*This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2008 proceedings.*

# Accurate and Efficient Traffic Monitoring Using Adaptive Non-linear Sampling Method

Chengchen Hu, Sheng Wang, Jia Tian, Bin Liu
Tsinghua University
Beijing, China, 100084
{hucc03,wangs04,tianj04}@mails.tsinghua.edu.cn
liub@tsinghua.edu.cn

Yu Cheng
Illinois Institute of Technology
Chicago, IL USA, 60616
cheng@iit.edu

Yan Chen
Northwestern University
Evanston, IL USA, 60208
ychen@northwestern.edu

***Abstract*—Sampling technology has been widely deployed in measurement systems to control memory consumption and processing overhead. However, most of the existing sampling methods suffer from large estimation errors in analyzing small-size flows. To address the problem, we propose a novel adaptive non-linear sampling (ANLS) method for passive measurement. Instead of statically configuring the sampling rate, ANLS dynamically adjusts the sampling rate for a flow depending on the number of packets having been counted. We provide the generic principles guiding the selection of sampling function for sampling rate adjustment. Moreover, we derive the unbiased flow size estimation, the bound of the relative error, and the bound of required counter size for ANLS. The performance of ANLS is thoroughly studied through theoretic analysis and experiments under synthetic/real network data traces, with comparison to several related sampling methods. The results demonstrate that the proposed ANLS can significantly improve the estimation accuracy, particularly for small-size flows, while maintain a memory and processing overhead comparable to existing methods.**

## I. Introduction

The Internet has been evolving into a common communication infrastructure supporting a variety of applications, which at the same time requires dedicated network management to necessary quality of service provision. *Passive traffic measurement* is very important to network management, which can provide various network status information including traffic matrix, packet length distributions, traffic volumes, session durations, etc., to be exploited for charging, engineering, managing, and securing the communication networks [1] [2]. With the continuous increasing of line speed and number of flows, per-flow passive measurement has become a challenging task due to the demanding requirements on both memory size and memory bandwidth. Off-the-shelf memory is either high speed or high capacity. Large capacity DRAM can hold more flow records but its low speed limits the sampling rate. Fast SRAM supports high speed sampling but is susceptible to overflow due to limited memory capacity. Thus, it is necessary to develop an efficient sampling method for compromising the above contradiction [3] [4] [5] [6]. There are two generic sampling approaches for passive measurement: *packet sampling* and *flow sampling*. The former samples each packet independently with a certain probability, while the latter samples packets at the granularity of flows (i.e., packets in different flows are sampled with different sampling rates). The passive measurement system/infrastructure typically consists of three components. A *monitoring component* tapped into the network link uses a sampling strategy to select packets and forwards them to a reporting component. The *reporting component* aggregates the packet information into flow records and exports them to a *remote data center and analysis system component*. The data center is equipped with high-density data storage, which makes the measurement results available to the analysis system for different applications. In this paper, we focus on the sampling strategy for the monitoring component and study how to design an efficient sampling scheme that enables precise estimations with a reasonable cost.

An efficient sampling method is expected to be applicable to different types of applications, where different sizes of flows may be of importance. For example, flow-level *usage accounting* is essential for management applications [4] [5] [7], e.g., usage-based charging/pricing, network planning, and traffic engineering. For usage accounting, the main target is to catch the *elephant flows* (i.e., the flows of large size). For *network security* applications, the flow-level traffic patterns often help reveal anomalies [8] [9] [10]. A typical scenario, a sharp increase of 40 bytes TCP flows with only one packet is probably caused by SYN flooding attacks or flash crowds. Unlike the usage accounting, network security applications require accurate estimation on *mice flows* (i.e., the flows of small size). It is the diverse application requirements that motivate us to develop a new sampling method, which should bound the estimation error for both small and large flows.

The existing static sampling (SS) methods (as adopted by [3] [11] [12] [13]) selects packets with the same sampling rate/probability $p$. It can be proved that the unbiased estimation value of the flow size $n$ is $c/p$ and the relative error of this estimation is $\sqrt{(1/p-1)/n}$, where $p$ is the static sampling rate, $c$ is the counter value for a sampled flow and $n$ is the flow size in terms of number of packets (See the proof in Appendix). From the results, we can see that the major problem of

This work is partly supported by NSFC (60573121, 60625201), China 973 program(2007CB310702),the Cultivation Fund of the Key Scientific and Technical Innovation Project, MoE, China (705003), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20060003058), Tsinghua Basic Research Foundation(JCpy2005054).

978-1-4244-2026-1/08/$25.00 

employing the static sampling method is its intolerably high relative error for small flows. For instance, the relative error will be 300% with $p = 0.1$ and $n = 1$. Using a larger $p$ can mitigate the relative error but lead to higher memory consumption, which conflicts with the purpose of sampling.

In this paper, we propose an adaptive non-linear sampling (ANLS) method for passive measurement. Instead of statically configuring the sampling rate, ANLS dynamically tunes the sampling rate for a flow depending on the number of packets having been samples, which is maintained by a *counter*. The intuition of ANLS is to use a large sampling rate for small flows and a small sampling rate for large flows. Specifically, this paper contributes in the following three aspects:

1) We provide the general principles guiding the selection of sampling function for sampling rate adjustment. The sampling rate is adjusted according to the counter value. There is no need to predict or estimate the flow size distribution.
2) We derive the unbiased flow size estimation, the bound of the relative error, and the bound of the required counter size for ANLS.
3) The performance of ANLS is thoroughly investigated through theoretic analysis and experiments under synthetic/real network data traces, with comparison to several related sampling methods. The results demonstrate that the proposed ANLS significantly improve the estimation accuracy, particularly for small-size flows, while maintaining a memory and processing overhead comparable to those of existing methods. Furthermore, flow size distribution has almost no impact on the estimation accuracy.

The rest of the paper is organized as follows. Section II reviews the related work. Section III presents the proposed ANLS method. Section IV demonstrates the properties of ANLS. Section V evaluates the performance of ANLS. Section VI gives the concluding remarks.

