Highly Compact Virtual Active Counters for Per-flow Traffic Measurement

You Zhou[†] Yian Zhou[†‡] Shigang Chen[†] Youlin Zhang[†]
[†]Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL, USA
[‡]Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA, USA
Email: youzhou@cise.ufl.edu yianzhou@google.com sgchen@cise.ufl.edu youlin@cise.ufl.edu

***Abstract*—Per-flow traffic measurement is a fundamental problem in the era of big network data, and has been widely used in many applications, including capacity planning, anomaly detection, load balancing, traffic engineering, etc. In order to keep up with the line speed of modern network devices (e.g., routers), per-flow measurement online module is often implemented by using on-chip cache memory (such as SRAM) to minimize per-packet processing time, but on-chip SRAM is expensive and limited in size, which poses a major challenge for traffic measurement. In response, much recent research is geared towards designing highly compact data structures for approximate estimation that can provide probabilistic guarantees for per-flow measurement. The state of art, called Counter Tree (CT), requires at least 2 bits per flow in memory consumption and more than 2 memory accesses per packet in processing time. In this paper, we propose a novel design of a highly compact and efficient counter architecture, called Virtual Active Counter estimation (VAC), which achieves faster processing speed (slightly more than 1 memory access per packet on average) and provides more accurate measurement results than CT under the same allocated memory. Moreover, VAC can perform well even with a very tight memory space (less than 1 bit per flow or even one fifth of a bit per flow). Theoretical analysis and experiments based on real network traces demonstrate the superior performance of VAC.**

I. Introduction

The rapid development of wired and wireless network arises enormous amount of data that flows on the Internet. Monitoring and measuring such big network data becomes a daunting task requiring tremendous resources, while has many important applications, including capacity planning, anomaly detection, load balancing, traffic engineering, etc [1]–[10]. One fundamental measurement problem over big network data is per-flow traffic measurement, which is to measure the *number of packets in each flow* (or called *flow size*) during a measurement period [11]–[19]. The network data is modeled as abstract *flows*, each representing a data subset defined depending on measurement requirements. Each flow is uniquely identified by a particular field in the packet header called *flow label*. For example, if the flow label is the source address in the packet header, then all the packets from the same source address constitute a flow. Therefore, we have per-source flows. Similarly, we can define per-destination flows, TCP flows, WWW flows, or any other specific flows.

To keep up with the line speed of modern network routers, per-flow measurement online module is often implemented by using on-chip cache memory (such as SRAM) to minimize per-packet processing time, but on-chip SRAM is expensive and limited in size, which poses a major challenge for per-flow traffic measurement. Moreover, today's network streams usually consist of millions of flows. It is infeasible to exactly measure each flow's size in modern routers [17], [20]. Therefore, much recent research is geared towards designing efficient data structures for approximate estimation that can provide probabilistic guarantees for per-flow traffic measurement [15]–[26].

One representative work is Count-Min [20], [21]. It demonstrated that giving each flow a separate counter in on-chip cache memory cannot scale today's big network data. By contrast, it uses hash functions to map flows to a frequency table of flows, and reduces memory overhead by sharing counters among different flows. Counter Braids (CB) [15], [16] improve on Count-Min sketch by improving the estimation accuracy in per-flow measurement. It maps each flow to k counters, and increases the counters by one for each packet. Multiple flows may be mapped to the same counter. Therefore, each counter represents one linear equation over the sizes of some flows. When there are enough counters, we may solve the equations for the flow sizes. However, CB performs $2k$ (occasionally $4k$) memory accesses for each packet, which may limit its throughput. Moreover, it cannot give converged results when the memory space is tight [17], [18].

Li et al. [17] proposed a counter sharing scheme called randomized counter sharing. To record a packet, it only needs to update one counter with two memory accesses and one hash computation. Its major drawback is that the measurement range is limited [18], e.g., no more than a few thousands for a typical implementation. The state of art is a two-dimensional counter sharing architecture called Counter Tree (CT) proposed by Chen et al. [18], where counters are not only shared by different flows but also among themselves. It achieves better memory efficiency than previous work and extends the estimation range. However, CT still requires more than 2 memory accesses per packet in processing time, and cannot work well under very tight memory space, e.g., less than 1 bit per flow.

Our Contribution: To further improve the memory and processing time efficiency, in this paper, we propose a novel counter architecture for per-flow traffic measurement. The main contributions are summarized as follows. First, we propose a new approximate per-flow measurement design based on virtual active counter arrays (VAC), which is highly compact in space and highly efficient in processing. To record a packet, VAC only requires one hash computation and one memory access in most cases (two in the worst case), while the previous approaches require 2 memory accesses or more. Second, VAC can handle large counting ranges without modifying preset parameters (which is required by the best prior work [18] in order to generate good measurement results when facing different traffic situations). It provides a more robust and flexible solution to

978-1-5386-4128-6/18/$31.00 ©2018 IEEE

meet real-life network traffic measurement demands. Third, we theoretically analyze the performance of VAC, and perform extensive experiments with real network traces to compare VAC with the best state of art. The experimental results demonstrate the superior performance of VAC.

II. Performance Metrics

In this paper, we evaluate the performance of a per-flow traffic measurement scheme based on following metrics, which are also used in [18], [25].

Memory overhead: The total memory required for the per-flow traffic scheme. As we explained in the introduction, the scheme should be made as compact as possible.

Processing time: The average time required to encode a packet. Generally, it is measured by the average number of memory accesses and the number of hash value computations as in [18].

Estimation accuracy: Let n be the actual size of a flow, and $\hat{n}$ be the estimated flow size. The estimation accuracy is determined by the relative bias $Bias(\frac{\hat{n}}{n})$ and relative standard error $StdErr(\frac{\hat{n}}{n})$:

$$\begin{aligned} Bias(\frac{\hat{n}}{n}) &= E(\frac{\hat{n}}{n}) - 1, \\ StdErr(\frac{\hat{n}}{n}) &= \frac{\sqrt{Var(\hat{n})}}{n}. \end{aligned} \tag{1}$$

III. Per-flow Measurement Based on Virtual Active Counter Arrays

A. Virtual Active Counter Arrays

When the memory space is limited and there are more flows than the counters, we will have to share counters among different flows. Count-Min [20] maps each flow to a number k of counters. Many flows may be mapped to the same counter. Each packet of a flow causes the k counters of the flow to increase by one. The smallest of the k counters will be used as an estimation of the flow size. However, the smallest counter may still be the sum of multiple flows. Therefore, this approach has a positive bias, which will become significant as the flow-to-counter ratio increases. For on-chip ASIC implementation, memory access can be the bottleneck for processing at the line speed. In Count-Min, each packet causes $2k$ memory accesses, one read and one write for every counter. In order to control the bias, the value of k has to be reasonably large. In comparison, Counter Braids [15], [16] increase k or occasionally $2k$ counters per packet, incurring $2k$ or $4k$ memory accesses. Randomized counter sharing [17] updates one counter per packet with 2 memory accesses. Counter Tree [18] updates one or (occasionally) more counters with 2 or more memory accesses; in the worst case, it has to update $O(\log m)$ counters, where m is the total number of counters used by all flows.

Although randomized counter sharing [17] incurs only 2 memory accesses per packet, it uses traditional counters and the counting range is limited by the counter size. If it increases counter size for range, the number of counters will be reduced, which hurts the accuracy in flow-size measurement. If it uses small counters for accuracy, the range will be reduced, which limits its capability of measuring large flows. Counter Tree [18] improves the range over randomized counter sharing with more memory accesses. This paper provides a new counting design that reduces per-packet overhead further down to 1 memory access for most packets, with 2 memory accesses in the worst case, which may potentially double the throughput that can be handled. In the meantime, we want the new design to use small counters to achieve large counting range.

