Origin-Destination Flow Measurement in High-Speed Networks

Tao Li Shigang Chen Yan Qiao
Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL, USA

***Abstract*—An origin-destination (OD) flow between two routers is the set of packets that pass both routers in a network. Measuring the sizes of OD flows is important to many network management applications such as capacity planning, traffic engineering, anomaly detection, and network reliability analysis. Measurement efficiency and accuracy are two main technical challenges. In terms of efficiency, we want to minimize per-packet processing overhead to accommodate future routers that have extremely high packet rates. In terms of accuracy, we want to generate precise measurement results with small bias and standard deviation. To meet these challenges, we design a new measurement method that employs a compact data structure for packet information storage and uses a novel statistical inference approach for OD-flow size estimation. We perform simulations to demonstrate the effectiveness of our method.**

I. Introduction

Traffic measurement in high-speed networks provides critical information for network management, resource allocation, and traffic engineering [1], [2], [3]. This paper focuses on the problem of *origin-destination (OD) flow measurement*. Consider two routers r_1 and r_2. We define the set of packets that first pass r_1 and then pass r_2 or first pass r_2 and then pass r_1 as an *origin-destination (OD) flow* of the two routers. The cardinality of the packet set is called the *OD flow size*. Our goal is to design an efficient method to measure the number of packets that traverse between two routers during a measurement period. It generally consists of two phases: One for online packet information storage and the other for offline OD-flow size computation. In the first phase, routers record information about arrival packets. In the second phase, each router reports its stored information to a centralized server, which performs the measurement of each OD flow based on the information sent from the origin/destination router pair.

The OD flow measurement is an important function in many network management applications such as capacity planning, traffic engineering, anomaly detection, and network reliability analysis [4], [5], [6]. For example, Internet service providers may use the OD-flow information between points of interest as a reference to align traffic distribution within the network. They may also study the OD-flow traffic pattern and identify anomalies that deviate significantly from the normal pattern. In the event of a DDoS attack, a sudden surge of OD flows from some routers to a common destination may give clues about where attack traffic is generated.

Measurement efficiency and accuracy are two main technical challenges. In terms of efficiency, we want to minimize the per-packet processing overhead to accommodate future routers that forward packets at extremely high rates. Increasingly faster line speed beyond OC-768's 40Gbps to reach tera bits per second [7] makes DRAM unsuitable for online traffic measurement functions due to its low speed. Instead, fast but more expensive SRAM is preferred. On-die SRAM may be shared by a number of critical functions for routing, scheduling, security, and traffic measurement. These functions have to take turns to access the SRAM. Therefore, it is extremely important to reduce per-packet processing overhead for any traffic measurement function.

Accuracy is another important design goal. In high-speed networks, we have to deal with a very large volume of packets. Yet, the available memory for packet information storage is limited by the size of SRAM. Therefore, it is unrealistic to store all packet-level information in order to achieve 100% accuracy. To solve this problem, some past research [8], [9], [10] uses widely available data such as link load, network routing, and configuration data to indirectly measure the OD flows. Cao, Chen and Bu [11] propose a quasi-likelihood approach based on a continuous variant of the Flajolet-Martin sketches [12]. However, none of them is able to achieve both efficiency and accuracy at the same time.

To meet these challenges, we design a new OD flow measurement method, which uses a compact bitmap data structure for packet information storage. At the end of a measurement period, bitmaps from all routers are sent to a centralized server, which examines the bitmaps of each origin/destination router pair and uses a statistical inference approach to estimate the OD flow size. The proposed method has three properties. First, its processing overhead is small and constant, only one hash operation and one memory access per packet. Second, it is able to achieve excellent measurement results, which will be demonstrated by simulations. Finally, its data storage is very compact. The memory allocation is less than 1 bit for each packet on average.

