Networking 2013 1569710625

Efficient Traffic Matrix Estimation for Data Center Networks

Yan Qiao, Zhiming Hu, Jun Luo
Nanyang Technological University, Singapore - 639798
Email: {yqiao, zhu007, junluo}@ntu.edu.sg

***Abstract*—We address the problem of estimating the real-time nature of traffic flows in data center networks, using the light-weight SNMP data. Unlike the problem of estimating the traffic matrix (TM) across origin-destination (OD) pairs in ISP networks, the traffic flows across servers or ToR (Top of Rack) switch pairs in data center networks are notoriously more irregular and volatile. Although numerous methods have been proposed in past several years to solve the TM estimation problem in ISP networks, none of them could be applied to data center networks directly. In this paper, we make the first step to decompose the data center topology to several clusters by leveraging the characteristics of prevailing data center architecture, which makes TM inference problems in data center networks easy to handle. We also state a lemma to obtain the coarse-grained traffic characteristics of these clusters unbiasedly. Two efficient TM inference algorithms are proposed based on the decomposed topology and the coarse-grained traffic information, which improves the state-of-the-art tomography methods without requiring any additional instrumentation. Finally, comparing with a recent representative TM inference algorithm through intensive simulations, the results show that, i) the data center TM inference problem could be well handled after the decomposition step, ii) our two algorithms outperforms the former one in both speed and accuracy.**

I. Introduction

As data center networks become increasingly central in cloud computing, both research and operations communities have begun to explore how to better design and manage them. The main topics include network structure design [1] [2] [3], traffic engineering [4], capacity planning [5], anomaly detection [6], etc. However, until recently, very little is known about the characteristics of traffic flows within data center networks. For instance, how do traffic volumes exchanged by two servers or ToR switches vary with time? Which server communicates to other servers the most in data center networks? Actually the real-time traffic matrix across servers or ToR switches is a critical input to all above network tasks. Lack of this information impedes both research and operations.

With the increasing demands for the detailed flow level information of data center networks, a few work started to study the flow characteristics of the data centers in their hands [7] [8] [9]. However, the main barrier for them is the difficulty in flow data collection, for the flow level instrumentation is unavailable in most data centers. Besides installing these additional modules requires substantial development and lots of administrative costs.

As the SNMP counters are ubiquitously available in all data center network devices, it is natural to question whether we could borrow the well known tomography methods, which use link level information (such as SNMP bytes counters) to infer the traffic matrices in ISP networks. Unfortunately both Kandula's experiments in real data center networks [9] and our simulations validate that all existing tomography based methods (which will be reviewed in Sec. II) perform poorly in data center networks. This is due to the irregular behavior of end-to-end flows in data center networks and the large quantity of redundant routes between each pair of servers or ToR switches.

In this paper, we demonstrate that the prevailing data center network topologies (including conventional data center architecture [10], Fat-Tree [11], VL2 [2], etc.) can be divided into several clusters, and the complexity of origin TM inference problem can be reduced accordingly. Based on that, we design two efficient algorithms to infer the traffic matrices across these clusters and ToRs within each cluster with high accuracy. Then we verify their performance in our experiments. More specifically, this paper makes the following contributions to the field of data center networking.

We decompose data center network into several clusters to deal with the large quantity of possible routes between OD pairs. By doing this, the complexity of the intractable inference problem can be dramatically reduced, and tomography methods are enabled to work on. We also state a lemma with its proof that the total traffic that exchanged intra/inter each clusters can be unbiasedly inferred from the link loads on switches. Such coarse-grained traffic characteristics are of great significance for the network administrators. For instance, clusters with much more intra traffic may be better designed, for the intra traffic often costs lower network and computational resources. And the administrators should pay more attention to the clusters that communicate a lot with other clusters, whose traffic may cause relative high network delay.

We propose two efficient algorithms to infer the detailed inter and intra clusters' traffic matrices. The first algorithm, which is more appropriate to infer the TMs without explicit structures, utilizes the coarse-grained traffic information to calculate a hypothesis flow volume on each route and then refine the assignments by a least square program. The second one models the inference problem as a state-space network which incorporates both the spacial and temporal structure of TM, and updates the states of TM elements whenever a new observation arrives.

Finally, we design several intensive experiments to validate the performances of our two proposals. Through comparing with a recent representative TM inference method, the experimental results show that our two algorithms outperform the former algorithms in both accuracy and speed, especially for the large scale TMs.

The rest of the paper is organized as follows: we start in Sec. II with the survey of related work; then the detailed problem formulation is described in Sec. III; in Sec. IV, we decompose the data center network topology to several clusters and state a lemma to figure out the coarse-grained traffic nature; we propose two efficient TM inference algorithms in Sec. V and Sec. VI, respectively; and evaluate them through simulations in Sec. VII; finally all the work in this paper is concluded in Sec. VIII.

II. Related Work

As data center networking has recently emerged as a topic of interest, there are numerous studies working on approaches for traffic engineering [4], anomaly detection [5], provisioning and capacity planning [6], etc. However, almost no existing work has devoted to the traffic measurement approaches, although the estimation of traffic flows is a critical input to all above network tasks.

Previous studies [7] [8] have exploited the traffic characteristics within data center networks. The former focuses on cloud data centers that host Web services as well as those running MapReduce, while the latter considers more generic data center networks such as enterprise and campus data centers. Both of them collected packet traces by attaching a dedicated packet sniffer to a SPAN port on the switches in data centers. It is an impractical task to instrument the entire data center network. Therefore, Benson in [8] selected a handful of locations at random per data center and installed sniffers on them.

Kandula et al. [9] studied the nature of data center traffic on a single MapReduce data center. They firstly measure the traffic on data center servers, providing socket level logs. They also question that whether can traffic matrix be inferred from link counters by tomography methods in data center networks as they perform in the ISP counterpart? If they do so, the barrier to understand the traffic characteristics of data centers will be lowered from the expensive instrumentation to analyzing the more easily available SNMP link counters. Unfortunately, they show with their evaluations that tomography performs poorly for data center traffic, due to the following reasons.

i) Most existing topography based methods model the traffic flows at the granularity of volumes exchanged by origin and destination pairs, assuming that there is only one route between an OD pair and the routing matrix will always be constant over time. However, this assumption may be violated in data center networks. There are a great number of redundant routes in data center networks to deal with the congestion, and choosing which route depends on the particular scheduling strategy within the network; ii) To address the under-determined problem in network topography, some methods make additional assumptions such as gravity traffic model [12] and sparsity maximization [13], both of which perform poorly in data center networks, since servers in data centers do not have the same behaviors as terminals in ISP networks; iii) Some proposals make use of historical flow data or portion of flow information to estimate the current flows [13] [14]. These methods suppose flow monitors (such as Netflow) are available in network-wide, and can be turned on if necessary. Actually most data center networks have not been instrumented any flow monitor tools; iv) Methods that exploit the spatio-temporal structure of traffic flows [13] often have high time and space complexity, for the elements in the TM were estimated simultaneously under the global constrains. When TM is in large scale, these inference algorithms cost extremely expensive time and space computational resources. Moreover, when new observations or requirements arrive, they need to start over again.