Active counters with adaptive sampling [27] can provide large counting range with small counter size through probabilistic counting [28], which will be briefly reviewed shortly. In a straightforward design, each flow is assigned a separate active counter. Each packet of the flow is recorded in the counter probabilistically, requiring a read of the counter but only occasionally a write-back. The number of counters is equal to the number of flows. (There also needs space overhead for an indexing structure that maps flows to counters.) This design will not work in tight on-chip cache memory when the number of bits is fewer than the number of flows; even though an active counter is compact and takes a smaller number of bits than a regular counter, one bit will not be enough.

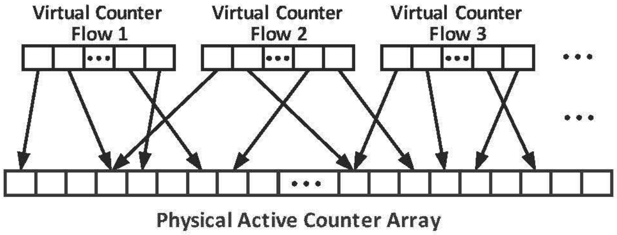


Fig. 1: An illustration of flows sharing counters from the physical active counter array.

To make our new design work under such tight space, counter-intuitively, we allocate an active counter array AC_f, consisting of a large number s of active counters, to each flow f. For each packet of the flow, we randomly pick a single counter $AC_f[i]$ from the array to probabilistically record the packet, where $0 \le i < s$. This is in contrast to Count-Min and Counter Braids which increase all k counters for each packet. For space compactness, the trick is that array AC_f is not physical, but logical. It is formed by pseudo-randomly taking s counters from a physical active counter array AC_* that is shared by all flows. AC_* is pre-constructed from the allocated memory for a traffic measurement function that monitors per-flow sizes. As illustrated in Figure 1, there is no limit on the number of flows that can be supported by AC_*. Each flow simply draws s counters from this common counter pool at random. When a new flow arrives, its virtual counter array is already implicitly embedded in AC_* under the mapping below.

Let m be the number of active counters in AC_*, where the ith counter is denoted as $AC_*[i], 0 \le i < m$. Consider an arbitrary flow f, we pseudo-randomly select the counters in AC_f as follows:

$$AC_f[i] = AC_*[H(f \oplus R[i])] \ , \ 0 \le i < s, \tag{2}$$

where $H(\cdot)$ is a hash function whose range is $[0, m)$, $\oplus$ is the XOR operator, and R is an array of s randomly-chosen constants. We stress that AC_f will not be explicitly constructed during online operations of recording packets. It is a logical construct that facilitates our description. Therefore, we call AC_f the *virtual active counter array* for flow f. Recording a packet from f is *logically* performed on a counter in AC_f, while *physically* it will be translated into an operation on a counter in AC_*, as we discuss later in the section.

We explain the reason of using a counter array (instead of a single counter) for each flow. If we use one counter per flow, in case when there are more flows than counters, each counter will have to be assigned to multiple flows and record the sum of the flows. We will not be able to know the individual flow sizes in the counter. If we consider a flow's size information is the noise to other flows in the same counter, the challenge is how to remove such noise. Now if we use a large number s of counters per flow, each counter carries a small portion of the flow's information. All flows share counters in AC_* uniformly at random. Any flow's information has an equal chance of causing noise to another flow. As all flows' information is split into small pieces and randomly placed in AC_*, the noise is spread out in AC_* roughly uniformly, if s is sufficiently large and each flow's size is negligibly small when comparing with the combined size of all flows. Such uniform noise can be measured statistically and removed. From an arbitrary flow's point of view, its virtual counter array AC_f has s counters from AC_* and thus carries an expected $\frac{s}{m}$ portion of the total noise in AC_*. The total noise is simply the sum of all other flows' sizes, which can be approximately computed by adding all counters in AC_*.

B. Active Counters

We briefly review active counters [28] in the context of this paper's per-flow size measurement, and make an improvement to it. The idea can be motivated from the scientific notation of a decimal number. Consider a number 123456789. With two significant figures, the number can be approximated as 1.2×10^8, with an error of 2.8%. It takes three decimal digits to store 1, 2 and 8, cutting the space requirement to one third. Now consider a 32-bit binary number 10101010101010101010101010101010. With five significant bits, the number can be approximated as 10101×2^{11011}, with an error of 1.6%. It takes 10 bits to store 10101 and 11011, with a range of $2^{5+2^5-1} = 2^{36}$.

In general, each active counter c consists of a coefficient α and an exponent β, which are a and b bits long, respectively. The value of the counter is

$$c = c.\alpha \times 2^{c.\beta}. \tag{3}$$

For example, suppose $a = 4$ and $b = 5$. If $c.\alpha = 1001$ and $c.\beta = 00101$, the counter value is $c = 288$. (We omit the subscript 2 for binary and 10 for decimal because the context typically make it obvious which base is used.)

Suppose we receive a new packet and want to record it in an active counter. We cannot simply increase $c.\alpha$ by one because that will actually increase the counter value by $2^{c.\beta}$. To circumvent this problem, we treat c as a probabilistic counter with a sampling probability of $\frac{1}{2^{c.\beta}}$. To add one, we increase $c.\alpha$ by one with a probability of $\frac{1}{2^{c.\beta}}$. That is, on average, $2^{c.\beta}$ arrival packets will cause $c.\alpha$ to increase by one.

When $c.\alpha$ has reached its largest value with all a bits being ones, i.e., 1...1, if we increase it by one, it overflows. In this case, $c.\alpha$ becomes 10...0, still a bits with the last zero cut off, and $c.\beta$ is increased by one. For example, suppose $c.\alpha = 1111$, $c.\beta = 00101$, and the counter value is 480. If we increase $c.\alpha = 1111$ by one, it will become 1000 and $c.\beta$ will become 00110. The counter value will become 512. The difference in counter value is 32. Recall that it takes $2^{c.\beta} = 32$ packets on average to cause $c.\alpha$ to increase by one. As the value of $c.\beta$ increases, the sampling probability $\frac{1}{2^{c.\beta}}$ decreases. Therefore, an active counter performs adaptive sampling.

We make an improvement to active counters as follows: Consider an example with $a = 4$ and $b = 5$. It is easy to see that once the value of $c.\alpha$ reaches 1000, its value will cycle from 1000 to 1111 and then back to 1000. The highest bit is always one. Therefore we do not have to store it. With this implicit bit of one, we can cut a from 4 to 3 without sacrificing the counting capacity. For example, if $c.\alpha = 001$ and $c.\beta = 00101$, the counter value is $(\mathbf{1}001) \times 2^{00101} = 288$, with the bold bit being implicit. One problem is that initially when $c.\alpha = 000$ and $c.\beta = 00000$, the counter value is $(\mathbf{1}000) \times 2^{00000} = 8$ while it should have been 0. Hence, when we determine the counter value, we should remove this initial value. In general, the formula for the value of the improved active counter is

$$c = c.\alpha \times 2^{c.\beta} + 2^{a+c.\beta} - 2^a, \tag{4}$$

where the second term reflects the implicit bit in the coefficient and the third term is the initial offset. With $c.\alpha = 001$ and $c.\beta = 00101$, the actual counter value should be $(\mathbf{1}001) \times 2^{00101} - 8 = 280$.

C. Online Recording Module

Our per-flow traffic measurement function includes an online recording module and an offline estimation module. The online module records the flow size information to the physical active counter array AC_* in real time, while the offline estimation module answers queries on flow sizes based on the data recorded from online module. Below we first describe the operations of the online module.

