Discount Counting for Fast Flow Statistics on Flow Size and Flow Volume

Chengchen Hu, *Member, IEEE*, Bin Liu, *Senior Member, IEEE*, Hongbo Zhao, Kai Chen, Yan Chen, *Member, IEEE*, Yu Cheng, *Senior Member, IEEE*, and Hao Wu

***Abstract*—A complete flow statistics report should include both flow size (the number of packets in a flow) counting and flow volume (the number of bytes in a flow) counting. Although previous studies have contributed a lot to the flow size counting problem, it is still a great challenge to well support the flow volume statistics 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 bytes 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 the network processor Intel IXP2850 for a performance test. Using only one microengine (ME) in IXP2850, the throughput can reach up to 11.1 Gb/s under a traditional traffic pattern. The throughput increases to 39 Gb/s when employing four MEs.**

***Index Terms*—Counter, flow statistics, network measurement, unbiased estimation.**

Manuscript received March 25, 2012; revised March 18, 2013; accepted May 21, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor P. Crowley. Date of publication July 22, 2013; date of current version June 12, 2014. This paper was supported in part by the NSFC under Grants 61272459 and 61221063, the 863 Plan under Grant 2013AA013501, the National Science and Technology Major Project under Grant No. 2013ZX03002003–004, the NSF under NeTS Award 1219116, and the Fundamental Research Funds for Central Universities. This paper was presented in part at the IEEE International Conference on Distributed Computing Systems (ICDCS), Genova, Italy, June 21–25, 2010.

C. Hu is with the MOE KLINNS Laboratory, Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710000, China (e-mail: China.huc@ieee.org).

B. Liu and H. Wu are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: liub@tsinghua.edu.cn; wuhao.thu@gmail.com).

H. Zhao is with MeshSr Co., Ltd., Nanjing 211100, China (e-mail: if_blue@sina.com).

K. Chen is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: kaichen@cse.ust.hk).

Y. Chen is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA (e-mail: ychen@northwestern.edu).

Y. Cheng is with the Department of Electrical and Computer Engineering Technology, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: wuhao.thu@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2013.2270439

I. INTRODUCTION

INTERNET becomes a critical infrastructure component of our global information-based society, but as it grows more elaborate, network operators spend ever more time to monitor and manage Internet. Passive measurement provides the network operators an efficient tool for charging, engineering, managing, and securing the communication networks. Instead of actively injecting probes into the network like active measurement does, passive measurement monitors the traffic traversing the measurement beacons without the disruption of the normal traffic [3], [20]. A passive measurement system/infrastructure typically consists of four components. A *measurement beacon* tapped into the network link uses a data collection strategy to do flow statistics and then forwards the measured flow information to a *reporting component*. The reporting component aggregates the flow information into flow records and exports them to a remote *storage system* after a specific measurement interval. The data center is equipped with high-density data storage, which makes the measurement results available to the last component, i.e., the analysis system, which is responsible to generate analyses for different applications. In this paper, we study the flow statistics method in the measurement beacon, which generates the basic flow information for passive measurement, and many other measures like flow distribution can be calculated from the flow size estimation result.

In this paper, we study the fast flow statistics that support 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). The continuous increase of link speed and the number of flows leaves two choices to us: One is keeping the statistics results in DRAM and trying to match the updates frequency to the counters with the input/output (I/O) bandwidth of DRAM [17], [18], [21], [24], and the other is the SRAM-based solution, whose key is to reduce the required counter size. Previous works propose a Hybrid SRAM&DRAM (SD) counter architecture [17], [18], [24], which slows down the updates to the counters in order to match the I/O speed of DRAMs. However, this solution also has its limitations on read access speed, significant communication traffic between SRAMs, and the extra pin connections [8], [10]. Based on modern fast DRAM, [21] and [23] proposed a randomized DRAM architecture that can harness the performance of fast DRAM offerings by interleaving counter updates to multiple memory banks. However, without any compressing on the statistics, this method still faces the risk to overflow their counters when counting flow bytes or needs a quite

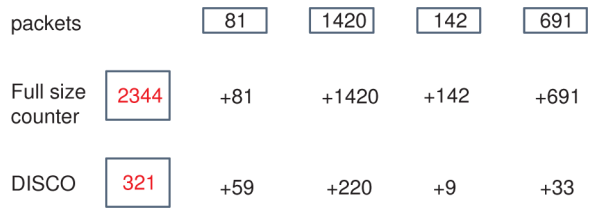


Fig. 1. Counting example of DISCO. For the four packets of length 81, 1420, 142, 691, a full-size counter is simply increased by the packet length, while DISCO increases with discounted values as 59, 220, 9, 33. The counter value is compressed seven times (2334/321) in this case.

large fast DRAM. Previous SRAM-based solution susually employed random sampling as a common approach to control the memory consumption of flow size statistics [2], [6], [7], [9]. However, simple extensions of sampling methods for flow volume counting will lead to awkward performance in accuracy or processing speed. Small Active Counters (SAC) [19] can be utilized to count flow byte in SRAMs, but needs an extra storage overhead to keep parameters for each counter and extra processing overhead to frequently renormalize the counter values.

In this paper, we propose a memory efficient and accurate flow statistics method named DIScount COunting (DISCO) to support both flow size and flow byte counting and provide both offline and online access to measurement results. The accessing speed of the up-to-date SRAM or fast DRAM is sufficient for per-packet processing, so the goal of DISCO is to compress the required counter bits so as to fit the counters in a small but fast memory (SRAM or small fast DRAM [21], [23]). 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. Fig. 1 illustrates how DISCO counter updates with a real trace segment input. 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 seven times (2334/321) in this case. In general, 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 required counter size is greatly compressed compared to a full-size counter-like SD solution. In this way, the technical challenge is how to determine Δ and its inverse estimation, and the proposed solution to this challenge is the major merit of this paper.

Especially, 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 sublinearly with the increase of the flow length, making the counters easily implementable in an SRAM for online 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 96 kb on-chip memory is required for both flow size and flow volume counting. When using one micro-engine (ME), the throughput can reach up to 11.1 Gb/s, and the throughput keeps increasing if more MEs are utilized.