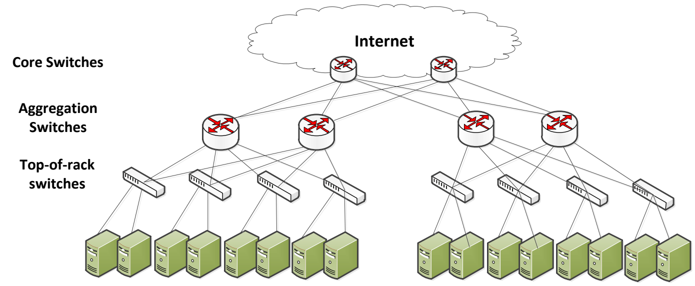


Fig. 1. An Example of Conventional Data Center Network Architecture (adopted from Cisco [10])

In this paper, we aim at designing an efficient tool to infer the nature of data center flows with high accuracy without requiring any additional instrumentations besides the ubiquitous SNMP data collected by switches. With the new powerful tool, the data center administrators could learn the real-time network traffic details at any moment they need.

III. Problem Formulation Background

We consider a typical data center network as shown in Fig. 1, consisting of ToR switches, aggregation switches and core switches connecting with Internet. We can poll the SNMP MIBs on the network switches for bytes/packets-in and bytes/packets-out at granularities ranging from 1 minute to 30 minutes. The SNMP data can also be interpreted as switch loads, which equals to the summation of volumes of flows that traverse the corresponding switches. The traffic volumes exchanged by servers or ToR switches over different time periods form a traffic matrix (or TM). In this paper, we aim at inferring the ToR TM from the switch loads.

We represent switches in the network as $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$, where m is the number of switches. Let $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_m\}$ denote the traffic loads collected by SNMP counters on the switches, and $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ denote the traffic flow volumes on the routes between ToR switch pairs, where n is the number of all available routes in data center networks. $X_i(t)$ and $Y_j(t)$ represent the corresponding traffic at discrete time t. The correlation between

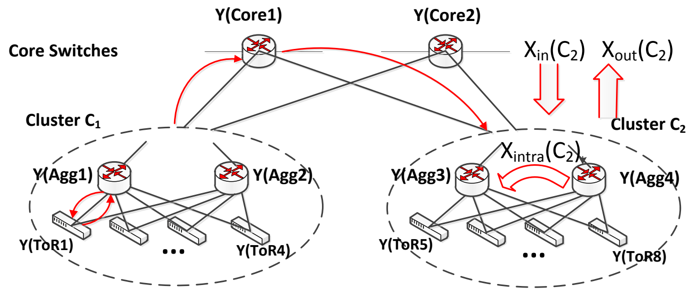


Fig. 2. Decomposing Fig. 1 to Two Clusters

$\mathbf{X}(t)$ and $\mathbf{Y}(t)$ can be formulated as

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

Here, $\mathbf{A}$ denotes the routing matrix, where each row represents a switch and each column represents the flows' routes. $A_{ij} = 1$ when the route of X_j traverses the switch S_i, and $A_{ij} = 0$ otherwise. Although Eqn. (1) is a typical linear system, it is very difficult to solve. Since the number of equations is much less than the number of unknown variables, the problem is highly ill-posed. Especially, the case for the conventional data center topologies (as shown in Fig. 1) makes the problem even worse—as many ToR switches connect to one or a few high-degree aggregation switches, the number of available switch measurements is small. For example in Fig. 1, the network consists of 8 ToR switches, 4 aggregation switches and 2 core switches. The number of possible routes between all ToR switches is more than 100, while the number of observations is only 14. Moreover, the number of routes grows dramatically with the network scale. When the numbers of corresponding switches double, the number of routes will grow up to thousands. Hence, inferring TMs of data center networks directly from Eqn. (1) is impractical. In the next section, we will introduce a novel methodology which turns the intractable problem into an easy-handle one.

IV. Data Center Topologies Decomposing

Due to the special architecture of prevailing data center networks, the TM across ToR switches can be decomposed to several smaller TMs. The possibility of the decomposition operation is based on the locally tree-like structure of data center topologies. For example in Fig.1 each ToR connects with two aggregations. Then the two aggregation switches together with 4 ToR switches can form a conditional independent cluster. That is to say, if we know the traffic flows that go in (or out of) Agg_1 and Agg_2, the traffic flows that go in (or out of) $ToR_1 \sim ToR_4$ are independent to the traffic flows going in (or out of) $ToR_5 \sim ToR_8$. Hence, Fig. 1 can be decomposed to two clusters as shown in Fig. 2. Therefore, the problem can be turned into inferring the TM across clusters (inter inference) and the TM across the ToR switches intra each cluster (intra inference), both of which are much more determined than the original problem.

Note that, although the inter and intra TMs that we aim to estimate can reveal most of the traffic characteristics of data center networks, we still could not obtain the traffic flows directly if their ends are in different clusters. Instead, we could learn from intra TM that how much traffic that originates from ToR_i and goes to another clusters and how much traffic that originates from another cluster and enters in ToR_j. We could also learn from inter TM the detailed traffic flows across clusters. We will carry the work on to estimate the TM across ToRs in different clusters in our future work based on the intra and inter TMs.

Actually, decomposing data center topology not only reduces the complexity of inference problem but also motivates us to figure out the coarse-grained traffic characteristics for each cluster, including the total traffic exchanged intra each cluster, the total traffic going out of each cluster and the total traffic entering into each cluster. In the rest part of this section, we first give some definitions then state Lemma IV.1 to demonstrate the coarse-grained traffic information of clusters can be inferred unbiasedly from the loads on switches.