The measurement is performed periodically. At the beginning of each measurement period, all counters in AC_* are initialized to zeros. For each arrival packet, the router extracts the flow label f from the packet header. It generates a random integer $q \in [0, s)$ to select a counter $AC_f[q]$ from the flow's virtual active counter array, and then increase the counter value by one probabilistically. Recall that AC_f is logical and not actually constructed during online operations. Hence, the actual operation is performed on the corresponding physical counter $AC_*[H(f \oplus R[q])]$, to which $AC_f[q]$ is mapped. More specifically, the router reads counter $AC_*[H(f \oplus R[q])]$, consisting of a coefficient and an exponent. It increases the counter by one with probability $\frac{1}{2^{AC_*[H(f \oplus R[q])].\beta}}$ as described in the previous subsection. The probability for writing the counter back is the same, which decreases exponentially as the exponent increases.

The online recording module only incurs small per-packet overhead, including one hash computation, one random-number generation, and at most two memory accesses: reading a counter and writing it back if the counter value is changed. Generally, when the total number of packets mapped to each counter is large, the sampling probability will be small for most packets. In this case, only a small portion of packets will trigger write-back memory accesses.

D. Offline Estimation Module

At the end of each measurement period, the active counter array AC_* stores the size information of all flows. It is offloaded to an offline server for long-term storage. The server uses AC_* to estimate the sizes of flows under query. We propose an offline virtual active counter estimation architecture (VAC) to estimate the flow sizes.

For an arbitrary flow f under query, we explicitly construct its virtual active counter array AC_f based on (2). Note that offline operations do not have the same overhead requirement as real-time online operations. Let n be the actual flow size of f, and n_v be the number of packets recorded by the virtual active counter array AC_f. Due to counter sharing, we know that n_v is the flow f's size plus the noise introduced by other flows. Let random variable Y be the number of 'noise' packets (from other flows) that are recorded by the s counters in AC_f. We have

$$Y = n_v - n. \tag{5}$$

Let N be the total size of all flows. The number of packets from flows other than f is $N - n$. This noise is spread on all m counters in AC_*. The chance for a noise packet to belong to AC_f is approximately $\frac{s}{m}$, when both s and m are sufficiently large. Hence, Y approximately follows a binomial distribution:

$$Y \sim Binom(N - n, \frac{s}{m}). \tag{6}$$

Hence, the expected number of noise packets in AC_f is $E(Y) = \frac{s(N-n)}{m}$. Applying (5), we have

$$\begin{aligned} E(n_v - n) &= \frac{s(N-n)}{m} \\ n &= \frac{ms}{m-s}(\frac{E(n_v)}{s} - \frac{N}{m}) \end{aligned} \tag{7}$$

We consider the aggregation of all flows as a grant flow whose size is N. We treat AC_* as the active counter array for the grand flow. Hence, we can estimate the value of N by adding the m counter values in AC_*. We denote the estimate as $\hat{N}$. From (4), we have

$$\hat{N} = \sum_{i=0}^{m-1} AC_*[i].\alpha \times 2^{AC_*[i].\beta} + 2^{a+AC_*[i].\beta} - 2^a, \tag{8}$$

where a is the number of bits in the coefficient $AC_*[i].\alpha$. In (7), replacing N with $\hat{N}$ and $E(n_v)$ with the instance value n_v, we have an estimator $\hat{n}$ for the flow size

$$\hat{n} = \frac{ms}{m-s}\Big(\frac{\hat{n}_v}{s} - \frac{\hat{N}}{m}\Big), \tag{9}$$

where the instance value n_v is estimated as $\hat{n}_v$ from the observed counter values in AC_f based on (4), i.e.,

$$\hat{n}_v = \sum_{i=0}^{s-1} AC_f[i].\alpha \times 2^{AC_f[i].\beta} + 2^{a+AC_f[i].\beta} - 2^a. \tag{10}$$

IV. VAC Performance Analysis

In this section, we perform thorough analysis on VAC with regard to its measurement accuracy. We first analyze the active counter array size estimator, which is used to estimate the number of packets recorded in an active counter array. Then we provide relative bias and relative standard error of our VAC approach.

A. Analysis of Active Counter Array Size Estimator

We first provide a theorem on the accuracy of active counter estimation, which can be proved by similar mathematical method in [27], [28]. (we omit the proof due to paper length limitation).

Theorem 1: Let n' be the number of packets that are recorded in an active counter with $(a+b)$ scheme, and $\hat{n}'$ be the estimate given by (4). Then,

$$E(\hat{n}') = n', \qquad StdErr(\tfrac{\hat{n}'}{n'}) = \tfrac{0.82}{\sqrt{2^a}}.$$

Based on this theorem, we now analyze the estimation accuracy of our active counter array size estimator. Suppose that $\mathcal{N}$ packets are recorded in an active counter array $\mathcal{C}$ with s active counters through online recording module. Let the random variable N_i $(0 \le i < s)$ be the number of packets recorded in the ith active counter of $\mathcal{C}$. Since each packet has the same probability $\frac{1}{s}$ to be mapped in any active counter of $\mathcal{C}$, the joint law of $N_0, N_1, \ldots, N_{s-1}$ follows a multinomial distribution with parameters $\mathcal{N}$ and $\mathbf{p}$, where $\mathbf{p} = (\frac{1}{s}, \ldots, \frac{1}{s})$. Its probability mass function is

$$\begin{aligned} &Pr(N_0 = n_0, N_1 = n_1, \ldots, N_{s-1} = n_{s-1}) \\ &= \begin{cases} \binom{\mathcal{N}}{n_0, n_1, \ldots, n_{s-1}} \frac{1}{s^{\mathcal{N}}} & \sum_{i=0}^{s-1} n_i = \mathcal{N}, \\ 0 & \text{otherwise,} \end{cases} \end{aligned} \tag{11}$$

where $n_0, n_1, \ldots, n_{s-1}$ are non-negative integers. The expected value and covariance matrix of N_0 ,. .., N_i, . . ., N_{s-1} are

$$\begin{aligned} &E(N_i) = \tfrac{\mathcal{N}}{s}, \quad Var(N_i) = \tfrac{\mathcal{N}}{s}(1 - \tfrac{1}{s}), \\ &Cov(N_i N_j) = -\tfrac{\mathcal{N}}{s^2},\ i \neq j. \end{aligned} \tag{12}$$

1) Relative Bias of $\hat{\mathcal{N}}$:

We first analyze the expectation of $\hat{\mathcal{N}}$. Since the expectation of $\hat{\mathcal{N}}$ satisfies

$$E(\hat{\mathcal{N}}) = E\big(E(\hat{\mathcal{N}}|N_0, N_1, \ldots, N_{s-1})\big), \tag{13}$$

thereby we have

$$\begin{aligned} E(\hat{\mathcal{N}}) =& \textstyle\sum_{n_0,\ldots,n_{s-1}} E(\hat{\mathcal{N}}|N_0 = n_0, \ldots, N_{s-1} = n_{s-1}) \\ &\cdot Pr(N_0 = n_0, \ldots, N_{s-1} = n_{s-1}) \\ =& \textstyle\sum_{n_0+\cdots+n_{s-1}=\mathcal{N}} \binom{\mathcal{N}}{n_0, \ldots, n_{s-1}} \frac{1}{s^{\mathcal{N}}} E(\hat{\mathcal{N}}|n_0, \ldots, n_{s-1}), \end{aligned} \tag{14}$$

where $E(\hat{\mathcal{N}}|n_0, \ldots, n_{s-1})$ is the abbreviation of $E(\hat{\mathcal{N}}|N_0 = n_0, \ldots, N_{s-1} = n_{s-1})$ for simplicity.

