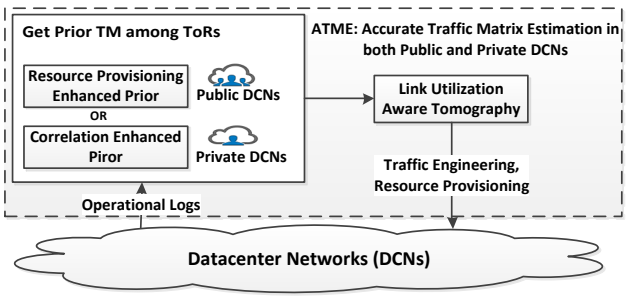


Fig. 4. The ATME architecture.

As shown in Fig. 4, our system ATME contains two algorithms in total: ATME-PB for public DCNs and ATME-PV for private DCNs. Both of them take two main steps to estimate the TM for DCN ToRs. They have different ways to compute the prior TM among ToRs, while share the same link utilization aware tomography process as the second step. More specifically, first of all, ATME calculates the prior TM among different ToRs based on SNMP link counts and some other operational information such as resource provisioning information in a public DCN or the service placement information in a private DCN motivated by **Observation 1**. We elaborate the first step in Sec. 5. Second, it eliminates the lowly utilized links to reduce redundant routes and narrows the searching space of potential TMs suggested by the load vector $\mathbf{y}$ according to **Observation 2**. After that, it takes the prior TM among ToRs and network tomography constrains as input and solve the optimization problem to estimate the TM. We discuss the second step later in Sec. 6.

5 Getting the Prior TM among ToRs

An accurate prior TM is a good beginning for our prior-based network tomography algorithm. In this section, we introduce two light-weighted methods to get the prior TM $\mathbf{x}'$ with the help of operational information in DCNs. More specifically, as only resource provisioning information is available in public DCNs, we use them to deduce the relationship between communication pairs. Since service placement information provides more information than resource provisioning information in private DCNs, we adopt service placement information instead to enhance the estimation accuracy of $\mathbf{x}'$ in private DCNs.

5.1 Computing the Prior TM among ToRs by Resource Provisioning Information in Public DCNs

In a public cloud datacenter, we can only know which part of VMs is occupied by whom, but we have no idea about how users will use their VMs for privacy issues. However we can still use the resource provisioning information, which specify the mappings between VMs and users, to infer the sparse prior TM among ToRs for the following reasons. In a multi-tenant datacenter or IaaS platform, the hardware resources are provisioned to different users, with users accessing only their own VMs. Thus the VMs belonging to one user may only communicate with each other and would not communicate with VMs occupied by other users. The volume of traffic between two ToRs can be computed by the volume of traffic among VMs (occupied by same uses) in these two racks. Therefore, the problem of computing the prior TM among ToRs can be converted to computing the volume of traffic among VMs belonging to the same user.

To better illustrate the algorithm details, here are some notations that will be used in the following sections. After analyzing the resource provisioning information, we can get a tuple set $\mathcal{T}$, with each tuple containing the userId, vmId and rackId, respectively. For instance, for a tuple $(i, j, k) \in \mathcal{T}$, it means that the i-th user is using the j-th VM located at the k-th rack. Here one VM can only be located in one rack at a certain moment. For simplicity, $\mathcal{T}_u$ denotes the set of VMs owned by the u-th user. All the VMs in the i-th rack is stored in $\mathcal{T}^i$. We also use $\mathcal{U}$ to denote the set of all the users in the public DCNs. Because the computation process also takes the VMs into account, we also need the total in/out bytes of every VM during a certain interval, which can be easily collected through the hypervisor (Domain 0) of VMs. We use v_i^{in} and v_i^{out} to denote in/out bytes of the i-th VM.

5.1.1 Building Blocks of ATME-PB

5.1.1.1 Deriving VM Locations: After analyzing the resource provisioning information, we can easily know the number of VMs and the locations of VMs owned by each user. Here for the location, we are only concerned with the index of the rack that one VM belongs to. For instance, if user_1 has two VMs (vm_1 (rack_1), vm_3 (rack_2)) and user_2 has one VM (vm_2 (rack_1)) allocated in a datacenter, we should get the following tuples after deriving the VM locations: (user_1, vm_1, rack_1), (user_2, vm_2, rack_1) and (user_1, vm_3, rack_2). In this example, $\mathcal{T}_1$ is (vm_1 (rack_1), vm_3 (rack_2)), which denotes the set of VMs owned by user_1, and $\mathcal{T}^1$ consists of (vm_1 (rack_1), vm_2 (rack_1)), which specifies the set of VMs located at rack_1.

5.1.1.2 Computing the TM among VMs in each cluster: There are roughly two steps in computing the TM among VMs. The first step is to group the VMs in $\mathcal{T}$ by user and to get $\mathcal{T}_u$ for all the users. Then in the second step, we need to compute the TM among VMs belonging to each user, given the total volume of traffic sent and received by each VM recorded by SNMP link counts during each interval. As we assume each VM will only communicate with other VMs that belong to the same user, a wise choice may be the gravity model [23], which is well suited to all-to-all traffic pattern. Therefore the volume of traffic from the a-th VM to the b-th VM e_{ab} can be computed by the gravity model as follows:

$$e_{ab} = v_a^{out} \frac{v_b^{in}}{\sum_{k \in \mathcal{T}_u} v_k^{in}}. \quad (2)$$

We conduct the same process for each group of VMs grouped by user and obtain the TM among VMs.

5.1.1.3 Computing Rack to Rack Prior: After getting the TM among VMs for each user, we then compute the rack to rack prior TM based on the locations of VMs. As we have computed the volumes of traffic among VMs and we also know the racks where VMs are, we can just sum up those volumes of traffic among VMs in different racks to get the estimated prior TM among ToRs. For example, if vm_1 and vm_2 belong to rack_1 and rack_2 respectively, then the volume of traffic from rack_1 to rack_2 will add the volume of traffic from vm_1 to vm_2.

