DISCO: Memory Efficient and Accurate Flow Statistics for Network Measurement

Chengchen Hu, Bin Liu, Hongbo Zhao
Computer Science and Technology Department
Tsinghua University
{huc,liub}@tsinghua.edu.cn; zhao-hb07@mails.tsinghua.edu.cn

Kai Chen, Yan Chen
Electrical Engineering and Computer Science Department
Northwestern University
{kchen,ychen}@northwestern.edu

Chunming Wu
Computer Science and Technology College
Zhejiang University
wuchunming@zju.edu.cn

Yu Cheng
Electrical and Computer Engineering Department
Illinois Institute of Technology
cheng@iit.edu

Abstract—**A basic task in network passive measurement is collecting flow statistics information for network state characterization. With the continuous increase of Internet link speed and the number of flows, flow statistics has become a great challenge due to the demanding requirements on both memory size and memory bandwidth in monitoring device. In this paper, we propose a DIScount COunting (DISCO) method, which is designed for both flow size and flow volume counting. For each incoming packet of length l, DISCO increases the corresponding counter assigned to the flow with an increment that is less than l. With an elaborate design on the counter update rule and the inverse estimation, DISCO saves memory consumption while providing an accurate unbiased estimator. The method is evaluated thoroughly under theoretical analysis and simulations with synthetic and real traces. The results demonstrate that DISCO is more accurate than related work given the same counter sizes. DISCO is also implemented on network processor Intel IXP2850 for performance test. Using only one MicroEngine (ME) in IXP2850, the throughput can reach up to 11.1Gbps under a traditional traffic pattern, and it increases almost linearly with the number of MEs employed.**

I. Introduction

In general, network measurement approaches can be classified into passive measurement and active measurement [3, 21]. The former measures the traffic traversing the traffic monitors without the disruption of the normal traffic, while the latter actively injects probe packets to infer the network status (e.g., available bandwidth, packet loss ratio, delay) via the inspections on the probe traffic's output. In this paper, we focus on passive measurement, which has been widely used to characterize the status of the network, *e.g.*, traffic matrix, packet length distributions, user session durations, and *etc*.

One of the most important components in a passive measurement system/infrastructure is the *monitoring component*. It is tapped into a high-speed network link and maintains a large number of counters for recording flow length statistics information. A complete flow length statistics report includes both *flow size counting* (which counts the number of packets in a flow) and *flow volume counting* or *flow byte counting* (which counts the number of bytes in a flow). Please note that, flow size distribution [5, 12, 22] cannot indicate flow-specific properties, *e.g.*, accurate size estimation for a particular flow or a subpopulation, which can be addressed by flow size estimation.

With the continuous increase of Internet link speed and the number of flows, fast and large memory is required to store the monitoring results. For example, the processing time per packet in a 40-Gbps link is only 12.8 ns in the worst case (considering only 64-byte packets). This makes it necessary to employ SRAMs and infeasible to use DRAMs only. However, due to tremendous flow volume and potential millions of in-process flows, a low density SRAM is susceptible to overflow the counters for network applications with fine measurement granularity [14]. The crux that off-the-shelf memory is either low speed or low capacity has posed a great challenge to flow statistics collection.

Generally, there are two categories of solutions in the literature to solve the problem. The first one sets full-size counters in DRAMs, and its key problem is how to slow down the updates to the counters in order to match the I/O (Input/Output) speed of DRAMs. Hybrid SRAM&DRAM (SD) counter architectures fall into this category [18, 19, 23]. The idea is to store lower-order bits of each counter in SRAMs and all the counter bits in DRAMs. SD solutions propose a good architecture to set measurement counters, but it also has limitations on 1) read access speed, 2) significant communication traffic between SRAMs and 3) DRAMs across the system bus, and extra pin connections.

The second one is the "SRAM-only counter" solutions and the main challenge is reducing the required counter size while providing accurate flow statistics. Random sampling is a common approach to control the memory consumption of flow size statistics [2, 6, 7, 9]. However, simple extensions of sampling methods for flow byte counting will lead to awkward performance in accuracy or processing speed. There is a need of a new SRAM-based method to support flow byte counting as well as flow size counting. Small Active Counters (SAC) [20] can be utilized to count flow byte in SRAMs, but

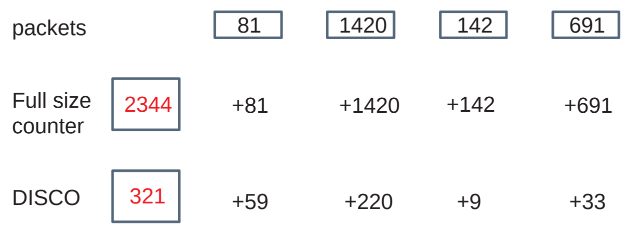

Fig. 1. An example of the counting process of DISCO. For the four packets of length 81, 1420, 142, 691, a full size counter is simply increased by the packet length; while DISCO increases with discounted values as 59, 220, 9, 33. The counter value is compressed 7 times (2334/321) in this case.

needs an extra storage overhead to keep parameters for each counter and extra processing overhead to frequently renormalize the counter values. Two recent proposals, BRICK [10] and CB [14], study the variable length counters to reduce the total memory requirements for measurement. BRICK/CB and the method proposed in this paper are complementary to each other and can work together to achieve further reduction on counter size.

To support both flow size and flow byte counting and provide both off-line and on-line access to measurement results, we propose a memory efficient and accurate flow statistics method named DIScount COunting (DISCO) which keeps the measurement results in SRAM only. The idea of DISCO is to regulate the counter value to be a real increasing concave function of the actual flow length (flow byte or flow size) n. Figure 1 illustrates how DISCO counter updates with a real trace segment input. For each incoming packet of l bytes, the counter is increased by a number Δ that is smaller than l. With the compact increase each time, the counter value, *i.e.*, the required counter size, is greatly compressed compared with a full size counter like SD solution. In this way, the technical challenge is how to determine Δ and its inverse estimation.

By successfully overcoming these challenges, we make the following contributions in this paper.

- We propose a flow statistics collection method for both flow size and flow byte counting with better accuracy than the related work under the same memory size. The memory consumption grows sub-linearly with the increase of the flow length, making the counters easily implementable in a SRAM for on-line access.

- We conduct theoretic analysis and extensive evaluations on real traces and synthetic data. The results validate the design of DISCO on the high accuracy and small memory consumption.

- We embed DISCO into Intel IXP2850 network processor for real implementation evaluation. The results indicate that only 96Kb on-chip memory is required for both flow size and flow volume counting. When using one MicroEngine(ME), the throughput can reach up to 11.1Gbps and the throughput keeps increasing if more MEs are utilized.