Let $\hat{n}_i$ $(0 \le i < s)$ be the estimate of the number of packets in the ith active counter by (4). Then, $\hat{\mathcal{N}} = \sum_{i=0}^{s-1} \hat{n}_i$. In addition, the estimate for each active counter only depends on the number of packets mapped to this counter. Hence, under the condition of $N_0 = n_0, \ldots, N_{s-1} = n_{s-1}$, the estimations of $\hat{n}_0$, . . ., $\hat{n}_{s-1}$ are independent. Therefore, combining Theorem 1, we have

$$\begin{aligned} &E(\hat{\mathcal{N}}|n_0, \ldots, n_{s-1}) = E(\textstyle\sum_{i=0}^{s-1} \hat{n}_i | n_0, \ldots, n_{s-1}) \\ &= \textstyle\sum_{i=0}^{s-1} E(\hat{n}_i | N_i = n_i) = \sum_{i=0}^{s-1} n_i. \end{aligned} \tag{15}$$

By combining (14) with above formula, we can calculate

$$E(\hat{\mathcal{N}})=\textstyle\sum_{n_0+\cdots+n_{s-1}=\mathcal{N}} \binom{\mathcal{N}}{n_0,\ldots,n_{s-1}} \frac{1}{s^{\mathcal{N}}} \sum_{i=0}^{s-1} n_i = \mathcal{N}. \quad (16)$$

Therefore, we have

$$Bias(\frac{\hat{\mathcal{N}}}{\mathcal{N}}) = \frac{E(\hat{\mathcal{N}})}{\mathcal{N}} - 1 = 0. \quad (17)$$

Hence, the estimator $\hat{\mathcal{N}}$ is unbiased for $\mathcal{N}$.

2) Relative Standard Error of $\hat{\mathcal{N}}$:

Next, we derive the variance and relative standard error of $\hat{\mathcal{N}}$. Similar to formula (14), we have

$$\begin{aligned} E(\hat{\mathcal{N}}^2) &= \textstyle\sum_{n_0,\ldots,n_{s-1}} E(\hat{\mathcal{N}}^2 | N_0 = n_0, \ldots, N_{s-1} = n_{s-1}) \\ &\quad \cdot Pr(N_0 = n_0, \ldots, N_{s-1} = n_{s-1}) \quad (18) \\ &= \textstyle\sum_{n_0+\cdots+n_{s-1}=\mathcal{N}} \binom{\mathcal{N}}{n_0,\ldots,n_{s-1}} \frac{1}{s^{\mathcal{N}}} E(\hat{\mathcal{N}}^2 | n_0, \ldots, n_{s-1}). \end{aligned}$$

Combining Theorem 1 and the estimation independence under the condition of $N_0 = n_0, \ldots, N_{s-1} = n_{s-1}$, we have

$$\begin{aligned} E(\hat{n}_i^2 | N_i = n_i) &= Var(\hat{n}_i | N_i = n_i) + E(\hat{n}_i | N_i = n_i)^2 \\ &= \left(\frac{0.67}{2^a} + 1\right) n_i^2, \ 0 \le i < s, \end{aligned}$$

$$\begin{aligned} E(\hat{n}_i \hat{n}_j | N_i = n_i, N_j = n_j) &= E(\hat{n}_i | N_i = n_i) E(\hat{n}_j | N_j = n_j) \\ &= n_i n_j, \text{ for } i \ne j. \quad (19) \end{aligned}$$

Hence, we can obtain

$$\begin{aligned} E(\hat{\mathcal{N}}^2 | n_0, \ldots, n_{s-1}) &= E\Big(\big(\sum_{i=0}^{s-1} \hat{n}_i\big)^2 | n_0, \ldots, n_{s-1}\Big) \\ = \sum_{i=0}^{s-1} E(\hat{n}_i^2 | N_i = n_i) &+ 2 \sum_{0 \le i < j < s-1} E(n_i n_j | N_i = n_i, N_j = n_j) \\ = \frac{0.67}{2^a} \textstyle\sum_{i=0}^{s-1} n_i^2 &+ \left(\textstyle\sum_{i=0}^{s-1} n_i\right)^2. \quad (20) \end{aligned}$$

By combining (18) with above formula, we have

$$\begin{aligned} E(\hat{\mathcal{N}}^2) &= \textstyle\sum_{n_0+\cdots+n_{s-1}=\mathcal{N}} \binom{\mathcal{N}}{n_0,\ldots,n_{s-1}} \frac{1}{s^{\mathcal{N}}} \Big(\frac{0.67}{2^a} \sum_{i=0}^{s-1} n_i^2 \\ &\quad + \big(\textstyle\sum_{i=0}^{s-1} n_i\big)^2\Big) = \mathcal{N}^2 + \frac{0.67s}{2^a} \left(\frac{\mathcal{N}}{s}(1 - \frac{1}{s}) + \frac{\mathcal{N}^2}{s^2}\right). \quad (21) \end{aligned}$$

Therefore, we have the variance of the estimator $\hat{\mathcal{N}}$:

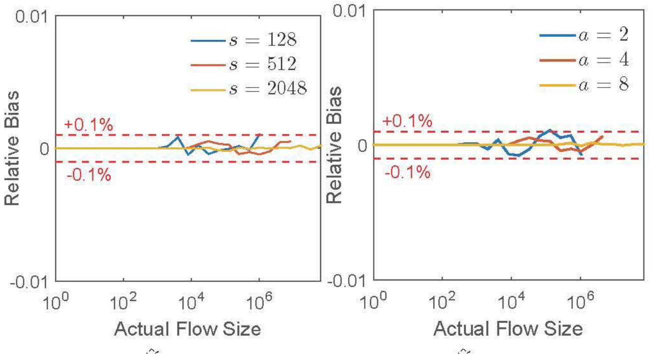
$$Var(\hat{\mathcal{N}}) = E(\hat{\mathcal{N}}^2) - E(\hat{\mathcal{N}})^2 = \frac{0.67}{2^a} \left(\mathcal{N}(1 - \frac{1}{s}) + \frac{\mathcal{N}^2}{s}\right).$$

According to the simulation results, when the total number $\mathcal{N}$ of packets is smaller than $s \times 2^{a-1}$, the sampling probability is 1 for almost all active counters. Thus, our estimator is very accurate in this case. Therefore, we only need to consider the estimation error resulted from small sampling probability when $\mathcal{N}$ is relatively large (i.e., $\mathcal{N} \gg s$). Then we can do the approximation:

$$Var(\hat{\mathcal{N}}) \approx \frac{0.67 \mathcal{N}^2}{s 2^a}.$$

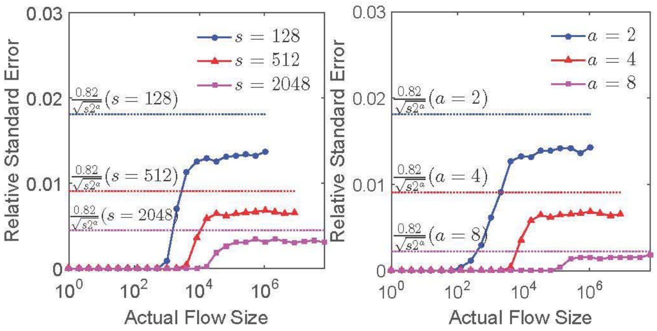
The relative standard error of the estimator $\hat{\mathcal{N}}$ is

$$StdErr(\frac{\hat{\mathcal{N}}}{\mathcal{N}}) = \frac{\sqrt{Var(\hat{\mathcal{N}})}}{\mathcal{N}} \approx \frac{0.82}{\sqrt{s 2^a}}. \quad (22)$$



(a) $Bias(\frac{\hat{\mathcal{N}}}{\mathcal{N}})$ with $a = 4$ (b) $Bias(\frac{\hat{\mathcal{N}}}{\mathcal{N}})$ with $s = 512$

Fig. 2: Relative bias of active counter array size estimator.



(a) $StdErr(\frac{\hat{\mathcal{N}}}{\mathcal{N}})$ with $a = 4$ (b) $StdErr(\frac{\hat{\mathcal{N}}}{\mathcal{N}})$ with $s = 512$

Fig. 3: Relative standard error of active counter array size estimator.