5.1.2 The Algorithm Details

We present the details of computing resource provisioning enhanced prior TM among ToRs with $\mathcal{U}$ and in/out bytes of each VM as the input in **Algorithm 1**, where q is the total number of VMs in the DCN. It returns the prior traffic vector among ToRs $\mathbf{x}'$. More specifically, in line 1, we get $\mathcal{T}$ from resource provisioning information as additional information. From line 2 to line 4, we compute the prior volume of traffic among different VMs belonging to the same user. For each user $u \in \mathcal{U}$, the volume of traffic from the a-th VM to the b-th VM is calculated by Eqn. (2), according to the gravity traffic model. We then present our new ways to compute the prior volume of traffic between the i-th rack and the j-th rack in lines 7–9. Here, line 7 calculates the volume of traffic from the i-th ToR to the j-th ToR $x'_{i \rightarrow j}$ by summing up the volumes of traffic from a-th VM to b-th VM e_{ab} that originating at the i-th ToR and ending at the j-th ToR. Line 8 calculates $x'_{j \rightarrow i}$ in the similar way. $x'_{i \leftrightarrow j}$ in line 9 denotes the total volumes across the i-th ToR and the j-th ToR that equals to the summation of $x'_{i \rightarrow j}$ and $x'_{j \rightarrow i}$. As the algorithm runs for every time instance t, we drop the time indices. The complexity of the algorithm, which is dominated by the part that computes e_{ab}, is $\mathcal{O}(|\mathcal{U}|\mathcal{T}_u^2)$. $\mathcal{T}_u$ is normally small, so the complexity is almost linear to the number of users. In other words, the approximate complexity is $\mathcal{O}(|\mathcal{U}|)$.

Algorithm 1: Compute Resource Provisioning Enhanced Prior TM among ToRs

Input: $\mathcal{U}$, $\{v_a^{out} | a = 1, \cdots, q\}$, $\{v_b^{in} | b = 1, \cdots, q\}$
Output: $\mathbf{x}'$

1 Get $\mathcal{T}$ by analyzing the resource provisioning information.
2 **forall the** $u \in \mathcal{U}$ **do**
3 **forall the** $a,\ b \in \mathcal{T}_u$ **do**
4 $e_{ab} = v_a^{out} * \frac{v_b^{in}}{\sum_{c \in \mathcal{T}_u} v_c^{in}}$
5 **for** $i = 1$ **to** n **do**
6 **for** $j = i + 1$ **to** n **do**
7 $x'_{i \rightarrow j} \leftarrow \sum_{a \in \mathcal{T}^i} \sum_{b \in \mathcal{T}^j} e_{ab}$
8 $x'_{j \rightarrow i} \leftarrow \sum_{a \in \mathcal{T}^j} \sum_{b \in \mathcal{T}^i} e_{ab}$
9 $x'_{i \leftrightarrow j} \leftarrow x'_{i \rightarrow j} + x'_{j \rightarrow i}$
10 **return** $\mathbf{x}'$

5.1.3 A Working Example

Here we give an example about how to estimate the TM among ToRs. As shown in Fig. 5, there are three users in total. The VMs owned by those users are listed below:

- user_1: $\mathrm{vm}_1(\mathrm{rack}_1), \mathrm{vm}_9(\mathrm{rack}_5), \mathrm{vm}_{11,12}(\mathrm{rack}_6)$,
- user_2: $\mathrm{vm}_4(\mathrm{rack}_2), \mathrm{vm}_6(\mathrm{rack}_3), \mathrm{vm}_{13,14}(\mathrm{rack}_7)$,
- user_3: $\mathrm{vm}_2(\mathrm{rack}_1), \mathrm{vm}_{10}(\mathrm{rack}_5), \mathrm{vm}_{15,16}(\mathrm{rack}_8)$.

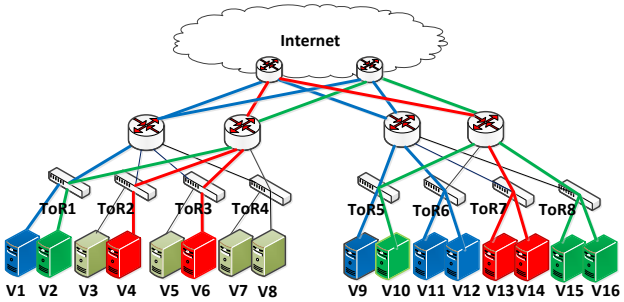


Fig. 5. Each color represent one user. Here there are totally three users. v3, v5, v7, v8 are not used by any user in this case.

Those information can be gathered in the process of resource provisioning for the cloud users. Here for simplicity, the volume of traffic that each vm sends out and receives is 10000 bytes and 1000 bytes for user1 and user3 in a certain interval, respectively. Then if we want to know the volume of traffic from ToR_1 to ToR_5, we should know the volume of traffic from v1 to v9 and the volume of traffic from v2 to v10, respectively. The volume of traffic from v1 to v9 is computed by the gravity model among v1, v9, v11 and v12. Therefore $e_{1,9} = 1000 * \frac{1000}{1000+1000+1000+1000} = 250$. We can also get $e_{2,10} = 100 * \frac{100}{100+100+100+100} = 25$. Thus based on our algorithm, the estimated prior volume of traffic from ToR_1 to ToR_5 is 275. Similarly, we can also compute the prior volume of traffic from ToR_5 to ToR_1.

5.2 Computing the Prior TM among ToRs by Service Placement Information in Private DCNs

In ATME-PB, we assume that only VMs/servers belonging to the same user may exchange information. However, it may not be the case if a user deploys different and irrelevant services on two VMs/servers. As we can also take advantage of service placement information in private DCNs, it is natural for us to utilize the service placement

information to derive more fine-grained relationship among communication pairs in private DCNs.

As stated in **Observation 1**, the TM among ToRs in DCNs is very sparse. According to the literature, as well as our experience with our own datacenter, 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 volume of the flows may be inferred by the number of instances of the shared services. Bodík *et al.* [8] has analyzed a medium scale DCN and claimed that only 2% of distinct service pairs communicates with each other. Moreover, several proposals such as [4], [14] 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 TMs among DCN ToRs. To better illustrate this phenomenon in our DCN, we show the placement of services in 5 racks using the percentage of servers occupied by individual services in each rack in Fig. 6(a), and we depict the traffic volumes exchanged among these 5 racks in Fig. 6(b). Clearly, the racks that host more common services tend to exchange greater volume of traffic (e.g., for racks 3 and 5, 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). Therefore, we propose to compute the prior TM among ToRs by service placement information in private DCNs.