It should be noted that, DISCO goes a big step beyond Adaptive Non-Linear Sampling (ANLS) in our previous paper to support flow byte counting [9]. Although we leverage the same unbiased estimator for DISCO and ANLS for the sake of same memory compression ratio, the counter update algorithms are quite different. ANLS counter is always increased by one for the sampled packets; while DISCO updates the counter for every packet, and the counter increment depends on the packet length as well as the counter value being accumulated, instead of always one. As we will be discussed in Section II and Section V, simple extensions on ANLS do not work for flow volume counting. The basic idea of DISCO is presented at [8] and this paper describes the detailed design, analysis and experiments on DISCO.

The rest of the paper is organized as follows. Section II reviews the related work. Section III presents the counter update algorithm and the unbiased estimation of DISCO. Section IV analyzes the properties of DISCO theoretically. Section V evaluates the performance of DISCO under real and synthetic traces. In Section VI, an implementation of DISCO is described and tested. Finally in Section VII, we conclude the paper.

II. Related Work

A. DRAM-based full-size counters

A combined SRAM&DRAM (SD) counter architecture is first proposed in [19]. The increments are first made only to SRAM counters, and the values of each SRAM counter is then committed to the corresponding DRAM counters before being overflow. The key problem of this architecture is the design of a Counter Management Algorithm (CMA), which determines the order of the SRAM counters to be flushed to DRAM counters [18, 19, 23]. While the contributions of the SD solution is significant for many application scenarios, it has its limitations. First, the read operation of SD can only be done on DRAM side and thus it is quite slow. Second, SD also significantly increases the amount of traffic between SRAM and DRAM across the system bus, which may lead to a serious bottleneck in real system implementation [10]. Third, it is a trend to integrate measurement functions into routers; however, SD needs a dedicated SRAM and a dedicated DRAM, which will consume extra pins connections, as well as board areas.

B. Sampling based method

Sampling based method selects packets with a probability and each selected packets will trigger a update to the counter [2, 4]. With a sampling rate of p, if c packets have been sampled in a n-packet flow train, the unbiased estimation of the total packets is $\hat{n} = c/p$. There is a number of variation of sampling based methods [1, 6, 9, 13], however, they are designed for only flow size counting, and there could be two extensions of it to possibly support flow volume counting.

The first extension (E1) is to increase the counter by the size of the sampled packets instead of always one in the setting of flow size statistics. Using the example in Figure 1, if E1 samples the first and the third packet, the counter is 81+0+142+0=223. However, it may also only sample the first and the fourth packet which increase the counter by 772. The inverse estimations from these two samples are 446 and 1544, respectively. Such method will easily mislead the estimation of the total traffic unless the packet length variation of each flow is rare. However, it is not the case as the examination on real trace as Section V demonstrated.

The second way (E2) to extend sampling based method is to view a packet of l bytes as l independent packets, *i.e.*, to trigger the sampling l times/rounds for the packet. Obviously, the unbiased estimation, relative error and memory consumption of such an extension are the same as original sampling method; however, the per-packet processing complexity is as large as $O(\bar{l})$ on average and as $O(l_{max})$ in the worst case, where $\bar{l}$ and l_{max} are the average and largest packet length, respectively.

ANLS is also a sampling based method proposed in our previous work [9], which improves the measurement accuracy for small flows. We extend ANLS in these two ways to ANLS-I (like E1) and ANLS-II (like E2). Taking ANLS-I and ANLS-II as illustration, we will use experiments to demonstrate in Section V that the extensions of sampling based methods work awkward for flow volume counting.

C. Small active counters

The term "active counter" is introduced in [20], which allows estimation on per-packet basis without DRAM access. Small Active Counters (SAC) is proposed to reduce the SRAM space needed for the statistic counters [20]. For a q-bit counter, it is divided into two parts, an estimation part A and an exponent part $mode$. The estimator of SAC is $\hat{n} = A \cdot 2^{r \cdot mode}$, where r is a global parameter for all the counters. When a packet of size l comes, SAC updates the counter with $l/2^{r \cdot mode}$ on average. If A overflows, SAC increases $mode$ and renormalize the counter. If $mode$ overflows, r is incremented and all the counters are re-normalized. SAC compresses the counter size with small error, but it needs to be improved for two main problems. First, SAC divides a counter into two parts and the $mode$ part of the counter is an extra overhead. Second, when r increases, SAC needs to renormalize all the counters and this renormalization will suspend the update of the counter and may cause possible loss of necessary packet updates.

III. DISCO:DISCOUNT COUNTING

DISCO is a probabilistic counting algorithm for flow length statistics. The counting algorithm consists of the two parts: the counter update part and the inverse estimation part. The former one determines the increase of the counter for an incoming packet of length l, while the latter one estimates the actual flow length from the counter value with the counter update rule[1]. For convenience, the main notations utilized in this paper are first illustrated in Table I.

[1]l is set to be one for flow size counting, and is set to be the packet length for flow volume counting.

A. Counter update

As mentioned in Section I, the goal of DISCO is to compress the required counter bits so as to fit the counters in SRAM. Suppose c is the counter value and n is the flow length. We regulate the relationship between flow size and counter value as $n = f(c)$ or $c = f^{-1}(n) = g(n)$. Specifically, DISCO uses such a function $f(\cdot)$ to control the increments of the counter value,

$$f(c) = \frac{b^c - 1}{b - 1}, \tag{1}$$

where $b > 1$ is a pre-defined constant parameter. It is obvious that $f(\cdot)$ is an increasing convex function and its inverse function $f^{-1}(\cdot)$ is an increasing concave function[2]. It means that the "growing" of the counter value will be slower than the linear increasing. If the counters could record decimal fraction, the problem would be simple. The counter could be just increased by $\Delta(c, l)$ from its previous value c when a packet of l bytes comes, where $\Delta(c, l) = f^{-1}(l + f(c)) - c$. And the actual flow length can be calculated from the counter value c by $f(c)$ with no error. Since there is no enough memory size to maintain decimal counters in SRAM, we could only rely on the integer counters. The error will be accumulated if one simply rounds or truncates $\Delta(c, l)$. Instead, we give a probabilistic counter update algorithm as illustrated in Algorithm 1. When counter value is c and a packet of l bytes comes, DISCO increases the counter by $\delta(c, l) + 1$ with probability of $p_d(c, l)$, and increases the counter by $\delta(c, l)$ with probability $1 - p_d(c, l)$, where $\delta(c, l)$ and $p_d(c, l)$ are defined as

$$\delta(c, l) = \lceil f^{-1}(l + f(c)) - c \rceil - 1; \tag{2}$$

$$p_d(c, l) = \frac{l + f(c) - f(c + \delta(c, l))}{f(c + \delta(c, l) + 1) - f(c + \delta(c, l))}. \tag{3}$$

Algorithm 1 Counter update algorithm

```
/* A packet of $l$ bytes comes*/
$v = rand(0, 1)$;
/* rand() generate a random variable between 0 and 1 */
calculate $\delta(c, l)$ as formulated in (2);
calculate $p_d(c, l)$ as formulated in (3);
if $v \le p_d(c, l)$ then
    $c = c + \delta(c, l) + 1$
else
    $c = c + \delta(c, l)$;
end if
```

Please note that, the larger the counter value and/or packet length is, the smaller the increase of a counter is. And it can be guaranteed that $0 \le p_d \le 1$[3].