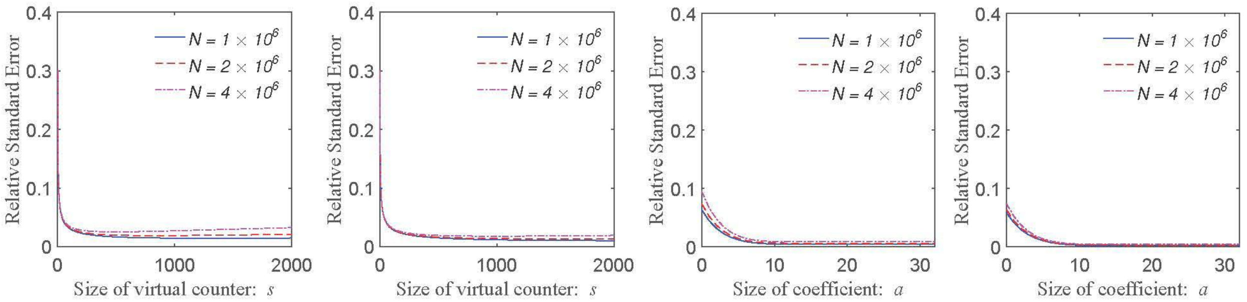
3) Simulation Results: We use some numerical results to illustrate the interplay between the different sources of estimation bias and estimation error. We run 1000 simulations for each case. The simulation results of relative bias are given in Figure 2. Clearly, the relative bias of the estimator is very small ($< 0.1\%$) with respect to different values of s, a and $\mathcal{N}$.

By (22), we know the measurement accuracy (i.e., relative standard error) is affected by two factors: one is the total number s of active counters in $\mathcal{C}$, the other is the size a of coefficient in the active counter. We analyze the impact of s and a in the estimator, and the corresponding simulation results are shown in Figure 3a and Figure 3b, respectively. The reference line $y = \frac{0.82}{\sqrt{s 2^a}}$ of theoretical relative standard error in (22) is also shown for each case.

Figure 3a shows that the estimation accuracy improves as the increase of s. Similar trends can be observed when the coefficient part a grows. With regard to the total packet number $\mathcal{N}$, in each figure, we can find that there is no error for our active counter array size estimator when $\mathcal{N}$ is small. Starting from a certain point, as $\mathcal{N}$ grows, the estimation error also increases. However, when $\mathcal{N}$ becomes large enough, the error reaches it upper bound and becomes stabilized. Clearly, the relative error of all simulation results is lower than the theoretical line, thereby formula (22) can be treated as an upper bound error. Through the theoretical analysis and the simulation results, we can see that our active counter array size estimator can estimate large range flow sizes with high accuracy.

B. VAC Estimation Accuracy

We now analyze the relative bias and relative standard error of VAC. According to (17) and (22) in the analysis of the active



(a) Flow size $n = 1 \times 10^4$ (b) Flow size $n = 2 \times 10^4$ (c) Flow size $n = 1 \times 10^4$ (d) Flow size $n = 2 \times 10^4$

Fig. 4: Relative standard error of VAC with respect to s, a, N and n. Memory size M = 0.25MB.

counter array size estimator, we have the following theorem.

Theorem 2: Let $\mathcal{N}$ be the number of packets that are mapped to an active counter array $\mathcal{C}$ of s active counters. We have

$$E(\hat{\mathcal{N}}) = \mathcal{N} \qquad StdErr(\tfrac{\hat{\mathcal{N}}}{\mathcal{N}}) \approx \tfrac{0.82}{\sqrt{s2^a}}.$$

1) Relative Bias:

Recall that n_v packets are recorded in the virtual active counter array AC_f, and random variable Y indicates the number of 'noise' packets that are mapped to the virtual active counter array AC_f. Combining (5) and Theorem 2, under the condition of $Y = l$, $l \in [0, N-n]$, we have

$$E(\hat{n}_v \,|\, Y = l) \approx n_v = n + l. \tag{23}$$

By (6) and (23),

$$\begin{aligned} E(\hat{n}_v) &= \textstyle\sum_{l=0}^{N-n} E(\hat{n}_v \,|\, Y = l) \cdot Pr(Y = l) \\ &\approx \textstyle\sum_{l=0}^{N-n} (n+l) \cdot \binom{N-n}{l} \cdot (\frac{s}{m})^l (1 - \frac{s}{m})^{N-n-l} \\ &= n + \frac{s(N-n)}{m}. \end{aligned} \tag{24}$$

The value of $\hat{N}$ is estimated based on active counter array AC_*. Hence $E(\hat{N}) = N$. Combining (9) with above, we have

$$\begin{aligned} E(\hat{n}) &= \tfrac{ms}{m-s}\big(\tfrac{E(\hat{n}_v)}{s} - \tfrac{E(\hat{N})}{m}\big) \approx \tfrac{ms}{m-s}\big(\tfrac{n + \frac{s(N-n)}{m}}{s} - \tfrac{N}{m}\big) \\ &= \frac{ms}{m-s}(\frac{n}{s} - \frac{n}{m}) = n. \end{aligned} \tag{25}$$

Therefore,

$$Bias(\tfrac{\hat{n}}{n}) = \tfrac{E(\hat{n})}{n} - 1 \approx 0. \tag{26}$$

Hence, the VAC estimator $\hat{n}$ is approximately unbiased.

2) Relative Standard Error:

Next we derive the relative standard error of $\hat{n}$. By Theorem 2 and (23), we have

$$\begin{aligned} &E\big(\hat{n}_v^2 \,|\, Y = l\big) = Var\big(\hat{n}_v^2 \,|\, Y = l\big) + E\big(\hat{n}_v \,|\, Y = l\big)^2 \\ &\approx \tfrac{0.67(n+l)^2}{s2^a} + (n+l)^2 = \big(\tfrac{0.67}{s2^a} + 1\big)(n+l)^2, \end{aligned} \tag{27}$$

$$Var(\hat{N}) \approx \tfrac{0.67N^2}{m2^a}. \tag{28}$$

Combining (6) with the above equations, we have

$$\begin{aligned} E\big(\hat{n}_v^2\big) &= \textstyle\sum_{l=0}^{N-n} E\big(\hat{n}_v^2 | Y = l\big) Pr(Y = l) \qquad (29) \\ &\approx \textstyle\sum_{l=0}^{N-n} \big(\frac{0.67}{s2^a} + 1\big)(n+l)^2 Pr(Y = l) \\ &= \big(\tfrac{0.67}{s2^a} + 1\big)\big((n + \tfrac{s(N-n)}{m})^2 + \tfrac{s(N-n)}{m}(1 - \tfrac{s}{m})\big). \end{aligned}$$

From (9) (24) (28) and (29), we have the variance of the estimator $\hat{n}$:

$$\begin{aligned} Var(\hat{n}) &= \big(\tfrac{ms}{m-s}\big)^2 \Big(\tfrac{Var(\hat{n}_v)}{s^2} + \tfrac{Var(\hat{N})}{m^2}\Big) \\ &\approx \big(\tfrac{m}{m-s}\big)^2 \Big(\tfrac{0.67}{s2^a}\Big(n + \tfrac{s(N-n)}{m}\Big)^2 \\ &+ \big(\tfrac{0.67}{s2^a} + 1\big)\tfrac{s(N-n)}{m}(1 - \tfrac{s}{m}) + \big(\tfrac{s}{m}\big)^2 \tfrac{0.67N^2}{m2^a}\Big). \end{aligned}$$

The relative standard error of the estimator $\hat{n}$ is

$$\begin{aligned} StdErr(\tfrac{\hat{n}}{n}) &= \tfrac{\sqrt{Var(\hat{n})}}{n} \approx \tfrac{m}{n(m-s)} \cdot \\ &\sqrt{\tfrac{0.67}{s2^a}\big(n + \tfrac{s(N-n)}{m}\big)^2 + \big(\tfrac{0.67}{s2^a} + 1\big)\tfrac{s(N-n)}{m}(1 - \tfrac{s}{m}) + \big(\tfrac{s}{m}\big)^2 \tfrac{0.67N^2}{m2^a}}. \end{aligned} \tag{30}$$

3) Numerical Results of $StdErr(\frac{\hat{n}}{n})$: The relative standard error (or error in short) $StdErr(\frac{\hat{n}}{n})$ is affected by several factors including N, n, s and a. We use some numerical results to illustrate the interplay between the different sources of estimation error.