II. Problem Statement and Performance Metrics

A. Problem Statement

Let S be a subset of routers of interest in a network. The problem is to measure traffic volume between any pair of routers in S. We model an origin-destination (OD) flow as the set of packets traverse between two routers (the undirectional case) or traverse from one router to the other (the directional case). Our goal is to measure the size of each OD flow in terms of number of packets.

978-1-4673-0775-8/12/$31.00 ©2012 IEEE

Consider the set of access routers on the perimeter of an ISP network. If each access router stores information about ingress packets (that enter the ISP network) and egress packets (that leave the ISP network) in separate data structures, we can figure out the size of an directional OD flow by comparing the information in the ingress data structure of the origin router and the information in the egress data structure of the destination router. On the other hand, if each access router stores information of all arrival packets in the same data structure, we can figure out the size of an undirectional OD flow by comparing the information in the data structures of both routers. The measurement method proposed in this paper can be applied to both cases even though our description uses the undirectional case for simplicity.

We consider two performance metrics, per-packet processing overhead and measurement accuracy, which are discussed below.

B. Per-packet Processing Overhead

The maximum packet throughput that an online measurement function can achieve is determined by the per-packet processing overhead of the function. In order to keep up with today's high-speed network, it is desirable to make the per-packet processing overhead as small as possible, especially when the SRAM and processing circuits are shared by other critical functions.

The per-packet processing overhead is mainly determined by the computational complexity and the number of memory accesses for each packet. When a router receives a packet, it needs to perform certain computation to determine the proper location for the information storage and at least one memory access for the storing operation. We will show that our OD flow measurement function is able to achieve extremely small per-packet processing overhead.

C. Measurement Accuracy

Let n_c be the OD flow size of an origin/destination router pair and $\hat{n}_c$ be the corresponding measurement result. The event for n_c to fall into the interval $[\hat{n}_c \cdot (1-\beta), \hat{n}_c \cdot (1+\beta)]$ with probability at least α specifies the measurement accuracy of our function, where α and β are pre-determined accuracy parameters, e.g., when $\alpha = 95\%$ and $\beta = 5\%$, it means the measurement result is constrained in the range $[0.95n_c, 1.05n_c]$ with probability 95%. A smaller value of β means better measurement results.

If the memory requirement and the processing speed for each packet are unlimited, we can achieve 100% measurement results. Otherwise, we have to compromise the measurement accuracy if the memory resource is not enough or the processing speed requirement is relatively stringent.

III. Origin-Destination Flow Measurement

We first describe a straightforward approach and discuss its limitations. We then motivate the bitmap idea that we use in this study. Finally we present our origin-destination flow measurement method (ODFM) in details.

A. A Straightforward Approach

A straightforward approach is for each router to store the information of all packets that pass it. In this way, when we want to measure the OD flow size of two routers, we only need to compare the two sets of packet information and count how many packets the two sets have in common, i.e., the cardinality of the intersection of the two sets. Clearly, storing information of all packets is unrealistic since the number of packets passing a router is huge in high-speed networks and it imposes an extremely large memory requirement on the router.

In order to reduce the memory requirement, we can store the signatures of packets instead. The signature of a packet is a hash value of the packet with a fixed length. When the length of the signature is long enough, e.g., 160 bits if using SHA-1 [13], the chance of two packets having the same signatures is negligibly small. Therefore, we can count the number of identical signatures that stored in the two routers to obtain the OD flow size. This enhancement can reduce the memory requirement to some extent. However, it is still not memory efficient. Suppose there are $1M$ packets that pass a router during a measurement period. When the length of the signature is 160 bits long, a router needs $20MB$ ($1M \times 160/8$) memory to store the information of all signatures, which is still too much in practise. Using smaller signatures cannot solve the problem, either. For example, if we reduce the signature length to just 16 bits, the memory requirement is still $2MB$, far higher than the goal of this paper, less than 1 bit per packet.