Taking Fig. 2 for example, we denote the loads on the core switches and aggregation switches as $\mathbf{Y}(Core)$ and $\mathbf{Y}(Agg)$, respectively. The bold letter means it is a set of variables. Practically, it is easy to distinguish the "in" and "out" traffic flows on ToR switches. We represent the "out" flows which come from servers as $\mathbf{Y}_{out}(ToR)$, and the "in" flows which are transferred to servers as $\mathbf{Y}_{in}(ToR)$. Obviously, the total loads on ToR switches $\mathbf{Y}(ToR) = \mathbf{Y}_{out}(ToR) + \mathbf{Y}_{in}(ToR)$.

Suppose the underlying data center network can be decomposed to s clusters which are grouped in $\mathbf{C} = \{C_1, C_2, \ldots, C_s\}$. $\mathbf{Y}(Agg_{C_i})$ denotes the traffic loads on the aggregation switches in cluster C_i, and $\mathbf{Y}_{in}(ToR_{C_i})$ ($\mathbf{Y}_{out}(ToR_{C_i})$) are the "in" ("out") loads on the ToRs within cluster C_i.

Lemma IV.1. *Suppose $X_{intra}(C_i)$ is the total traffic exchanged intra the cluster C_i, $X_{in}(C_i)$ is the total traffic entering C_i from other clusters, and $X_{out}(C_i)$ is the total traffic going out of C_i. Then the following equations hold.*

$$X_{intra}(C_i) = \sum_{ToR_k \in ToR_{\mathbf{C_i}}} (Y_{in}(ToR_k) + Y_{out}(ToR_k)) - \sum_{Agg_j \in Agg_{\mathbf{C_i}}} Y(Agg_j) \tag{2}$$

$$X_{in}(C_i) = \sum_{Agg_j \in Agg_{\mathbf{C_i}}} Y(Agg_j) - \sum_{ToR_k \in ToR_{\mathbf{C_i}}} Y_{out}(ToR_k) \tag{3}$$

$$X_{out}(C_i) = \sum_{Agg_j \in Agg_{\mathbf{C_i}}} Y(Agg_j) - \sum_{ToR_k \in ToR_{\mathbf{C_i}}} Y_{in}(ToR_k) \tag{4}$$

Proof: From the ToRs' "out" traffic aspect, we have

$$\sum_{ToR_k \in ToR_{\mathbf{C_i}}} Y_{out}(ToR_k) = X_{out}(C_i) + X_{intra}(C_i) \tag{5}$$

Similarly, from the ToRs' "in" traffic aspect, we have

$$\sum_{ToR_k \in ToR_{\mathbf{C_i}}} Y_{in}(ToR_k) = X_{in}(C_i) + X_{intra}(C_i) \tag{6}$$

Combining Eqn. (5) and Eqn. (6), we have

$$\sum_{ToR_k \in ToR_{\mathbf{C_i}}} (Y_{in}(ToR_k) + Y_{out}(ToR_k)) = X_{in}(C_i) + X_{out}(C_i) + 2 \cdot X_{intra}(C_i) \tag{7}$$

As we have known that the counters on aggregation switches Agg_{C_i} can only be triggered by the traffic exchanged within the cluster, the traffic going out of the cluster and the traffic entering into the cluster, which means

$$\sum_{Agg_j \in Agg_{\mathbf{C_i}}} Y(Agg_j) = X_{intra}(C_i) + X_{out}(C_i) + X_{in}(C_i) \tag{8}$$

Thus we have

$$\sum_{ToR_k \in ToR_{\mathbf{C_i}}} (Y_{in}(ToR_k) + Y_{out}(ToR_k)) = \sum_{Agg_j \in Agg_{\mathbf{C_i}}} Y(Agg_j) + X_{intra}(C_i) \tag{9}$$

Eqn. (9) can also be written as

$$X_{intra}(C_i) = \sum_{ToR_k \in ToR_{\mathbf{C_i}}} (Y_{in}(ToR_k) + Y_{out}(ToR_k)) - \sum_{Agg_j \in Agg_{\mathbf{C_i}}} Y(Agg_j) \tag{10}$$

Thus Eqn. (2) has been proven. It is not difficult to learn from (5), (6) and (8) that both (3) and (4) in Lemma IV.1 are also true. ■

The coarse-grained traffic nature of clusters is unbiased and simple in calculation. It is significant to the data center administrators and network designers for making more convincing decisions based on the real-time traffic information.

V. TM Inference Method Based on Coarse-grained Traffic Characteristics

In this section, we propose an efficient TM inference algorithm based on the coarse-grained traffic characteristics (TMBCT). The new algorithm first calculates a hypothesis set of TM elements based on the coarse-grained traffic from Lemma IV.1 and then refines the hypothesis using the constraints of switch observations by least square program.

A. Hypothesis for TM Elements

The hypothesis for inter TM elements are calculated based on the gravity traffic model [15]. As we mentioned in Sec. II, traffic flows across servers or ToR switches in data center networks are revealed to violate the gravity traffic model [9], since the traffic flows are often irregular and burst. However, the data center network after being decomposed operates in a different situation. We found that traffic flows across clusters are relatively smooth and steady. Moreover, the clusters and cores are likely to play the same roles as the terminals and routers in the Internet. The sub-networks intra clusters have the similar situation as well. Thus, in this section we set the hypothesis for TMs using the gravity traffic model, which will be demonstrated feasible through the experiments in Sec. VII.

At their simplest, gravity models are based on the assumption of a simple proportionality relationship [16]

$$X_{ij} \propto Y_i^{out} \cdot Y_j^{in} \tag{11}$$

Where X_{ij} denotes the traffic from the i^{th} end to the j^{th}. Y_i^{out} denotes the total traffic going out at the i^{th} end, while Y_j^{in} denotes the total traffic entering at the j^{th} end.