In ATME-PV, we use service placement information recorded by controllers of a private datacenter as the extra information. Suppose there are r services running in a DCN, we can then get the *service placement matrix* $S = [s_{ij}]_{i=1,\cdots,r;j=1,\cdots,n}$ 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 λ_j the number of servers belonging to the j-th rack.

5.2.1 Building Blocks of ATME-PV

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 information. We use $corr_{ij}$ to quantify the correlation between the i-th and the j-th ToRs, and we calculate it as follows:

$$corr_{ij} = \sum_{k=1}^{r} [(s_{ki} \times s_{kj})/(\lambda_i \times \lambda_j)] \quad i, j = 1, \cdots, n, \quad (3)$$

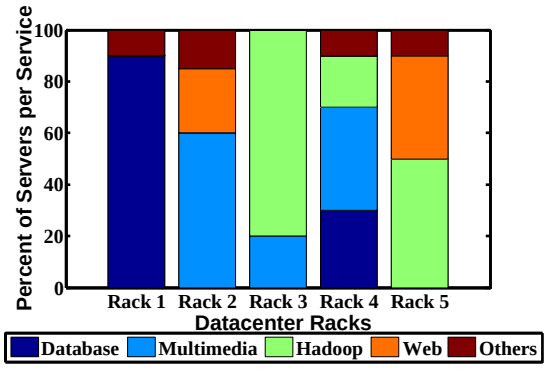
where the concerning quantities are derived from the service placement information.

In the second step, we derive a new way to compute the prior TM among ToRs based on the correlation coefficient among ToRs and the total in/out bytes of the ToRs during a certain interval. More specifically, we first compute $x_{i \leftrightarrow j}$ as the volume of traffic between ToR_i and ToR_j by the following procedure based on the correlation coefficients.

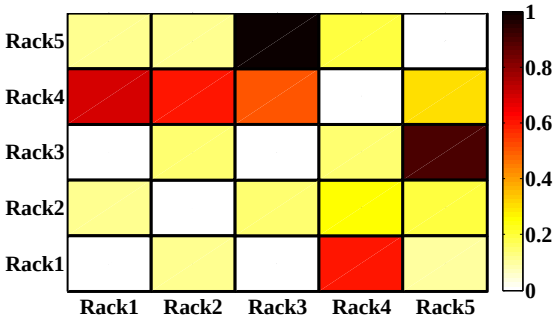
$$\begin{aligned} x'_{i \rightharpoonup j} &= ToR_i^{out} \times \frac{corr_{ij}}{\sum_{k=1}^{n} corr_{ik}} & i, j = 1, \cdots, n, \\ x'_{i \leftrightarrow j} &= x'_{i \rightharpoonup j} + x'_{j \rightharpoonup i} & i, j = 1, \cdots, n. \end{aligned}$$

Due to symmetry, $x_{i \rightharpoonup j}$ can also be computed through ToR_j^{in} in similar ways.

As our TM estimation takes the time dimension into account (to cope with the volatile DCN traffics), one may wonder whether the correlation coefficient $[corr_{ij}]$ has 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 [8]. Therefore, once $[corr_{ij}]$ is 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 quits. Even under those circumstances, we only need to re-compute the coefficients among the ToRs that are affected by service changes.



(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 placements in (a).

Fig. 6. The correlations between traffic and service in our datacenter.

5.2.2 The Algorithm Details

We show the pseudocode of calculating correlation enhanced prior TM in **Algorithm 2**. This algorithm takes service placement matrix S and the ToR SNMP counts as the main inputs, and it also returns the prior traffic vector among ToRs $\mathbf{x}'$. After computing the correlation coefficients in line 1, we compute the volume of traffic exchanged between the i-th and j-th ToR using ToR_i^{out}, ToR_j^{out} and the computed correlation coefficients in lines 4–6. The complexity of the algorithm is $\mathcal{O}(n^2)$, where n is the number of racks

in the datacenter. As n is generally small, the computation times are acceptable as we will see in the evaluations.

Algorithm 2: Compute Correlation Enhanced Prior TM among ToRs

Input: S, $\{ToR_i^{out} | i = 1, \cdots, n\}$
Output: $\mathbf{x}'$

```
[corr_ij] ← Correlation(S)
for i = 1 to n do
    for j = i + 1 to n do
        x'_{i→j} ← ToR_i^out * corr_ij / (Σ_{1≤k≤n} corr_ik)
        x'_{j→i} ← ToR_j^out * corr_ij / (Σ_{1≤k≤n} corr_kj)
        x'_{i↔j} ← x'_{i→j} + x'_{j→i}
return x'
```

5.2.3 A Working Example

Fig. 7 presents an example to illustrate how ATME-PV works. The three colors represent three services deployed in the datacenter 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 among the ToR pairs are shown in Table 2. More specifically, ToR_2 is related to ToR_3 and

TABLE 2
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

ToR_7 by a coefficient of 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. Similarly, we can compute the traffic volume that ToR_7 sends to ToR_2. Then we add the traffic of two directions together to get the traffic volumes between ToR_2 and ToR_7. A similar situation applies to ToR_2 and ToR_3. The estimated prior TM is then fed to the final estimation, as discussed later in Sec. 6.

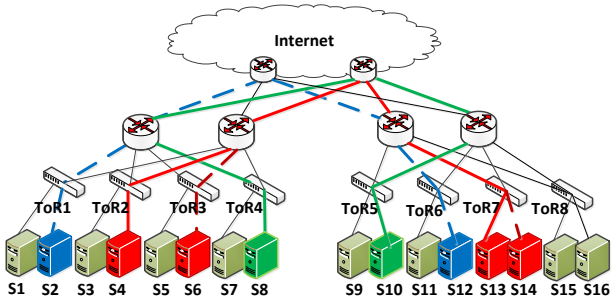


Fig. 7. Four different line styles represent four flows and three different colors represent three services.

6 Link Utilization Aware Network Tomography

In this section, we first propose to eliminate the links with low utilizations to turn the network tomography problem in DCNs into a more determined one. We then compute the prior volumes of traffic on the routes in DCNs and feed it to the network tomography constrained optimization problem.