It is worth noting that DISCO goes a big step beyond our previous work, Adaptive Non-Linear Sampling (ANLS) [9], which is used for flow size counting only, and in this paper, DISCO is developed to further support both flow volume and flow size statistics. 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 Sections II and V, simple extensions on ANLS do not work for flow volume counting. The experiment results demonstrate that DISCO is fast, accurate, and memory-efficient. We use SRAM (or fast DRAM[1]) to implement counters, and one read/write operation on the counter per-packet is not a big concern, but counter width is the issue that our approach mostly addresses. The basic idea of DISCO is presented at its previous conference version [8]. This paper gives more theoretic analysis on the relative error and memory cost of DISCO and describes the detailed information about the implementation.

The rest of the paper is organized as follows. Section II reviews the related work. Section III presents the detailed design of DISCO, including the architecture, 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. In Section VII, we conclude the paper.

II. Related Work

A. Dram-Based Full-Size Counters

A combined SD counter architecture is first proposed in [18]. The increments are first made only to SRAM counters, and the values of each SRAM counter are 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 [17], [18], [24]. While the contribution 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 the 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. Leveraging modern fast DRAM, it is proposed a randomized DRAM

[1]When fast DRAM is employed, complementary mechanism exploring burst input/output may be needed to guarantee throughput [21].

architecture in [21], [23], which can harness the performance of fast DRAM offerings by interleaving counter updates to multiple memory banks. However, without any compressing on the statistics, this method still faces the risk to overflow their counters when counting flow bytes or needs a quite large fast DRAM.

B. Sampling-Based Method

Sampling-based method selects packets with a probability, and each selected packet will trigger an update to the counter [2], [4]. With a sampling rate of p, if c packets have been sampled in an n-packet flow train, the unbiased estimation of the total packets is $\hat{n} = c/p$. There are a number of variations of sampling-based methods [1], [5], [6], [9], [12], [22], 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 Fig. 1, if E1 samples the first and the third packets, the counter is $81+0+142+0 = 223$. However, it may also only sample the first and the fourth packets, 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 the 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 awkwardly for flow volume counting.

C. Elaborate Counter Organization

Traditional counting system configures all the counters as the same size, and this implementation is not efficient for flow length counting. A recent work in [10] proposed Bucketized Rank Index Counter (BRICK) to organize efficient "variable-length" counters. The basic idea of BRICK is intuitive and is based on statistical multiplexing, which bundles groups of a fixed number (say 64) of counters randomly selected from the array into buckets. BRICK allocates just enough bits to each counter in the sense that if its current value is c_i, BRICK allocates $\lfloor lg(c_i) \rfloor + 1$ bits to it. Counter Braids (CB) [13] is another novel counter organization for accurate flow measurement, which builds a hierarchy of counters braided via random graphs in tandem. CB allows the sharing of counter bits, and thus the required counter bits are reduced. The motivations and gains of BRICK and CB are different from our solution in this paper. BRICK and CB achieve the memory compression by organizing the counters with statistical multiplexing, but they do not compress the size for each single counter. DISCO is a statistical counter-updating algorithm to save memory consumption of each counter. In this paper, DISCO uses a uniform length for each counter for easy presentation and understanding, but in fact, we can combine BRICK/CB with DISCO, i.e., we organize the counters with BRICK/CB and update the counter using DISCO. In this way, we will achieve more memory savings.

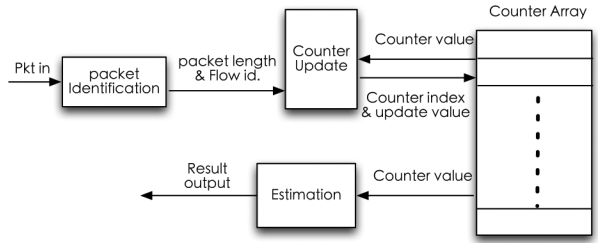


Fig. 2. Architecture of DISCO. For simple illustration, only one directional line is drawn to show the main signals between any two modules.

D. Small Active Counters

The term "active counter" is introduced in [19], which allows estimation on a per-packet basis without DRAM access. Small Active Counters (SAC) is proposed to reduce the SRAM space needed for the statistic counters [19]. 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 renormalizes the counter. If $mode$ overflows, r is incremented, and all the counters are renormalized. 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 counter update and cause potential loss of necessary packet updates.

III. DISCO: DISCOUNT COUNTING

A. Architecture

A descriptive architecture of DISCO is depicted in Fig. 2. An incoming packet is first inputted to a "packet identification" module for extraction of the flow ID and the packet length. The flow ID is a number used to identify the different flows, and the packet length is set to be the bytes of the packet for flow bytes statistics or to be one for flow length statistics.

The packet length is paged to a counter array, which contains one counter for each flow. When paging the packet length into the counter array, a probabilistic counting algorithm is proposed to compress the counter for flow statistics. The packet identification process is a packet/flow classification task, which has been extensively discussed in the literature [25]. We study in this paper the counting algorithm, which consists of two parts:

TABLE I
NOTATIONS

Notations	Descriptions
b	a predefined parameter, $b > 1$
c	the counter value
l	the bytes of an incoming 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
m	the number of packets in a flow
n	the actual flow length
$\hat{n}$	the estimated flow length
$T(S)$	traffic amount that lets counter value to be S
e	Coefficient of variation of $T(S)$

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.[2]

B. Counter Update

For convenience, the main notations utilized in this paper are first illustrated in Table I.

As mentioned in Section I, the goal of DISCO is to compress the required counter bits so as to fit the counters in a fast but small memory. 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 predefined 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.[3] In this way, the "growing" of the counter value will be slower than the linear increasing, and thus the counting is scalable. Although other increasing convex functions may be used as $f(\cdot)$, our analysis and simulation show that (1) is a good formula.

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 $\Delta(c, l) = f^{-1}(l + f(c)) - c$. The actual flow length can be calculated from the counter value c by $f(c)$ with no error. Since there is not 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

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

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

Please note that the larger the counter value and/or packet length is, the smaller the increase of a counter is. It is guaranteed that $0 \leq p_d \leq 1$.[4]

Algorithm 1: Counter update algorithm

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

Theorem 1: When δ and p_d are defined as (2) and (3), it holds that $0 \leq p_d \leq 1$.