1) Hypothesis for Inter TM: Since we have known the out (in) traffic volumes of all clusters from Lemma IV.1, the hypothesis of inter TM elements $X^H(C_{ij})$ can be formulated by

$$X^H(C_{ij}) = X_{out}(C_i) \cdot \frac{X_{in}(C_j)}{\sum_{C_k \in \mathbf{C}} X_{in}(C_k)} \tag{12}$$

Suppose N_c is the number of core switches. Since there is only one hop on each route between cluster pairs, there are N_c routes from C_i to C_j. We define a set of weights $\mathbf{w} = \{w_1, \ldots, w_{N_c}\}$ for the routes between two clusters, where $\sum_{i=1}^{N_c} w_i = 1$. For example, in Fig. 2, there are two routes from C_1 to C_2 : $C_1 \rightarrow Core_1 \rightarrow C_2$ and $C_1 \rightarrow Core_2 \rightarrow C_2$. If we suppose they have the equal weight, then $w_1 = w_2 = \frac{1}{2}$. Thus the traffic allocated on the k^{th} route is $w_k \cdot X^H(C_{ij})$.

2) Hypothesis for Intra TM: Similar to inter traffic flows, we also model the intra traffic exchanged by ToR switches as gravity traffic model. Different from inter traffic, there is a portion of traffic that goes out of (or enters into) the cluster. Therefore, we partition the total out (or in) loads on ToR switches into two parts: traffic going to (or coming from) ToRs outside the cluster (denoted by $\mathbf{X}(ToR^{out})$ or $\mathbf{X}(ToR^{in})$) and traffic going to (or coming from) ToRs within the cluster (equal to $\mathbf{X}_{out}(ToR) - \mathbf{X}(ToR^{out})$ or $\mathbf{X}_{in}(ToR) - \mathbf{X}(ToR^{in})$). We also define a parameter θ_i^{out} as the possibility that traffic goes out of cluster C_i, and θ_i^{in} as the possibility of traffic comes from other clusters, which can be calculated by

$$\theta_i^{out} = \frac{X_{out}(C_i)}{\sum_{ToR_j \in ToR_{\mathbf{C_i}}} X_{out}(ToR_j)} \tag{13}$$

and

$$\theta_i^{in} = \frac{X_{in}(C_i)}{\sum_{ToR_j \in ToR_{\mathbf{C_i}}} X_{in}(ToR_j)} \tag{14}$$

The hypothesis for $X(ToR_i^{out})$ and $X(ToR_i^{in})$ are

$$X^H(ToR_i^{out}) = \theta_i^{out} \cdot X_{out}(ToR_i) \tag{15}$$

and

$$X^H(ToR_i^{in}) = \theta_i^{in} \cdot X_{in}(ToR_i) \tag{16}$$

For the intra traffic flows, let $X^H(ToR_{ij})$ denote the hypothesis for traffic volumes from ToR_i to ToR_j within the cluster C_k, which can be calculated by

$$X^H(ToR_{ij}) = (1 - \theta_i^{out}) X_{out}(ToR_i) \cdot \frac{(1 - \theta_i^{in}) X_{in}(ToR_j)}{X_{intra}(C_k)} \tag{17}$$

Suppose there are N_a aggregation switches in cluster C_k. Then the number of available routes between ToR_i and ToR_j within a cluster is N_a as well, for there is only one hop on each route. We define $\mathbf{v} = \{v_1, \ldots, v_{N_a}\}$ as the weights of routes that go through the corresponding aggregation switches, where $\sum_{i=1}^{N_a} v_i = 1$. The traffic allocated on the k^{th} route of each hypothesis set is $v_k \cdot X^H(ToR_{ij})$, $v_k \cdot X^H(ToR_i^{out})$ and $v_k \cdot X^H(ToR_i^{in})$. As well as $\mathbf{w}$, the assignments to $\mathbf{v}$ depend on the routing strategy applied to the underlying network.

B. Refine the Hypothesis TMs by Least Square Program

The intuition to refine the hypothesis of inter and intra TM elements is finding the solution that is most close to the hypothesis and subject to the switch loads we have observed. This problem can be formulated as a least square program.

For the inter TM, the program can be formulated as

$$\begin{aligned} \text{Minimize} \quad & ||\mathbf{X(C)} - \mathbf{X^H(C)}|| \\ \text{s.t.} \quad & \mathbf{A^{inter}X(C)} = [\mathbf{Y}(Core), \mathbf{Y}_{out}(C), \mathbf{Y}_{in}(C)] \end{aligned} \tag{18}$$

where $\mathbf{X}(C)$ is the set of directed traffic flows from one cluster to another, and $\mathbf{X}^H(C)$ is its hypothesis value. $\mathbf{A}^{inter}$ is the routing matrix where each column represents a directed route between a cluster pair. $\mathbf{Y}(Core)$ is the set of loads on the core switches, $\mathbf{Y}_{out}(C)$ and $\mathbf{Y}_{in}(C)$ are the sets of out and in traffic of each cluster.

For the intra TM of cluster C_i, the program can be formulated as

$$\begin{aligned} \text{Minimize} \quad & ||\mathbf{X(C_i)} - \mathbf{X^H(C_i)}|| \\ \text{s.t.} \quad & \mathbf{A^{C_i}X(C_i)} = [\mathbf{Y}(Agg_{C_i}), \mathbf{Y}(ToR_{C_i})] \end{aligned} \tag{19}$$

where, $\mathbf{X(C_i)}$ consists of the set of directed traffic volumes exchanged by ToR switch pairs within cluster C_i, the traffic from each ToR switches going out of C_i, and the traffic entering each ToR switch in C_i from other clusters. $\mathbf{X}^H(C_i)$ is the corresponding hypothesis values. $\mathbf{A^{C_i}}$ is the routing matrix, where each column represents a directed route between two ToR switches within C_i. Note that we denote a flow route from ToR_i going out of the cluster by a column that only contains the source ToR_i and the aggregation switches, and similarly with the flow entering ToR_i from other clusters. $\mathbf{Y}(Agg_{C_i})$ is the set of loads on aggregation switches. $\mathbf{Y}(ToR_{C_i})$ consists of total in and out traffic loads on ToR switches in C_i, the total traffic entering in and going out of C_i, and the total traffic exchanged intra C_i. $||\cdot||$ in Eqn. (17) and Eqn. (18) is the L_2 norm of the vector (i.e., the Euclidean distance).