6.1 Eliminating Lowly Utilized Links and Computing Prior Vector

This step is motivated by **Observation 2**, which states that there are plenty of lowly utilized links in DCNs. As we all know, there are many redundant routes between any two ToR switches in DCNs. Thus in the perspective of network tomography, the number of available measurements (link counts) is much smaller than the number of variables (routes). To this end, we eliminate the lowly utilized links to turn the original network tomography problem into a more determined one. More specifically, 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 Eqn. (1) can be substantially reduced, resulting in a more determined tomography problem. On one hand, this threshold 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 or running speed of algorithms. In our experiments, we shall try different values of the threshold to see the trade-off between these two sides.

Fig. 8 is the result of reducing lowly utilized links through thresholding, hence we can estimate the traffic volumes on the remaining routes from one ToR to another. In order to compute the prior vector $\bar{\mathbf{x}}$ (we omit time slice t, so the TM at time slot t is a vector), we estimate the traffic volumes on each route by dividing the total number of bytes between two ToRs, which are also stored in $\mathbf{x}'$ and can be computed by **Algorithm 1** or **Algorithm 2**, equally on every path connecting them. The reason for this equal share is the widely used ECMP [17] in DCNs; it by default selects routing paths between two switches with equal probability on each. The computed prior vector $\bar{\mathbf{x}}$ will give us a good start in solving a quadratic programming problem to determine the final estimation.

6.2 Combining Prior TM with Network Tomography constraints

Here we provide more details on the computation involved in getting the final estimation, which is also a QuadProgram. Basically, we want to obtain $\mathbf{x}$ that is as close as possible to the prior $\bar{\mathbf{x}}$ but also satisfies the tomographic constrains. This problem can be formulated as follows:

$$\text{Minimize} \qquad \|\mathbf{x} - \bar{\mathbf{x}}\| + \|A\mathbf{x} - \mathbf{y}\| \tag{4}$$

where $\|\mathbf{x} - \bar{\mathbf{x}}\|$ is the distance between the final solution and the prior, $\|A\mathbf{x} - \mathbf{y}\|$ is the deviation from the tomographic constrains, and $\|\cdot\|$ is L_2-norm of a vector.

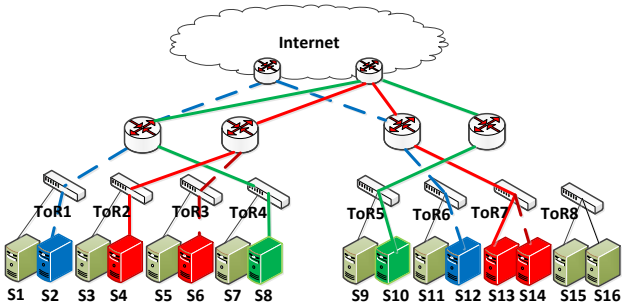


Fig. 8. After reducing the lowly utilized links in Fig. 7

Algorithm 3: Link Utilization-aware Network Tomography

Input: A, $\mathbf{b}$, $\mathbf{y}$, θ, $\mathbf{x}'$
Output: $\hat{\mathbf{x}}$

1 **for** $k = 1$ **to** m **do**
2 **if** $y_k/b_k \leq \theta$ **then**
3 **forall the** $r \in \mathcal{P}_{ij}$ **do**
4 **if** r *contains* l_k **then**
5 $\mathcal{P}_{ij} \leftarrow \mathcal{P}_{ij} - \{r\}$; Adjust A, $\mathbf{x}$ and $\mathbf{y}$
6 **for** $i = 1$ **to** n **do**
7 **for** $j = i + 1$ **to** n **do**
8 **forall the** $r \in \mathcal{P}_{ij}$ **do** $\bar{x}_r \leftarrow x'_{i \leftrightarrow j}/|\mathcal{P}_{ij}|$;
9 $\hat{\mathbf{x}} \leftarrow \mathsf{QuadProgram}(A, \bar{\mathbf{x}}, \mathbf{y})$
10 **return** $\hat{\mathbf{x}}$

To tackle this problem, we first compute the deviation of prior values $\tilde{\mathbf{y}} = \mathbf{y} - A\bar{\mathbf{x}}$, then we solve the following constrained least square problem in Eqn.(5) to obtain the $\tilde{\mathbf{x}}$ as the adjustments to $\bar{\mathbf{x}}$ for offsetting the deviation $\tilde{\mathbf{y}}$.

$$\begin{aligned} \text{Minimize} \quad & \|A\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\| \\ \text{s.t.} \quad & \beta\tilde{\mathbf{x}} \geq -\bar{\mathbf{x}} \end{aligned} \tag{5}$$

We use a tunable parameter β, $0 \leq \beta \leq 1$ 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 $\hat{\mathbf{x}} = \bar{\mathbf{x}} + \beta\tilde{\mathbf{x}}$. According to our experience, we take $\beta = 0.8$ to give a slightly more bias towards the prior.

6.3 The Algorithm Details

We summarize the link utilization aware network tomography in **Algorithm 3**. It takes routing matrix A, the vector of link capacities $\mathbf{b}$, link counts vector $\mathbf{y}$, threshold of link utilization θ and the prior TM among ToRs $\mathbf{x}'$ as the main inputs. Its output is the vector of final estimations of the traffic volume on each path among ToRs $\hat{\mathbf{x}}$. In particular, we first check each of the links to see whether their utilizations are below θ (lines 2). If so, we remove the paths which contain such links from the path set $\mathcal{P}_{ij}$ (includes all paths between the i-th ToR and the j-th ToR), and adjust the matrix A, vector $\mathbf{x}$ and $\mathbf{y}$ by removing the corresponding rows and components (line 5). Here, the utilization of link k is computed by y_k/b_k, where y_k is the load on link k, and b_k is the link's bandwidth. Then for each of the ToR pairs (i, j), and the loads on the remaining paths in $\mathcal{P}_{ij}$ are calculated by averaging the total traffic across the two ToRs $x'_{i \leftrightarrow j}$ (line 8). 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 A (line 9).

Obviously, The dominant running time of the algorithm is spent on $\mathsf{QuadProgram}(A, \bar{\mathbf{x}}, \mathbf{y})$, whose main component Eqn. (5) 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 [24].

7 Evaluation

In this section, we evaluate ATME-PB and ATME-PV with both hardware testbed and extensive simulations.