B. ODFM: Motivation and Overview

We design a bitmap based OD flow measurement method that is able to solve the problems of the above approach. Instead of storing the signatures of packets, each router maintains a bit array with a fixed length and initially all bits in the array are set to zero. When the router receives a packet, it pseudo-randomly maps the packet to one bit of the array by a hash operation and sets the bit to one. At the end of the measurement period, we measure the OD flow size of two routers by comparing their bit arrays. Since a packet always uses the same hash function to choose a bit in the arrays for all routers and the size of each bit array is fixed within a measurement period, it will map to the same location in the bit arrays of any routers it has passed. Therefore, if a packet enters router r_1 and exits from r_2 or the other way around, its corresponding bit in these two bit arrays must be both set to one. Based on this observation, we can take a bitwise AND operation of the two bit arrays and count the number of ones in the combined bit array to measure the OD flow.

Note that this approach may introduce the overestimation problem, which could lead to an inaccurate measurement result. Suppose two packets, called p_1 and p_2, map to the same location j by the hash function, while p_1 passes one router and p_2 passes the other. In this case, the jth bit of both bit arrays of the two routers will be set to one. When we compare the two bit arrays, we will falsely treat p_1 and p_2 as the same packet and overestimate the OD flow size. However, there is a nice property of this scheme: Because the bit for each packet is randomly picked in the bit array, the event for any two packets to choose the same bit in the array has an equal probability to happen. When the number of packets and the size of the bit array are

large enough, this event occurs in the bit array uniformly at random and the overestimation problem can be solved through statistical analysis. This property enables us to design a compact yet accurate measurement method. Moreover, in our scheme, a router only needs to perform one hash operation and one memory accesses per packet, which is very efficient and feasible for high-speed networks.

C. ODFM: Storing the Packet Information

ODFM consists of two components: one for storing the packet information into routers, the other for measuring the OD flow of any two routers. This subsection presents the first component and the second one will be described in the next subsection.

At the beginning of the measurement period, each router maintains a bit array B with a fixed length m. Initially each bit in B is set to zero. The ith bit in the array is denoted as $B[i]$. When a router receives a packet p, it pseudo-randomly picks one bit in B by performing a hash operation $H(p)$ and set the bit to one, where $H(..)$ is a hash function whose output range is $[0..m-1]$. More specifically, to store the packet p, ODFM performs the following assignment:

$$B[H(p)] := 1. \tag{1}$$

Actually a router does not have to perform the hash operation on all the content of a packet. In the network layer, a packet can be uniquely identified by its IP header, which stores the packet label information, i.e., source IP address and destination IP address and so on. For two packets that are fragments of some original, larger packet, although they share the same source/destination IP addresses and identification number, their fragmentation offset values are different. Therefore, a router only needs to perform the hash operation ($H(p)$) on the IP header of a packet, which can further reduce the hash computational complexity and improve the processing speed. This enhancement can be also applied to the straightforward approach in Section III-A.

It is worth noting that a router only needs to perform one hash operation and sets one bit in its bit array per packet, which is very simple, efficient, and can be easily implemented in high-speed routers.

D. ODFM: Measuring the Size of Each OD Flow

At the end of the measurement period, all routers will report its bit array to a centralized server, e.g., the network management center, which performs the offline measurement. ODFM employs the maximum likelihood estimation (MLE) [14] to measure the OD flow of any two routers based on their bit arrays. Let S_1 and S_2 be the set of packets that pass the two routers r_1 and r_2. Let n_1 and n_2 be the cardinalities of S_1 and S_2, respectively, i.e., $n_1 = |S_1|, n_2 = |S_2|$, n_c be the number of common packets that r_1 and r_2 share, i.e., the OD flow size of the two routers, which is the value that we want to measure in this study. Figure 1 illustrates the relationship of n_1, n_2, and n_c. Obviously, we have $n_c = |S_1 \cap S_2|$. Let B_1 and B_2 be the two bit arrays of r_1 and r_2, U_1 and U_2 be the number of '0's in B_1 and B_2, respectively, V_1 and V_2 be the percentage of bits in B_1 and B_2 whose values are zero. Clearly, $V_1 = \frac{U_1}{m}$ and $V_2 = \frac{U_2}{m}$.