VI. Adaptive TM Inference Method based on Linear State-space Model

The TM inference method based on coarse-grained traffic characteristics simply calculates a solution using the observations on current time slice. The historical observations and solutions would not be used once a new observation set arrives. However, motivated by the studies in [13] and [14], we found in our experiments that the traffic flows over multiple time periods also have a spatial and temporal structure as the flows in ISP networks. To leverage this characteristics of TM structure, we model the TM inference problem as a linear state-space network which always incorporates both historical

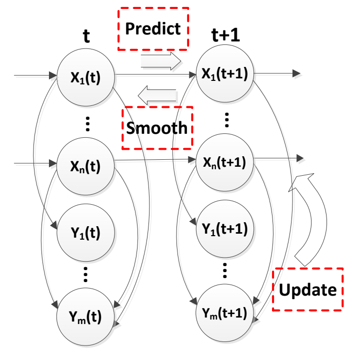


Fig. 3. Linear Space-state Model for TM Inference Problem

and fresh observations. We then design an efficient adaptive inference algorithm based on the model to infer the TM with a spatio-temporal structure.

A. Modeling

The linear state-space network is defined as

$$\begin{cases} X(t+1) = \mathbf{F} \cdot X(t) + Q(t) \\ Y(t+1) = \mathbf{A} \cdot X(t+1) + V(t+1) \end{cases} \tag{20}$$

where $\mathbf{F}$ is a known matrix, correlating the state $X(t+1)$ with its state on the last discrete time t. Q is an i.i.d. Gaussian process with covariance matrix σ_Q, indicating the uncertainty of the relationship. $Y(t+1)$ is the set of observations on time $t+1$, which correlates $X(t+1)$ by matrix $\mathbf{A}$. V is also an i.i.d. Gaussian process with covariance matrix σ_V, representing the observation noise.

In our TM inference problem, we represent the inter (intra) traffic flow on each route as a state, and the switch loads together with the coarse-grained traffic of clusters as the observations. The state-space network for the TM inference problem is shown in Fig. 3, where $X_i(t)$ denotes the traffic on the i^{th} route of inter (intra) networks, and $Y_j(t)$ is the j^{th} observation. The arrow connecting $X_i(t)$ and $Y_j(t)$ means $X_i(t)$ contributes the total traffic of $Y_j(t)$. Our goal is to estimate the states of $X_1(t) \sim X_n(t)$, $t = 1, \ldots, T$, given the observations $Y_1(t) \sim Y_m(t)$, $t = 1, \ldots, T$, where n is the number of possible routes and m is the number of observations.

B. A Variant Kalman Filtering Algorithm

Kalman filter is one of the best inference algorithms for linear state-space network. It can achieve an optimal solution to (20) under the assumption that the probability density of the state at every time step is Gaussian. In our model, we assume traffic state $X_i(t)$ follows a Gaussian distribution $N(\mu_i(t), \sigma_i(t))$. We treat $\mu_i(t)$ as the estimation of state $X_i(t)$, and $\sigma_i(t)$ as the uncertainty on the estimation. In the rest of the paper, we use $X_i(t)$ uniformly to represent both the state and its mean value. Obviously, the observation $Y_j(t)$ follows a multivariate Gaussian distribution.

The Kalman filtering includes two steps: predict and update. The former predicts the states of traffic flows in the next time slice based on their current states, and the latter updates the predicted states with the new observations. In our traffic inference method, we add a smooth step to the Kalman filter to adjust the states on preceding time slices using their new updated states in order to refine the TM structure. The relationship of the three steps are shown in Fig. 3.

1) Predict Step: In the first step, the states of traffic flows on the next time slice are calculated by

$$\mathbf{X}(t+1) = \mathbf{F}\mathbf{X}(t) \tag{21}$$

where $\mathbf{F}$ is the correlation among traffic flows on adjacent time slices. The diagonal elements of $\mathbf{F}$ indicate a temporal correlation within a single flow, while the off-diagonal elements capture the spatial correlation across different traffic flows.

Meanwhile, the covariance of the states is calculated by

$$\sigma(t+1) = [\sigma^2(t) + \sigma_Q^2(t)]^{\frac{1}{2}} \tag{22}$$

where, $Q(t)$ denotes the uncertainty about the matrix $\mathbf{F}$. In our experiments, we simply set $\mathbf{F}$ to a unit matrix and $Q(t)$ with zero mean and relative large covariance, which presents a promising result.

2) Update Step: In this step, the states of flows will be updated using the new observations. To learn the gap between the states we expected and their ground truth, we first calculate the deviation between the observations we expected and their true values by

$$\mathbf{I}(t+1) = \mathbf{Y}(t+1) - \mathbf{A}\mathbf{X}(t+1) \tag{23}$$

Here, $\mathbf{I}(t+1)$ is called measurement innovation, which we can use to adjust the predicted state of traffic flows by

$$\mathbf{X}(t+1) = \mathbf{X}(t+1) + \mathbf{K}(t+1)\mathbf{I}(t+1) \tag{24}$$

Here, $\mathbf{K}(t+1)$ is called Kalman gain matrix, which is chosen by minimizing the posteriori error covariance $E(\bar{\mathbf{X}}(t+1) - \mathbf{X}(t+1))^2$, where $\bar{\mathbf{X}}(t+1)$ is the real states of the traffic flows.

Through some regular linear algebra steps, the Kalman gain $\mathbf{K}(t+1)$ can be written as

$$\mathbf{K}(t+1) = [\sigma^2(t+1)\mathbf{A}^T] \cdot [\mathbf{A}\sigma^2(t+1)\mathbf{A}^T + \sigma_V^2(t+1)]^{-1} \tag{25}$$

where $\mathbf{A}^T$ is the transpose of $\mathbf{A}$. The physical meaning of the update step is that: we try to find an optimal state for the traffic flow, which is a tradeoff between the predicted state and the linear-equation constraints. If the uncertainty about the predicted state $\mathbf{X}$ is small enough, the Kalman filter will incline to the predicted state rather than the observation constraints, and vice versa. The filter keeps refining the uncertainty of the states by measuring the performance of their prediction.