[2]A real-valued function f defined on an interval is called convex, if for any two points x and y in its domain and any λ in [0,1], we have $f(\lambda x + (1 - \lambda)y) \le \lambda f(x) + (1 - \lambda)f(y)$. A real-valued function f defined on an interval is called concave, if for any two points x and y in its domain and any t in λ in [0,1], we have$f(\lambda x + (1 - \lambda)y) \ge \lambda f(x) + (1 - \lambda)f(y)$.

[3]we use p_d and $p_d(c, l)$, δ and $\delta(c, l)$ interchangeably in the rest of the paper.

TABLE I
TABLE OF NOTATIONS

Notations	Descriptions
m	total number of packets
n	the actual flow length
b	a predefined parameter, $b > 1$
c	counter value
$c_i, 0 < i \le m$	counter value after the arrival of the ith packet
l	the bytes of an incoming packet
$l_i, 0 < i \le m$	length of the ith packet
$\delta(c, l)$	counter increment with c and l
$p_d(c, l)$	probability for counter update according to c and l
$f(c)$	unbiased estimation
$\hat{n}$	the estimated flow length
t	the number of possibilities of the counter values after $m-1$ packets
s	the number of possibilities of the counter values after m packets
$u_j, 1 \le j \le t$	possible counter value after $m-1$ packets
$v_i, 1 \le i \le s$	possible counter value after m packets
P_j	Probability that $c_{m-1} = u_j$
Q_i	Probability that $c_m = v_i$
θ	uniform integer increment size
$T(S)$	traffic amount that lets the counter value to be S
e	coefficient of variation of $T(S)$

B. Estimation from counter value

With the counter update rule described above, we can estimate the actual flow length with an unbiased estimator $f(c)$, where c is the counter value. Prior to the proof on the unbiased estimation, we first describe a general scenario of counting process. Without loss of generality, we concentrate on a single counter and suppose that, during a measurement interval, there are m packets whose packet lengths are $l_1, l_2, \cdots, l_m$ ($l_i, i = 1, 2, \cdots, m$, can be positive integer), respectively. The counter value is updated to c_i after the arrival of ith packet. Learned from Algorithm 1, there are two possible choices for the probabilistic update of the counter when a packet comes. Therefore, after the arrival of the $(m-1)$th packet, the counter value can be one of the $t = 2^{m-1}$ values. Denote these possible counter values as $u_1, u_2, \cdots, u_t$. For $\forall j, 1 \le j \le t$, the probability $p(c_{m-1} = u_j)$ is denoted as P_j. Similarly, after the arrival of the mth packet, the counter value will have $s = 2^m$ possibilities, denoting as $v_1, v_2, \cdots, v_s$. And for $\forall i, 1 \le i \le s$, the probability $p(c_m = v_i)$ is denoted as Q_i. The following equations holds:

$$v_{2j-1} = u_j + \delta(u_j, l_m); \tag{4}$$

$$v_{2j} = u_j + \delta(u_j, l_m) + 1; \tag{5}$$

$$Q_{2j-1} = P_j[1 - p_d(u_j, l_m)]; \tag{6}$$

$$Q_{2j} = P_j p_d(u_j, l_m). \tag{7}$$

Theorem 1: If c is the counter value, $f(c)$ is an unbiased estimation for DISCO.

Proof:

From the general counting scenario described above, if $E[f(c_m)] = \sum_{i=1}^{m} l_i$, then $f(c)$ is an unbiased estimation for DISCO.

Denote $F_i = E[f(c_i)], \forall i = 1, 2, \cdots, m$, then we have,

$$\begin{aligned}
F_m &= \sum_{j=1}^{s} (f(v_j)Q_j) \\
&= \sum_{j=1}^{t} (f(v_{2j-1})Q_{2j-1} + f(v_{2j})Q_{2j}) \\
&= \sum_{j=1}^{t} [f((u_j) + \delta(u_j, l_m) + 1)(P_j p_d(u_j, l_m)) \\
&\quad + f(u_j + \delta(u_j, l_m))P_j(1 - p_d(u_j, l_m))] \\
&= \sum_{j=1}^{t} P_j\{p_d(u_j, l_m)[-f(u_j + \delta(u_j, l_m)) \\
&\quad + f(u_j + \delta(u_j, l_m) + 1)] + f(u_j + \delta(u_j, l_m))\} \\
&= \sum_{j=1}^{t} P_j[l_m + f(u_j)](Substitute(3)) \\
&= l_m + F_{m-1}.
\end{aligned} \tag{8}$$

The counter value is zero when the first packet of size l_1 comes, therefore,

$$p_d(0, l_1) = \frac{l - f(\delta)}{f(\delta+1) - f(\delta)}; \tag{9}$$

$$\delta(0, l_1) = \lceil f^{-1}(l_1) \rceil - 1; \tag{10}$$

$$F_1 = p_d(0, l_1)f(\delta + 1) + (1 - p_d(0, l_1))f(\delta) = l_1. \tag{11}$$

Combine (8) and (11), the following equation holds by a mathematical induction argument.

$$F_m = E[f(c_m)] = \sum_{i=1}^{m} l_i. \tag{12}$$

The assertion of the theorem follows. ■

IV. Theoretical Analysis

A. Analysis on variation and error

Denote $T(S)$ as the random variable that represents the total traffic amount needed to let the counter value be S. We analyze the coefficient of variation of $T(S)$ which reflects the relative error of the estimation with the assumption of uniform integer increment size $\theta > 0$. The coefficient of variation is defined as

$$e[T(S)] = \sqrt{\frac{Var[T(S)]}{E^2[T(S)]}}. \tag{13}$$

Theorem 2: With the uniform integer traffic increments $\theta > 0$, the coefficient of variation of $T(S)$ is,

$$e = \begin{cases} \sqrt{\frac{(b-1)(b^S-b)}{(b+1)(b^S-1)}}, \theta = 1; \\ \sqrt{\frac{(b-1)[b^{2x}(b^{2S-2x}-1)-\theta b^x(b^{S-x}-1)(b+1)]}{(b+1)[b^x(b^{S-x}-1)+(b-1)\theta]^2}}, \theta > 1. \end{cases} \tag{14}$$

Proof:

We define every incoming of θ traffic as a trial, and $G(p_c)$ is a variable that describes the number of trials needed to make one increment of the counter when the current counter value is c. Since the traffic volume is θ in each trial, the traffic required to let the counter size increase from c to $c+1$ is $\theta G(p_c)$. Obviously, $G(p_c)$ is a geometric random variable, *i.e.*, $E[G(p_c)] = 1/p_c$ and $Var[G(p_c)] = (1-p_c)/p_c^2$, where the probability $p_c = \frac{\theta}{f(c+1)-f(c)} = \frac{\theta}{b^c}$.