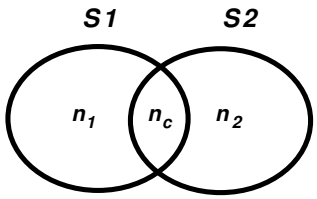

Fig. 1. The relation between two routers r_1 and r_2

The measurement consists of two steps. In the first step, we compute the cardinality of S_1 (i.e., n_1) and the cardinality of S_2 (i.e., n_2) based on B_1 and B_2, respectively. In the second step, we take a bitwise AND operation of B_1 and B_2 to generate a new bit array, denoted as B_c, to compute the OD flow size n_c. Let U_c be the number of '0's in B_c, V_c be the percentage of bits in B_c whose values are zero. Clearly, $V_c = \frac{U_c}{m}$. We compute n_c based on B_c and the results obtained in previous step, i.e., the values of n_1 and n_2.

1) Measure n_1 and n_2: The number of packets that a router receives during a measurement period can be easily obtained by adding a counter whose initial value is set to zero. When it comes a new packet, the router simply increases the counter by one. In this way, we can obtain the exact values of n_1 and n_2, which we will use to measure n_c in the following subsection.

2) Measure n_c: After n_1 and n_2 are obtained, we take a bitwise AND operation of B_1 and B_2, denoted as B_c, to measure n_c. More specifically, we have

$$B_c[i] = B_1[i] \ \& \ B_2[i], \quad \forall i \in [0..m-1]. \tag{2}$$

For an arbitrary bit b in B_c, it is '0' if and only if the following two conditions are both satisfied. First, it is not chosen by any packet in $S_1 \cap S_2$. If b is chosen by a packet $p \in S_1 \cap S_2$, we know the corresponding bits in both B_1 and B_2 will be set to '1'. Therefore, b will be '1'. Second, it is either not chosen by any packet in $S_1 - S_2$ or not chosen by any packet $S_2 - S_1$. If it is chosen by both a packet $p_1 \in S_1 - S_2$ and a packet $p_2 \in S_2 - S_1$, the corresponding bits in both B_1 and B_2 will be also set to '1'. As a result, b will be '1'. For the first condition, a packet in $S_1 \cap S_2$ has probability $\frac{1}{m}$ to set b to '1', which means the probability for b not to be set by this packet is $1 - \frac{1}{m}$. As Figure 1 shows, $n_c = |S_1 \cap S_2|$. Therefore, the probability for b not to be set to '1' by any packet in $S_1 \cap S_2$ is $(1 - \frac{1}{m})^{n_c}$. Similarly, the probability for it not to be chosen by any packet in $S_1 - S_2$ is $(1 - \frac{1}{m})^{n_1 - n_c}$ and the probability for it not to be chosen by any packet in $S_2 - S_1$ is $(1 - \frac{1}{m})^{n_2 - n_c}$. As a result, the probability $q(n_c)$ for b to remain '0' in B_c is

$$\begin{aligned} q(n_c) &= (1 - \frac{1}{m})^{n_c}\{1 - (1 - (1 - \frac{1}{m})^{n_1 - n_c}) \\ &\qquad \times (1 - (1 - \frac{1}{m})^{n_2 - n_c})\} \\ &= (1 - \frac{1}{m})^{n_1} + (1 - \frac{1}{m})^{n_2} - (1 - \frac{1}{m})^{n_1 + n_2 - n_c} \end{aligned} \tag{3}$$

Each bit in B_c has a probability $q(n_c)$ to be '0'. The observed number of '0' bits in B_c is U_c. Therefore, the likelihood function

for this observation to occur is given as follows:

$$L = q(n_c)^{U_c} \times (1 - q(n_c))^{m-U_c} \tag{4}$$

Following the standard process of maximum likelihood estimation, we find an optimal value of n_c that can maximize the above likelihood function. Namely, we want to find

$$\hat{n}_c = \arg\max_{n_c}\{L\} \tag{5}$$

To find $\hat{n}_c$, we take a logarithm operation to both sides of (4).