In some cases, the Kalman gain matrix calculated by Eqn. (25) does not make any physical sense in our inference model, for substituting it into Eqn. (24) may lead to negative traffic volumes. Therefore, we modify (25) to a least square program by

$$\begin{aligned} \text{Minimize} \quad & ||\mathbf{K}(t+1)[\mathbf{A}\sigma^2(t+1)\mathbf{A}^T] - \sigma^2(t+1)\mathbf{A}^T|| \\ \text{s.t.} \quad & \mathbf{K}(t+1)\mathbf{I}(t+1) > -\mathbf{X}(t+1) \end{aligned} \tag{26}$$

The uncertainty of the states is updated by

$$\sigma(t+1) = [(\mathbf{I}(t+1) - \mathbf{K}(t+1)\mathbf{A})\sigma^2(t+1)]^{\frac{1}{2}} \tag{27}$$

3) Smooth Step: In the last step, we use the new updated states of traffic flows to smooth the preceding states backward. The smooth process is formulated as

$$\hat{\mathbf{X}}(t) = \mathbf{X}(t) + \mathbf{J}(t+1)(\mathbf{X}(t+1) - \mathbf{F}\cdot\mathbf{X}(t)) \tag{28}$$

Here, $\hat{\mathbf{X}}(t)$ represent the smoothed states of flows on the t time slice. Similar to $\mathbf{K}$, $\mathbf{J}$ is called Kalman smooth gain, which can be calculated by

$$\mathbf{J}(t+1) = [\sigma^2(t)\cdot\mathbf{F}^T]\cdot[\mathbf{F}\cdot\sigma^2(t)\cdot\mathbf{F}^T + \sigma_Q^2(t+1)]^{-1} \tag{29}$$

where, $\mathbf{F}^T$ is the transpose of $\mathbf{F}$.

We also modify Eqn. (29) to be a least square problem to eliminate the unreasonable results.

$$\begin{aligned} \text{Minimize} \quad & ||\mathbf{J}(t+1)[\mathbf{F}\sigma^2(t)\mathbf{F}^T + \sigma_Q^2(t+1)] - \sigma^2(t)\mathbf{F}^T|| \\ \text{s.t.} \quad & \mathbf{J}(t+1)\cdot[\mathbf{X}(t+1) - \mathbf{F}\cdot\mathbf{X}(t)] > -\mathbf{X}(t) \end{aligned} \tag{30}$$

The uncertainty of the states in last discrete time should also be refined by

$$\hat{\sigma}(t) = [\sigma^2(t) + \mathbf{J}(t+1)(\sigma^2(t) - \sigma_Q^2(t+1))\mathbf{J}^T(t+1)]^{\frac{1}{2}} \tag{31}$$

Consider the scalability of the method, we only smooth the flow states of the direct precursor rather than passing the smooth step back to the beginning. In practice, the effect of smooth operation becomes much weaker when it passes to the last but one discrete time.

The coarse traffic based method and the linear state-space based method are applicable to different network episodes. The former one is more appropriate for the network whose TM does not have an explicit structure, while the latter performs better in the opposite situation.

C. Algorithm Pseudocode

Algorithm 1 describes the adaptive TM inference algorithm based on linear state-space model (TMBLS). We set the initial input states of traffic flows $\mathbf{X}(0)$ as the hypothesis of TM on the first time slice, which is calculated by the method described in Sec. V-A. The algorithm first predicts the state of $\mathbf{X}(t)$ based on $\mathbf{X}(t-1)$, and then updates the state of $\mathbf{X}(t)$ by the observation $\mathbf{Y}(t)$. Finally, we perform the smooth backwards to the flow states $\mathbf{X}(t-1)$ using their corresponding new states. Note that, algorithm 1 unifies both inter and intra TM inference by substituting the input parameters by the inter or intra TM parameters.

VII. Simulation

A. Experiment Setup

Topology: We adopt the conventional data center architecture [10] to conduct our experiments, although our methods are also applicable to other data center architectures such as Fat-Tree [11], VL2 [2] and so on. The topology is a medium scale data center with 32 ToR switches, 16 aggregation switches and

Algorithm 1 Adaptive TM Inference Algorithm Based on Linear State-space Model)

```
procedure TMINFERENCE(X(0), Y(1 : k), A, F, σ(0), σ_Q(0), σ_V(0),T)
    for each t ∈ 1 : T do
        [X(t)]← Predict(X(t − 1), σ(t − 1),σ_Q(t − 1))
        [X(t),σ(t)]
        ← Update(X(t),Y(t),σ(t), σ_V(t))
        [X(t − 1),σ(t − 1)]
        ← Smooth(X(t−1),X(t),σ(t−1),σ(t),σ_Q(t−1))
    end for
end procedure
```

8 core switches. For a rack, there are 20 servers connecting to each ToR switch, where the link capacities are set to be 1Gbps. At the meantime, the capacities of the links between switches are also 1Gbps.

Traffic generation: We generate the traffic flows based on the study of the traffic characteristics of data center networks [7] [8] [9]. Specifically, we randomly select $1 \sim 10$ server(s) under each ToR switch, and let them generate flows to all of other servers; the numbers of packets within the flows that go out of the cluster follow the distribution $Log-Normal(4,1)$, while the numbers of packets for the flows exchanged within a cluster follow $Log-Normal(10,1)$; the size of each packet is around 1400 bytes. We use TCP flows to simulate the real data center traffic since most of the data center traffic is TCP traffic [17]. For routing strategies, we use both ECMP (equal cost multiple path) and NIx-Vectors [18] (a source routing traffic engineering method which aims to decrease the link congestions). However, we found very little difference between the two routing strategies. Therefore, we only report the results for the prevailing ECMP strategy, and the weights of routes between two ends are set to be equal.

Data Collections: We record the total number of packets that enter and go out of each switch in the network every 5 minutes. We also record the total packets of flows on each route in the corresponding time periods as the ground truth.

B. Algorithms and Metrics

We implement our two TM inference algorithms: TMBCT and TMBLS, together with a recent representative TM inference algorithm—Sparsity Regularized Matrix Factorization (SRMF for short) [13] which leverages the Spatio-temporal structure of traffic flows, and utilizes the compressive sensing method to infer the missing data in TM by sparsity maximization. The TM inference problem that we aim to address is a special case that all elements in TM are missing. In our experiments, the SRMF algorithm can not provide a result in a limited time (more than 24 hours) when apply it directly to the data center tomography problem. For the number of columns of the origin routing matrix is more than 15000 while the number of its rows is only 56. Therefore, we apply all the three algorithms on the decomposed data center topology, and reconfigure the parameters of the TM inference problem based on Lemma IV.1.

We quantify the performance of the three algorithms from three aspects: The cumulative distribution of relative error (CDRE), the mean relative error (MRE), and the computing time.

The relative error (RE) is formulated as

$$RE_i = \frac{|X_i - \hat{X}_i|}{X_i} \tag{32}$$

Here X_i denotes the true TM element and $\hat{X}_i$ is the corresponding estimated value. The cumulative distribution of relative error indicates the percentage of accurate results (or deviated results).