## II. Related work

A pioneering work on statistical traffic sampling was published in [3], which uses static sampling to estimate the packet size distribution in a backbone network. The primary flow-level measurement tool used by network operators nowadays is NetFlow [14], which resorts to packet sampling (known as sampled NetFlow [13]), to handle the large traffic volume and diversity in high speed links. Considering the multi-hop feature of most flows, the work [11] [12] deployed the sampling system in a distributed manner for the purpose of passive measurement. The "sample and hold" method was introduced in [7], which uses a small and fast memory to process every packet in a real-time manner. This method is used to capture large flows but not for small flows. CATE was proposed in [15] which estimates the proportion of each flow by making multiple comparisons for each arrival and counting the number of coincidences. This method is accurate for media-size and large-size flows but is not accurate for small-size flows.

In the context of adaptive sampling, several mechanisms are introduced for different purposes. Better NetFlow (BNF) was proposed in [6] to improve memory utilization by an adaptive linear sampling method. A relatively large sampling rate is configured at the beginning of the measurement interval and will adaptively decrease when possible memory overflow is detected. A size-dependent sampling (SDS) mechanism was presented in [5]. A flow whose size is larger than $z$ is always selected, while the flow with size $x < z$ is sampled with probability $x/z$. The authors in [16] provided an important theorem specifying the minimum number of packet samples required to be sampled to guarantee the expected relative error, and they also proposed an adaptive random sampling (ARS) method. However, to utilize their theorem, it is required to first estimate the total packet amount using a linear auto-regressive (AR) prediction model. The accuracy and the implementation complexity of ARS are greatly restricted by the determination of the AR model parameters. All the above methods optimize on either the memory size or accuracy for medium to large flows, while the relative error in estimating small flows is considerably large.

Many previous works estimated the original flow size distribution from sampled flow statistics [4] [17] [18], or using a data stream algorithm with "lossy data structure" [19]. The flow size distribution is one of the most fundamental statistics from which we can deduce many other statistics, such as the total number of flows and the average flow size. However, the flow size distribution can not indicate flow-specific properties, e.g., accurate size estimation for a particular flow or a subpopulation, which is to be addressed in this paper.

TABLE I
Table of notations

| Notations | Descriptions |
|---|---|
| $c$ | the counter value |
| $c_t$ | the counter value in time $t$ |
| $p$ | the static sampling rate |
| $P(c)$ | the sampling rate when the counter value is $c$, using ANLS |
| $f(c)$ | the sampling function which is used to calculate $P(c)$ |
| $b$ | the parameter of ANLS defined in $f(c)$ |
| $n$ | the actual flow size |
| $\hat{n}$ | the estimation of flow size |
| $Q_i(n)$ | the probability that $c = i$ when actual flow size is $n$ |
| $k$ | the parameter of CATE method |
| $p_f$ | the final sampling rate of BNF |
| $u$ | the parameter of the sampling function defined in Section IV |

## III. Adaptive non-linear sampling method

For convenience, we summarize the main notations used in this paper in Table I, where the counter value and flow size are in terms of number of packets. With static sampling method of rate $p$, the counter value $c_t$ will be refreshed upon an packet arrival after time interval $t'$, according to the following expression

$$c_{t+t'} = \begin{cases} c_t + 1 & \text{with probability } p; \\ c_t & \text{with probability } 1-p. \end{cases} \quad (1)$$

The ANLS is proposed to replace the static sampling rate $p$ in (1) with a function $P(c)$ over the counter value $c$. It is expected that $P(c)$ diminishes with the increasing of $c$. Specifically, $P(c)$

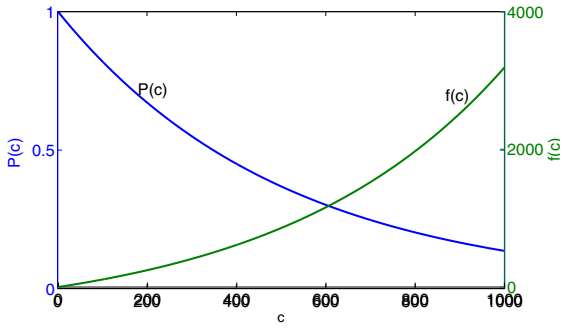


Fig. 1. An example of $f(c)$ and $P(c)$.

for the proposed ANLS is calculated as

$$P(c) = 1/[f(c+1) - f(c)], \tag{2}$$

where $f(c)$ is a sampling function to be selected according to the following general principles.

*Definition 1:* Sampling function $f(c), c \geq 0$, is defined as a function satisfying the following conditions:

1) a real increasing convex function;
2) $f(0) = 0$ and $f(1) = 1$;
3) $f(c) < f(c+1) \leq bf(c) + 1$ with $b > 1$ and $c > 0$.

Now, given a pre-defined $f(c), c \geq 0$, we could adaptively tune the sampling rate depending on the counter value. With the convexity, it is not difficult to check that $c \uparrow\rightarrow [f(c+1) - f(c)] \uparrow\rightarrow P(c) \downarrow$. Namely, the sampling rate decreases as the counter value (as well as the flow size) increases. A nice feature of ANLS compared to existing work is that the sampling rate is adjusted according to the counter value and there is no need to predict or estimate the flow size distribution. In Fig. 1, we illustrate an example of $f(c)$ and $P(c)$.

## IV. Properties of ANLS

In this section, we theoretically investigate the properties of adaptive non-linear sampling from the perspectives of accuracy, memory consumption and processing overhead, with sampling function selected according to *Definition 1*.

### A. Accuracy

The accuracy in estimating the flow size can be examined through two aspects: unbiased estimation and bounded relative error.

*1) Unbiased estimation:*

*Theorem 1:* Under the ANLS method, $\hat{n}(c) = f(c)$ is an unbiased estimation of the flow size $n$.