$$\ln L = U_c \times \ln q(n_c) + (m - U_c) \times \ln(1 - q(n_c)) \tag{6}$$

We then differentiate the above equation:

$$\begin{aligned}\frac{d\ln L}{dn_c} &= (\frac{U_c}{q(n_c)} - \frac{m - U_c}{1 - q(n_c)}) \times q'(n_c) \\ &= (\frac{U_c}{q(n_c)} - \frac{m - U_c}{1 - q(n_c)}) \times \ln(1 - \frac{1}{m}) \\ &\quad \times (1 - \frac{1}{m})^{n_1+n_2-n_c},\end{aligned} \tag{7}$$

since according to (3), we have

$$\begin{aligned}q'(n_c) &= \frac{dq(n_c)}{dn_c} \\ &= \ln(1 - \frac{1}{m}) \times (1 - \frac{1}{m})^{n_1+n_2-n_c}.\end{aligned} \tag{8}$$

In order to compute $\hat{n}_c$, we set the right side of (7) to zero, i.e.,

$$(\frac{U_c}{q(n_c)} - \frac{m - U_c}{1 - q(n_c)}) \times \ln(1 - \frac{1}{m}) \times (1 - \frac{1}{m})^{n_1+n_2-n_c} = 0 \tag{9}$$

Since neither of $\ln(1 - \frac{1}{m})$ and $(1 - \frac{1}{m})^{n_1+n_2-n_c}$ could be 0 when m is positive, we have

$$\frac{U_c}{q(n_c)} - \frac{m - U_c}{1 - q(n_c)} = 0. \tag{10}$$

Applying (3) to (10), we have

$$\begin{aligned}(1 - \frac{1}{m})^{n_1} + (1 - \frac{1}{m})^{n_2} - (1 - \frac{1}{m})^{n_1+n_2-n_c} &= \frac{U_c}{m} \\ &= V_c.\end{aligned} \tag{11}$$

In above equation, m, n_1, and n_2 are all known values, and V_c can also be computed when the packets information are recorded. As a result, we can measure n_c in the following formula:

$$n_c = n_1 + n_2 - \frac{\ln((1 - \frac{1}{m})^{n_1} + (1 - \frac{1}{m})^{n_2} - V_c)}{\ln(1 - \frac{1}{m})}. \tag{12}$$

TABLE I
NUMBER OF MEMORY ACCESSES AND NUMBER OF HASH OPERATIONS PER PACKET WITH $n_1 = 6,000,000$ AND $n_2 = 6,000,000$

	memory accesses	hash operations	constant?
ODFM	1	1	Yes
QMLE	1.50	2	No

TABLE II
NUMBER OF MEMORY ACCESSES AND NUMBER OF HASH OPERATIONS PER PACKET WITH $n_1 = 6,000,000$ AND $n_2 = 300,000$

	memory accesses	hash operations	constant?
ODFM	1	1	Yes
QMLE	1.56	2	No

TABLE III
NUMBER OF MEMORY ACCESSES AND NUMBER OF HASH OPERATIONS PER PACKET WITH THE VALUES OF n_1 AND n_2 ARE RANDOMLY ASSIGNED BETWEEN 100,000 AND 10,000,000

	memory accesses	hash operations	constant?
ODFM	1	1	Yes
QMLE	1.22	2	No

IV. SIMULATIONS

We evaluate the performance of our method ODFM by simulations in this section. We compare ODFM with the most related work, QMLE [11]. For fair comparison, we assign the same amount of memory to ODFM and QMLE. We compare them in terms of online processing overhead and measurement accuracy.

Simulations are performed under system parameters, n_1, n_2, and n_c. For an origin-destination router pair, n_1 is the number of packets that one router receives during the measurement period, and n_2 is the number of packets that the other router receives. Parameter n_c is the actual OD flow size. The amount of memory used is set to be 1 MB.