Suppose the allocated memory is $M = 0.25$MB and a particular flow size of $n = 10^4$. The active counters utilize the (4 + 4) scheme ($a = 4$, $b = 4$) by default. Hence, the default number of active counters is $m = \frac{M}{a+b} = 2^{18}$. We first analyze the impacts of the number of counters in each virtual active counter array s and the total size of all flows N, and the corresponding numerical results of relative standard error in (30) with respect to s and N for different curves are shown in Figure 4a and Figure 4b. Clearly, when N increases, the relative standard error increases too. At the beginning, the relative standard error drops quickly as s grows. When s becomes further larger, the estimation accuracy improves slowly as the error caused by noise increases. Combining these two observations, we find that when s is relatively large, the error for VAC is very small.

Figure 4c and Figure 4d show the numerical results of the relative standard error in (30) with respect to a (the number of bits in the coefficient part of the active counters). Clearly, the relative standard error drops quickly as a grows. In addition, when a becomes further larger (e.g., 10), the error becomes very small and stabilized, which can also be predicted by Theorem 2 with its factor of improvement being $\frac{1}{\sqrt{2^a}}$.

From Figure 4a to Figure 4b, we observe that the relative standard error decreases as the flow size n of the target flow grows we increase. Clearly, the relative standard error is smaller for flows of larger sizes. The same trends are observed from Figure 4c to Figure 4d.

V. Experimental Evaluation

In this section, we compare the performance of our VAC scheme and the most related work CT [18] through extensive experiments using real network traces captured by the main gateway of our university. The performance metrics in Section II are employed in our experimental evaluation, including memory overhead, processing time, and estimation accuracy.

A. Experiment Setup

Without loss of generality, we consider TCP flows and measure the size for each flow. The network trace used in our experiments contains $126,569,701$ packets generated by $11,453,043$ flows. Hence, the average flow size is 11.05 packets per flow.

We conduct two sets of experiments to evaluate the performance of our scheme. In our first experiment set, we evaluate the processing time and measurement accuracy of CT and VAC with regard to the memory size of the physical active counter array. For CT, we use the same settings as Chen et al. [18] did in their experiments: degree with value 3, virtual counter size with value 100, and node counter size fixed to 4 bits. For VAC, the active counters utilize (4+4) scheme, and we fix the virtual active counter array size s to 512. The available memory size M for CT and VAC is varied from 0.25MB, 0.5MB, 1MB to 2MB. In our second experiment set, we evaluate the impact of the number of counters in each virtual active counter array s and the number of bits in the coefficient part of the active counters a on the estimation accuracy of VAC. We fix the memory size M to 1MB, and vary the values of s and a to observe its accuracy.

B. Processing Time

We first compare the performance of CT and VAC in terms of processing time in our first experiment set. Generally, the processing overhead is measured by the average number of memory accesses and the number of hash value computations [17], [18]. The average results of encoding a packet are shown in Table I. CT requires more than 2 memory accesses per packet and 1 hash computation in processing time. As we mentioned in Section III-C, VAC requires 1 memory access to read the value in one active counter, and only needs to write it back with corresponding adaptive sampling probability, which becomes smaller as the total number of packets mapping to this active counter grows. Therefore, VAC only needs slightly more than 1 memory access per packet on average, which is consistent to the results in Table I. Moreover, as the available memory space decreases, the average number of memory accesses of VAC also decreases. This is because each active counter is shared by more flows, which incurs higher sampling probability and thereby reduces the updating frequency. Clearly, VAC can achieves faster processing speed than CT.

TABLE I: Comparison of processing time for encoding a packet by CT and VAC.

memory size (MB)	number of memory accesses		number of hash computations	
	CT	VAC	CT	VAC
0.25	2.13	1.11	1	1
0.5	2.13	1.19	1	1
1	2.12	1.31	1	1
2	2.11	1.48	1	1

C. Estimation Accuracy w.r.t Memory Overhead

In the first set of experiments, we also evaluate the estimation accuracy of CT and the proposed scheme VAC with regards to the memory overhead M. The experiment results are presented in Figure 5-6, where the allocated memory in each scheme is 0.25MB in the first plot, 0.5MB in the second plot, 1MB in the third plot, and 2MB in the fourth plot. Since the total number of TCP flows is above 10 million, average memory per flow is about 0.2 bit in the first plot, 0.4 bit in the second plot, 0.8 bit in the third plot, and 1.6 bits in the fourth plot.

The experiment results of CT are shown in the four plots of Figure 5, where each point in each plot represents a particular flow, with the x coordinate being the actual flow size n and the y coordinate being the estimated flow size $\hat{n}$. The equality line $y = x$ is also shown in each plot for reference. Clearly, the closer a point is to the equality line, the better the measurement is. From Figure 5, we observe that CT cannot produce accurate result when the allocated memory is tight (i.e., 0.2 bit per flow). When we increase the memory size, CT starts to work as the points become more clustered towards the equality line. However, the estimate results of CT are still not accurate enough. The four plots in Figure 6 show the experiment results of VAC. We observe that VAC can produce accurate estimates for all flows even under tight memory as shown in the first plot. As expected, VAC becomes more accurate when more memory space is available. We also discover that although the relative standard errors for small flows can be large, the absolute errors of these small flows remain relatively small. Hence, VAC can still be useful for small flows.

The four plots in Figure 7 show the relative bias in the estimations for CT and VAC under different memory availability. In each plot, we can see that the relative bias decreases rapidly toward 0 as the actual flow size increases. We also observe that VAC has smaller relative bias than CT. The similar observation is made in Figure 8, where the performance metrics is the relative standard error. Therefore, VAC can yield more accurate per-flow traffic measurement than CT.

D. Impact of Virtual Active Counter Array Size s

In the second set of experiments, we first study the impact of the virtual active counter array size s on the estimation accuracy of VAC, while the available memory size M is fixed to 1MB. We repeat the experiments in Figure 6 with memory overhead M = 1MB, and vary the value of s from 512 to 128, 256 and 2048, respectively. The corresponding estimation results are presented in the first three plots of Figure 9, while the fourth plot shows the relative standard errors with different values of s. We observe that, as s increases, the estimation accuracy increases for large flows but decreases for small flows. The above observations are consistent with our analysis and numerical results in Section IV-B3. Hence, it is practical to choose a virtual active counter array size s of 512.