*Proof:* Let $Q_i(n)$ denote the probability that counter value $c$ equals $i$ when current actual flow size is $n$. We have,

$$Q_i(n) = \prod_{j=0}^{i-1} P(j) \sum_{\alpha_0+\ldots+\alpha_i=n-i} (1-P(0))^{\alpha_0} \ldots (1-P(i))^{\alpha_i} \tag{3}$$

$$Q_i(n) = Q_{i-1}(n-1)P(i-1) + Q_i(n-1)(1-P(i)). \tag{4}$$

where $\alpha_j, j = 0, \cdots, i-1$ represents the number of unsampled packets between the $j$th and the $(j+1)$th sampled packets, and $\alpha_i$ represents the number of unsampled packets after the $i$th sample until the end of the flow. Moreover, it is not difficult to see that $Q_1(1) = 1$ and $Q_0(n) = Q_n(n-1) = 0$.

Let $F(n)$ denote the expectation of $\hat{n}(c)$ when the actual flow size is $n$, we have

$$F(n) = E[\hat{n}(c)] = \sum_{i=0}^{n} f(i)Q_i(n) \tag{5}$$

Based on (4) and (5), we can further have

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

According to (2), $f(i+1) - f(i) = 1/P(i)$. Thus,

$$F(n) - F(n-1) = \sum_{i=1}^{n-1} Q_i(n-1) = 1. \tag{6}$$

$$F(n) = \sum_{i=1}^{n} [F(i) - F(i-1)] + F(0) = n. \tag{7}$$

That is,

$$E[\hat{n}(c)] = E[f(c)] = F(n) = n. \tag{8}$$

which represents an unbiased estimation of the flow size. ■

*2) Bounded relative error:*

*Theorem 2:* Using $\hat{n}(c) = f(c)$ as the unbiased estimation, the relative error is upbounded by $\sqrt{\frac{b-1}{2} - \frac{b-1}{2n}}$.

*Proof:* Let $H(n)$ denote the expectation of $f^2(c)$ when the flow size is $n$. We have

$$H(n) = E[f^2(c)] = \sum_{i=0}^{n} f^2(i)Q_i(n). \tag{9}$$

Thus, from (4) and (9), we get,

$$\begin{aligned}
&H(n) - H(n-1) \qquad (10)\\
&= \sum_{i=1}^{n} (f^2(i))[Q_{i-1}(n-1)P(i-1) + Q_i(n-1)(1-P(i))] \\
&\quad - \sum_{i=1}^{n-1} (f^2(i))Q_i(n-1)[P(i) + (1-P(i))] \\
&= \sum_{i=2}^{n} (f^2(i))Q_{i-1}(n-1)P(i-1) - \sum_{i=1}^{n-1} (f^2(i))Q_i(n-1)P(i) \\
&= \sum_{i=1}^{n-1} (f^2(i+1))Q_i(n-1)P(i) - \sum_{i=1}^{n-1} (f^2(i))Q_i(n-1)P(i).
\end{aligned}$$

Since $f(i+1) - f(i) = 1/P(i)$ and $Q_0(n-1) = 0$, we have

$$H(n) - H(n-1) = \sum_{i=1}^{n-1} Q_i(n-1)[f(i+1) + f(i)]. \quad (11)$$

Applying *Definition 1* and *Theorem 1*, we obtain,

$$\begin{aligned} H(n) - H(n-1) &\leq (1+b)\sum_{i=1}^{n-1} Q_i(n-1)f(i) + 1 \\ &= (1+b)F(n-1) + 1 \\ &= (1+b)(n-1) + 1 \end{aligned}$$

and therefore

$$\begin{aligned} H(n) &= \sum_{i=1}^{n} [H(i) - H(i-1)] + H(0) \\ &\leq \frac{(b+1)n^2 - (b-1)n}{2}. \end{aligned} \quad (12)$$

The variation of adaptive non-linear sampling is then computed as,

$$Var[\hat{n}(c)] = H(n) - F^2(n) \leq \sqrt{\frac{(b-1)}{2} - \frac{(b-1)}{2n}}, \quad (13)$$

and the relative error of ANLS can be upbounded by,

$$\frac{\sqrt{Var[\hat{n}(c)]}}{n} \leq \sqrt{\frac{(b-1)}{2} - \frac{(b-1)}{2n}}. \quad (14)$$

■

*Theorem 2* tells that the relative error is zero when $n$ is one. The relative error increases with the increment of $n$, but converges to $\sqrt{(b-1)/2}$ when $n \to \infty$. The relative error decreases as $b$ diminishes, while $b$ should be larger than one as described in *Definition 1*.

To give an intuitive illustration, we select one specific sampling function according to *Definition 1* as

$$f(c) = [(1+u)^c - 1]/u, 0 < u < 1, \quad (15)$$

where $u$ is a constant parameter. It can be easily proved that (15) satisfies *Definition 1* by setting $b = 1 + u$. From *Theorem 1*, it is known that $\hat{n}(c) = [(1+u)^c - 1]/u$ is an unbiased estimation with (15) adopted as the sampling function. In this case, we can further obtain the accurate relative error instead of an upper bound.

*Theorem 3:* when the sampling function is $f(c) = [(1+u)^c - 1]/u$, the relative error of the unbiased estimation is $\sqrt{(1-1/n)u/2}$.

*Proof:* From (15), we have,

$$\frac{f(i+1) - f(i) - 1}{u} = \frac{1}{u}[(1+u)^i - 1] = f(i).$$

Consequently, (11) is equivalent to

$$\begin{aligned} H(n) &- H(n-1) \\ &= \sum_{i=0}^{n-1} Q_i(n-1)\{2f(i) + 1 + [f(i+1) - f(i) - 1]\} \\ &= \sum_{i=0}^{n-1} Q_i(n-1)(2f(i)+1) + u\sum_{i=0}^{n-1} Q_i(n-1)f(i) \\ &= 2(n-1) + 1 + u(n-1). \end{aligned}$$

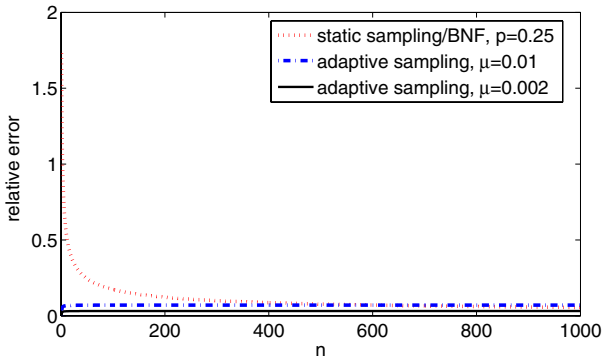


Fig. 2. Theoretical results of relative error.