Proof: Start the proof from the definition of δ in (2)

$$\begin{aligned} \delta &= \lceil f^{-1}(l + f(c)) - c \rceil - 1 \\ f^{-1}(l + f(c)) - c - 1 \leq \delta &\leq f^{-1}(l + f(c)) - c \\ \delta + c \leq f^{-1}(l + f(c)) &\leq c + \delta + 1. \end{aligned}$$

Since $f(\cdot)$ is an increasing function, we have

$$\begin{aligned} f(c + \delta) \leq l + f(c) &\leq f(c + \delta + 1) \\ 0 \leq \frac{l + f(c) - f(c + \delta)}{f(c + \delta + 1) - f(c + \delta)} &\leq 1. \end{aligned} \tag{4}$$

Namely, $0 \leq p_d \leq 1$. ■

C. Flow Size Counting

The above algorithm is obviously suitable for flow volume counting, and when the packet length of each packet is viewed as one, DISCO counts the flow size. In this way, $l = 1$ and

$$\delta(c, 1) = \lceil f^{-1}(l + f(c)) - c \rceil - 1 = 0 \tag{5}$$

$$p_d(c, 1) = \frac{1}{f(c + 1) - f(c)}. \tag{6}$$

Therefore, the counting process of DISCO can be presented as $c \leftarrow c + 1$ with probability $p(c)$, where c is the counter value and $p(c) = 1/[f(c + 1) - f(c)]$.

D. 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

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

[3] 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 $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y)$. 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 $f(\lambda x + (1 - \lambda)y) \geq \lambda f(x) + (1 - \lambda) f(y)$.

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

counter and suppose that, during a measurement interval, there are m packets whose packet lengths are $l_1, l_2, \ldots, l_m$ ($l_i, i = 1, 2, \ldots, m$ can be positive integer), respectively. The counter value is updated to c_i after the arrival of the 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 $u_1, u_2, \ldots, u_t$. 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 m th packet, the counter value will have $s = 2^m$ possibilities, denoted as $v_1, v_2, \ldots, v_s$. Ror $\forall i, 1 \le i \le s$, the probability $p(c_m = v_i)$ is denoted as Q_i. The following equations hold:

$$v_{2j-1} = u_j + \delta(u_j, l_m) \tag{7}$$

$$v_{2j} = u_j + \delta(u_j, l_m) + 1 \tag{8}$$

$$Q_{2j-1} = P_j[1 - p_d(u_j, l_m)] \tag{9}$$

$$Q_{2j} = P_j p_d(u_j, l_m). \tag{10}$$

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

$$\begin{aligned} F_m &= \sum_{j=1}^{} (f(v_j)Q_j) \\ &= \sum_{j=1}^{t} (f(v_{2j-1})Q_{2j-1} + f(v_{2j})Q_{2j}) \\ &= \sum_{j=1}^{t} [f((u_j) + \delta(u_j, l_m) + 1)(P_j p_d(u_j, l_m)) \\ &\quad + f(u_j + \delta(u_j, l_m))P_j(1 - p_d(u_j, l_m))] \\ &= \sum_{j=1}^{t} P_j\{p_d(u_j, l_m)[-f(u_j + \delta(u_j, l_m)) \\ &\quad + f(u_j + \delta(u_j, l_m) + 1)] + f(u_j + \delta(u_j, l_m))\} \\ &= \sum_{j=1}^{t} P_j[l_m + f(u_j)] \text{ (Substitute (3))} \\ &= l_m + F_{m-1}. \end{aligned} \tag{11}$$

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

$$p_d(0, l_1) = \frac{l - f(\delta)}{f(\delta + 1) - f(\delta)} \tag{12}$$

$$\delta(0, l_1) = \lceil f^{-1}(l_1) \rceil - 1 \tag{13}$$

$$F_1 = p_d(0, l_1) f(\delta + 1) + (1 - p_d(0, l_1)) f(\delta) = l_1. \tag{14}$$

Combining (11) and (14), the following equation holds by a mathematical induction argument:

$$F_m = E[f(c_m)] = \sum_{i=1}^{m} l_i. \tag{15}$$

The assertion of the theorem follows. ∎

IV. Properties

A. Variation and Error

Denote S as the random variable counter value (of a corresponding flow) after a number of packets (of the same flow) and $T(S)$ as the estimation of the total traffic amount from the counter value S. Since the coefficient of variation (COV) is an indication of relative error, we analyze $T(S)$ in this paper to formulate relative error. COV is defined as

$$e = \mathrm{COV}(T(S)) = \sqrt{\frac{\mathrm{Var}[T(S)]}{E^2[T(S)]}}. \tag{16}$$

Lemma 1: Given two flows (or packet sequences) with the same volume. Suppose that the first flow has $m + 1$ packets, whose packet lengths are $l_1, l_2, \ldots, l_{m-1}, l_{m1}, l_{m2}$, and the second packet sequence is $l_1, l_2, \ldots, l_{m-1}, l_m = l_{m1} + l_{m2}$. Using DISCO to count and estimate the volumes of these two flows, the COV of the packet sequence one is larger than the COV of the second flow.

Proof: With DISCO, the counter is updated $m + 1$ times or m times for each packet in the first or the second packet sequence. The same as the analysis in Section III-D, after the $(m-1)$th update (no matter the first flow or the second flow), the counter value can be one of the total $t = 2^{m-1}$ values, which is denoted as $u_1, u_2, \ldots, u_t$. In addition, the probability $p(c_{m-1} = u_j)$ is denoted as P_j.

As to the first flow, on the condition that the counter value is u_j with probability P_j after the $(m-1)$th packet, there are (at most) four possibilities on counter values after two more counter updates (the mth and the $(m+1)$th packet). There are (at most) four possible counter values; we denote them as w_{4j-3}, w_{4j-2}, w_{4j-1}, w_{4j}, $j = 1, 2, \ldots, t$. The probability that the counter value equals w_{4j-3}, w_{4j-2}, w_{4j-1}, or w_{4j} can be formalized as the following:

$$Q_{4j-3} = P_j[1 - p_d(u_j, l_{m1})][1 - p_d(v_{2j-1}, l_{m2})] \tag{17}$$

$$Q_{4j-2} = P_j[1 - p_d(u_j, l_{m1})]p_d(v_{2j-1}, l_{m2}) \tag{18}$$

$$Q_{4j-1} = P_j p_d(u_j, l_{m1})[1 - p_d(v_{2j}, l_{m2})] \tag{19}$$

$$Q_{4j} = = P_j p_d(u_j, l_{m1}) p_d(v_{2j}, l_{m2}). \tag{20}$$

Since the first $m - 1$ packets in the second flow are of same length as the first flow, we also use u_j and P_j to denote the counter value after the $(m-1)$th packet and its corresponding probability for the second flow. After the mth update for the second flow with l_m, the counter value will have two possibilities, denoted as $v_{2j-1}^{(m)}$, $v_{2j}^{(m)}$. The probability $p(c_m = v_i^{(m)})$ is denoted as $Q_i^{(m)}, \forall i, i = 2j-1, 2j$. Thus, we have

