Adaptive Non-Linear Sampling Method for Accurate Flow Size Measurement

Chengchen Hu, *Member, IEEE,* Bin Liu, *Senior Member, IEEE,*
Sheng Wang, Jia Tian, Yu Cheng, *Senior Member, IEEE,*
Yan Chen, *Member, IEEE*

Abstract

Sampling technology has been widely deployed in network measurement systems to control memory consumption and processing overhead. However, most of the existing methods suffer from large errors for the estimation of small-size flows, which is important for many applications like Worm detection. To address this problem, we propose an adaptive non-linear sampling (ANLS) method for flow size estimation. Instead of statically pre-configuring the sampling rate, ANLS dynamically adjusts the sampling rate for each flow according to the value of a corresponding counter. A smaller sampling rate is utilized when the counter value is large, while a larger sampling rate is employed for a smaller counter. In this paper, the unbiased flow size estimation, the relative error, and the required counter size are investigated through theoretical analysis and experimental evaluations. The analysis and the experiments demonstrate that the proposed ANLS can significantly improve the estimation accuracy (particularly for small-size flows), and save memory consumption, while maintaining processing overhead comparable to existing methods. Moreover, we validate the design of ANLS by implementing it in an FPGA-based prototype, and the prototype is capable of measuring traffic whose throughput is up to 26.5 Gbps.

I. Introduction

In order to provide network status information for controlling, engineering, managing, and securing the communication networks, network measurement systems have been developed,

This paper was presented in part at IEEE INFOCOM 2008, Phoenix, USA.

Chengchen Hu is with Department of Computer Science and Technology, Xi'an Jiaotong University, China.Bin Liu, Sheng Wang and Jia Tian are with Department of Computer Science and Technology, Tsinghua University, China. Yu Cheng is with Department of Electrical and Computer Engineering Technology, Illinois Institute of Technology, USA. Yan Chen is with Department of Electrical Engineering and Computer Science, Northwestern University, USA.

which generally can be classified into passive and active measurement [1]. The former approach passively monitors traffic by analyzing packets passing through the traffic monitors, while the latter one actively injects probe packets into the network to infer the network status (*e.g.*, available bandwidth, packet loss ratio, delay) by analyzing the output of probe traffic. In this paper, we study the flow size estimation method that generates the flow information for passive measurement.

With the continuous increase of Internet link speed and number of flows, measuring the flow size has become a challenging task due to the demanding requirements on both memory size and memory bandwidth. For example, the processing time for a 40-Gbps OC-768 link is only 8 ns in the worst case (considering only 40-byte packets). This makes it necessary to employ SRAMs and it is infeasible to use only DRAM. However, due to their low density, SRAMs are susceptible to overflow for measurement applications [2]. Sampling technology is widely used as an efficient method for resolving the above contradiction [3, 4]. The simplest way to sample packets is the static sampling (SS) method, which selects packets with the same sampling rate/probability p for all the flows during the measurement interval. Denote c to be the counter value for a sampled flow, and n to be the flow size in terms of number of packets. It can be proven that the unbiased estimate of the flow size n is c/p [5], where p is the sampling rate for SS. Since the coefficient of variation $\frac{\sqrt{Var[\hat{n}(c)]}}{n}$ is an indication of relative error, it is used in this paper to formulate relative error. Therefore, the relative error is $\sqrt{(1/p-1)/n}$ for SS [5]. Obviously, from these results, the major issue with SS is that small-size flows cannot be estimated accurately, *e.g.*, the relative error will be 300% when $p = 0.1$ and $n = 1$.

Using a larger p can mitigate the relative error but lead to higher memory consumption, which contradicts the purpose of sampling. Even if it is possible to increase the sampling rate, almost all the increased samples come from the large flows, which have little measurement accuracy improvement on small and medium flows. However, the measurement results in [6] show that about 80% of the traffic flows has a size less than 2 packets. For such small flows, most of the existing related methods will lead to significant estimation error as we will demonstrate in Section VI.

Flow size estimation is employed in many network measurement applications. These applications have different requirements on accuracy, and an efficient flow size estimation method is expected to be applicable to the entire flow size spectrum. Some examples are given as follows.

- Detecting worm spanning/ super spreader/flooding [7, 8]. A sharp increase of TCP flows with only one 40-byte packet is probably caused by SYN flooding attacks or flash crowds. For this application, the estimation accuracy of small flows is very crucial.
- Identifying the heavy hitters or the dominant flows [9, 10]. Since the large flows are the major contributor of traffic volume, its measurement accuracy is quite important.
- Providing the statistics of flow size distribution or per-flow traffic. Such an application requires accurate measurement on the whole flow size spectrum for both small flows and large flows.

The motivation of this paper is to develop a sampling method, which could bound the estimation error for both small and large flows, so as to fit the requirements of diverse applications as illustrated above. In particular, we make the following contributions:

- We conduct a sampling-based flow size estimation method: Adaptive Non-Linear Sampling (ANLS). The merit of the method is that the sampling rate is adjusted according to the counter value and no prediction or estimation of flow size distribution is required beforehand.
- We derive analysis on unbiased flow size estimation, relative error, and required counter size for ANLS. The performance of ANLS is also investigated under synthetic and real network traces. ANLS significantly improves the estimation accuracy, particularly for small-size flows. Furthermore, the memory consumption of ANLS is the smallest under the same mean estimation accuracy, compared to related work.
- We design and implement a prototype system of ANLS. With one FPGA (Field Programmable Gate Array) chip and two small SRAMs, the prototype can deal with the measurement task on a 26.5 Gbps link. With SRAMs running on a higher frequency, say 250 Mhz, we can easily implement ANLS for OC-768 highly loaded links.