Therefore,

$$H(n) = n^2 + \frac{n(n-1)}{2}u, \quad (16)$$

and the variation and relative error can be obtained as,

$$Var[\hat{n}(c)] = H(n) - (F(n))^2 = \frac{n(n-1)}{2}u. \quad (17)$$

$$\frac{\sqrt{Var[\hat{n}(c)]}}{n} = \frac{\sqrt{\frac{n(n-1)}{2}u}}{n} = \sqrt{\frac{(1-1/n)}{2}u} \quad (18)$$

■

With *Theorem 3* and *Theorem 6* (see Appendix), we can examine the relative error of ANLS and static sampling versus the flow size $n$, as shown in Fig. 2. Better NetFlow (BNF) [6] adaptively adjusts the sampling rate, but it samples all the flows with the same sampling rate. BNF can be viewed as adaptive linear sampling since it adjusts sampling rate linearly. If the final sampling rate of BNF in a sampling interval is $p_f$, the relative error of BNF is the same as the relative error of static sampling with sampling rate $p_f$. In other words, in theory, the relative error curve of BNF is the same as that of static sampling (as shown in Fig. 2) with sampling rate $p_f$. The advantage of BNF over static sampling is that it could find a proper sampling rate automatically to control the memory consumption (this is the motivation of BNF). From the figure, we observe that 1) for the static sampling method, the relative error is quite large for small $n$ as we demonstrated before; 2) For the ANLS method, the relative error is almost the same for different values of $n$; 3) The relative error of ANLS decreases as parameter $u$ diminishes.

From the Lemma 4 in [15], we can calculate the theoretical relative error of CATE as follows,

$$\frac{\sqrt{(1-p_f^2)(1+2(2k-1)p_f/(1+p_f))}}{2p_f\sqrt{Nk}} \quad (19)$$

The theoretical results of CATE and ANLS are listed in Table II. When we compute the results, the parameter for CATE is set as $k = 1000$, and the parameter for ANLS is configured as $u = 0.002$. In this comparison, $N = 10^6$, and the traffic proportions for large-size flows, medium-size flows, and small-size flows are $0.1 \sim 0.2$, $0.0001 \sim 0.0002$, and $10^{-7}$, respectively. The results indicate that ANLS is more accurate

than CATE for medium-size and small-size flows. It is a little bit less accurate for large-size flows for ANLS but the accuracy is reasonably acceptable.

TABLE II
THEORETICAL RELATIVE ERROR COMPARISON BETWEEN CATE WITH $k = 1000$ AND ANLS WITH $\mu = 0.002$.

| Flow size | Relative error of CATE | Relative error of ANLS |
|---|---|---|
| large-size | $0.0018 \sim 0.0028$ | $\leq 0.0316$ |
| medium-size | $0.1061 \sim 0.1871$ | $\leq 0.0316$ |
| small-size | 158 | $\leq 0.0316$ |

### B. Memory consumption

There are two parts of memory usage, the sample counters and the precomputed mapping table for $P(c)$ (equation 2), respectively. The former one dominates the memory usage.

*1) Memory for sample counters:* When the actual flow size is $n$, the expected counter value in ANLS can be calculated as

$$E[c(n)] = \sum_{i=1}^{n} Q_i(n) i \tag{20}$$

for which we have the following theorem.

*Theorem 4:* The expected counter value $E[c(n)]$ is up-bounded by $f^{-1}(n)$, where $f^{-1}(n)$ is the inverse function of $f(c)$.

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

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

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

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

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

Substituting (8) into (23), we obtain,

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

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

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

■

The sampling function is specified in (15), and the expected counter value of adaptive non-linear sampling method can be accurately calculated by (3) and (20). We compare this calculated value 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 (15): the exact gap is very small and the relative gap is approximately on the order of $10^{-4}$ or below. The counter values of the static sampling method and ANLS are shown in Fig. 4. When the flow size is $n$, the expected counter value for static sampling is obviously $np$. The counter value for adaptive non-linear sampling is larger than the one for static sampling when $n$ is small, but it becomes much smaller than the one for static sampling when $n$ grows. Please

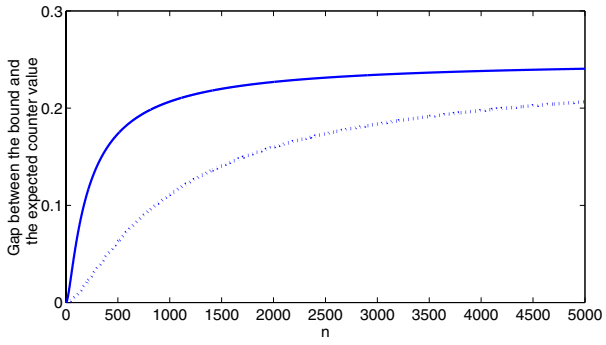


Fig. 3. Gap between the bound and the expected counter value.

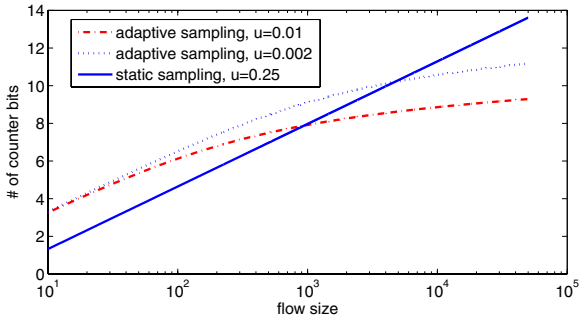


Fig. 4. Counter bits required for different sampling methods.

note that when we design the counter system, the width of the counter entry is determined by the largest counter value. Therefore, while keeping the same number of entries, ANLS consumes a smaller amount of memory than static sampling.