Case 1)

If $\theta = 1$, $G(p_c)$ is a geometric random variable, *i.e.*, $E[G(p_c)] = 1/p_c$ and $Var[G(p_c)] = (1-p_c)/p_c^2$, where the probability $p_c = \frac{\theta}{f(c+1)-f(c)} = \frac{\theta}{b^c}$. We have:

$$E[T(S)] = E[\sum_{c=0}^{S-1} \theta G(p_c)] = \sum_{c=0}^{S-1} b^c = \frac{b^S-1}{b-1} = f(S); \tag{15}$$

$$\begin{aligned} Var[T(S)] &= Var[\sum_{c=0}^{S-1} \theta^2 G(p_c)] = \sum_{c=0}^{S-1} \frac{1-p_c}{p_c^2} \\ &= \sum_{c=0}^{S-1} b^{2c}(1-1/b^c) = \sum_{c=0}^{S-1} b^{2c} - \sum_{c=0}^{S-1} b^c \\ &= \frac{b^{2S}-1}{b^2-1} - \frac{b^S-1}{b-1}. \end{aligned} \tag{16}$$

Substitute (15)(16) into (13),

$$e = \sqrt{\frac{(b-1)(b^S-b)}{(b+1)(b^S-1)}}. \tag{17}$$

Case 2)

If $\theta > 1$, the counter value increases to x after the first trial, where $f(x) \le \theta \le f(x+1)$. From the second trial, $G(p_c)$ is also a geometric random variable, *i.e.*, $E[G(p_c)] = 1/p_c$ and $Var[G(p_c)] = (1-p_c)/p_c^2$, where the probability $p_c = \frac{\theta}{f(c+1)-f(c)} = \frac{\theta}{b^c}$. Therefore,

$$E[T(S)] = \theta + E[\sum_{c=x}^{S-1} \theta G(p_c)] = \theta + \frac{b^x(b^{S-x}-1)}{b-1}; \tag{18}$$

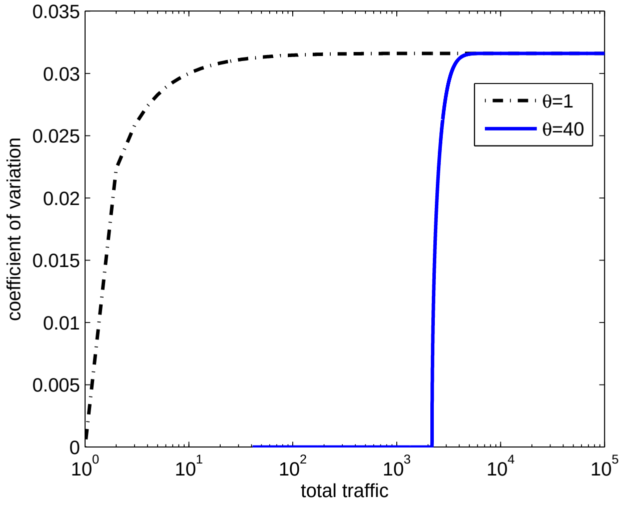

Fig. 2. Coefficient of variation vs. flow length when $b = 1.002$, with different increments.

$$\begin{aligned} Var[T(S)] &= Var[\sum_{c=x}^{S-1} \theta^2 G(p_c)] = \sum_{c=x}^{S-1} \theta^2 \frac{1-p_c}{p_c^2} \\ &= \sum_{c=x}^{S-1} b^{2c}(1-\theta/b^c) = \sum_{c=x}^{S-1} b^{2c} - \sum_{c=x}^{S-1} \theta b^c \\ &= b^{2x}\frac{b^{2S-2x}-1}{b^2-1} - \theta b^x \frac{b^{S-x}-1}{b-1}. \end{aligned} \tag{19}$$

Substitute (18)(19) into (13),

$$e = \sqrt{\frac{(b-1)[b^{2x}(b^{2S-2x}-1)-\theta b^x(b^{S-x}-1)(b+1)]}{(b+1)[b^x(b^{S-x}-1)+(b-1)\theta]^2}}. \tag{20}$$

Combine case 1) and case 2), (14) follows. ■

From Theorem 2, we can derive the following corollary.

Corollary 1: $\forall \theta > 0$,e is bounded by $\sqrt{\frac{b-1}{b+1}}$.

Proof:

Obviously, e monotonously increases when S increases. If $\theta = 1$, divide both numerator and denominator of (17) by b^S and let $S \to \infty$, we get $e \to \sqrt{\frac{b-1}{b+1}}$. If $\theta > 1$, divide both numerator and denominator of (20) by b^{2S} and let $S \to \infty$, we get $e \to \sqrt{\frac{b-1}{b+1}}$. ■

Figure 2 depicts the relationship between coefficient of variation and total traffic according to Theorem 2. No matter $\theta = 1$ or $\theta > 1$, coefficient of variation increases to a same bounded value as indicated in Corollary 1. In the figure,$b = 1.002$, so the bound is 0.0316. This bound is increased with the increment of b as demonstrated in Figure 3.

B. Analysis on memory cost

When the actual flow length is n, the expected counter value is not equal to $f^{-1}(n)$. In fact, it is bounded by $f^{-1}(n)$.

Theorem 3: An upper bound of expected counter value $E[c(n)]$ is $f^{-1}(n)$, where $f^{-1}(n)$ is the inverse function of $f(c)$.

Proof:

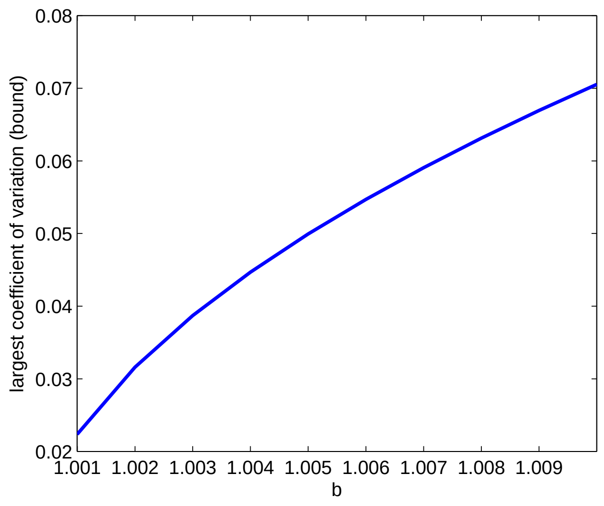

Fig. 3. Coefficient of variation vs. the parameter b. It is demonstrated that smaller b leads to smaller Coefficient of variation, *i.e.*, the relative error.