ANLS employs fast but small SRAM to keep the counters, so the processing speed on SRAM is not a constraint and the challenging issue is how to control the memory consumption without compromising the estimation accuracy. To address the problem, ANLS maintains a counter for each flow and tunes the sampling rate of the incoming packets according to the counter value of the flow it belongs to. Previous literatures employed prediction [11] or inverse estimate [12] method to "guess" the total packet count or flow size (distribution) in the next measurement interval and then determined sampling rate according to their estimate. However, with prediction technology, their methods are already not accurate in their input phase. Unlike these methods, our method adjusts the sampling rate based on the accurate counter value, which is an accumulative count of sampled packets in the current measurement interval, *i.e.*, ANLS decreases the sampling rate with the increase of counter value.

The rest of the paper is organized as follows. After the review of related work in Section II, we present how ANLS works in Section III. Section IV demonstrates the properties of ANLS and Section V describes a reference implementation. Section VI evaluates the performance of ANLS under synthetic data and real traces. And finally in Section VII, we conclude the paper.

II. Related work

Traditional counting systems provisions all counters with the same size resulting in inefficient for flow length counting. To accord with the "80-20" feature mentioned in Section I, recent work in [13] proposes BRICK to organize efficient "variable-length" counters with the idea of statistical multiplexing. Counter Braids (CB) [2] is another novel counter organization for accurate flow measurement, which builds a hierarchy of counters braided via random graphs in tandem. BRICK and CB reduce the total memory size by utilizing the features/relationship among all the counters, while ANLS reduces the total memory size by compressing every single counter. In fact, ANLS and BRICK/CB are complementary to each other and can be combined to get further reduction on counter size.

A combined SRAM&DRAM (SD) counter architecture was first proposed in [14]–[16]. The increments are first made only to SRAM counters, and the values of each SRAM counter is then committed to the corresponding DRAM counters before overflowing. SD architecture has its limitations. First, the read accesses of SD can only be done on the DRAM side and thus is quite slow. Second, SD also greatly increases the traffic between SRAM and DRAM across the system bus, which may lead to a serious bottleneck in system design [13]. Third, it is a trend to integrate measurement functions into routers; however, SD needs a SRAM and a DRAM, which will consume extra pins' connections and board areas.

In order to use only SRAM for flow size statistics, sampling is a simple but efficient way. A pioneering work on statistical sampling of network traffic was published in [17], which uses static sampling for the purpose of measuring on the NSFNET backbone. Presently, the primary flow-level measurement tool used by network operators is NetFlow [18], which resorts to packet sampling to handle the large traffic volume and diversity in high speed links. In the context of adaptive sampling, several methods were introduced for different purposes. BNF was proposed in [4], where a relatively large sampling rate is configured at the beginning and will adaptively decrease when possible memory overflow is detected. The SDS mechanism was presented in [19]. A flow whose size is larger than z is always selected by SDS, while the flow with size $x < z$ is sampled with probability x/z. ARS tunes the sampling rate by first predicating how many packets would arrive in the next measurement interval based on a linear auto-regressive (AR) prediction model [11]. The accuracy and complexity of ARS are greatly restricted by the operations to determine the AR model parameters. SGS also adjusts the sampling rate [12]. In SGS, each incoming packet is first hashed to increase one of the slots in a "sketch" and another counter for flow statistics is then updated based on estimated flow size from the sketch. SGS needs one more SRAM to keep the sketch, and its accuracy is penalized by the hashing conflicts in sketch. CATE proposed in [20] estimates the proportion of each flow by making multiple comparisons for each arrival and counting the number of coincidences. The closest work to our study is [21], which proposes a technique to count large number in small registers. Unlike that work, we provide a general family of sampling functions and provide the analysis on it. We also carry out empirical experiments to demonstrate the potential of ANLS.

III. Adaptive non-linear sampling method

The estimation error and memory consumption are major evaluation metrics in the design of an efficient sampling scheme. It is preferred that the estimation error, in the ideal case, keeps steady independent of the flow size to be estimated. The memory consumption is another constraint for flow size statistics, and an ideal one achieves a scalable increase (sub-linear increase) with the increase of flow size. To obtain the ideal curves, we propose ANLS for flow size estimation.

Before presenting how ANLS works, we first describe SS. With a static sampling of rate p, the counter value c will be refreshed upon a packet arrival according to the following expression,

$$c = \begin{cases} c+1 & \text{with probability } p; \\ c & \text{with probability } 1-p. \end{cases} \tag{1}$$

The proposed ANLS replaces the static sampling rate p in (1) with a function $P(c)$ over the counter value c as the following, where $P(c)$ is expected to diminish with the increase of c.

$$c = \begin{cases} c+1 & \text{with probability } P(c); \\ c & \text{with probability } 1-P(c). \end{cases} \tag{2}$$

Specifically, $P(c)$ for the proposed ANLS is calculated as

$$P(c) = 1/[f(c+1) - f(c)], \tag{3}$$

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 based 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 increases. A unique feature of ANLS compared to existing work is that the sampling rate is adjusted according to the counter value other than the prediction on packet size distribution [11] and packet count or estimate on flow size [12]. The sampling rate needs to be calculated on the arrival of a packet. To avoid computational processing overhead, the value of $P(c)$ could be pre-computed and stored in a $P(c)$ table, which will be shown to be quite small in Section IV-B2.