$$v_{2j-1}^{(m)} = u_j + \delta(u_j, l_m) \tag{21}$$

$$v_{2j}^{(m)} = u_j + \delta(u_j, l_m) + 1 \tag{22}$$

$$Q_{2j-1}^{(m)} = P_j[1 - p_d(u_j, l_m)] \tag{23}$$

$$Q_{2j}^{(m)} = P_j p_d(u_j, l_m). \tag{24}$$

We denote $D = E[f^2(w)]$ for the flow one and $D' = E[f^2(v^{(m)})]$ for the flow two. We also denote e

and e' as the COVs for the first and the second flow, respectively. According to the counter Algorithm 1, $f(c)$ is convex, therefore there can be only three possible cases.

1) In the first case, $w_{4j} = w_{4j-2} = v_{2j}^{(m)}$, $w_{4j-1} = w_{4j-3} = v_{2j-1}^{(m)}$, $v_{2j-1}^{(m)} + 1 = v_{2j}^{(m)}$. We have

$$D = \sum_j f^2(w_{4j})(Q_{4j} + Q_{4j-2}) + f^2(w_{4j-1})(Q_{4j-1} + Q_{4j-3}) \quad (25)$$

$$D' = \sum_j f^2(w_{4j})Q_{2j-1}^{(m)} + f^2(w_{4j-1})Q_{2j}^{(m)}. \quad (26)$$

Substituting (3) and (17)–(20) into (25) and substituting (3) and (21)–(24) into (26), we have

$$D = \sum_j \left\{ f^2(w_{4j}) \left(\frac{l_{m1} + l_{m2} + f(u_j) - f(w_{4j-1})}{f(w_{4j}) - f(w_{4j-1})} \right) + f^2(w_{4j-1}) \left(1 - \frac{l_{m1} + l_{m2} + f(u_j) - f(w_{4j-1})}{f(w_{4j}) - f(w_{4j-1})} \right) \right\} = D'.$$

Therefore, in this case, the variation and the coefficient of the variation of the two flows are the same.

2) In the second case, $w_{4j} = w_{4j-1} + 1 = w_{4j-2} + 1 = w_{4j-3} + 2$, $w_{4j} = v_{2j}^{(m)}$, $w_{4j-1} = w_{4j-2} = v_{2j-1}^{(m)}$

$$D = \sum_j \{ f^2(w_{4j})Q_{4j} + f^2(w_{4j-1})(Q_{4j-1} + Q_{4j-2}) + f^2(w_{4j-3})Q_{4j-2} \} \quad (27)$$

$$D' = \sum_j f^2(w_{4j-1})Q_{2j-1}^{(m)} + f^2(w_{4j})Q_{2j}^{(m)}. \quad (28)$$

It is easy to obtain that $D > D'$ by substituting (3) and (17)–(20) into (27) and substituting (3) and (21)–(24) into (28).

3) In the third case, $w_{4j} = w_{4j-1} + 1 = w_{4j-2} + 1 = w_{4j-3} + 2$, $w_{4j-1} = w_{4j-2} = v_{2j}^{(m)}$, $w_{4j-3} = v_{2j-1}^{(m)}$. D is the same as the formulation in (27), and D' can be calculated as above

$$D' = \sum_j f^2(w_{4j-3})Q_{2j-1}^{(m)} + f^2(w_{4j-2})Q_{2j}^{(m)}. \quad (29)$$

Again, substitute (3) and (21)–(24) into (29), and it is obvious that $D > D'$ in this case.

Since $e = \sqrt{(D - n^2)/n^2}$ and $e' = \sqrt{(D' - n^2)/n^2}$, $e \geq e'$ in all the cases. In other words, the relative error (or COV) of the first flow is larger than the second flow. ■

Theorem 3: The coefficient of variation of DISCO is bounded by $\sqrt{(1 - 1/n)(b - 1)/2}$, where n is the actual flow volume.

Proof: Suppose a packet sequence of a flow is of length $l_1, l_2, \ldots, l_m$, and the total traffic volume is $n = \sum_{i=1}^{m} l_i$. Using DISCO to count this original sequence, the coefficient of variation is e.

Also, we have another packet sequence that has n packets. Each packet in this packet sequence is length of one. Counting such a packet sequence with unit-size packets using DISCO, the coefficient of variation is e''.

The expected estimations ($\hat{n}$) of these two counting processes are the same, i.e., $E(\hat{n}) = n$. We have $e \leq e''$ according to a simple deduction from Lemma 1.

Now let us calculate e'' first. $Q_i(n)$ denotes the probability that counter value c equals i when current actual flow size is n. Since in this case each packet length is thought to be one, we simply use $p(c)$ to represent $p_d(c, 1)$. We have

$$Q_i(n) = \prod_{j=0}^{i-1} p(j) \sum_{\alpha_0 + \cdots + \alpha_i = n - i} (1 - p(0))^{\alpha_0} \ldots (1 - p(i))^{\alpha_i} \quad (30)$$

$$Q_i(n) = Q_{i-1}(n-1)p(i-1) + Q_i(n-1)(1 - p(i)). \quad (31)$$

Let D_n denote the expectation of $f^2(c)$ when the current flow size is n

$$D_n = E(f^2(c)) = \sum_{i=0}^{n} f^2(i)Q_i(n). \quad (32)$$

Thus, from (31) and (32), we get

$$\begin{aligned} D_n - D_{n-1} &= \sum_{i=1}^{n} (f^2(i))[Q_{i-1}(n-1)p(i-1) + Q_i(n-1)(1-p(i))] \\ &\quad - \sum_{i=1}^{n-1} (f^2(i))Q_i(n-1)[p(i) + (1-p(i))] \\ &= \sum_{i=2}^{n} (f^2(i))Q_{i-1}(n-1)p(i-1) - \sum_{i=1}^{n-1} (f^2(i))Q_i(n-1)p(i) \\ &= \sum_{i=1}^{n-1} (f^2(i+1))Q_i(n-1)p(i) - \sum_{i=1}^{n-1} (f^2(i))Q_i(n-1)p(i). \end{aligned}$$