*2) Memory for mapping tables:* To avoid the high computation overhead, we pre-compute the values of $P(c)$ and store it in a table. For on-line operation, we only need a single memory access to read out the required $P(c)$. Considering the worst case of a fully loaded OC-48 link, which contains only one flow with all 40-Byte packets. In this scenario, we could compute the flow length $n$ in a one-minute measurement interval, and then the counter value will not exceed $f^{-1}(n) < 10000$ (actually it is about 9992). We store a 16-bit for each $P(c)$ where $c \leq 10000$. Therefore, the extra table to keep the mapping table $P(c)$ is only 160kb. Such amount of memory is not large compared with that for counters and in the evaluations of Section V, we focus on the counter memory usage.

### C. Processing overhead

The processing overhead is the computation cost of processing each packet, including the memory accesses and CPU operations.

There are five steps for the general sampling model. (i) the flow classification module picks up the flow ID from the incoming packet, (ii) the flow sampling module decides whether to sample the packet or not. If yes, (iii) it fetches the counter address from the flow table, (iv) gets the counter value using the address, and (v) writes back the updated value to counter. Otherwise, drop the packet and wait for the next

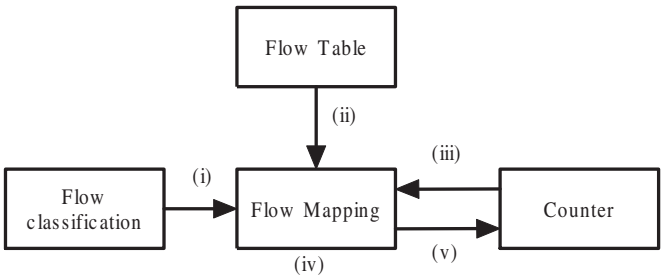


Fig. 5. System processing model of ANLS method.

packet. The ANLS model is in Fig. 5. (ii) fetches the counter address from the flow table, (iii) uses the address to read out the counter value, (iv) based on the counter value, the flow sampling module decides whether to sample and update the counter or not. From the model it is clear that we can implement ANLS using a five-stage pipeline. Thus the most time-consuming stage determines the processing speed. We analyze the processing overhead of these five stages below.

*Classification stage* ((i) in Fig. 5). Before deciding whether to sample a packet or not, the associated flow ID needs to be identified. Such a flow classification is also required by other flow sampling methods, like BNF and SDS. As the flow classification issue has been extensively discussed in the literature [20], we here ignore the detailed descriptions due to space limitations.

*Address fetching stage* ((ii) in Fig. 5). The processing is a table lookup operation on an on-chip SRAM.

*Sampling rate computation stage* ((iv) in Fig. 5). A concern of ANLS is that the sampling rate needs to be calculated on the arrival of each packet. However, the computational complexity of $P(c)$ should not be a big issue when ANLS is implemented in the real system by hardware. The value of $P(c)$ could be pre-computed and stored in a table. Thus we only need a direct address lookup on a table maintained by a small (on-chip) SRAM.

*Memory I/O stages* ((iii) and (v) in Fig. 5). Since the sampling method is utilized, the statistical results can be kept in a SRAM. The operation speed in this stage is determined by the access time on SRAM.

From the analysis above, we find that the processing bottleneck is mainly due to SRAM operations. Suppose the frequency of SRAM is 200 MHZ and, in the worst case, that each packet is 40 bytes. The I/O throughput of SRAM could match up to a 32 Gbps link speed. Actually, the processing overhead for sampling can be reduced greatly if the measurement function is implemented by hardware in a router. An intuitive explanation is that the processing for flow measurement should be much simpler than all the other processing functions in the router. By comparing with the other tasks' processing in a router, the measurement processing module should not be a big concern. Note that the flow measurement module can be a by-pass/parallel unit with all other data-path components in a router. Therefore, we can turn to pay more attention to the estimation accuracy and memory consumption for flow measurement.

## V. Performance Evaluations

In this section, we compare ANLS with other existing approaches including SS, BNF [6], SDS [5], CATE [15], and ARS [16] in terms of accuracy, memory cost, and processing overhead by performing two sets of experiments in the evaluation comparisons: 1) employing synthetic traces to test the different methods and 2) utilizing real IP data traces from NLANR [21] to validate our observations. All the results in this section are obtained by configuring the sampling function of ANLS as the specific form in (15). Furthermore, we discuss the processing overhead of each method and the attack resilience of ANLS in this section.

### A. Experiments and results on synthetic data

In order to examine the effects of flow size distribution on ANLS, we generate synthetic data for experiments. Suppose that we measure a fully loaded OC-48 (2.5 Gbps) link with a one-minute measurement interval. The required memory is calculated as the number of entries multiplied by the bit width of the entry, since each entry is of same width in real implementation as we mentioned before. The counter width is determined by the largest flow to avoid overflow. Please note that different sampling approaches vary in the number of entries and entry width. We first generate the flows whose sizes follow Pareto distribution (the shape parameter is 1.053 and the scale parameter is 4). We also synthesize data flows with an exponentially distributed size (the location parameter $\lambda = 500$, *i.e.*, the mean flow size is 500), and with uniformly distributed size between 1 and 1000.

The detailed results under different flow size distributions are depicted in Table III, Table IV and Table V. From the tables we observe that ANLS provides the most accurate estimation and that different flow size distributions have almost no effect on the average relative error (in fact, the average relative error of ANLS is only determined by the parameter $u$ as we demonstrated in Section IV).