IV. Analysis of ANLS Properties

In this section, we investigate the properties of ANLS for accuracy, memory consumption with sampling function selected according to *Definition 1*.

A. Unbiased estimation and relative error

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

Theorem 2: Using $\hat{n}(c) = f(c)$ as the unbiased estimation, the coefficient of variation is upper bounded by $\sqrt{\frac{b-1}{2} - \frac{b-1}{2n}}$.

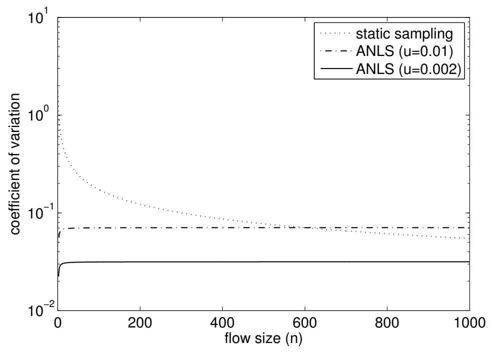

Fig. 1. Theoretical results of the coefficient of variation.

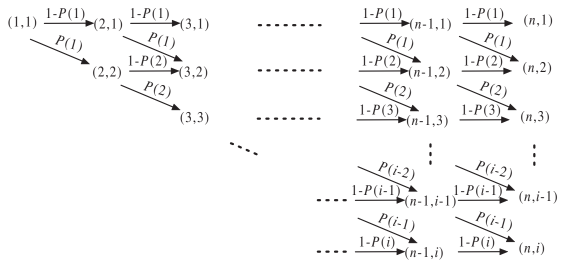

Fig. 2. Transition of the states.

Due to page limitations, we omit the proof of these two theorems and please refer to Theorem 1 and Theorem 2 in [5] for detailed proofs[1].

As mentioned in Section I, the coefficient of variation $\frac{\sqrt{Var[\hat{n}(c)]}}{n}$ in *Theorem 2* is used to analyze and indicate relative error. The coefficient of variation is zero when n is one. It increases with the increment of n and converges to $\sqrt{(b-1)/2}$ when $n \to \infty$. In addition, the relative error or the coefficient of variation 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

$$f(c) = [(1+u)^c - 1]/u, 0 < u < 1, \tag{4}$$

where u is a constant parameter. Obviously, (4) 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 when (4) is adopted as the sampling function. In this case, we can further obtain the accurate coefficient of variation instead of an upper bound.

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

Proof: See the proof of Theorem 3 in [5] for details. ■

With *Theorem 3* and the coefficient of variation of SS mentioned in Section I, we can examine the coefficient of variation of ANLS and SS versus the flow size n as depicted in Fig. 1. Better

[1]For the proofs, (13) in [5] should be $\frac{(b-1)n^2}{2} - \frac{(b-1)n}{2}$.

NetFlow (BNF) [4] adaptively adjusts the sampling rate, but it samples all the flows with the same sampling rate. 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, the relative error curve of BNF is the same as that of SS with sampling rate p_f (as shown in Fig. 1). The advantage of BNF over SS is that it could find a proper sampling rate automatically to fit the memory size. From the figure, we observe that 1) the relative error of SS is quite large for small n as we demonstrated before; 2) the relative error of ANLS is almost the same for different values of n; 3) the relative error of ANLS decreases as parameter u diminishes.

B. Memory consumption

There are two parts of memory usage: the counters to store counting results and the pre-computed mapping table to store the $P(c)$ values.

1) Memory usage for Counters: We first calculate probabilistic bound of counter value and then formulate its expected (average) bound.

Let us first denote (n, i) as the state that counter value c equals i when current actual flow size is n and $Q_i(n)$ as the probability in state (n, i). Then, we can define a metric to evaluate the probabilistic bound as the following, which means that the counter value will not exceed $B_\alpha(n)$ with probability of α, where

$$B_\alpha(n) = \sup\{y | \sum_{c=1}^{y} Q_c(n) \le \alpha, 0 \le \alpha \le 1\}. \tag{5}$$

To derive the numerical solution of $B_\alpha(n)$ given α and n, we analyze the Markov chain whose state transitions are depicted in Fig. 2. State (n, i) can be only transited from $(n-1, i-1)$ with probability $P(i-1)$ and from $(n-1, i)$ with probability $1 - P(i)$ after one incoming packet, namely,

$$Q_i(n) = Q_{i-1}(n-1)P(i-1) + Q_i(n-1)(1 - P(i)). \tag{6}$$

We first calculate $Q_i(n)$ using (6) as indicated in Fig. 2 and it is not difficult to see the initial value that $Q_1(1) = 1$, $Q_0(n) = 0$ when $n > 0$ and $Q_i(n) = 0$ if $i > n$.

Next, we increase y from the expected counter value $f^{-1}(n)$ (as demonstrated in [5]) to n by one each time and compute $\sum_{c=1}^{y+1} Q_c(n)$. We stop the increase of y once $\sum_{c=1}^{y} Q_c(n) \ge \alpha$ and then the value of y at the stop point is $B_\alpha(n)$.

Besides the probabilistic bound, we also investigate the average bound of the memory consumption.

Theorem 4: An upper bound of expected counter value $E[c(n)]$ is $f^{-1}(n)$, and $f^{-1}(n)$ is an increasing concave function when $f(c)$ is chosen from *Definition 1* (the ideal memory consumption curve as mentioned in Section I).