7.1 Experiment Settings

We implement ATME-PB and ATME-PV together with two representative TM inference algorithms:

- Tomogravity [34] is known as a classical TM estimation algorithm that performs well in IP networks. In contrast to ATME, it assumes traffic flows in the networks follow the gravity traffic model, and traffic exchanged by two ends is proportional to the total traffic on the two ends.
- Sparsity Regularized Matrix Factorization (SRMF for short) [35] 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 ATME-PB and ATME-PV 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:

$$\text{RE}_i = |x_i - \hat{x}_i|/x_i, \tag{6}$$

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:

$$\text{RMSE} = \sqrt{\frac{1}{n_x}\sum_{i=1}^{n_x}(x_i - \hat{x}_i)^2}, \tag{7}$$

$$\text{RMSRE}(\tau) = \sqrt{\frac{1}{n_\tau}\sum_{i=1, x_i > \tau}^{n_x}\left(\frac{x_i - \hat{x}_i}{x_i}\right)^2}. \tag{8}$$

Similar to [34], we use τ to pick up the relative large traffic flows since larger flows are more important for engineering DCNs. n_x is the number of elements in the ground truth $\mathbf{x}$ and n_τ is the number of elements $x_i > \tau$.

7.2 Testbed Evaluation of ATME-PB

7.2.1 Testbed Setup

We use a testbed with 10 switches and about 300 servers as shown in Fig. 9 for our experiments, and the architecture for this testbed is a conventional tree similar to the one in Fig. 1. The testbed hosts a variety of services and part of which has been shown in Fig. 6(a). We gather the resource provisioning information and SNMP link counts for all switches. We also record the flows exchanged among servers by using Linux iptable in each server (not a scalable approach) to form the ground truth. The data are all collected every 5 minutes. The capacities of links are all 1Gbps.

(a) The outside view of our DCN. (b) The inside view of our DCN.

Fig. 9. Hardware testbed with 10 racks and more than 300 servers.

7.2.2 Testbed Results

Fig. 10(a) depicts the relative errors of the three algorithms. As we can see in this figure, our algorithm can accurately infer about 80% of TM elements, while the two other competitive algorithms can only infer less than 60% of them. We can also clearly see that about 99% of percent of inference results of our algorithm has the relative error less than 0.5. An intuitive explanation for this is that our algorithm can clearly separate the traffic into many groups by user in the multi-tenant cloud datacenter. Consequently, it is closer to the real traffic patterns and is more suitable for the assumptions of gravity model after clustering. Therefore, our algorithm can get a more accurate prior TM and final estimated TM than the state-of-art algorithms.

We then present the RMSRE of the algorithms in Fig. 10(b). Clearly we can see that our algorithm has the lowest RMSRE as the flow size increases. When the flow size is less than 4000Mbit (500MBytes), the RMSRE is stable with the flow size, and it starts to decrease after the flow size is greater than 500MBytes, which demonstrates that our algorithm performs even better when handling elephant flows in the network.

7.3 Testbed Evaluation of ATME-PV

7.3.1 Testbed Setup

We use the same testbed as stated in Sec. 7.2, and we also use the Linux iptable in each server to collect the real TM as the ground truth. Besides all the SNMP link counts in the servers and switches, we also gather the service placement information in the controller nodes of the datacenter. All the data are collected every 5 minutes.

7.3.2 Testbed Results

Fig. 11(a) plots the CDF of REs of the three algorithms. Clearly, ATME-PV performs significantly better than the other two: it can accurately estimate the volumes of more than 78% of traffic flows. As the TM of our DCN may not be of low rank, SRMF performs similarly to tomogravity.

We then study these algorithms with respect to the RMSREs in Fig. 11(b). It is natural to see that the RMSREs of all three algorithms are non-increasing with τ, because estimation algorithms are all subject to noise for the light traffic flows, but they normally performs better for heavy traffic flows. However, ATME-PV 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 conduct extensive simulations with larger DCNs in *ns*-3.

7.4 Simulation Evaluation of ATME-PB

7.4.1 Simulation Setup

We adopt both the conventional datacenter architecture [20] and fat-tree architecture [1] in our simulations. For the conventional tree, there are 32 ToR switches, 16 aggregation switches, and 3 core switches; for fat-tree, we use $k = 8$ fat-tree with the same number of ToR switches as the conventional tree, but with 32 aggregation switches, 16 core switches. The link capacities are all set to be 1Gbps. We could not conduct simulations on BCube [13] 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 take the simulated datacenter as a multi-tenant environments, so there are many users in the datacenter and all the users are sending or receiving traffic in their own VM/servers independently. In our simulations, we record the resource provisioning information, which are used to enhance the network tomography results.

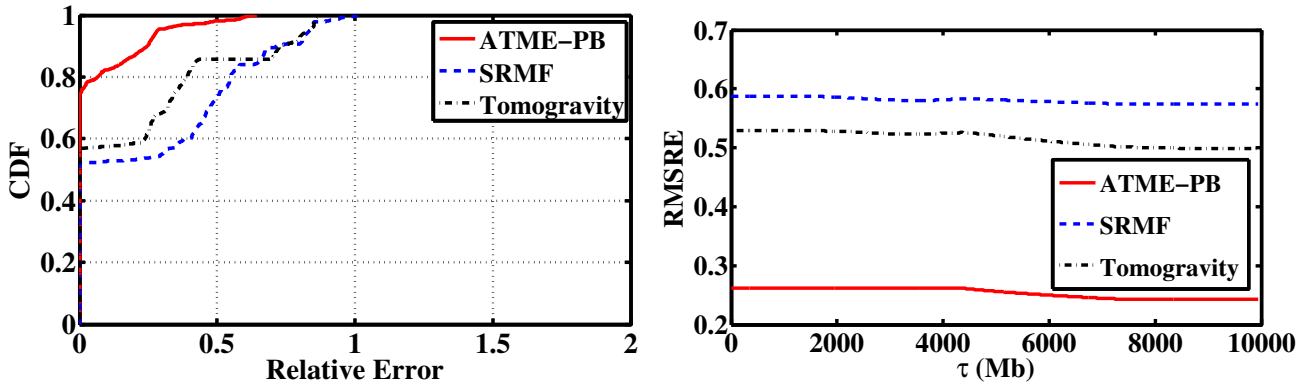
We install both on-off and bulk-send applications in *ns*-3. The packet size is set to be 1400 bytes (varying the packet size has little effect on the performance of our scheme in our experiments), and the flow sizes are randomly generated but still follows the characteristics of real DCNs [6], [11], [22]. For instance, 10% of the flows contributes to about 90% of the total traffic in a DCN [2], [12]. We use TCP flows in our simulations [3], and apply the widely used ECMP [17] 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 the 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.