TABLE III
Memory and relative error comparison under synthetic data generated from Pareto distribution.

| Methods | Parameters | Average relative error | Memory |
|---|---|---|---|
| ANLS | $u = 0.01$ | 0.07 | 4.49 Mb |
| SS | $p = 0.1$ | 0.96 | 5.17 Mb |
| BNF | $M = 256k$ | 1.38 | 4.44 Mb |
| SDS | $z = 1000$ | 0.99 | 10.03 Mb |

TABLE IV
Memory and relative error comparison under synthetic data generated from exponential distribution.

| Methods | Parameters | Average relative error | Memory |
|---|---|---|---|
| ANLS | $u = 0.01$ | 0.07 | 2.10 Mb |
| ANLS | $u = 0.2$ | 0.31 | 1.41 Mb |
| SS | $p = 0.1$ | 0.50 | 2.46 Mb |
| BNF | $M = 256k$ | 1.48 | 1.05 Mb |
| SDS | $z = 1000$ | 0.95 | 3.90 Mb |

TABLE V
MEMORY AND RELATIVE ERROR COMPARISON UNDER SYNTHETIC DATA GENERATED FROM UNIFORM DISTRIBUTION.

| Methods | Parameters | Average relative error | Memory |
|---|---|---|---|
| ANLS | $u = 0.001$ | 0.07 | 503.08 kb |
| SS | $p = 0.1$ | 0.96 | 493 kb |
| BNF | $M = 256k$ | 0.82 | 223.36 kb |
| SDS | $z = 1000$ | 0.699 | 744.49 kb |

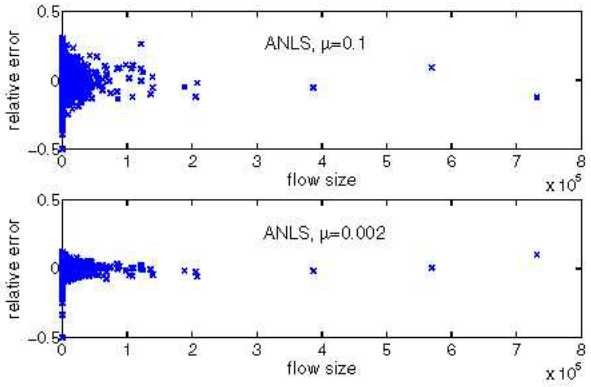


Fig. 6. ANLS relative error results on real NLANR trace.

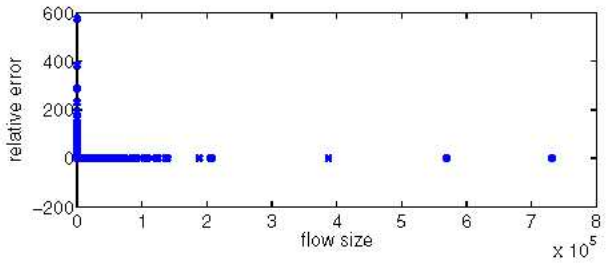


Fig. 7. relative error BNF($M = 20k$) or SS ($p = 1/20$) on NLANR trace.

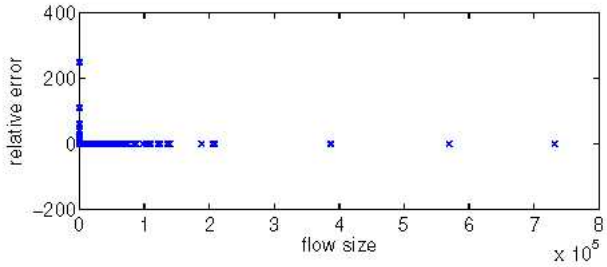


Fig. 8. SDS relative error with $z = 1000$ on NLANR trace.

Table III demonstrates the results for Pareto distributed flow size. Even when BNF (the $M$ parameter in the table is the expected flow entry for BNF) is furnished with a larger amount of memory comparable to ANLS, say 4.44Mb, its average relative error is almost 20 times worse than ANLS. Table IV, for the experiments on exponentially distributed synthetic data, shows that ANLS needs a slightly larger amount of memory than BNF but will provide tens of times more accurate measurement results. When a flow is generated with an uniformly distributed size between 1 and 1000, ANLS has an average relative error that is over 10 times better than BNF at a cost of about two times as much memory as BNF, as shown in Table V.

The average relative error and required memory size of SDS have no advantage over ANLS. Since SDS optimizes the statistic of large flows, it requires a lot of memory to record large flows. Compared with BNF, SDS has more accurate results but requires a larger amount of memory.

Since CATE and ARS depend on the packet arrival process, we do not use synthetic data to evaluate these two methods. The comparisons with them will be presented in Section V-B using real traces.

### *B. Experiments and results on real traces*

When we use an OC-192 real trace published in [21] as the experiment input, the results of ANLS are illustrated in Fig. 6, which shows the accuracy of ANLS for both small flows and large flows. Fig. 6 also clearly validates *Theorem 3*, which claims that a smaller $u$ is helpful to control the relative error.

We also apply other sampling methods to analyze the real trace and depict the results in Fig. 7 to Fig. 10. All these methods demonstrate a large relative error for small flows. In [16], a perfect theoretical theorem is provided to guide the sampling method. However, to practically benefit from the theorem, we should have a pre-knowledge of the flow length distribution. For this reason ARS, which employs an AR model to predict flow length distribution before deciding the sampling rate, is proposed. This method has the potential flaw that the accuracy is greatly limited by the AR model. We test ARS on a real trace using a $AR(1)$ prediction model and show the results in Fig. 10. Note that even when we use the actual data for the initial input to the $AR(1)$ model, the relative error for small flows is still larger than ANLS as shown in Fig. 6.

Besides the comparison of accuracy, we further illustrate the memory sizes of all the approaches in Fig. 11. It is shown that BNF consumes the least amount of memory, while SDS requires the largest amount of memory due to its optimization on large flows. The memory requirement for ANLS/ARS/CATE is similar. From the experiments on real traces, we found that the memory consumed by CATE is not as small as expected in [15]. The difference probably comes from the assumption in [15] that the packet arrival is uniform. In the experiment we observe many bursts of small flows, which will also make records in the coincidence count table of CATE. Please note that, the above memory expense corresponds to different sampling accuracy. The corresponding of relative errors of BNF, ANLS, ARS, SDS and CATE are 1.82, 0.21, 1.96, 1.186 and 262.11.