Since $f(i+1) - f(i) = 1/p(i)$ and $Q_0(n-1) = 0$

$$D_n - D_{n-1} = \sum_{i=1}^{n-1} Q_i(n-1)[f(i+1) + f(i)]. \quad (33)$$

From (1), we have

$$f(i+1) - f(i) = b^i = f(i)(b-1) + 1. \quad (34)$$

Consequently, (33) is equivalent to

$$\begin{aligned} D_n - D_{n-1} &= \sum_{i=0}^{n-1} Q_i(n-1)\{2f(i) + [f(i+1) - f(i)]\} \text{ (insert (34))} \\ &= \sum_{i=0}^{n-1} Q_i(n-1)(2f(i)) + (b-1)\sum_{i=0}^{n-1} Q_i(n-1)f(i) + 1 \\ &= 2(n-1) + (b-1)(n-1) + 1 = (b+1)(n-1) + 1. \end{aligned}$$

Since $D_1 = 1$, we have

$$D_n = (b+1)\frac{n(n-1)}{2} + n. \tag{35}$$

The variation and coefficient of variation can be formulated by

$$\mathrm{Var}[\hat{n}(c)] = D_n - n^2 = \frac{n(n-1)}{2}(b-1). \tag{36}$$

$$\frac{\sqrt{\mathrm{Var}\hat{n}(c)}}{n} = \frac{\sqrt{\frac{n(n-1)}{2}(b-1)}}{n} = \sqrt{\frac{(1-1/n)}{2}(b-1)}. \tag{37}$$

Therefore, $e \le e'' = \sqrt{\frac{(1-1/n)}{2}(b-1)}$. ■

The relative error is zero when n is one. The coefficient of variation decreases as b diminishes and increases with the increment of n, but converges to $\sqrt{(b-1)/2}$ when $n \to \infty$.

B. 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 4: An upper bound of expected counter value $E[c(n)]$ is $f^{-1}(n)$, where $f^{-1}(n)$ is the inverse function of $f(c)$, and $f^{-1}(n)$ is an increasing concave function when $f(c)$ is chosen from (1).

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

$$f(x) \ge f(y) + (x-y)f_r'(y) \qquad \forall x, y > 0 \tag{38}$$

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

$$f(c) \ge f(E[c]) + (c - E[c])f_r'(E[c]) \tag{39}$$

$$E[f(c)] \ge E[f(E[c]) + (c - E[c])f_r'(E[c])]. \tag{40}$$

From Theorem 2, $E(f(c)) = n$, then we obtain

$$E[f(c)] = n \ge f(E[c]). \tag{41}$$

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

$$E[c(n)] \le f^{-1}(n). \tag{42}$$

Since $f(c)$ is an increasing function, its inverse function $f^{-1}(n)$ is also an increasing function.

As defined in (1), $f(c)$ is an increasing convex function. Hence, for $\forall c_1 < c_2$, we have

$$n_1 = f(c_1) < f(c_2) = n_2 \tag{43}$$

$$\frac{f(c_1) + f(c_2)}{2} > f\left(\frac{c_1 + c_2}{2}\right). \tag{44}$$

Since $f^{-1}(n)$ is an increasing function, we have the following inequality from (44):

$$f^{-1}\left(\frac{f(c_1) + f(c_2)}{2}\right) > f^{-1}\left(f\left(\frac{c_1 + c_2}{2}\right)\right) = \frac{c_1 + c_2}{2}. \tag{45}$$

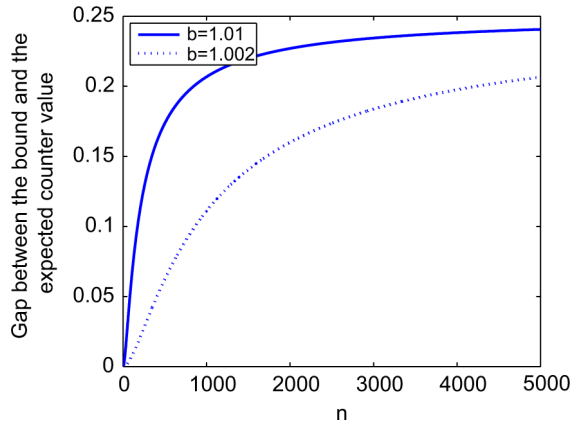


Fig. 3. Gap between the bound and the expected counter value.

Substituting (43) into (45), we have $\forall n_1 < n_2$

$$f^{-1}\left(\frac{n_1 + n_2}{2}\right) > \frac{f^{-1}(n_1) + f^{-1}(n_2)}{2}. \tag{46}$$

Therefore, $f^{-1}(n)$ is an increasing concave function if $f(n)$ is an increasing convex function. ■

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 4 and plot the gap between them in Fig. 3. The figure shows that the bound in Theorem 4 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.

Theorem 3 depicts the relationship between parameter b and the relative error, and Theorem 4 describes how b determines the memory cost. With the two theorems, we can derive b given a constraint on relative error or memory cost.

V. 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 [19]) and k bits to keep the estimation part (named as A in [19]). Therefore, the counter size of SAC is $S_{\mathrm{sac}} = s + k$ and in all our experiments s is set to be 3. In the simulations, we adjust the value of parameter b for DISCO according to Theorem 4 so as to make the counter size be the expected size. The parameter s for SAC is also tuned to keep the total counter size of SAC S_{sac} the same as DISCO's counter size for fair comparisons.

We study how the accuracy changes with the increment of counter size based on the real trace input. Relative error R is

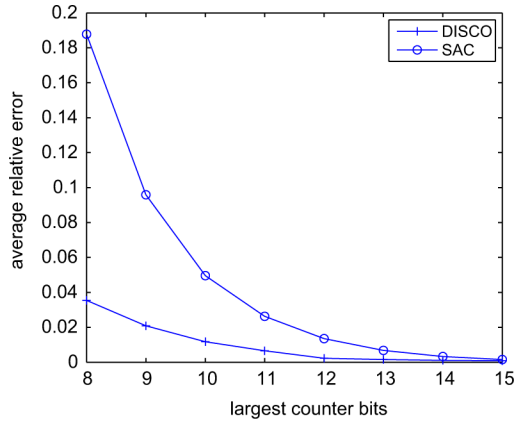


Fig. 4. Average relative error for flow volume counting.