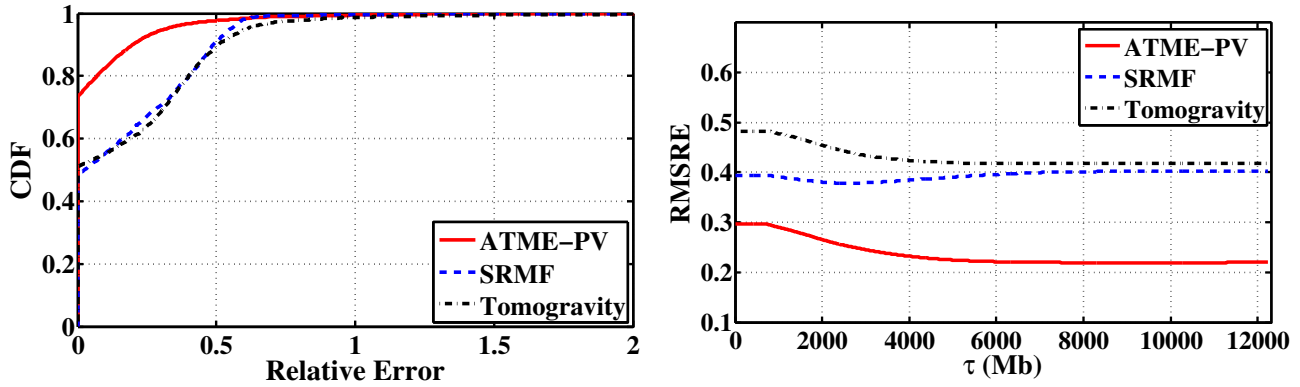
7.4.2 Simulation Results

We set θ to be 0.001. In Fig. 12(a), we plot the CDF of relative errors of the three algorithms under conventional tree



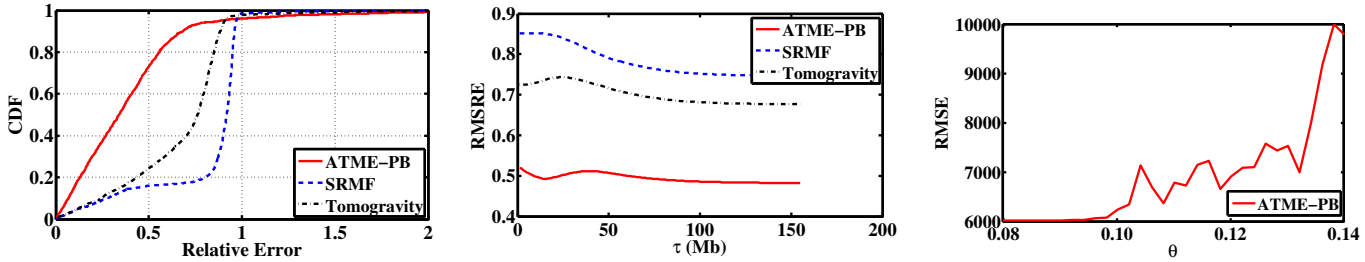
(a) The CDF of RE. (b) The RMSRE under different τ

Fig. 10. The CDF of RE and RMSRE of ATME-PB and two baselines on testbed.



(a) The CDF of RE. (b) The RMSRE under different τ

Fig. 11. The CDF of RE and RMSRE of ATME-PV and two baselines on testbed.

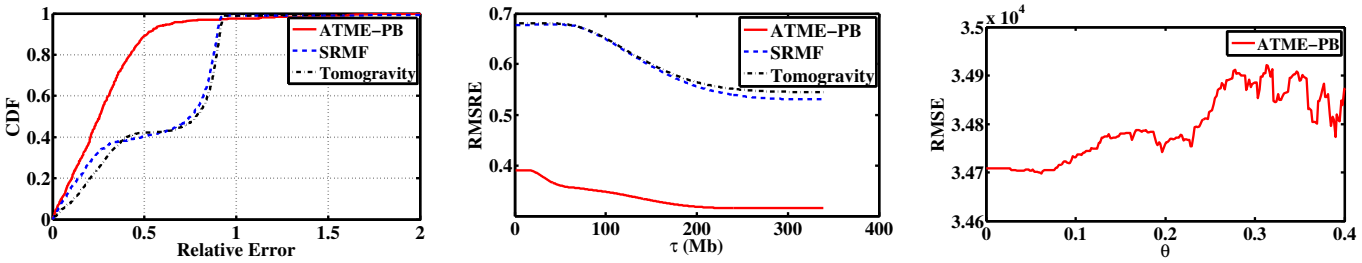


(a) The CDF of RE (b) The RMSRE under different τ (c) The RMSE under different θ.

Fig. 12. The CDF of RE (a), the RMSRE (b), and the RMSE (c) of ATME-PB and two baselines for estimating TM under tree architecture.

architecture. Our algorithm has the lowest relative errors when compared with the other two state-of-art algorithms. More specifically, about 80% of the relative errors are less than 0.5. While for the other two algorithms, about 80% of the relative errors is bigger than 0.5. We draw RMSREs of the three algorithms under different threshold of flow size in Fig.12(b). In this figure, all the three algorithms show declining trends with the increasing size of flows. However, our algorithm still performs the best among the three algorithms. The reason for these two figures is that no matter how the traffic changes in datacenter, our algorithm can accurately identify the communication groups by the easily collected resource provisioning information. When tomogravity fails to get a good prior TM, a bad final estimation would be obtained. For SRMF, it may get the TMs, which are much more sparse than the ground truth due to the rank minimization approach. We also present how the RMSEs change with the threshold θ of link utilization. As we can see that, the curve is stable when θ is smaller than 0.10 and becomes fluctuant afterwards. As removing the lowly utilized links can decrease the running time of the algorithm, it is a good trade off between accuracy and running speed if we set the θ properly (less than 0.10 in this case).