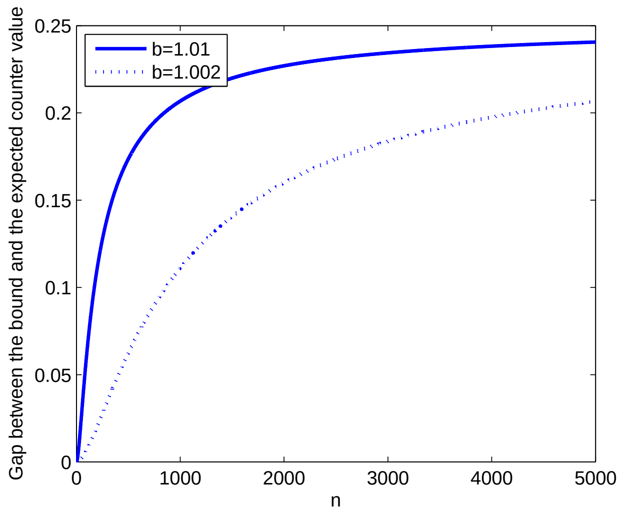

Fig. 4. Gap between the bound and the expected counter value.

As indicated in (1), $f(c)$ is a convex function, which satisfies

$$f(x) \geq f(y) + (x-y)f'_r(y), \forall x, y > 0 \tag{21}$$

where $f'_r(\cdot)$ is the derivative of $f(\cdot)$ on the right. Now, let $x = c$ and $y = E[c]$. We get,

$$f(c) \geq f(E[c]) + (c - E[c])f'_r(E[c]) \tag{22}$$

$$E[f(c)] \geq E[f(E[c]) + (c - E[c])f'_r(E[c])]. \tag{23}$$

From Theorem 1, $E(f(c)) = n$, then we obtain,

$$E[f(c)] = n \geq f(E[c]) \tag{24}$$

Since $f(c)$ is an increasing function, we can have

$$E[c(n)] \leq f^{-1}(n) \tag{25}$$

■

We run DISCO under different flow lengths for 50 times, and calculate the expected (average) counter value for each flow size. We compare these values with the bound indicated in *Theorem 3* and plot the gap between them in Fig. 4. The figure shows that the bound in *Theorem 3* is a tight one for the specific sampling function defined in (1): the absolute gap is quite small and the relative gap (absolute gap divided by n) is approximately on the order of 10^{-4} or even below.

C. Relationship with ANLS

The counting process of ANLS can be presented as $c \leftarrow c+1$ with probability $p(c)$, where c is the counter value. $p(c) = 1/[f(c+1) - f(c)]$, where $f(c)$ is any real increasing convex function satisfying $f(0) = 0, f(1) = 1$, $f(c) < f(c+1) \leq bf(c) + 1$ (b is a predefined parameter and $b > 1$). ANLS is designed only for packet number counting. The corresponding counter is increased by one when a packet is sampled. When DISCO is used to count packet number, *i.e.*, the length of every packet is viewed as one ($l = 1$). In this way, DISCO is equivalent to ANLS since the $f(c)$ defined in (1) satisfies the ANLS conditions described in [9].

V. Simulated Evaluation

In this section, we present the experiment configurations and results when DISCO is adopted to count flow volume and flow size.

A. Simulation settings

As mentioned in Section I, SAC is the only method in literature that can be implemented on SRAM for both flow volume and flow size counting, so numerical comparisons on estimation accuracy and memory consumptions between SAC and DISCO are investigated.

For each counter, SAC needs s bits to record the exponent part of the estimator (named as *mode* in [20]) and k bits to keep the estimation part (named as *A* in [20]). Therefore, the counter size of SAC is $S_{sac} = s + k$ and in all our experiments k is set to be 3.

We study how the accuracy changes with the increment of counter size based on the real trace input. Relative error R is defined as the absolute value of the distance between the real flow length and the estimated flow length, *i.e.*, $R = \frac{|\hat{n}-n|}{n}$. We introduce average relative error, maximum relative error and optimistic relative error for accuracy evaluation.

- *Average relative error* $\bar{R}$ is the mean value of R over all the counters.
- *Maximum relative error* R_{max} is the largest R over all the counters, which is a descriptor of the worst case.
- *α-Optimistic relative error* $R_o(\alpha)$ indicates the probability guarantees of the relative error, which can be formulated as

$$R_o(\alpha) = \sup\{r | Pr\{R \leq r\} \geq \alpha\} \tag{26}$$

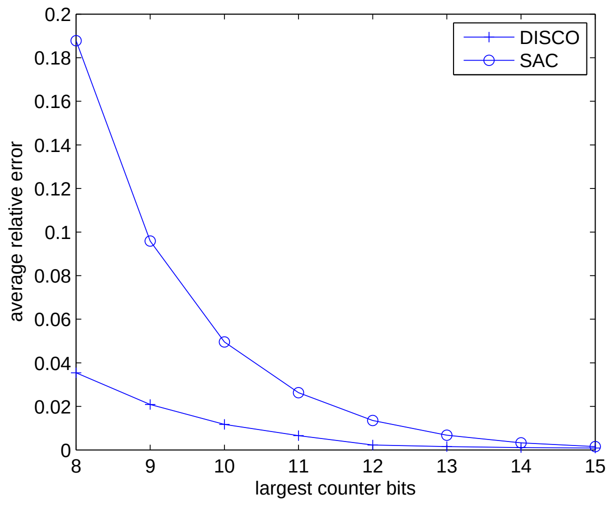

Fig. 5. Average relative error for flow volume counting.

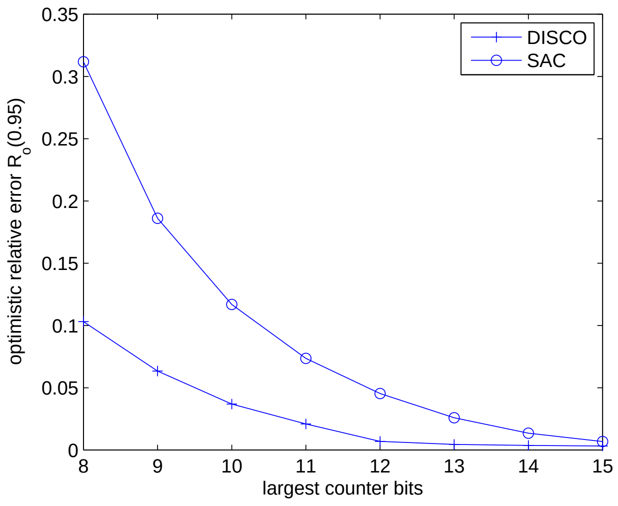

Fig. 7. Optimistic relative error ($\alpha(95)$) for flow volume counting.