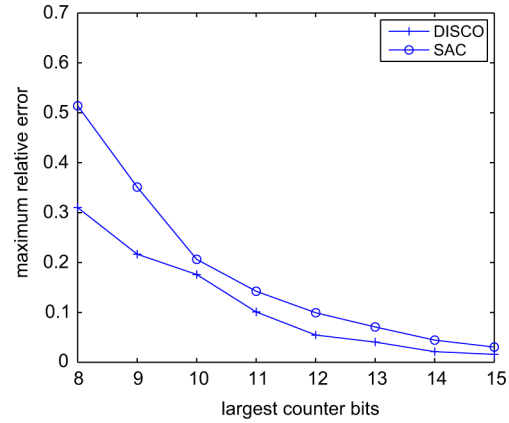


Fig. 5. Maximum relative error for flow volume counting.

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 three metrics 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

$$R_o(\alpha) = \sup\{r | \Pr\{R \le r\} \ge \alpha\}. \quad (47)$$

B. Simulation Results

The performance behaviors of DISCO and SAC are first investigated under a real trace for flow volume counting. The real trace on OC-192 link is obtained from NLANR [15], which represents totally 40 GB traffic volume. In this real trace, the number of flows is 100 728, and the average flow size is 409.5 kB.

Fig. 4 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 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). Fig. 5 shows the maximum relative error and indicates the similar trends as Fig. 4. It is demonstrated that DISCO is more accurate than SAC even in the worst case. Fig. 6 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 Fig. 7 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,

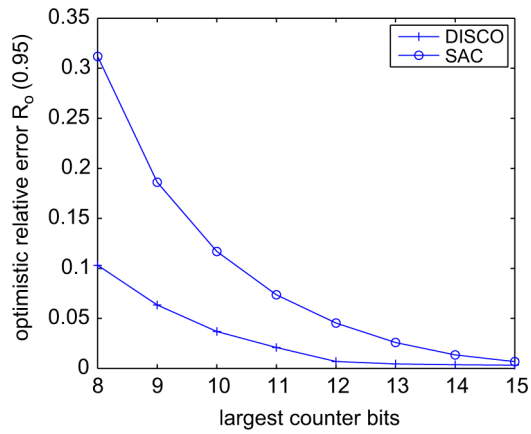


Fig. 6. Optimistic relative error $(\alpha(95))$ for flow volume counting.

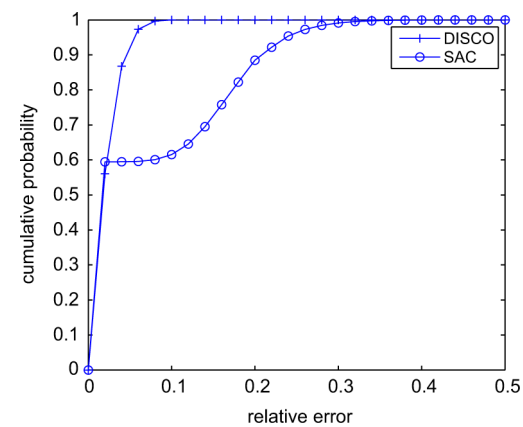


Fig. 7. Cumulative probability distribution of relative error.

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 Fig. 8. The larger the flow volume, the

TABLE II
EXPERIMENT RESULTS UNDER DIFFERENT TRAFFIC SCENARIOS

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

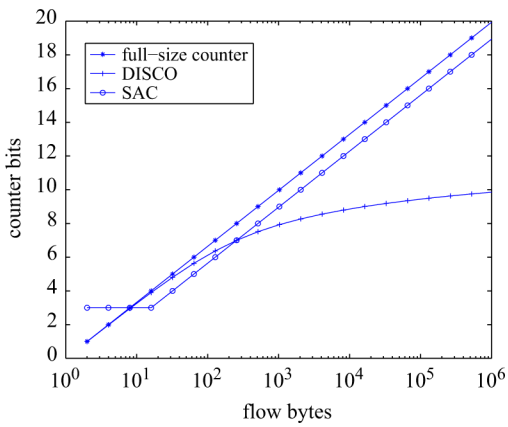


Fig. 8. Counter bits required under different flow volume. The parameter b of DISCO is 1.01. The parameter s of SAC is 2, i.e., two bits are required to keep the exponential part of the SAC estimate. The renormalization of SAC is not applied in the figure.

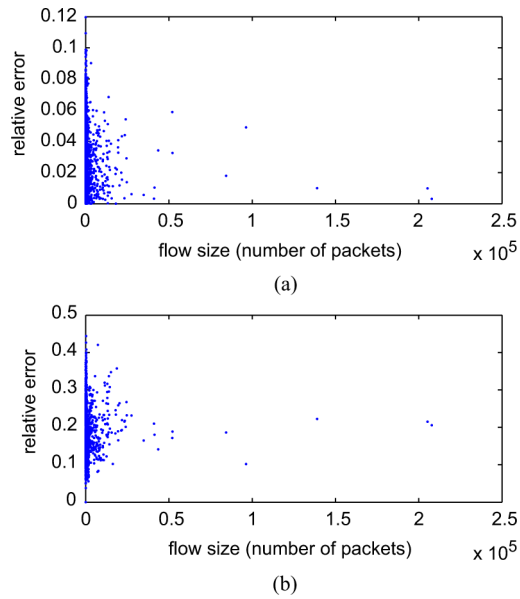


Fig. 9. Relative error of each flow for flow size counting. (a) Results for DISCO where the parameter b is set to be 1.002. (b) Results for SAC where the parameter s is set to be 8. The two methods consume the similar counter size.

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. Fig. 8 also demonstrates that DISCO is scalable for the potential dramatic increase of flow volume in the Internet.

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 DISCO is equivalent to ANLS. Fig. 9 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.

- 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 5.2 kB 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 82.7 kB 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 83.6 kB traffic in this scenario.

Table II illustrates three snapshots when the counter sizes are set to be 8, 9, 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 Section II do not work well for flow volume counting. To be fair, we compare DISCO to ANLS-I and ANLS-II given the same memory size, i.e., all use 10-bit counters for each flow.

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

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

If ANLS-I is utilized, the relative errors are too large to be acceptable as indicated in Table III, compared to 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 10 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

DISCO employs relatively more complicated math operations including exponent, logarithm, and randomness, while SAC only uses exponent and randomness. Therefore, SAC should be faster than DISCO when implemented in a real system. Since the implementation details are not mentioned in [19], we did not implement SAC in order not to potentially degrade its performance due to our simple implementation design. Instead of directly comparing the throughput of DISCO and SAC, we check whether DISCO can achieve wire speed in a core network by implementing it on the Intel network processor IXP2850 platform [11], [14]. IXA SDK 4.0 simulation environment is employed for performance validation.