We also set θ to be 0.001 in the fat-tree case. We draw the CDF of relative errors of the three algorithms under fat-tree architecture in Fig. 13(a). Here our algorithm still has the best performance among the three algorithms. About 90% of



(a) The CDF of RE (b) The RMSRE under different τ (c) The RMSE under different θ.

Fig. 13. The CDF of RE (a), the RMSRE (b), and the RMSE (c) of ATME-PB and two baselines for estimating TM under fat-tree architecture.

TABLE 3
The Computing Time (seconds) of ATME-PB, Tomogravity and SRMF under Different Scales of DCNs (Fat-tree)

Switches	Links	Routes	Computing Time			
			ATME-PB		Tomo-gravity	SRMF
			θ =0	θ =0.1		
80	256	7360	4.90	3.60	4.28	251.12
125	500	28625	48.08	40.10	45.32	-

TABLE 4
The Computing Time (seconds) of ATME-PV, Tomogravity and SRMF under Different Scales of DCNs (Tree)

Switches	Links	Routes	Computing Time			
			ATME-PV		Tomo-gravity	SRMF
			θ =0.001	θ =0.01		
51	112	5472	0.54	0.51	2.54	1168.22
102	320	46272	8.12	7.81	73.59	-

the relative errors are smaller than 0.5. The corresponding percentage for the other two algorithms is about 40%. In Fig. 13(b), we can see that the RMSRE of our algorithm decreases from 0.4 and approximates 0 with the increase of the size of flows. Finally, we also depict how RMSE changes with θ in Fig. 13(c). In this figure, the RMSE is stable when θ is lower than 0.1 and increases slowly with θ after that, which also demonstrates that removing some lowly utilized links will not decrease the accuracy of our algorithm. While we will see that it can decrease the running time instead if we set θ properly, as shown in Tab. 3.

Tab. 3 lists the computing time of the three algorithms under fat-tree architecture. Obviously, ATME-PB also performs faster than both tomogravity and SRMF with proper threshold settings. SRMF often cannot deliver a result for several hours when the topology is big. If we slightly increase θ, we may further reduce the computing time, as shown in Tab. 3. In other words, our proposal, ATME-PB, can run even faster without sacrificing accuracy by setting the threshold θ properly as we can see in the table and Fig. 13(c).

7.5 Simulation Evaluations of ATME-PV

7.5.1 Simulation Setup

The simulation setup is almost the same with the setup in Sec. 7.4: we simulate datacenters with conventional tree and fat-tree architecture by *ns*-3. The differences are that we randomly deploy services in the DCN and record the service placement information.

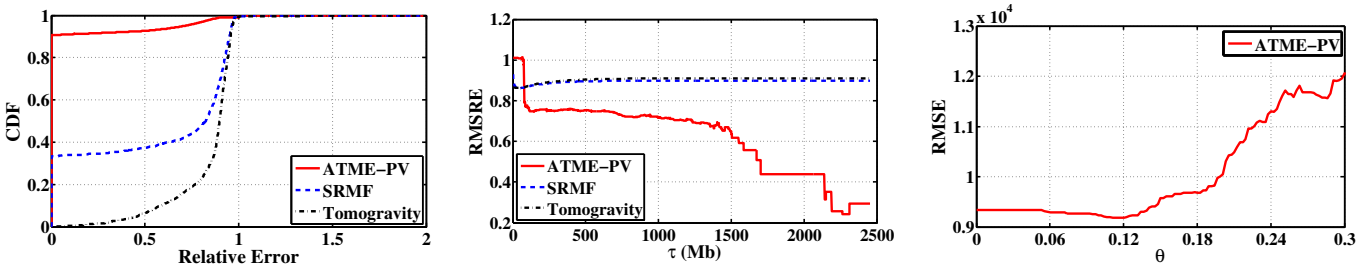
7.5.2 Simulation Results

Fig. 14(a) compares the CDF of REs of the three algorithms under conventional tree architecture and we set $\theta = 0.001$. We can clearly see that ATME-PV has much smaller relative errors. The advantage of ATME-PV over the other two algorithms stems from the fact that ATME-PV 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 the traffic in DCNs does lead to low ranked TM. The worse performance of SRMF (compared with ATME-PV) may be its over-fitting of the sparsity in eigenvalues, according to [22].

We then study the RMSREs of the three algorithms under different τ in Fig. 14(b). Again, ATME-PV exhibits the lowest RMSRE and a (expectable) reducing trend with the increase of τ, while the other two remain almost constant with τ. In Fig. 14(c), we then study how the RMSE changes with the threshold θ of link utilizations. 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.

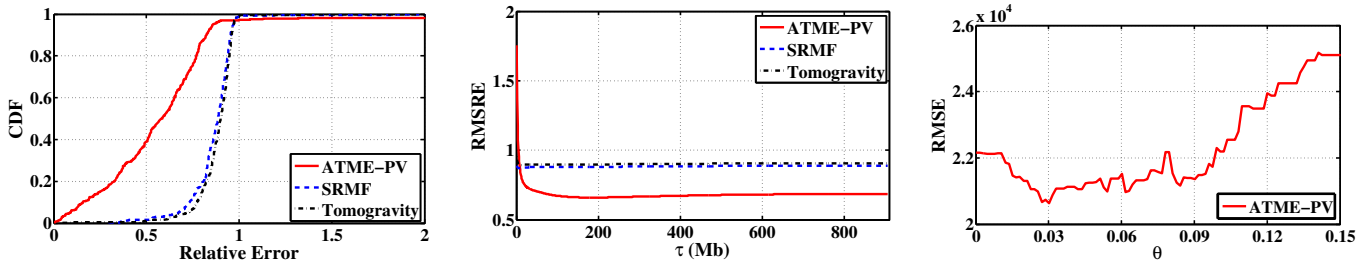
Fig. 15 evaluates the same metrics as Fig. 14 but under fat-tree architecture, which has even more redundant routes. We set $\theta = 0.001$. Since TM in fat-tree DCNs is far more sparse, the errors are evaluated only against the non-zero elements in TM. In general, ATME-PV retains its superiority over others in both RE and RMSRE. The effect of θ becomes more interesting in Fig. 15(c) (compared with Fig. 14(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. 6.1: it trades the estimation accuracy of light flows for that of heavy flows.