In the first set of simulations, we let $n_1 = 6,000,000$, $n_2 = 6,000,000$ or $300,000$. We vary n_c from 100 to 50,000. We use ODFM and QMLE to measure the flow size, and compare it with n_c to see how accurate the measurement is.

In the second set of simulations, we model a more realistic scenario, where n_1, n_2, and n_c are randomly chosen. The values of n_1 and n_2 are randomly selected from the range of $[100,000, 10,000,000]$, and the value of n_c is randomly selected from $[100, 50,000]$ in each simulation run.

A. Processing Overhead

Per-packet processing overhead of a measurement method is mainly determined by the number of memory accesses and the number of hash operations for each packet. Table I shows the averaged results when $n_1 = 6,000,000$, $n_2 = 6,000,000$, and n_c varies from 100 to $50,000$. ODFM requires only 1 hash operation and 1 memory access (memory write) for each packet, which is the optimal. QMLE requires more per-packet processing overhead. It incurs 1.50 memory accesses and 2 hash operations on average. Furthermore, per-packet processing overhead of ODFM is constant, while QMLE requires variable per-packet processing overhead, which is undesirable in practice. Table II presents similar results with $n_2 = 300,000$. Table III

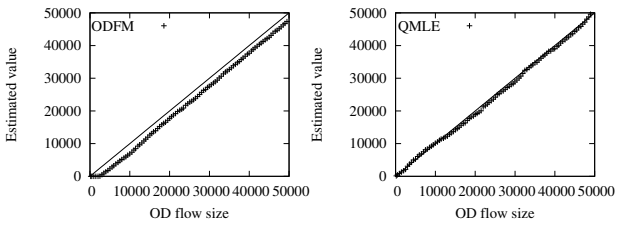

Fig. 2. • *Left Plot*: estimation results by ODFM when $n_1 = 6,000,000$ and $n_2 = 6,000,000$. • *Right Plot*: estimation results by QMLE when $n_1 = 6,000,000$ and $n_2 = 6,000,000$.

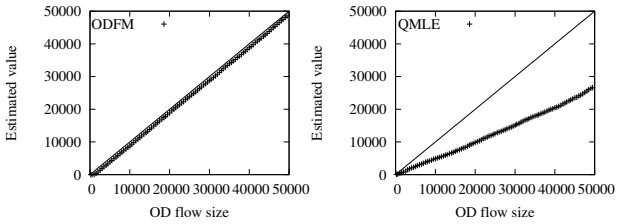

Fig. 3. • *Left Plot*: estimation results by ODFM when $n_1 = 6,000,000$ and $n_2 = 300,000$. • *Right Plot*: estimation results by QMLE when $n_1 = 6,000,000$ and $n_2 = 300,000$.

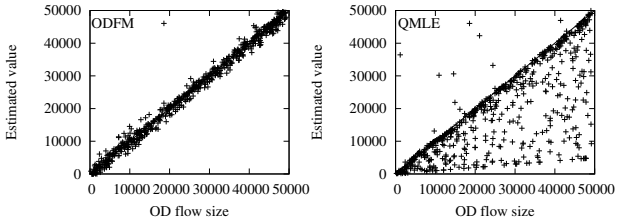

Fig. 4. • *Left Plot*: estimation results by ODFM when the values of n_1 and n_2 are randomly assigned between 100,000 and 10,000,000. • *Right Plot*: estimation results by QMLE when the values of n_1 and n_2 are randomly assigned between 100,000 and 10,000,000.

shows the results when the values of n_1 and n_2 are randomly chosen in the range $[100,000, 10,000,000]$ and the value of n_c is randomly chosen in the range of $[100, 50,000]$.