E. Impact of Active Counter Coefficient Size a

Finally, we study the impact of the coefficient size a in active counters on the estimation accuracy of VAC while fixing the available memory size M to 1MB and the value of s to 512. We repeat the experiments in Figure 6 with memory overhead M =

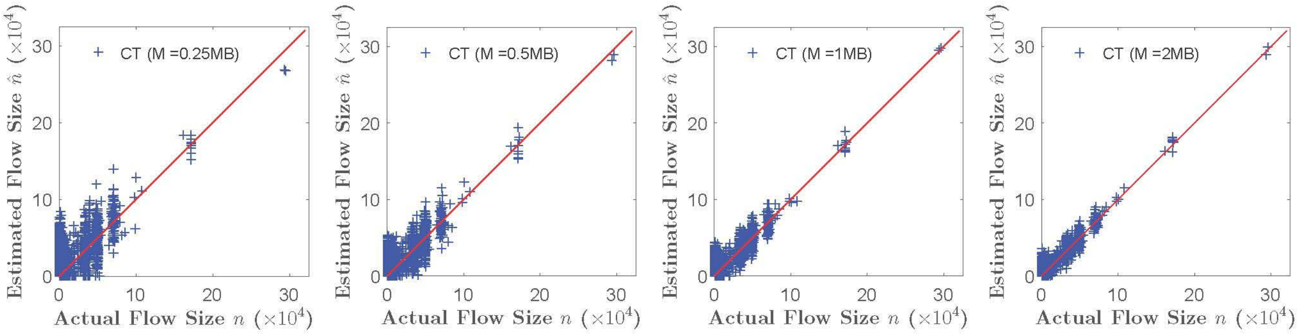


Fig. 5: Estimation results by CT when memory size M = 0.25MB, 0.5MB, 1MB, and 2MB.

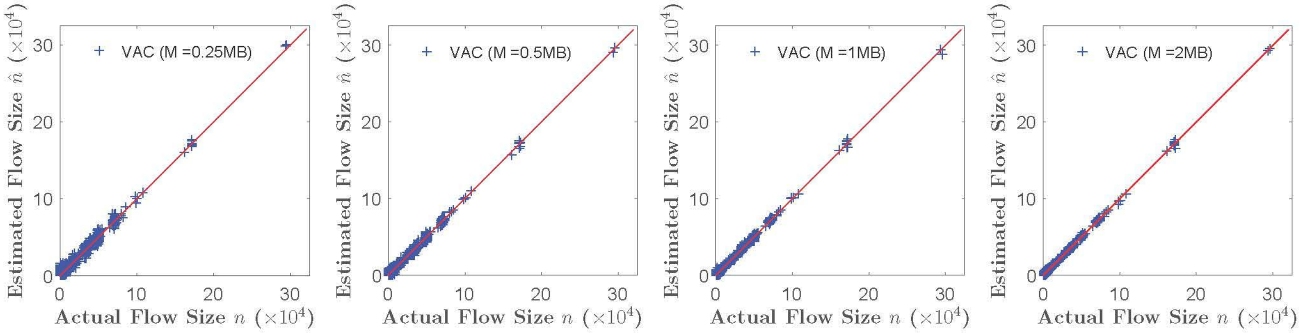


Fig. 6: Estimation results by VAC with $s = 512$ when memory size M = 0.25MB, 0.5MB, 1MB, and 2MB.

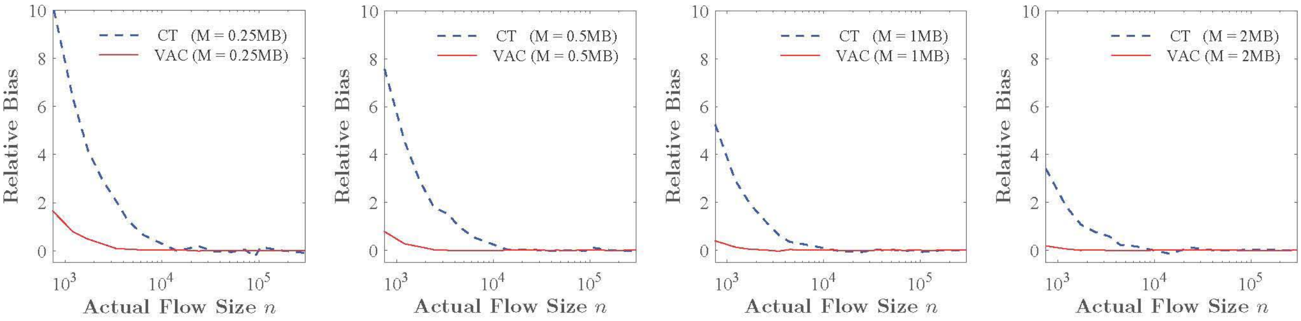


Fig. 7: Relative bias $Bias(\frac{\hat{n}}{n})$ when memory size M = 0.25MB, 0.5MB, 1MB, and 2MB.

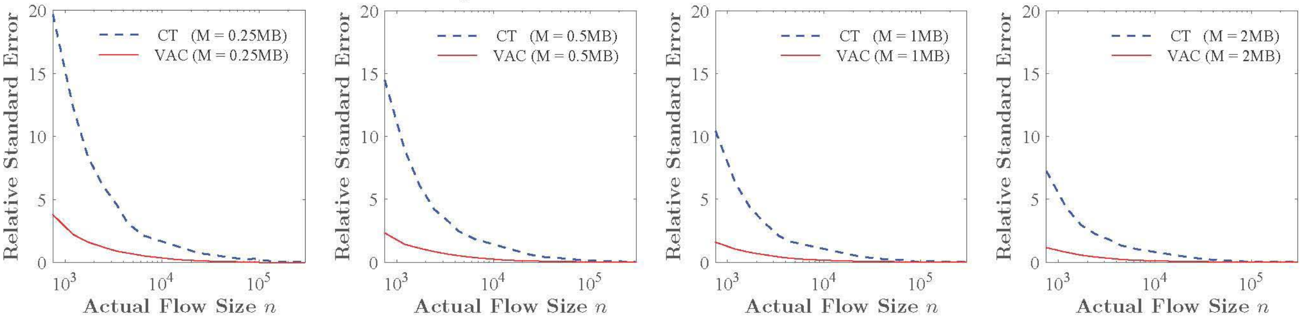


Fig. 8: Relative standard error $StdErr(\frac{\hat{n}}{n})$ when memory size M = 0.25MB, 0.5MB, 1MB, and 2MB.

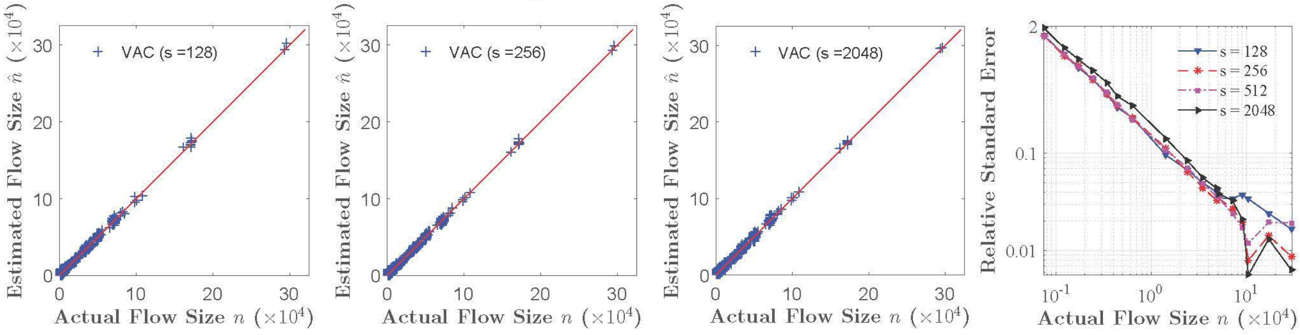


Fig. 9: Estimation results and relative standard errors of VAC with different values of s.