Proof: The proof that the upper bound of expected counter value $E[c(n)]$ is $f^{-1}(n)$ is demonstrated in the proof of Theorem 4 of [5]. And since $f(c)$ is an increasing function, obviously, its inverse function $f^{-1}(n)$ is also an increasing function [22].

■

By taking a logarithm computation, we calculate the counter width (number of counter bits) needed to record a flow size. We show the average bound and the 99% bound ($B_{0.99}(n)$) of the counter width in Fig. 3, compared to the average counter width for SS. The counter width for ANLS is larger than the one for SS when n is small, but it becomes much smaller than the one for static sampling when n grows. For simple implementation of counters, as most related work do, we consider fixed size counter whose width is determined by the largest counter value. Therefore, while keeping the same number of entries, ANLS consumes a smaller amount of memory than SS. Please note that, to maximize saving from small flows, variable size counters could also be used as recently mentioned in [13]. The figure also indicates that the probabilistic bound $B_{0.99}(n)$ increases sublinearly with n and is only a little bit larger than the expected counter size. Since it is much easier to compute the expected value than the numerical results of probabilistic bound, the expected counter size is an acceptable approximation to the actual counter size for ANLS.

Furthermore, we use the probability $P[|c(n)-E(c(n))| \geq t]$ as a concentration of measure [23] to indicate how the counter value is concentrated around its expectation. Given n, we first calculate $Q_i(n)$ using (6). Then the expected value of $c(n)$ and the concentration of measure can be obtained from the following two equations:

$$E[c(n)] = \sum_{i=1}^{n} iQ_i(n), \tag{7}$$

$$P[|c(n) - E(c(n))| \geq t] = \sum_{c\in\{|c-E(c)|\geq t\}} Q_c(n). \tag{8}$$

In figure 4, we depict the function $f(t) = P[|c(n) - E(c(n))| \geq t]$ when n equals 100, 1000 and 10000. We observe from the figure that the probability $P[|c(n) - E(c(n))| \geq t]$ is almost equal to zero when the difference t from expected value is larger than 20. Therefore, an additional one bit to the expected counter size is enough for practical use since the overflow probability is negligible[2].

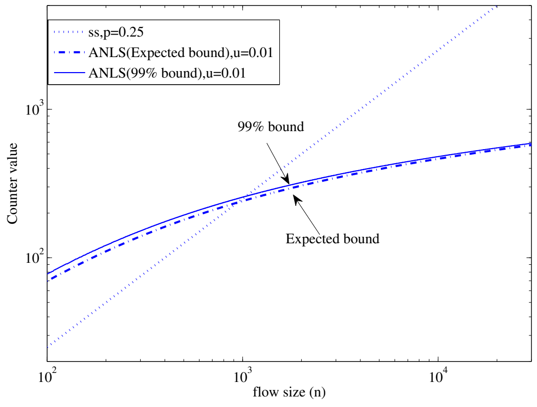

Fig. 3. Counter bits required for different sampling methods.

2) Memory usage for the $P(c)$ table: Consider the worst case of a fully loaded OC-48 link, which contains only one flow with all 40-Byte packets. In this scenario, the flow length $n = \frac{2.5\times10^9 b/s\times 60s}{8\times 40b} = 468,750,000$ in a one-minute measurement interval, and then the counter value will not exceed $f^{-1}(n) < 6883$ ($f(n)$ is in the form of (4) and $u = 0.002$) as indicated in *Theorem 4*. An extra table to keep $P(c)$ only consumes about 111 Kb ($16b \times 6883 \approx 111Kb$) memory. In addition, the $P(c)$ table's memory size does not increase linearly with the link speed. For one-minute measurement on an OC-192 link, the counter will not exceed 7577 using the same sampling function and the $P(c)$ table size is about 122 Kb. The increase of the precomputed table size is much slower than the increase of link speed. Similarly, enlarging the measurement interval also introduces a sublinear increase on the $P(c)$ table since a four-minute interval on OC-48 link also requires a 122 Kb table (only 1.1 times than that in one-minute interval). The amount of memory required for the $P(c)$ table is not large compared with the memory for

[2]Although $log(n)$-bit counters is required to guarantee zero overflow probability in theorem, by adding one bit to the expected counter size, the probability to overflow can be negligible in practical.

counters.

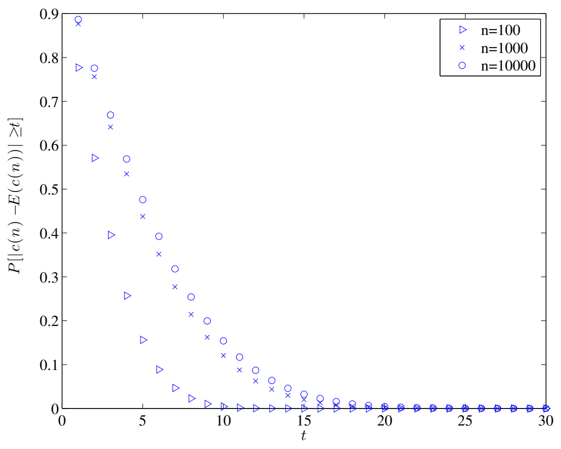

Fig. 4. Concentration of measure for the counter value

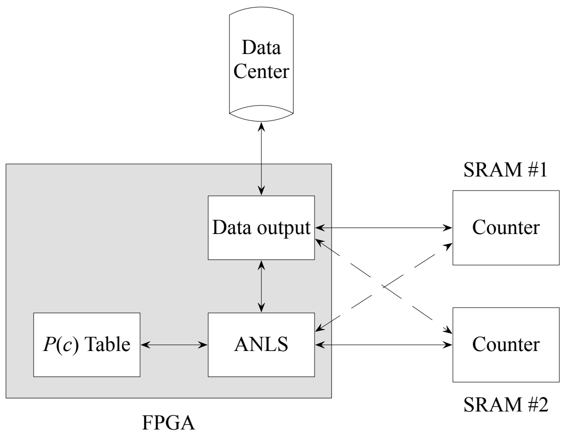

Fig. 5. Prototype system of ANLS.