B. Measurement Accuracy

Figures 2-3 present the measurement results of ODFM and QMLE. Each figure consists of two plots. Each point in the left plot (ODFM) or the right plot (QMLE) represents an OD flow. The x-axis is the actual flow size n_c, and the y-axis is the estimated value $\hat{n}_c$. We also show the equality line, $y = x$, for reference. Clearly, the closer a point is to the equality line, the better the estimation result is. The two figures present the following results.

As shown in the left plot of Figure 2, when the values of n_1 and n_2 are the same, ODFM has a small bias in its measurement, which is understandable because it is well known that the maximum likelihood estimation may produce small bias under certain parameter settings. The right plot shows that QMLE performs better and produces almost perfect results. However, this is only part of the story. When the values of n_1 and n_2 are different, as shown in Figure 3 where $n_1 = 6,000,000$ and $n_2 = 300,000$, ODFM performs nearly perfectly, while QMLE produces large bias. As the difference between n_1 and n_2 widens, the bias of QMLE becomes larger, whereas the performance of ODFM is actually improved. Now the question is which case is closer to the reality, n_1 and n_2 having close values or diverse values? It is the latter, as we will show in the next section.

Figure 4 compares the performance of ODFM and QDFM when n_1 and n_2 are randomly picked in the range $[100,000, 10,000,000]$. Clearly, ODFM outperforms QMLE by a wide margin. The reason is that randomly-selected values of n_1 and n_2 tend to be very different than being close to each other.

V. Conclusions

This paper proposes a new method for OD flow measurement which employs the bitmap data structure for packet information storage and uses statistical inference approach to recover packet information. Our method not only requires smaller per-packet processing overhead but also achieves much more accurate results, when comparing with the existing approach. We implement simulations to demonstrate the superior performance of our method.

VI. Acknowledgements

This work was supported in part by the US National Science Foundation under grant CNS-1115548. We would also like to thank the anonymous reviewers for their constructive comments.

References

[1] J. Cao, Y. Jin, A. Chen, T. Bu, and Z. Zhang, "Identifying High Cardinality Internet Hosts," *Proc. of IEEE INFOCOM*, 2009.

[2] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani, "Counter Braids: A Novel Counter Architecture for Per-Flow Measurement," *Proc. of ACM SIGMETRICS*, 2008.

[3] X. Dimitropoulos, P. Hurley, and A. Kind, "Probabilistic Lossy Counting: An Efficient Algorithm for Finding Heavy Hitters," *ACM SIGCOMM Computer Communication Review*, 2008.

[4] M. Thorup M. Roughan and Y. Zhang, "Traffic engineering with estimated traffic matrices," *Proc. of Internet Measurement Conference (IMC)*, 2003.

[5] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of PCA for traffic anomaly detection," *Proc. of ACM SIGMETRICS*, 2007.

[6] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," *Proc. of ACM SIGCOMM*, 2002.

[7] W. David Gardner, "Researchers Transmit Optical Data At 16.4 Tbps," *InformationWeek*, February 2008.

[8] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An informationtheoretic approach to traffic matrix estimation," *Proc. of ACM SIGCOMM*, 2003.

[9] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "Estimating Point-to-Point and Point-to-Multipoint Traffic Matrices: An Information-Theoretic Approach," *IEEE/ACM Transactions on Networking*, vol. 10, no. 10, 2005.

[10] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale ip traffic matrices from link loads," *Proc. of ACM SIGMETRICS*, 2003.

[11] J. Cao, A. Chen, and T. Bu, "A Quasi-Likelihood Approach for Accurate Traffic Matrix Estimation in a High Speed Network," *Proc. of IEEE INFOCOM*, 2008.

[12] G. Flajolet, "Probabilistic counting," *Proc. of Symp. on Fundations of Computer Science (FOCS)*, 1983.

[13] National Institute of Standards and Technology, "FIPS 180-1: Secure Hash Standard," *http://csrc.nist.gov*, 1995.

[14] G. Casella and R. L. Berger, "Statistical Inference," *2nd edition, Duxbury Press*, 2002.