### *C. Processing overhead*

The processing overhead can be measured by the number of memory access and CPU operations, and we summarize the results of different methods in Table VI.

As discussed in Section IV, for each packet, ANLS needs one read operation, one write operation (update) on the counter, and a further memory read operation to get the pre-

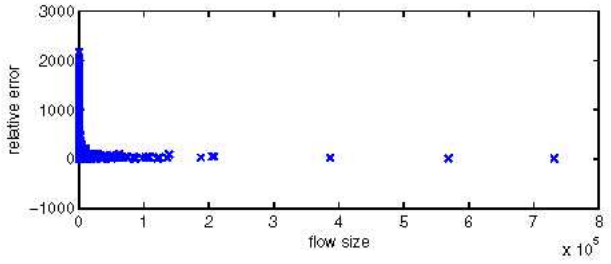


Fig. 9. CATE relative error with $k = 1000$ on NLANR trace.

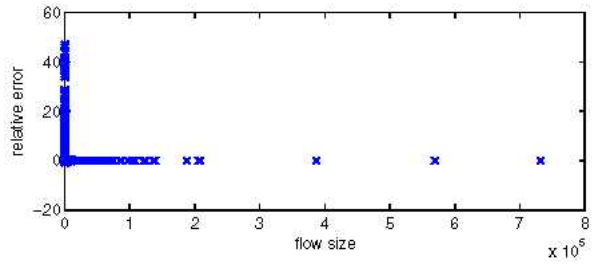


Fig. 10. ARS relative error on NLANR trace. Actual data is used for initial input of AR prediction model.

computed sampling rate.

Considering the implementation of BNF, it needs an additional CPU operation for re-normalization. Although re-normalization will not block the accounting process, it may delay the report process to the remote data collector if the re-normalization is not completed at the end of the measurement interval. Additionally, to determine the sampling rate, BNF needs to keep several histogram bins, which also consume memory. On the arrival of a packet, the related histogram should be updated, and all the histograms must be refreshed when a re-normalization process is activated. If a packet is sampled, BNF needs one write and one read operation and when the sampling is adjusted, BNF need two more memory accesses. In most cases, a total of 4 memory accesses are required.

SDS uses a minimum and division computation to decide the sampling rate and employs a maximum computation for re-normalization. For each packet, SDS requires one write operation and one read operation on the memory. To implement CATE, $k$ comparisons need to be done for each incoming packet. It can be deployed with a CAM, which requires one memory access. Two more memory accesses (one write and one read) are needed if there is a hitting in the comparison. ARS utilizes an $AR(n), (n > 1)$ model, which increases the memory consumption linearly with $n$. Furthermore, to determine the parameters of the $AR(n)$ model, we need to solve $n$ linear equations, and its computational complexity is a bit high if $n$ gets large. A nice feature of ANLS compared to ARS is that the sampling rate is adjusted according to the counter value and that no pre-knowledge on the flow size distribution is required. Two memory accesses (one write and one read) are needed to update the counter.

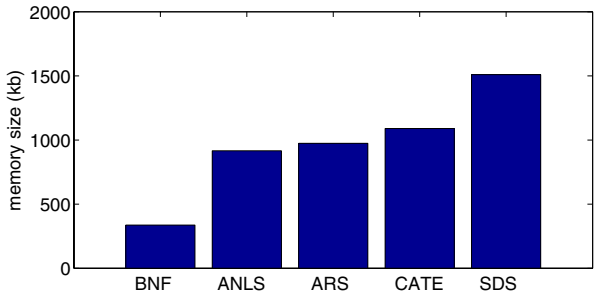


Fig. 11. Memory comparison of different approaches on NLANR trace. The corresponding of relative errors of BNF, ANLS, ARS, SDS and CATE are 1.82, 0.21, 1.96, 1.186 and 262.11

TABLE VI
PROCESSING OPERATIONS PER PACKET OF DIFFERENT METHODS.

| Methods | ANLS | SS | BNF | SDS | CATE | ARS |
|---|---|---|---|---|---|---|
| memory access | 3 | 2 | 4 | 2 | 3 | 2 |
| CPU operation | 0 | 0 | 1 | 2 | 0 | 1 |

### D. Attack resilience

ANLS keeps records for small flows. Although few resources are needed to record each flow, one may be concerned with the performance of ANLS when an attacker launches DoS attacks towards ANLS system with large number of small flows. We use a trace file collected by NLANR during the spread of the Slammer worm in January 2003 to test the attack resilience of ANLS. Since the average traffic rate of the original trace is not very large, we scale down the time stamp in each packet so that the flows will fully utilize the links of 100Mbps and 1 Gbps respectively. When the measurement interval is set as 5 seconds, the required memory size is shown in table VII, which implies that ANLS is resilient to DoS attacks.

TABLE VII
RESILIENCE TO ATTACK TRACES.

| traffic load | 100 Mbps | 1 Gbps | flow entries |
|---|---|---|---|
| memory size | 134 kb | 441 kb | 16549 |

### E. Summary

From the above results, the design spaces of different sampling methods can be summarized in Table VIII, where $A$, $B$, $C$ or $D$ is used to indicate that the performance of a method is $excellent$, $good$, $acceptable$, or $bad$ for a certain metric. ANLS bounds the error for both large and small flows. The analysis of the relative error and the upper bound for counter size, given in Section IV, can also be exploited to tackle the tradeoff in case that the memory constraint or error constraint is given. In fact, from Fig. 2 and Fig. 4, we could find out that, increasing $u$ will decrease memory requirement relatively quickly while slightly increasing the relative error. For real implementation, $u$ can not be arbitrarily large since it is limited by the constraints of the implementation and expected error.