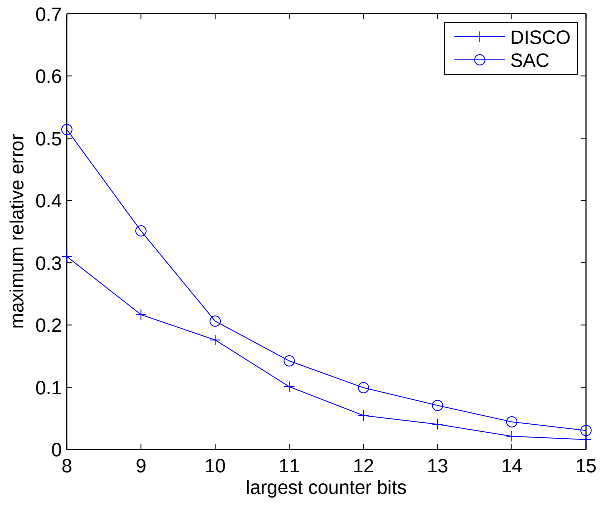

Fig. 6. Maximum relative error for flow volume counting.

B. Simulation results

The performance behavior of DISCO and SAC is first investigated under a real trace for flow volume counting. The real trace on OC-192 link is obtained from NLANR [16] which represents totally 40G bytes traffic volume. In this real trace, the number of flows is 100,728 and the average flow size is 409.5K bytes.

Figure 5 depicts the relationship between average relative error and counter size when SAC and DISCO are used to count flow volume. It is as expected that the average relative error $\bar{R}$ decreases with the increase of counter size for both two methods. We observe from the figure that, the average relative error of DISCO is smaller than SAC with the same counter size. The margin between the two error curves becomes smaller when the counter size increases. The reason is that the relative error for both SAC and DISCO should converge to zero when the counter size is set to be large enough as a full-size counter (like SD). Figure 6 shows the maximum relative error and indicates the similar trends as Figure 5. It is demonstrated that DISCO is more accurate than SAC even in the worst case. Figure 7 depicts the 0.95-optimistic relative error curves for the two methods. The relative error of 95% of the counters should be under the 0.95-optimistic error curve for each counting method. Obviously, DISCO provides better probabilistic guarantees of relative error than SAC.

The cumulative probability function of relative error using the real trace is investigated and the result is shown in Figure 8 with the snapshot of 10-bit counters. Under DISCO, for 90% of the flows, the flow volume estimation error is less than 0.04 and the estimation error of all the flows is less than 0.15. However, when employing SAC, these two numbers are increased to 0.22 and 0.4, respectively.

The compression ratio of the counter size is also studied. Although full-size SD counters do not have estimation errors, its counter value increases linearly with the increase of flow length (the slope is one). With a small estimation error, SAC or DISCO only consumes a smaller counter for the statistics of a large flow. Without renormalization, the counter value of SAC increases linearly with a slope that is less than one and the counter increment of DISCO is an increasing convex function of the flow size/bytes as shown in Figure 9. The larger the flow volume, the larger the memory efficient gain achieved by using DISCO. As indicated in (1), $f(0) = 0$ and $f(1) = 1$, the memory consumption of DISCO will not be larger than SD and SAC, even for the smallest flow. Figure 9 also demonstrates that DISCO is scalable for the potential dramatic increase of flow volume in the Internet.

TABLE II
EXPERIMENT RESULTS UNDER DIFFERENT TRAFFIC SCENARIOS. R IS THE RELATIVE ERROR AND S IS THE COUNTER SIZE IN BITS

Scenarios	Metric	SAC	DISCO	SAC	DISCO	SAC	DISCO
Scenario 1	Average relative error	0.089	0.052	0.045	0.031	0.025	0.016
	counter bits	8	8	9	9	10	10
Scenario 2	Average relative error	0.177	0.096	0.091	0.079	0.054	0.038
	counter bits	8	8	9	9	10	10
Scenario 3	Average relative error	0.143	0.097	0.094	0.063	0.061	0.041
	counter bits	8	8	9	9	10	10
Real trace Scenario	Average relative error	0.177	0.035	0.105	0.021	0.054	0.012
	counter bits	8	8	9	9	10	10

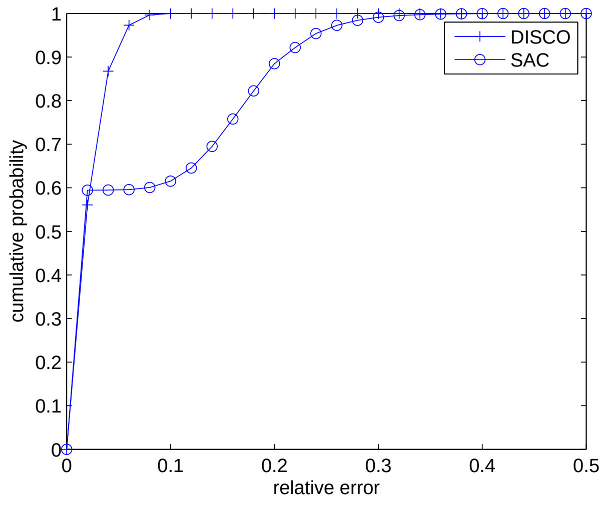

Fig. 8. Cumulative probability distribution of relative error.

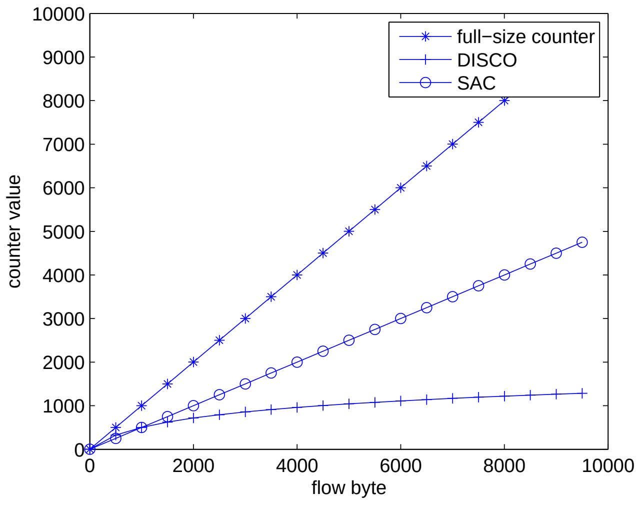

Fig. 9. Counter bits required under different flow volume.

Similar experiments are also conducted to study the performance of SAC and DISCO when they are used to count the flow size, *i.e.*, the number of packets in a flow. In this case, SAC is actually the same as Better NetFlow (BNF) [6] and as shown in Section IV-C, DISCO is equivalent to ANLS. Figure 10 plots the average relative error of estimated flow size for each flow under the same counter size, which indicates that DISCO is more accurate than SAC given the same memory resources.