The mean relative error (MRE) introduced by [19] can be calculated by

$$MRE = \frac{1}{N_\varepsilon} \cdot \sum_{X_i > \varepsilon} \frac{|X_i - \hat{X}_i|}{X_i} \tag{33}$$

The sum is taken over the flow volumes larger than a threshold ε, and N_ε is the number of elements that larger than ε. In our experiments, we set ε to 10 packets, which can shield most minor flows that we do not concern, such as some small ACK flows. The algorithm's MRE indicates the global deviation of the estimation from the ground truth.

For the computing time, we concern the time period from the time we input the inter and intra inference parameters to the algorithms, to the time they output their estimation for all TM elements. All three algorithms are implemented by Matlab (R2012a) on 6-core Intel Xeon CPU @2.93GHz, with 12GB of memory and the Mac OS X 10.6.5(10H574).

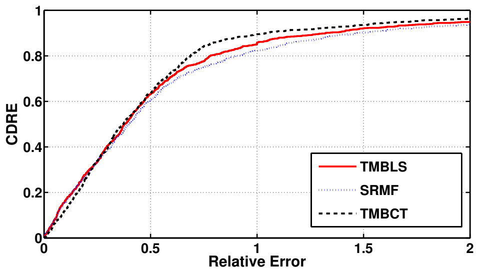
All the figures and tables below show results averaged over 10 runs.

C. Simulation Results

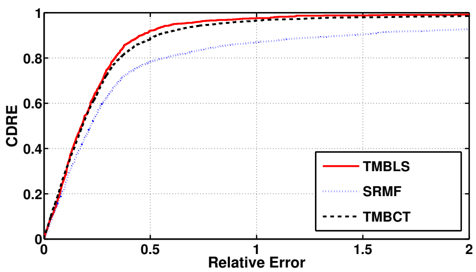
1) The Cumulative Distribution of Relative Error: Fig. 4 and Fig. 5 compare the CDRE of the three algorithms inferring the TMs with different numbers of time slices. From the figures we can see that both our two algorithms outperform SRMF on all kinds of networks. The promising results are partially due to our algorithms make use of the coarse-grained flow information obtained by Lemma IV.1. Such as TMBCT computes the hypothesis flow values based on the coarse-grained data on every time slice, and TMBLS utilizes the traffic information as its initial input.

From the figures, we can also find that the three algorithms perform differently in intra TM inference and inter TM inference: TMBLS and SRMF perform better in intra cluster inference than in inter inference, while TMBCT performs in the opposite way. This implies traffic exchanged within each cluster has a relatively explicit structure than the traffic exchanged inter clusters. And the inter TM inference problem may be more appropriate for gravity traffic model.

We can see from Fig. 5, when the number of time slices increases, the gap between SRMF and our two algorithms

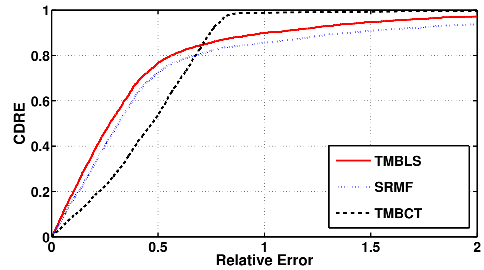


(a) Inference for Inter TM

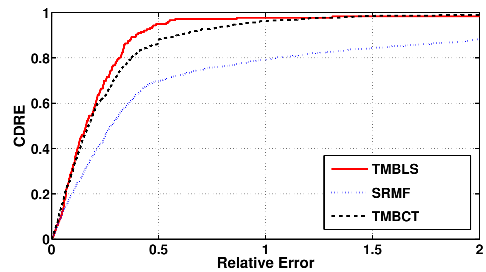


(b) Inference for Intra TM

Fig. 4. The Relative Error of the Three Algorithms for Inferring TM With 3 Time Periods



(a) Inference for Inter TM

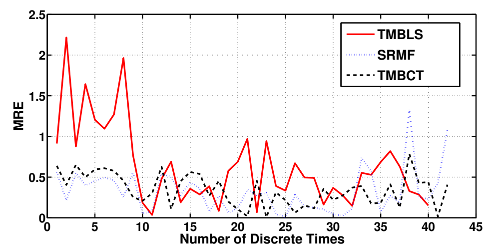


(b) Inference for Intra TM

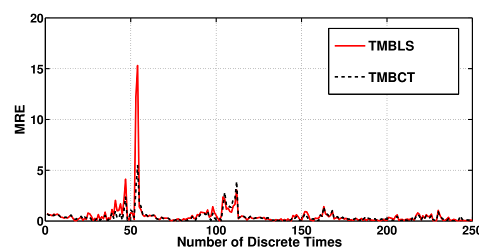
Fig. 5. The Relative Error of the Three Algorithms for Inferring TM With 25 Time Periods

becomes even larger. That is to say, the performance of SRMF degrades as TM elements expands, and it can not handle the large quantity of missing data well. TMBLS performs even better when TM gets larger, for the reason that it adaptively updates the old TM elements when new observation arrives, and it keeps refining the flow states and their uncertainty in order to make a reasonable tradeoff between the estimations and the observations. Thus as the time goes on, it could capture the spacial and temporal structure of traffic flows more accurately.

2) The Mean Relative Error: Fig. 6 compares the MRE of the three algorithms. From the two figures we can see that although TMBLS algorithm can accurately estimate most of the TM elements in some cases, its average accuracy fails to lead the two algorithms. That is to say a small part of TMBLS's results have a relative large deviation. However, the MRE of TMBLS is going to converge as the number of time slices increases. TMBCT has the best performance on MRE, as it always recomputes the hypothesis flow volumes using the new observations and the unbiased coarse-grained traffic information. Therefore, the results with large deviation could be adjusted timely when the new observations arrive. In Fig. 6(a), SRMF performs better at the beginning of the the time period. However, its MRE tends to be larger as the time passes. For the reason that in data center network, the larger TM with more time slices may not as sparse as the corresponding smaller one. Therefore, the deviation between the sparsest TM and the ground truth TM becomes larger when time goes on. Moreover, the computing time of SRMF is also unacceptable in large TM inference, that is why we did not present the curve of SRMF in the long term period in Fig. 6(b).