TABLE VIII
DESIGN SPACES FOR DIFFERENT METHODS.

| Methods | ANLS | SS | BNF | SDS | CATE | ARS |
|---|---|---|---|---|---|---|
| Accuracy for small flows | A | D | D | D | D | C |
| Accuracy for media/large flows | B | B | B | A | A | B |
| memory | B | B | A | C | B | B |
| processing | C | A | B | C | C | C |

## VI. CONCLUSION

We have proposed an adaptive non-linear sampling method (ANLS) for passive measurement with the purpose of mitigating the high relative error for small events introduced by static sampling. The basic idea is to sample a small flow with a large sampling rate and to sample a large flow with a small sampling rate. ANLS has unbiased estimation and bounded relative error for the flow size estimation, and bounded counter size which implies small memory consumption. In particular, ANLS significantly improves the estimation accuracy for small flows compared to existing methods, while maintaining similar memory size and processing overhead. ANLS tunes the sampling rate according to the counter value, and no prediction or estimation of the flow size distribution is required. The experimental results show that the proposed sampling method obtains a better tradeoff between relative error and memory size consumption, in comparison to existing sampling methods. In addition, flow size distribution has almost no effect on the estimation accuracy.

## APPENDIX
## ESTIMATION AND ERROR OF STATIC SAMPLING

*Theorem 5:* $\hat{n}(c) = c/p$ is an unbiased estimation of the flow size $n$ under the static sampling method.

*Proof:* From the definition of the expected value of $\hat{n}(c)$, we have,

$$\begin{aligned} E[\hat{n}(c)] &= \sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} \frac{i}{p} \\ &= n \sum_{i=1}^{n} \frac{(n-1)!}{(i-1)!(n-i)!} p^{i-1}(1-p)^{n-i} \\ &= n(p+1-p)^{n-1} = n. \end{aligned}$$

■

*Theorem 6:* $\sqrt{(1/p-1)/n}$ is the relative error of unbiased estimation under the static sampling method.

*Proof:*

$$\begin{aligned} E[\hat{n}^2(c)] &= \sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} (\frac{i}{p})^2 \\ &= \frac{n}{p} \sum_{i=1}^{n} \frac{i(n-1)!}{(i-1)!(n-i)!} p^{i-1}(1-p)^{n-i} \\ &= \frac{n}{p} \sum_{i=0}^{n-1} \frac{(i+1)(n-1)!}{i!(n-i)!} p^{i}(1-p)^{n-i-1} \\ &= \frac{n}{p}[(n-1)p+1] = n^2 + n(1/p-1). \end{aligned}$$

From the definition, we have the relative error

$$\frac{\sqrt{Var[\hat{n}(c)]}}{n} = \sqrt{(1/p-1)/n}.$$

■

## REFERENCES


[1] K. Claffy and S. McCreary. Internet measurement and data analysis: Passive and active measurement. [Online]. Available: http://www.caida.org/outreach/papers/1999/Nae4hansen/Nae4hansen.html

[2] G. Varghese and C. Estan, "The measurement manifesto," *ACM SIGCOMM Computer Communication Review*, vol. 34, pp. 9–14, 2004.

[3] K. C. Claffy, G. C. Polyzos, and H.-W. Braun, "Application of sampling methodologies to network traffic characterization," in *ACM SIGCOMM 1993*, 1993, pp. 194–203.

[4] N. Duffield, C. Lund, and M. Thorup, "Estimating flow distributions from sampled flow statistics," in *ACM SIGCOMM 2003*, 2003, pp. 325–336.

[5] ——, "Learn more, sample less: Control of volume and variance in network measurement," *IEEE Trans. Inform. Theory*, vol. 51, pp. 1756–1775, 2005.

[6] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," in *ACM SIGCOMM 2004*, 2004, pp. 245 – 256.

[7] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *ACM SIGCOMM 2002*, 2002, pp. 323 – 336.

[8] D. Brauckhoff, B. Tellenbach, A. Wagner, A. Lakhina, and M. May, "Impact of traffic sampling on anomaly detection metrics," in *ACM SIGCOMM IMC 2006*, 2006, pp. 159 – 164.

[9] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?" in *ACM SIGCOMM IMC 2006*, 2006, pp. 165 – 176.

[10] K. Ishibashi, R. Kawahara, T. Mori, T. Kondoh, and S. Asano, "Effect of sampling rate and monitoring granularity on anomaly detectability," in *10th IEEE Global Internet Symposium*, 2007.

[11] C. Hu, B. Liu, Z. Liu, S. Gao, and D. O. Wu, "Optimal deployment of distributed passive measurement monitors," in *ICC 2006*, vol. 2, 2006, pp. 621 – 626.

[12] K. Suh, Y. Guoy, J. Kurose, and D. Towsley, "Locating network monitors: Complexity, heuristics, and coverage," in *INFOCOM 2005*, vol. 1, 2005, pp. 351–361.

[13] Cisco. Sampled netflow data sheet. [Online]. Available: http://www.cisco.com/en/US/products/ps6601/products_data_sheet09186a0080081201.html

[14] ——. Cisco ios netflow data sheet. [Online]. Available: http://www.cisco.com/en/US/products/ps6601/products_data_sheet0900aecd80173f71.html

[15] H. Fang, M. Kodialam, T. V. Lakshman, and Z. Hui, "Fast, memory-efficient traffic estimation by coincidence counting," in *INFOCOM 2005*, vol. 3, 2005, pp. 2080 – 2090.

[16] B.-Y. Choi, J. Park, and Z.-L. Zhang, "Adaptive packet sampling for flow volume measurement," University of Minnesota, MA, Tech. Rep. TR 02-040, Dec. 2002.

[17] N. Hohn and D. Veitch, "Inverting sampled traffic," in *ACM SIGCOMM IMC 2003*, vol. 14, 2003, pp. 68–80.

[18] L. Yang and G. Michailidis, "Sampled based estimation of network traffic flow characteristics," in *INFOCOM 2007*, 2007.

[19] A. Kumar, M. S. amd J. J. Xu, and J. Wang, "Data streaming algorithms for efficient and accurate estimation of flow size distribution," in *ACM SIGMETRICS 2004*, 2004, pp. 177–188.

[20] 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, pp. 947–961, 2006.

[21] NLANR. Passive measurement and analysis (pma). [Online]. Available: http://pma.nlanr.net