Besides the experiments under the real trace, we employ other three synthetic traffic scenarios for evaluations. They are:

- Scenario 1. Each flow has x packets, where x is a random variable following Pareto distribution. The shape parameter is 1.053 and the scale parameter is 4. The packet length (bytes in a packet) follows truncate exponential distribution between 40 and 1500 with location parameter $\lambda = 100$. On average, a flow has 48.99 packets and 5.2K bytes traffic in this scenario.

- Scenario 2. Each flow has x packets, where x is a random variable following Exponential distribution with location parameter of 800. The packet length follows truncate exponential distribution between 40 and 1500 with location parameter $\lambda = 100$. On average, a flow has 778.30 packets and 82.7K bytes traffic in this scenario.

- Scenario 3. Each flow has x packets, where x is a random variable following Uniform distribution between 2 and 1600. The packet length follows truncate exponential distribution between 40 and 1500 with location parameter $\lambda = 100$. On average, a flow has 772.01 packets and 83.6K bytes traffic in this scenario.

Table II illustrates three snapshots when the counter sizes are set to be 8 bits, 9 bits and 10 bits, respectively, for both SAC and DISCO. Since the counter memory is determined by the largest counter value for the fixed-length counter system, in this paper, we use the *largest counter bits* for evaluation. From the experiments, we observe that 1) the accuracy can be improved with the increases of counter size, and 2) DISCO is also more accurate than SAC even if their counter sizes are configured to be the same. In other words, DISCO consumes less counter size with the same accuracy as SAC.

Although DISCO converges to ANLS when it is used to flow size counting, simple extensions of ANLS presented in

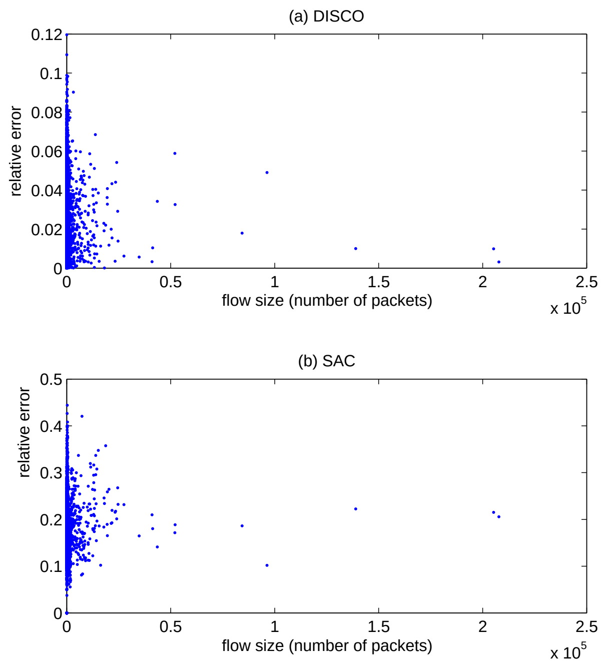

Fig. 10. The relative error of each flow for flow size counting. (a) is the results for DISCO and (b) is the results for SAC.

TABLE III
EXPERIMENTAL RESULTS FOR ANLS-I

	pkt. len. var.>10	average relative error
Scenario 1	100%	11.09
Scenario 2	100%	6.23
Scenario 3	100%	18.15
real trace	100%	6.26

Section II do not work well for flow volume counting. To be fair, we compare DISCO with ANLS-I and ANLS-II given the same memory size, *i.e.*, all use 10-bit counters for each flow. If ANLS-I is utilized, the relative errors are too large to be acceptable as indicated in Table III, compared with the results of DISCO shown in Table II. The large relative error of ANLS-I is caused by the large variations of the packet length. For example, the variation is larger than 10 for 62.78% of the flows in real trace and for 100% of other three synthetic traces. The mean variation over all the flows in each trace scenario is in the magnitude of 10^3-10^4. In addition DISCO is at least ten times faster than ANLS-II. The execution time ratio of DISCO over ANLS-II is illustrated in Table IV. It increases with the growth of the average flow length in different scenarios.

VI. IMPLEMENTATION AND PERFORMANCE TEST

In order to give a more comprehensive evaluation on DISCO, we have implemented DISCO on Intel network processor IXP2850 platform [11, 15]. IXA SDK 4.0 simulation environment is employed for performance validation.

TABLE IV
RATIO BETWEEN EXECUTION TIME OF ANLS-II AND DISCO

Scenario 1	Scenario 2	Scenario 3	real trace
15.03	28.34	31.53	189.88

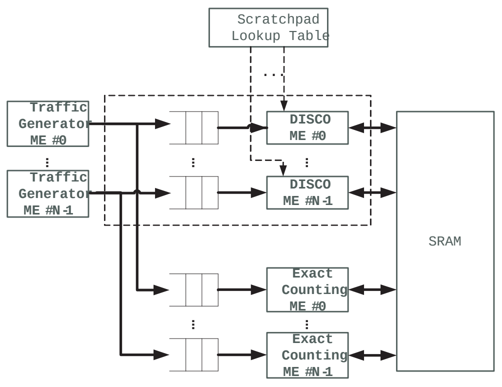

Fig. 11. Implementation of DISCO and the test-bench on IXP 2850.

The architecture of DISCO implementation and its test-bench, is depicted in Fig. 11. Four IXP2850 MicroEngines (ME) are utilized to function as traffic generators (TGEN). In order to mimic ultra high traffic input rate, TGEN only generates packet handlers instead of the whole packets. Each packet handler contains the flow ID and the packet length. The packet handlers are first forwarded to a specific "Scratchpad Ring", which is typically used as packet handler FIFO in IXP2850. Next to the packet handler FIFO, four MEs are equipped with DISCO logic (Algorithm 1) to update counters. In order to check the accuracy, an exact counting element is also designed and a copy of each synthetic packet handler is passed to it.

$\log_b(X)$ and b^X are required to obtain δ and p_d in (2) and (3). However, IXP2850 does not have instructions to deal with logarithm and power computation directly. We pre-compute $\log_b(X)$ and b^X, and then use a lookup table to get its value when a logarithm or an exponentiation operation occurs. The logarithm table and power table are combined into one "Log & Exp" table in our implementation. For each 32-bit entry of the table, the leftmost 20 bits are used for power computation and the rightmost 12 bits are employed to keep logarithm results. There is no need to keep too many table entries for very large X and we only store $3K$ entries for $\log_b(X)$ and b^X, $X \leq 3072$ and the memory of the pre-computation table is 96Kb with 3K entries. With simple shift and sum operation, we could calculate the values for $X > 3072$.