V. Prototype Implementation

We implement an ANLS prototype using a FPGA (ALTERA Stratix EPS80) chip and two SRAMs (CYPRESS CY7C1372C, 18Mb) as shown in Fig. 5. The ANLS module in the figure implements the proposed algorithm using a pre-computed $P(c)$ table mentioned above, and the data output module is used to export the resulting statistics to the data center. We employ two alternating SRAMs in the prototype system for seamless measurement. When one SRAM becomes full, it will be paged to data center and the other one can process new records. In other words, these two SRAMs store flow counting information in alternating measurement epochs.

We configure each SRAM with 512k entries (counters). The width of each entry is 32 bits, where the leftmost 18-bit is used to store flow ID[3] and the rightmost 14-bit is employed as a counter. As mentioned in Section IV-B, ANLS assigns a counter for every distinct flow during the measurement epoch, but when all the counters are assigned or one of the counters becomes full, the switching of SRAM is triggered. By investigating backbone traces, the time to fill up all the 512k entries or to overflow the 14-bit counter is enough to page the counter values from the inactive SRAM. Taking the real trace (collected on an OC-192 link [24]) used in Section VI as

[3]Flow ID is a number to index and identify the flows.

illustration, we only observe 100728 flows in a ten-minute interval. In addition, for a ten-minute measurement interval on an OC-192 link, the packet number is $\frac{10Gbps\times 60s\times 10}{8\times 40b}$ in the worst case (one flow with only 40-Byte packets); thus the expected counter value 7577 if we set $u = 0.002$ in (4) and 13-bit counter is enough to count 7577. Adding one more bit to the all the counters to avoid overflow, we implement our prototype with 14-bit counters.

The ANLS module is the key function of the prototype system and we implement ANLS using a five-stage pipeline (developed by Verilog-HDL), where each step above is a stage in the pipeline. In each stage, the processing overhead (including the memory accesses and CPU operations) is presented as follows.

Stage (1) is the flow classification stage. 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 [25], we omit here the detailed descriptions because it is out of the scope of this paper.

Stage (2) is the counter address fetching stage. We implement this function using an efficient hardware hash function from the H_3 hash function class [26].

Stage (3) is the counter value reading stage. It is a read operation on SRAM.

Stage (4) is the sampling stage. As we mentioned above, the value of $P(c)$ could be pre-computed and stored in a $P(c)$ table. Thus we only need a direct address lookup on a table maintained by a small (on-chip) SRAM. In our implementation, the table has 10000 entries, each of which occupies 16-bit to keep the value of $P(c)$ for a specific $c \leq 10000$. Therefore, the total memory size for the $P(c)$ table is only 160 Kb and it is implemented by FPGA's on-chip memory. Furthermore, in this stage, a pseudo-random number is generated using LFSR (Linear Feedback Shift Register) [27], which can return a random number in one cycle. The LFSR deployed in our prototype is of 16 bits, and the characteristic polynomial is $x^{16} + x^{14} + x^{13} + x^{11} + 1$.

Stage (5) is the counter value writing-back stage. Counter value is updated and written back if necessary.

It is clear that the processing bottleneck is the SRAM operations. In the worst case, each packet is only 40 bytes. The operation frequency of SRAM in our prototype is 166 Mhz, therefore, the I/O throughput of SRAM could match up to 26.5 Gbps, and it means the prototype can handle the measurement task on a 26.5 Gbps link. If we choose SRAMs that could run on a higher frequency, say 250 Mhz (it is not difficult to obtain such SRAMs in today's FPGA/ASIC market), we can implement ANLS for fully loaded OC-768 links.

VI. Evaluations

In this section, we compare ANLS with other existing approaches including SS, BNF [4], SDS [19], CATE [20], ARS [11] and SGS [12] in terms of accuracy, memory, and processing overhead by performing two sets of experiments: 1) adopting synthetic traces to test the different methods and 2) utilizing real IP data traces from NLANR [24] to validate our observations. All the results in this section were obtained by configuring the sampling function of ANLS according to the specific form in (4). 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 different flow size estimation methods, we generate synthetic data for three traffic scenarios where the experiments are performed.

- *Pareto distribution* traffic scenario. The distribution of the flows sizes obeys a Pareto distribution where the shape parameter is 1.053 and the scale parameter is 4. The average flow size and the maximum flow size in this distribution are 81 and 3,295,575, respectively.
- *Exponential distribution* traffic scenario. The flow sizes are exponentially distributed with location parameter $\lambda = 100$. In this scenario, the mean flow size is 100 and the maximum flow size is 1416.
- *Bi-modal uniform distribution* traffic scenario. The flow sizes are uniformly distributed between 5 and 15 with probability 0.05, between 7500 and 8500 with probability 0.95. It is obvious that the average flow size and the maximum of flow size are 8000 and 8500, respectively.