(a) Short Term for the Three Algorithms



(b) Long Term for the Two Algorithms

Fig. 6. The Mean Relative Error for the Three (Two) Algorithms Over Different Terms of Time Periods

3) Computing Time: Table I lists the computing times of the three algorithms. We can see from Table I, TMBCT finishes the inference much faster than the other two algorithms, and its computing time seems not be affected much by the TM scale. SRMF runs relatively faster when the scale of TM is small. However, its computing time increases sharply when TM becomes larger. When the number of time periods goes up to 100, SRMF cannot finish the inference within a day. Although TMBLS runs relatively slower at the beginning, its computing time grows linearly with the TM scale, which is one of the most valuable advantages of TMBLS. That is to say, TMBLS

TABLE I
THE COMPUTING TIME OF THE THREE ALGORITHMS (SECONDS)

Number of time slices	inter TM					intra TM				
	5	10	24	42	100	5	10	24	42	100
TMBCT	0.01	0.01	0.03	0.05	0.18	0.10	0.34	0.40	0.56	0.81
TMBLS	12.59	22.88	50.81	89.46	200.90	5.58	11.85	26.73	53.46	102.54
SRMF	45.79	176.84	1528.71	6341.20	-	2.02	26.45	147.06	470.12	-

can be applied to on-line TM inference problem, where it incorporates the new SNMP information and adjusts the direct precursor at each round. We could estimate from Table I that each round of updating (adjusting the last column and adding a new column) needs about 2 seconds and 1.3 seconds for inter TM inference and intra TMs inference, respectively. Therefore, although TMBLS requires 200.90 seconds to infer the inter TM over 100 time periods, actually it only takes 2 seconds to update the 99 time-period TM. On the contrary, SRMF algorithm can only be applied to the off-line inference problem as it needs to start over on each discrete time period.

It is notable that TMBLS and SRMF compute much faster in intra TM inference problem than in the inter counterpart, for the number of possible routes between all cluster pairs is much larger than their intra routes. Therefore, their speeds are correlative with the TM scale. However TMBCT performs exactly the opposite way, due to its speed only correlates with the total number of TM elements. And the total elements in intra TM over all clusters are much more than the elements in the inter TM. This also implies that decomposing the inference problem into several smaller one greatly helps the structure based methods to handle the problem.

VIII. CONCLUSION

In this paper we enabled the tomography based methods to handle the TM inference problem for data center networks from easy-collected SNMP data. We argued that the prevailing data center network topologies can be decomposed into several clusters. Hence the complexity of inference problem can be dramatically reduced. We also stated a lemma to demonstrate the coarse-grained traffic characteristic of each cluster can be calculated unbiasedly using the SNMP data. Two efficient algorithms were proposed to infer the structured and unstructured TMs, respectively. Through comparing our two algorithms with a recent representative TM inference method, the experimental results showed that our two algorithms outperform the former one in both accuracy and efficiency. Moreover, our methods can make the online inference through updating the TM elements on the last time period within a few seconds, while the former method requires hours to start over a new inference.

ACKNOWLEDGMENT

This work is supported in part by AcRF Tier 1 Grant RG 32/09.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, *A Scalable, Commodity Data Center Network Architecture* SIGCOMM, pages 63C74, 2008.

[2] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, *VL2: a scalable and flexible data center network* SIGCOMM, 2009.

[3] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, *BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers* SIGCOMM 2009 Conference on Data Communication, Barcelona, Spain, August 17 - 21 2009.

[4] M. Al-Fares, S. Radhakrishnan, B. Raghavan, W. College, N. Huang, and A. Vahdat, *Hedera: Dynamic flow scheduling for data center networks* NSDI 2010, San Jose, CA, USA, April 2010.

[5] P. Gill, N. Jain, N. Nagappan, *Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications* SIGCOMM11, August 15-19, 2011, Toronto, Ontario, Canada.

[6] J. W. Jiang, T. Lan, S. Ha, M. Chen, M. Chiang, *Joint VM Placement and Routing for Data Center Traffic Engineering* INFOCOM12, 25-30 March 2012, Princeton, NJ, USA.

[7] T. Benson, A. An, A. Akella, M. Zhang, *Understanding Data Center Traffic Characteristics* WREN09, August 21, 2009, Barcelona, Spain.

[8] T. Benson, A. Akella, A. Akella, D. A. Maltz, *Network Traffic Characteristics of Data Centers in the Wild* IMC10, November 1C3, 2010, Melbourne, Australia.

[9] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, R. Chaiken, *The Nature of Datacenter Traffic: Measurements & Analysis* IMC09, November 4C6, 2009, Chicago, Illinois, USA.

[10] Cisco Data Center Infrastructure 2.5 Design Guide, http://www.cisco.com/application/pdf/en/us/guest/netsol/ns107/c649/ccmi/newlinegration/underline/space09186a008073377d.pdf.

[11] C. E. Leiserson,*Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing,* IEEE Trans. Computer, vol. 34, no. 10, pp. 892C901, Oct. 1985.

[12] Y. Zhang, M. Roughan, N. Duffield, A. Greenberg, *Fast Accurate Computation of LargeScale IP Traffic Matrices from Link Loads* SIGMETRICS03, June 10C14, 2003, San Diego, California, USA.

[13] M. Roughan, Y. Zhang, W. Willinger, L. Qiu, *Spatio-Temporal Compressive Sensing and Internet Traffic Matrices (Extended Version)* IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 20, NO. 3, JUNE 2012

[14] A. Soule1, A. Lakhina2, N. Taft3, K. Papagiannaki, *Traffic Matrices: Balancing Measurements, Inference and Modeling* SIGMETRICS05, June 6C10,2005, Banff,Alberta,Canada.

[15] J. Kowalski and B. Warfield, *Modeling traffic demand between nodes in a telecommunications network,* in ATNAC95, 1995.

[16] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, *Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning (extended abstract)* ACM SIGCOMM Internet Measurement Workshop, 2002.

[17] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, *Data center TCP (DCTCP)* SIGCOMM '10 Proceedings of the ACM SIGCOMM 2010 conference New York, NY, USA, 2010

[18] George F. Riley, Mostufa H. Ammar and Ellen W. Zegura *Efficient Routing Using NIx-Vectors* IEEE Workshop on High Performance Switching and Routing, pp: 25C27, 2001, Atlanta, GA, USA.

[19] Anders Gunnar, Mikael Johansson and Thomas Telkamp *Traffic Matrix Estimation on a Large IP Backbone—A Comparison on Real Data* IMC04 October 25C27, 2004, Taormina, Sicily, Italy.