Prior to presenting the experimental results, we first describe the traffic pattern generated for performance tests. There are 2560 flows generated, where 20% of flows carries 80% of the traffic volume[4]. The packet length is uniformly distributed

[4]It is well known today that, Internet exhibits an "80-20" feature for its traffic [17], *i.e.*, 80% of Internet packets are generated by 20% of the flows.

TABLE V
THROUGHPUT ON IXP 2850 PLATFORM

Burst len.	Pkt Len.	# ME	error	Throughput
1	64-1kB	4	0.013	39.0Gbps
1	64-1kB	2	0.013	22.0Gbps
1	64-1kB	1	0.013	11.1Gbps
1-8	64-1kB	4	0.007	104.8Gbps
1-8	64-1kB	2	0.007	55.3Gbps
1-8	64-1kB	1	0.007	28.6Gbps

between 64B and 1KB. We first check the situation where burst length of any flow is only one, *i.e.*, any two packets from a same flow are intersected by packets of other flows. We enable 1, 2 and 4 MEs in this experiment and the results are shown in the first half of Table V. The throughput with only one ME reaches up to 11.1Gbps with a relative error of 0.013 and it is competent enough to serve for flow statistics on majority of the Internet backbone links. In addition, the throughput increases slightly smaller than the linear increase of the number of MEs.

Real traffic often shows burst of flows, *i.e.*, a number of back-to-back packets from a same flow comes continuously. In this case, the performance can be improved by delaying the update to SRAM counters. Instead of updating the counter for each incoming packet, counter is increased at the end of each burst period. A small naive on-chip counter is first used to fully record the flow length in a burst before its possible overflow. When a burst is over, the counter value is viewed as the bytes from a single packet and Algorithm 1 is used to update the counter. We check the performance improvement for this modification on processing. When the burst-length is a uniform random number between 1 and 8, the throughput is increased by about 2.5 times and the relative error is reduced to a half value. Considering the worst case where all the packets are 64B and arrive without burst, 8 MEs are needed to achieve 10Gbps throughput. Table lookup and counter update on SRAM are the main operations of DISCO. One write and a read operation on SRAM using IXP 2850 takes about 186 ns, and the time can be approximately reduced to 10-20 ns using FGPA/ASIC to implement operations on SRAM. Therefore, the performance of DISCO can be roughly improved ten times when porting the implementation to a FPGA/ASIC design.

VII. CONCLUSION

Acquiring both the flow size and the flow byte statistics in a same algorithm with improved accuracy and low memory occupation is always a target when implementing in real network equipments. In this paper we have proposed a DIScount COunting (DISCO) method to achieve this goal by an elaborate design of the counter update rule and the unbiased estimator. We theoretically model the DISCO algorithm and give a systemic analysis on its accuracy and counter/memory requirements. Extensive experimental evaluations with real traces and synthetical data validate the theoretical results. A real implementation is made on the Intel IXP2850 network processor with an inspiring outcome that only 96Kb memory is required and a throughput of 11.1 Gbps can be achieved by only using one MEs. The throughput increases almost linearly when multiple MEs are employed. This makes DISCO performance/cost effective for practical applications.

ACKNOWLEDGMENT

This work is supported by NSFC (60903182, 60873250, 60625201), 973 project (2007CB310702), Tsinghua University Initiative Scientific Research Program and open project of State Key Laboratory of Networking and Switching Technology (SKLNST-2008-1-05).

REFERENCES

[1] B.-Y. Choi, J. Park, and Z.-L. Zhang. Adaptive random sampling for load change detection. In *ACM SIGMETRICS 2002*, pages 272 – 273, 2002.
[2] Cisco. Sampled netflow data sheet. http://www.cisco.com.
[3] K. Claffy and S. McCreary. Internet measurement and data analysis: Passive and active measurement. http://www.caida.org.
[4] K. C. Claffy, G. C. Polyzos, and H.-W. Braun. Application of sampling methodologies to network traffic characterization. In *ACM SIGCOMM 1993*, pages 194–203, 1993.
[5] N. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. In *ACM SIGCOMM 2003*, pages 325–336, 2003.
[6] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a better netflow. In *ACM SIGCOMM 2004*, pages 245 – 256, 2004.
[7] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *ACM SIGCOMM 2002*, pages 323 – 336, 2002.
[8] C. Hu, B. Liu, and K. Chen. Poster: Compressing flow statistic counters. In *IEEE ICNP 2009 (poster)*, 2009.
[9] C. Hu, S. Wang, J. Tian, B. Liu, Y. Cheng, and Y. Chen. Accurate and efficient traffic monitoring using adaptive non-linear sampling method. In *INFOCOM 2008*, Phoenix, USA, 2008.
[10] N. HUA, B. Lin, J. J. Xu, and H. C. Zhao. Brick: A novel exact active statistics counter architecture. In *ANCS 2008*, 2008.
[11] E. J. Johnson and A. R. Kunze. *IXP2400/2800 Programming*. Intel Press, 2003.
[12] A. Kumar, M. S. amd J. J. Xu, and J. Wang. Data streaming algorithms for efficient and accurate estimation of flow size distribution. In *ACM SIGMETRICS 2004*, pages 177–188, 2004.
[13] A. Kumar and J. Xu. Sketch guided sampling – using on-line estimates of flow size for adaptive data collection. In *IEEE INFOCOM'06*, 2006.
[14] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani. Counter braids: A novel counter architecture for per-flow measurement. In *ACM SIGMETRICS*, 2008.
[15] U. R. Naik and P. R. Chandra. *Designing High-Performance Networking Applications*. Intel Press, 2004.
[16] NLANR. Passive measurement and analysis (pma). http://pma.nlanr.net.
[17] K. Psounis, A. Ghosh, B. Prabhakar, and G. Wang. SIFT: a simple algorithm for trucking elephant flows and taking advantage of power laws. In *the 43rd Allerton Conference on Communication, Control, and Computing*, 2005.
[18] S. Ramabhadran and G. Varghes. Efficient implementation of a statistics counter architecture. In *ACM SIGCOMM'03*, 2003.
[19] D. shah, S. Iyer, B. Prabhakar, and N. McKeown. Maintaining statistics counters in router line cards. *IEEE Micro*, 22(1):76–81, 2002.
[20] R. Stanojevic. Small active counters. In *IEEE INFOCOM'07*, 2007.
[21] G. Varghese and C. Estan. The measurement manifesto. *ACM Computer Communication Review*, 34:9–14, 2004.
[22] L. Yang and G. Michailidis. Sampled based estimation of network traffic flow characteristics. In *INFOCOM 2007*, 2007.
[23] Q. Zhao, J. J. Xu, and Z. Liu. Design of a novel statistics counter architecture with optimal space and time efficiency. In *ACM SIGMETRICS 2006*, 2006.