We compare the experimental results and the theoretical value of ANLS's relative error in Fig. 6. Please note that both axes in the figure are in logarithmic scaling. The experimental value in the figure means the gap between unbiased estimation and real flow size, divided by the real flow size while the theoretical value in the figure is the mean relative error of all the flows calculated from *Theorem 3*. The results under three traffic scenarios show that the theoretical value well meets the experimental value. Since relative error R is approximately equal to $\sqrt{u/2}$

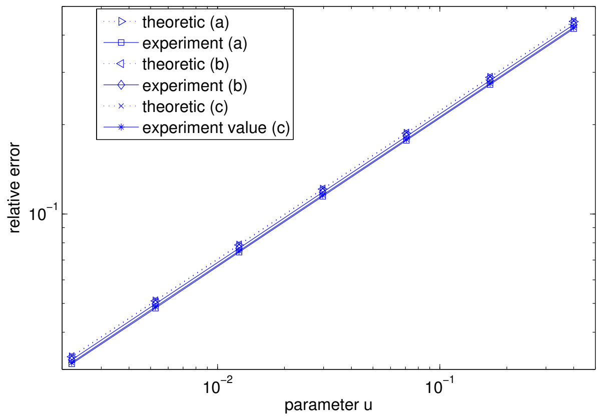

Fig. 6. Comparison between theoretical value and experimental value of ANLS relative error. (a) Pareto distribution traffic scenario; (b) Exponential distribution traffic scenario;(c) Bi-modal distribution traffic scenario

TABLE I

MEMORY AND RELATIVE ERROR COMPARISON UNDER PARETO DISTRIBUTION TRAFFIC SCENARIO.

ANLS	parameter (u)	0.4000	0.1682	0.0707	0.0297	0.0125	0.0053	0.0022
	relative error	0.42	0.27	0.18	0.11	0.07	0.05	0.03
	memory size	2.78 Mb	3.31 Mb	3.87 Mb	4.45 Mb	5.03 Mb	5.61 Mb	6.18 Mb
SS	parameter (p)	1/500	1/250	1/100	1/50	1/10	1/5	1/2
	relative error	7.11	5.06	3.17	2.23	0.96	0.64	0.32
	memory size	62.4 Kb	132.71 Kb	349.78 Kb	734.50 Kb	4.26 Mb	9.98 Mb	10.67 Mb
BNF	parameter (M)	3163	10000	17783	31623	100000	177828	316228
	relative error	26.33	13.16	9.31	6.58	3.05	2.15	1.32
	memory size	28.19 Kb	109.13 Kb	211.83 Kb	408.26 Kb	1.51 Mb	2.86 Mb	5.52Mb
SDS	parameter (z)	10000	3162.2	1000	316.2	100	31.62	10
	relative error	1.88	1.76	1.59	1.30	0.89	0.44	0.14
	memory size	850.86 Kb	1.47 Mb	2.52 Mb	4.20 Mb	6.67 Mb	9.52 Mb	11.11 Mb
CATE	parameter (K)	1	10	100	500	1000	5000	10000
	relative error	3.74	3.70	3.68	3.65	3.63	3.55	3.50
	memory size	11.18 Mb	12.89 Mb	14.59 Mb	15.26 Mb	16.31 Mb	17.01 Mb	18.02 Mb
SGS	parameter (ϵ)	0.14	0.12	0.10	0.08	0.06	0.04	0.02
	relative error	0.86	0.79	0.69	0.55	0.35	0.15	0.02
	memory size	8.15 Mb	8.31 Mb	8.58 Mb	8.90 Mb	9.40 Mb	10.20 Mb	12.63 Mb

(*Theorem 3*), the trends between $\log R$ and $\log u$ should follow a linear relationship, which is also indicated in Fig. 6.

The detailed comparisons among different methods under three synthetic flow size distributions are made to validate our method. The required memory is calculated as the number of entries multiplied by the counter width of the entry, since each entry is of the same width in real implementation as we mentioned before. The counter width is determined by the largest flow to avoid overflow and different sampling approaches vary in the entry number and entry width. The results are similar under all the three traffic scenarios and only the results for Pareto distribution are illustrated in Table I due to the page limitation. All the methods produce a more accurate result given a larger memory. Furthermore, we observe that ANLS provides the most accurate estimate using a memory with a same or even smaller 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, its average relative error is almost tens of times worse than ANLS. As indicated in Table I, when memory size of ANLS is 5.03 Mb, the relative error is only 0.07, which is nearly 20 times more accurate than BNF with 5.52 Mb memory size (the relative error for BNF is 1.32). The average relative error and required memory size of CATE have no advantage over ANLS. For CATE, the burst Poisson arrival packets generate a large number of flow counter entries in memory, and so it consumes a larger memory. CATE is accurate for large flows, but for small flows, the error is large. Since there are a large number of small flows, the average relative error is still large. The Poisson arrival process of packets may be another possible reason for the large relative error of CATE (The packet arrivals are supposed to be smooth by CATE in the original paper [20]). SGS is still less accurate than ANLS given similar memory size. The entries of the sketch for SGS method is 100k in the experiment. The relatively small gain on accuracy improvement is not worth the extra cost on memory size, if a large sketch is employed.

B. Experiments and results on real traces

We first use a real OC-192 trace [24] for experiments. There are 26,942,526 packets belonging to 100,728 flows. The maximum flow has 720,192 packets and the minimum flow has only one packet. The cumulative distribution of flow size is illustrated in Table II, and it demonstrates that most of the flows are small ones.