Tab. 4 lists the computing time of the three algorithms under conventional tree architecture. Obviously, ATME-PV performs much faster than both tomogravity and SRMF. While both ATME-PV 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. 4. In summary, our



(a) The CDF of RE (b) The RMSRE under different τ (c) The RMSE under different θ.

Fig. 14. The CDF of RE (a), the RMSRE (b), and the RMSE (c) of ATME-PV and two baselines for estimating TM under tree architecture.



(a) The CDF of RE (b) The RMSRE under different τ (c) The RMSE under different θ.

Fig. 15. The CDF of RE (a), the RMSRE (b), and the RMSE (c) of ATME-PV and two baselines for estimating TM under fat-tree architecture.

algorithm has both a higher accuracy and faster running speed compared to the two state-of-art algorithms.

8 Conclusion

To meet the increasing demands for detailed traffic characteristics in DCNs, we make the first step towards estimating the TM among ToRs in both public and private DCNs, relying only on the easily accessible SNMP counters and the datacenter operational information. We pioneer in applying tomographic methods to DCNs by overcoming the barriers of solving the ill-posed linear system in DCNs for TM estimation. We first obtain two major observations on the rich statistics of traffic data in DCNs. The first observation reveals that the TMs among ToRs of DCNs are extremely sparse. The other observation demonstrates that eliminating part of lowly utilized links can potentially increase both overall accuracy and the efficiency of TM estimation. Based on these two observations, we develop a new TM estimation system ATME, which is applicable to most prevailing DCN architectures without any additional infrastructure supports. We validate ATME with both hardware testbed and simulations, and the results show that ATME outperforms the other two well-known TM estimation methods on both accuracy and efficiency. Particularly, ATME can accurately estimate more than 80% traffic flows in most cases with far less computing time.

Although both several recent proposals [8], [30] and our testbed experiments revealed the facts that different services rarely communicate with each other and communications could only happen within the same user's VMs. Some special cases that violates these assumptions actually exist. In our future work, such special cases that fail to follow the two assumptions will be considered. We will try to figure out the correlations among different services and the VMs belonging to different users using learning methods. Besides, we are also interested in combining network tomography with direct measurements offered by software defined network (SDN) to derive a hybrid network monitoring scheme. The Initial results have been reported in [18].

References

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

[2] 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.

[3] 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*, pages 63–74, 2010.

[4] H. Ballani, P. Costa, T. Karagiannis, and A. I. Rowstron. Towards Predictable Datacenter Networks. In *Proc. of ACM SIGCOMM*, pages 242–253, 2011.

[5] D. Belabed, S. Secci, G. Pujolle, and D. Medhi. On Traffic Fairness in Data Center Fabrics. In *Proc. of IEEE CloudNet*, 2014.

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

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

[8] 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*, pages 431–442, 2012.

[9] Y. Cui, H. Wang, X. Cheng, D. Li, and A. Yla-Jaaski. Dynamic Scheduling for Wireless Data Center Networks. *Parallel and Distributed Systems IEEE Transactions on*, 24(12):2365–2374, 2013.

[10] 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*, pages 1629–1637, 2011.

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

[12] 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*, pages 51–62, 2009.

[13] 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*, pages 63–74, 2009.

[14] 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 CoNEXT*, pages 15:1–15:12. ACM, 2010.

[15] 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*, pages 38–49, 2011.

[16] K. Han, Z. Hu, and J. Luo. RUSH: RoUting and Scheduling for Hybrid Data Center Networks. In *Proc. of IEEE INFOCOM*, 2015.

[17] C. Hopps. Analysis of an Equal-Cost Multi-Path Algorithm, 2000.

[18] Z. Hu and J. Luo. Cracking Network Monitoring in DCNs with SDN. In *Proc. of IEEE INFOCOM*, 2015.

[19] Z. Hu, Y. Qiao, and J. Luo. CREATE: CoRrelation Enhanced trAffic maTrix Estimation in Data Center Networks. In *Proc. of IFIP Networing*, pages 1–9, 2014.

[20] C. D. C. Infrastructure. 2.5 Design Guide, 2007.

[21] 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*, pages 2876–2880, 2012.

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

[23] J. P. Kowalski and B. Warfield. Modelling Traffic Demand between Nodes in a Telecommunications Network. In *Proc. of ATNAC95*. Citeseer, 1995.

[24] Y. Luo and R. Duraiswami. Efficient Paraller Non-Negative Least Square on Multi-core Architectures. *SIAM Journal on Scientific Computing*, 33(5):2848–2863, 2011.

[25] 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*, pages 934 – 942, 2014.

[26] 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*, 38(2):69–74, 2008.

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

[28] 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*, pages 7:1–7:13, 2012.

[29] 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*, pages 362–373, 2005.

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

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

[32] N. L. M. Van Adrichem, C. Doerr, and F. A. Kuipers. OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–8, 2014.

[33] M. Yu, L. Jose, and R. Miao. Software defined traffic measurement with OpenSketch. In *Proc. of USENIX NSDI*, pages 29 – 42, 2013.

[34] 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*, pages 206–217, 2003.

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

Zhiming Hu received his BS degree in computer science from Zhejiang University, China, in 2011. He is currently a PhD candidate at the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests are big data, datacenter networking and cloud computing.

Yan Qiao is a lecturer in School of Information and Computer, Anhui Agriculture University. She worked as a post-doctoral fellow in School of Computer Engineering, Nanyang Technological University. She received her Ph.D. degree of computer science from Beijing University of Posts and Telecommunications in 2012. She now focuses on fault diagnosis and network monitoring in IP networks and datacenter networks.

Jun Luo received his BS and MS degrees in Electrical Engineering from Tsinghua University, China, and the PhD degree in Computer Science from EPFL (Swiss Federal Institute of Technology in Lausanne), Lausanne, Switzerland. From 2006 to 2008, he has worked as a post-doctoral research fellow in the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. In 2008, he joined the faculty of the School of Computer Engineering, Nanyang Technological University in Singapore, where he is currently an associate professor. His research interests include wireless networking, mobile and pervasive computing, applied operations research, as well as network security. More information can be found at http://www3.ntu.edu.sg/home/junluo.