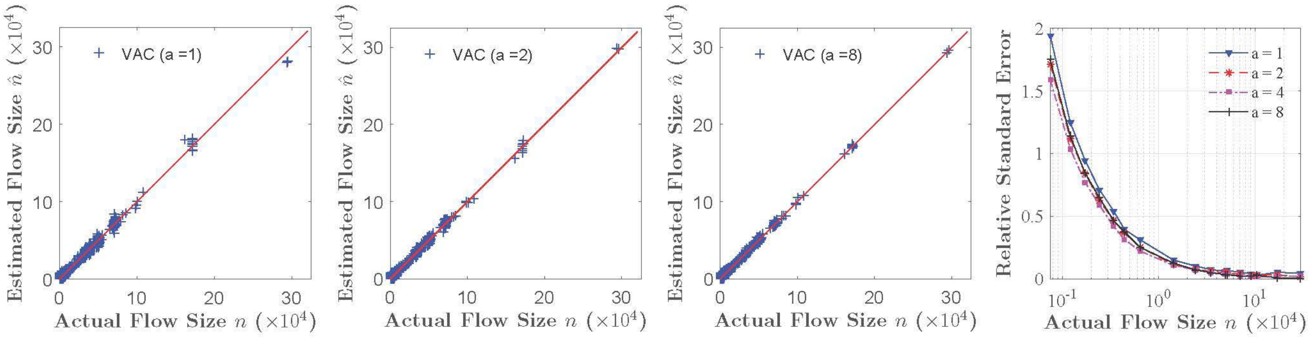


Fig. 10: Estimation results and relative standard errors of VAC with different values of a.

1MB, and vary the value of a from 4 to 1, 2 and 8, respectively. The corresponding estimation results are presented in the first three plots of Figure 10, while the fourth plot shows the relative standard errors with different values of a. We observe that when a is relatively small at 1, the estimation accuracy is worse when comparing with $a = 2$ or $a = 4$. This observation is consistent with our analysis in Section IV-B. However, when a becomes large enough ($a = 8$), the relative standard errors are larger than when $a = 4$. This is because the total number m of active counters drops as a increases, which incurs more noise. Therefore, it is practical to choose a coefficient size a of 4.

VI. Conclusion

This paper proposes a highly compact and efficient counter architecture VAC for per-flow traffic measurement through active counter sharing. VAC achieves faster processing speed (slightly more than 1 memory access per packet) and provides more accurate measurement results on per-flow size estimation than the existing work. Moreover, VAC can yield good estimates even under a very tight memory space (less than 1 bit per flow or even one fifth of a bit per flow). Extensive experiments with real network trace data as well as rigorous theoretical analysis demonstrate the superior performance of VAC.

VII. Acknowledgment

This work is supported in part by the National Science Foundation under grants STC-1562485 and CNS-1719222, and a grant from Florida Center for Cybersecurity.

References

[1] C. Estan, G. Varghese, and M. Fish, "Bitmap Algorithms for Counting Active Flows on High-Speed Links," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 925–937, 2006.

[2] S. Heule, M. Nunkesser, and A. Hall, "HyperLogLog in Practice: Algorithmic Engineering of a State of the Art Cardinality Estimation Algorithm," *Proc. of ACM EDBT*, pp. 683–692, 2013.

[3] Y. Zhou, S. Chen, Z. Mo, and Q. Xiao, "Point-to-Point Traffic Volume Measurement through Variable-Length Bit Array Masking in Vehicular Cyber-Physical Systems," *Proc. of IEEE ICDCS*, pp. 51–60, 2015.

[4] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "SCREAM: Sketch Resource Allocation for Software-defined Measurement," *Proc. of ACM CoNEXT*, 2015.

[5] Y. Zhou, Z. Mo, Q. Xiao, S. Chen, and Y. Yin, "Privacy-Preserving Transportation Traffic Measurement in Intelligent Cyber-physical Road Systems," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 5, pp. 3749–3759, 2016.

[6] R. Ben-Basat, G. Einziger, R. Friedman, and Y. Kassner, "Heavy Hitters in Streams and Sliding Windows," *Proc. IEEE INFOCOM*, 2016.

[7] Y. Zhou, Y. Zhou, S. Chen, and O. P. K, "Limiting Self-Propagating Malware Based on Connection Failure Behavior through Hyper-Compact Estimators," *arXiv preprint arXiv:1602.03153*, 2016.

[8] Q. Xiao, S. Chen, Y. Zhou, M. Chen, J. Luo, T. Li, and Y. Ling, "Cardinality Estimation for Elephant Flows: A Compact Solution Based on Virtual Register Sharing," *IEEE/ACM Transactions on Networking*, 2017.

[9] S. Chen, Y. Zhou, and S. Chen, "Efficient Hierarchical Traffic Measurement in Software-Defined Datacenter Networks," *Proc. of IEEE International Conference on Cloud Computing*, pp. 163–170, 2017.

[10] Q. Xiao, Y. Zhou, and S. Chen, "Better with Fewer Bits: Improving the Performance of Cardinality Estimation of Large Data Streams," *Proc. of IEEE INFOCOM*, pp. 1–9, 2017.

[11] D. Shah, S. Iyer, B. Prabhakar, and N. McKeown, "Analysis of a Statistics Counter Architecture," in *Hot Interconnects 9, 2001.* IEEE, 2001, pp. 107–111.

[12] S. Ramabhadran and G. Varghese, "Efficient Implementation of a Statistics Counter Architecture," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, 2003, pp. 261–271.

[13] Q. Zhao, J. Xu, and Z. Liu, "Design of a Novel Statistics Counter Architecture with Optimal Space and Time Efficiency," *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 1, pp. 323–334, 2006.

[14] A. Kumar, J. Xu, and J. Wang, "Space-Code Bloom Filter for Efficient Per-Flow Traffic Measurement," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2327–2339, 2006.

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

[16] Y. Lu and B. Prabhakar, "Robust Counting Via Counter Braids: An Error-Resilient Network Measurement Architecture," *Proc. of IEEE INFOCOM*, April 2009.

[17] T. Li, S. Chen, and Y. Ling, "Per-flow Traffic Measurement through Randomized Counter Sharing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1622–1634, 2012.

[18] M. Chen and S. Chen, "Counter Tree: A Scalable Counter Architecture for Per-Flow Traffic Measurement," *Proc. of IEEE ICNP*, November 2015.

[19] Y. Zhou, Y. Zhou, S. Chen, and Y. Zhang, "Per-flow Counting for Big Network Data Stream over Sliding Windows," *Proc. of IEEE/ACM International Symposium on Quality of Service*, pp. 1–10, 2017.

[20] G. Cormode and S. Muthukrishnan, "An Improved Data Stream Summary: the Count-Min Sketch and Its Applications," *Proc. of LATIN*, 2004.

[21] C. Estan and G. Varghese, "New Directions in Traffic Measurement and Accounting," *Proc. of ACM SIGCOMM*, August 2002.

[22] T. Mori, M. Uchida, R. Kawahara, J. Pan, and S. Goto, "Identifying Elephant Flows through Periodically Sampled Packets," *Proc. of ACM IMC*, pp. 115–120, 2004.

[23] N. Kamiyama and T. Mori, "Simple and Accurate Identification of High-rate Flows by Packet Sampling," *Proc. of IEEE INFOCOM*, April 2006.

[24] Y. Yao, S. Xiong, J. Liao, M. Berry, H. Qi, and Q. Cao, "Identifying Frequent Flows in Large Datasets through Probabilistic Bloom Filters," *Proc. IEEE IWQOS*, pp. 279–288, 2015.

[25] Y. Zhou, Y. Zhou, M. Chen, Q. Xiao, and S. Chen, "Highly Compact Virtual Counters for Per-Flow Traffic Measurement through Register Sharing," *Proc. of IEEE Globecom*, 2016.

[26] Y. Zhou, Y. Zhou, M. Chen, and S. Chen, "Persistent Spread Measurement for Big Network Data Based on Register Intersection," *Proc. of ACM SIGMETRICS*, 2017.

[27] P. Flajolet and L. Chesnay, "On Adaptive Sampling," *Computing*, vol. 43, no. 4, pp. 391–400, 1990.

[28] R. Stanojevic, "Small Active Counters," *Proc. of IEEE INFOCOM*, pp. 2153–2161, 2007.