The results of ANLS are illustrated in Fig. 7, which demonstrate the accuracy of ANLS for both small flows and large flows. We also apply other sampling methods to analyze the real trace and depict the results in Fig. 8, respectively. All these methods demonstrate a large relative error for small flows. A nice theorem is provided to guide the sampling method. However, to practically

TABLE II

CUMULATIVE FLOW SIZE DISTRIBUTION OF THE REAL TRACE.

flow size (packets)	≤ 1	≤ 10	≤ 100	≤ 1000	≤ 10000
percentage	0.07	0.3499	0.8363	0.97	0.9956

benefit from the theorem, we should have a pre-knowledge of the flow length distribution. For this reason, ARS employs an AR (Auto-Regressive) model to predict flow length distribution before deciding the sampling rate. This method has the potential flaw that the accuracy is greatly limited by the AR model [28]. We test ARS on a real trace using a $AR(1)$ prediction model. 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. The parameter z of SDS is 1000 in this experiment, and this setting means SDS maintains full size counter for the flow that is larger than 1000. Thats why the error of SDS is zero when the flow size is larger than 1000. For the evaluation on SGS, the size of sketch is also set to be 100k and as the same as the example in [12] , $\epsilon = 0.1$ here.

Besides the comparison of accuracy, we illustrate the corresponding memory sizes of all the approaches. The corresponding relative errors of BNF, ANLS, ARS, SDS and SGS are 1.82, 0.21, 1.96, 1.186 and 0.69. It is shown that, ANLS only consumes more memory than BNF, and requires less memory than other methods[4]. ANLS is the most accurate method with the relative error of 0.21, while the relative error of BNF is 1.82. In other words, the accuracy is improved by 9 times at the cost of a 2.5 times memory in this experiment setting.

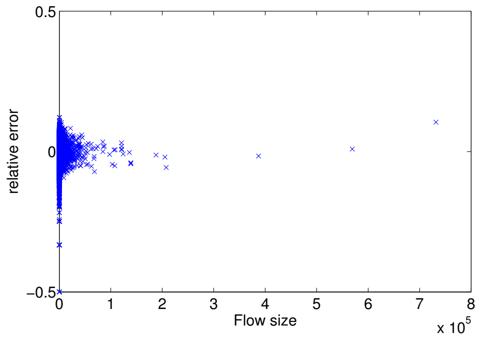

Fig. 7. Relative error of ANLS on real NLANR trace.

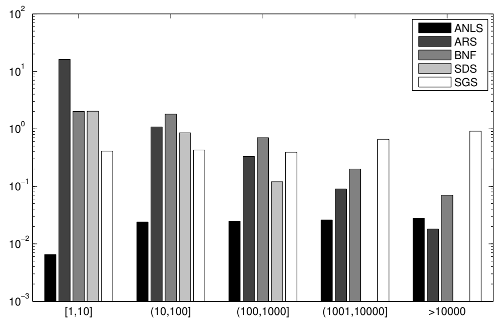

Fig. 8. Relative error under real trace .

[4]The memory cost of the $P(c)$ table for ANLS is not considered in this figure. Adding a $P(c)$ table with 122 Kb, the memory consumption of ANLS is a little bit larger than the one of ARS.

C. Attack resilience

Although few resources are needed to record each flow, one may be concerned with the performance of ANLS when an attacker launches DoS attacks on an 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[5], the required memory size for the 100Mbps link and the 1Gbps link are 134kb and 441kb, respectively. Please refer to [29] for how to effectively and adaptively adjust ANLS settings during an attack.

D. Processing overhead

The processing overhead can be measured by the number of memory access and CPU operations, and results of different methods in the worst case are summarized in Table III.

TABLE III

PROCESSING OPERATIONS PER PACKET OF DIFFERENT METHODS IN THE WORST CASE.

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

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 fetch the pre-computed sampling rate.

Considering the implementation of BNF, it needs an additional CPU operation for re-normalization. Although the re-normalization will not block the counting process, it may delay the report process to the remote data collector if the re-normalization is not completed at the end of a measurement interval. Additionally, to determine the sampling rate, BNF needs to keep several histogram bins, which also consumes 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

[5]When we scale down the time stamp to mimic a higher speed trace, we find the life-time of the revised trace being too short to use a one-minute interval as the previous experiment.

activated. If a packet is sampled, BNF needs one write and one read operation. BNF needs two more memory accesses when the sampling is adjusted. In the worst case, 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 becomes high if n gets large.

SGS first updates the sketch with one memory read and one memory write operation, and then if the packet is sampled according to the sketch value (one CPU operation is required to calculate the sampling rate), one more read and one more write operation on the counters are needed.

E. Summary

From the above results, the design space of different flow size estimation methods can be summarized in Table IV. Compared with other methods, ANLS possesses the following features.

- ANLS bounds the relative error for both large and small flows, and flow size distribution has almost no effects on relative error.
- To exhibit the same estimation accuracy, ANLS requires the smallest memory size compared to another methods.
- ANLS is very practical for real implementations and its processing overhead of ANLS is comparable to other existing methods.

VII. Conclusion

In order to control the high relative error for small flows introduced by static sampling, we have proposed an adaptive non-linear sampling method (ANLS) for flow size estimation in this paper. The basic idea of ANLS is to automatically adjust sampling rate for each flow according to the

TABLE IV

DESIGN SPACE FOR DIFFERENT METHODS.