The architecture of the DISCO implementation and its test-bench is depicted in Fig. 10. Four IXP2850 MEs 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 a 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. Only one external SRAM is used in the implementation. The IXP2850 itself can handle the potential I/O conflicts between multiple MEs to at most four parallel SRAMs. The random number is generated by an instruction provided by the IXP network processor.

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

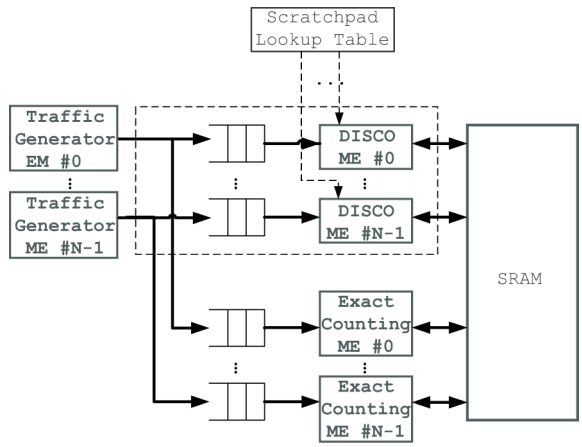


Fig. 10. Implementation of DISCO and the test-bench on IXP 2850.

IXP2850 does not have instructions to calculate logarithm and power computation directly. We precompute $\log_b(X)$ 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 3 K entries for $\log_b(X)$ and b^X, $X \leq 3072$, and the memory of the precomputation table is 96 kb with 3 K entries. For $X > 3072$, $\log_b(X)$ can be calculated using shift and sum operations by

$$\log_b X = \log_b \frac{X}{2^e} + e \log_b 2, \frac{X}{2^e} \leq 3072. \tag{48}$$

The following formulation helps us obtain b^X when $X > 3072$:

$$b^X = \prod_{i=1}^{k} b^{x_i}, x_i \leq 3072,\ X = \sum_{i=1}^{k} x_i. \tag{49}$$

The pseudocodes for logarithm and power computation are in Algorithms 2 and 3. For $X \leq 3072$, a direct table lookup is performed to get the logarithm or power of X; otherwise for $X > 3072$, (48) and (49) are used to decompose the computation. Algorithm 2 introduces implementation errors, but the evaluation in Table V shows acceptable false since the ceiling function in (2) mitigates the wipe error. A larger table to keep the precomputing result of log operation would decrease the chance of wiping off least significant bits. Also, the implementation uses 3 K entries, which could store the value for $X = 1$ to $X = 3 * 1024 = 3072$.

Algorithm 2: $\log_b X$

$t = X, e = 0$
while $t >= 3072$ **do**
  $e++;$
  $t \gg 1;$ /* bit shift. Divide t by 22 */
end while
lookup table for $z1 = \log_b t$ and $z2 = \log_b 2$
return $z1 + e * z2$

Algorithm 3: b^X

$t = X, k = 0;$
while $t >= 3072$ **do**
  $k++;$
  $t = t - 3072;$
end while
lookup table for $z1 = \log_b t$ and $z2 = \log_b 3072$
$z = 1;$
while $k > 0$ **do**
  $k--;$
  $z = z * z2$
end while
return $z * z1$

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 carry 80% of the traffic volume.[5] The packet length is uniformly distributed between 64 B and 1 kB. 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 one, two, and four 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 11.1 Gb/s with a relative error of 0.013, and it is competent enough to serve for flow statistics on the 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, the 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 64 B and arrive without burst, eight MEs are needed to achieve 10 Gb/s 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 take 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 10 times when porting the implementation to a FPGA/ASIC design.

[5]It is well known today that Internet exhibits an "80–20" feature for its traffic [16], i.e., 80% of Internet packets are generated by 20% of the flows.

VII. Conclusion

Acquiring both the flow size and the flow byte statistics in the same algorithm with improved accuracy and low memory occupation is always a target when implementing in real network equipment. In this paper, we have proposed a 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 synthetic data validate the theoretical results. A real implementation is made on the Intel IXP2850 network processor with an inspiring outcome that only 96 kb memory is required and a throughput of 11.1 Gb/s can be achieved by only using one ME. The throughput increases almost linearly when multiple MEs are employed. This makes DISCO performance/cost-effective for practical applications.

Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] B.-Y. Choi, J. Park, and Z.-L. Zhang, "Adaptive random sampling for load change detection," in *Proc. ACM SIGMETRICS*, 2002, pp. 272–273.
[2] Cisco, San Jose, CA, USA, "Sampled Netflow data sheet," [Online]. Available: http://www.cisco.com
[3] K. Claffy and S. McCreary, "Internet measurement and data analysis: Passive and active measurement," [Online]. Available: 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, pp. 194–203.
[5] A. Cvetkovski, "An algorithm for approximate counting using limited memory resources," *Perf. Eval. Rev.*, vol. 35, pp. 181–190, Jun. 2007.
[6] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," in *Proc. ACM SIGCOMM*, 2004, pp. 245–256.
[7] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proc. ACM SIGCOMM*, 2002, pp. 323–336.
[8] C. Hu, B. Liu, H. Zhao, K. Chen, Y. Chen, C. Wu, and Y. Cheng, "Disco: Memory efficient and accurate flow statistics for network measurement," in *Proc. ICDCS*, Washington, DC, USA, 2010, pp. 665–674.
[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 *Proc. IEEE INFOCOM*, Phoenix, AZ, USA, 2008, pp. 26–30.
[10] N. Hua, B. Lin, J. J. Xu, and H. C. Zhao, "Brick: A novel exact active statistics counter architecture," in *Proc. ANCS*, 2008, pp. 89–98.
[11] E. J. Johnson and A. R. Kunze, *IXP2400/2800 Programming*. Santa Clara, CA, USA: Intel Press, 2003.
[12] A. Kumar and J. Xu, "Sketch guided sampling—using on-line estimates of flow size for adaptive data collection," in *Proc. IEEE INFOCOM*, 2006, pp. 1–11.
[13] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani, "Counter braids: A novel counter architecture for per-flow measurement," in *Proc. ACM SIGMETRICS*, 2008, pp. 121–132.
[14] U. R. Naik and P. R. Chandra, *Designing High-Performance Networking Applications*. Santa Clara, CA, USA: Intel Press, 2004.

[15] NLANR, "Passive measurement and analysis (PMA)," [Online]. Available: http://pma.nlanr.net
[16] K. Psounis, A. Ghosh, B. Prabhakar, and G. Wang, "Sift: A simple algorithm for trucking elephant flows and taking advantage of power laws," in *Proc. 43rd Allerton Conf. Commun., Control, Comput.*, 2005.
[17] S. Ramabhadran and G. Varghes, "Efficient implementation of a statistics counter architecture," in *Proc. ACM SIGCOMM*, 2003, pp. 261–271.
[18] D. shah, S. Iyer, B. Prabhakar, and N. McKeown, "Maintaining statistics counters in router line cards," *IEEE Micro*, vol. 22, no. 1, pp. 76–81, Jan.–Feb. 2002.
[19] R. Stanojevic, "Small active counters," in *Proc. IEEE INFOCOM*, 2007, pp. 2153–2161.
[20] G. Varghese and C. Estan, "The measurement manifesto," *Comput. Commun. Rev.*, vol. 34, pp. 9–14, 2004.
[21] H. Wang, H. Zhao, B. Lin, and J. Xu, "DRAM-based statistics counter array architecture with performance guarantee," *IEEE/ACM Trans. Netw.*, vol. 20, no. 4, pp. 1040–1053, 2012.
[22] X. Wang, X. Li, and D. Loguinov, "Modeling residual-geometric flow sampling," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1808–1816.
[23] H. C. Zhao, H. Wang, B. Lin, and J. J. Xu, "Design and performance analysis of a DRAM-based statistics counter array architecture," in *Proc. 5th ACM/IEEE ANCS*, New York, NY, USA, 2009, pp. 84–93.
[24] Q. Zhao, J. J. Xu, and Z. Liu, "Design of a novel statistics counter architecture with optimal space and time efficiency," in *Proc. ACM SIGMETRICS*, 2006, pp. 323–334.
[25] K. Zheng, H. Che, Z. Wang, B. Liu, and X. Zhang, "DPPC-RE: TCAM-based distributed parallel packet classification with range encoding," *IEEE Trans. Comput.*, vol. 55, no. 8, pp. 947–961, Aug. 2006.

Chengchen Hu (S'04–M'09) received the B.S. degree in automation from Northwestern Polytechnical University, Xi'an, China, in 2003, and the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2008.

He worked as an Assistant Research Professor with Tsinghua University from 2008 to 2010 and is now an Associate Professor in the MOE Key Laboratory for Intelligent Networks and Network Security, Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China. His recent research interests include computer networking systems, network measurement and monitoring, cloud data center networks, and software defined networking.

Dr. Hu serves in the organization committee and technique program committee of several conferences, e.g., INFOCOM, IWQoS, GLOBECOM, ICC, etc.

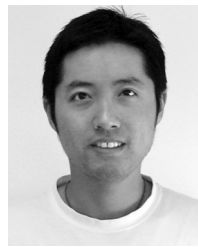
Bin Liu (M'03–SM'12) was born in 1964. He received the M.S. and Ph.D. degrees in computer science and engineering from Northwestern Polytechnical University, Xi'an, China, in 1988 and 1993, respectively.

He is now a Full Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research areas include high-performance switches/routers, network processors, high-speed security, and greening the Internet.

Prof. Liu has received numerous awards from China, including the Distinguished Young Scholar of China and won the inaugural Applied Network Research Prize sponsored by ISOC and IRTF in 2011.

Hongbo Zhao received the B.S. degree from the Beijing Institute of Technology, Beijing, China, in 2007, and the M.S. degree from Tsinghua University, Beijing, China, in 2010, both in computer science and technology.

He worked as an Engineer with Ericsson Research, Beijing, China, from 2010 to 2013. He is now working with MeshSr Co., Ltd., Nanjing, China.



Kai Chen received the Ph.D. degree in computer science from Northwestern University, Evanston, IL, USA, in 2012.

He is an Assistant Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. His research interests include networked systems design and analysis, data center networks, and cloud computing. He is interested in finding simple yet deep and elegant solutions to real-world networking and systems problems.

Yan Chen (M'03) received the Ph.D. degree in computer science from the University of California, Berkeley, CA, USA, in 2003.

He is an Associate Professor with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA. Based on Google Scholar, his papers have been cited over 5000 times. His research interests include network security, measurement, and diagnosis for large-scale networks and distributed systems.

Dr. Chen has chaired several conferences such as the ACM CCS, SecureComm, IEEE IWQoS, and IEEE GLOBECOM, and NGN, and has served track/area chairs for WWW and IEEE CNS. He won the Department of Energy (DoE) Early CAREER Award in 2005, the Department of Defense (DoD) Young Investigator Award in 2007, and the Microsoft Trustworthy Computing Awards in 2004 and 2005 with his colleagues. His paper won the Best Paper nomination in ACM SIGCOMM 2010.

Yu Cheng (S'01–M'04–SM'09) received the B.E. and M.E. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2003.

From 2004 to 2006, he was a Postdoctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. Since 2006, he has been with the Department of Electrical and Computer Engineering, Illinois Institute of Technology (IIT), Chicago, IL, USA, and now as an Associate Professor. His research interests include next-generation Internet architectures and management, wireless network performance analysis, network security, and wireless/wireline interworking.

Dr. Cheng served as a Co-Chair for the Wireless Networking Symposium of IEEE ICC 2009, a Co-Chair for the Communications QoS, Reliability, and Modeling Symposium of IEEE GLOBECOM 2011, a Co-Chair for the Signal Processing for Communications Symposium of IEEE ICC 2012, a Co-Chair for the Ad Hoc and Sensor Networking Symposium of IEEE GLOBECOM 2013, and a Technical Program Committee (TPC) Co-Chair for WASA 2011. He is an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and New Books and Multimedia Column Editor for *IEEE Network*. He received a Postdoctoral Fellowship Award from the Natural Sciences and Engineering Research Council of Canada (NSERC) in 2004 and a Best Paper Award from the conferences QShine 2007 and ICC 2011. He received the National Science Foundation (NSF) CAREER Award in 2011 and IIT Sigma Xi Research Award in the junior faculty division in 2013.

Hao Wu received the B.S. degree in information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2011, and is currently pursuing the Ph.D. degree in computer science and technology at Tsinghua University, Beijing, China.

His research interests include caching algorithm in CCN and packet switching algorithms.