Methods	ANLS	SS	BNF	SDS	CATE	ARS	SGS
Accuracy for small flows	Excellent	Bad	Bad	Bad	Bad	Fair	Fair
Accuracy for media/large flows	Good	Good	Good	Excellent	Excellent	Good	Fair
Memory consumption	Good	Good	Excellent	Fair	Good	Good	Good
Processing overhead	Fair	Excellent	Good	Fair	Fair	Fair	Fair

counter size so as to sample a small flow with a large sampling rate and to sample a large flow with a small sampling rate. Note that no prediction or estimation of the flow size distribution is required. ANLS has unbiased estimation and bounded relative error for the flow size estimation, and bounded counter size which implies small memory consumption. Furthermore, we find a broad category of sampling functions that can be used for ANLS. The theoretic and experimental results demonstrate that, ANLS significantly improves the estimation accuracy for small flows compared to the existing methods, while maintaining smaller memory size and comparable processing overhead. ANLS can also obtain a better tradeoff between relative error and memory consumption and the flow size distribution has almost no effect on the estimation accuracy.

REFERENCES

[1] G. Varghese, C. Estan, The measurement manifesto, ACM Computer Communication Review 34 (2004) 9–14.

[2] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, A. Kabbani, Counter braids: A novel counter architecture for per-flow measurement, in: ACM SIGMETRICS, 2008.

[3] N. Duffield, C. Lund, M. Thorup, Estimating flow distributions from sampled flow statistics, in: ACM SIGCOMM'03, 2003, pp. 325–336.

[4] C. Estan, K. Keys, D. Moore, G. Varghese, Building a better netflow, in: ACM SIGCOMM'04, 2004, pp. 245 – 256.

[5] C. Hu, S. Wang, J. Tian, B. Liu, Y. Cheng, Y. Chen, Accurate and efficient traffic monitoring using adaptive non-linear sampling method, in: INFOCOM'08, Phoenix, USA, 2008.

[6] X. Guan, T. Qin, W. Li, P. Wang, Dynamic feature analysis and measurement for large-scale network traffic monitoring, IEEE Transactions on Information Forensics and Security 5 (4) (2010) 905 –919.

[7] Z. Chen, C. Ji, Measuring network-aware worm spreading ability, in: IEEE INFOCOM'07, 2007.

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

[9] N. G. Duffield, C. Lund, M. Thorup, Charging from sampled network usage, in: ACM SIGCOMM IMW 2001, 2001, pp. 245 – 256.

[10] C. Estan, G. Varghese, New directions in traffic measurement and accounting, in: ACM SIGCOMM'02, 2002, pp. 323 – 336.

[11] B.-Y. Choi, J. Park, Z.-L. Zhang, Adaptive random sampling for load change detection, in: ACM SIGMETRICS'02, 2002, pp. 272 – 273.

[12] A. Kumar, J. Xu, Sketch guided sampling – using on-line estimates of flow size for adaptive data collection, in: IEEE INFOCOM'06, 2006.

[13] N. HUA, B. Lin, J. J. Xu, H. C. Zhao, BRICK: A novel exact active statistics counter architecture, in: ANCS 2008, 2008.

[14] D. shah, S. Iyer, B. Prabhakar, N. McKeown, Maintaining statistics counters in router line cards, IEEE Micro 22 (1) (2002) 76–81.

[15] S. Ramabhadran, G. Varghes, Efficient implementation of a statistics counter architecture, in: ACM SIGCOMM'03, 2003.

[16] Q. Zhao, J. J. Xu, Z. Liu, Design of a novel statistics counter architecture with optimal space and time efficiency, in: ACM SIGMETRICS'06, 2006.

[17] K. C. Claffy, G. C. Polyzos, H.-W. Braun, Application of sampling methodologies to network traffic characterization, in: ACM SIGCOMM'93, 1993, pp. 194–203.

[18] Cisco, Cisco ios netflow data sheet, http://www.cisco.com.

[19] N. Duffield, C. Lund, M. Thorup, Learn more, sample less: Control of volume and variance in network measurement, IEEE Trans. Inform. Theory 51 (2005) 1756–1775.

[20] F. Hao, M. S. Kodialam, T. V. Lakshman, H. Zhang, Fast, memory-efficient traffic estimation by coincidence counting, in: IEEE INFOCOM'05, 2005.

[21] R. Morris, Counting large numbers of events in small registers, Commun. ACM 21 (10) (1978) 840–842.

[22] L. S. Husch, Visual calculus, http://archives.math.utk.edu/visual.calculus/0/inverse.6/index.html.

[23] M. Luczak, Concentration of measure for markov chains with local transitions, in: Workshop New random geometries and other recent developments in probability, 2009.

[24] NLANR, Passive measurement and analysis (pma), http://pma.nlanr.net.

[25] K. Zheng, H. Che, Z. Wang, B. Liu, X. Zhang, DPPC-RE: TCAM-based distributed parallel packet classification with range encoding, IEEE Trans. Comput. 55 (2006) 947–961.

[26] M. V. Ramakrishna, E. Fu, E. Bahcekapili, Efficient hardware hashing functions for high performance computers, IEEE Trans. Comput. 46 (12) (1997) 1378–1381.

[27] M. John, S. Smith, Application-Specific Integrated Circuits, Addison-Wesley, 1997, Ch. 14.7.1.

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

[29] K. He, C. Hu, J. Jiang, Y. Zhou, B. Liu, A^2C: Anti-attack counters for traffic measurement, in: IEEE GLOBECOM 2010, 2010.