# Online Identifying Elephant Flows through a Scalable Non-uniform Sampling Algorithm

Cheng Guang, Gong Jian[1]
*the School of Computer Science and Engineering, Southeast University,*
*Nanjing, P.R.China, 210096*
*gcheng@njnet.edu.cn*

***Abstract*—Elephant flows contribute a large portion of the total traffic volume despite being relatively few in the number of flows. Thus, it is very important to identify these elephant flows in traffic engineering, network operation and management. One can easily keep counters for a few elephant flows using a small amount of fast memory (SRAM). Thus a reasonable goal is to devise an algorithm that identifies elephant flows using a limited SRAM memory. In this paper, we propose a novel approach to tackle this issue by using an identifying Elephant flows based on a Scalable Non-uniform Sampling algorithm (ESNS) to make the monitoring system self-adjustable to the varying monitored traffic. The paper use CERNET traces to compare the performance the ESNS algorithm with the sample and hold algorithm, and sampled algorithm. The experiment shows that the ESNS has better precision than PSH and sampled algorithm under the same flow memory size and configures.**

***Index Terms*—Elephant Flow, Sample and Hold, Measurement, Adaptive systems.**

## I. INTRODUCTION

Many studies [1, 2] have revealed that flow statistics exhibit strong heavy-tail behaviors in various networks. Fang [3] shows that 9% of the flows between AS pairs account for 90% of the byte traffic between all AS pairs. For one trace used in our study, about 0.08% of all flows contributed more than 57.3% of the total traffic volume. This characteristic is often referred to as the elephant and mice phenomenon that most mice flows only have a small number of packets, while a very few elephant flows have a large number of packets. Elephant flows contribute a large portion of the total traffic volume despite being relatively few in the number of flows. Elephant flows are defined as those that send more than a given threshold (0.1% of the total) during a given measurement interval.

The impact of elephant flows on network performance is significant. Thus, it is very important to identify these elephant flows in traffic engineering, network operation and management. By quickly identifying elephant flows, network operators can immediately take appropriate actions against individual hosts or networks generating these flows. Updating per flow in DRAM is already impossible with today's line speeds. One can easily keep counters for a few elephant flows using a small amount of fast memory (SRAM). Thus a reasonable goal is to devise an algorithm that identifies elephant flows using a limited memory.

To identify elephant flows, the primary method used for flow level measurement by IP backbone operators is to keep per flow counters in a large, slow DRAM, such as NetFlow [4]. The processing overhead can be alleviated using sampling algorithm, such as Sampled NetFlow [5]. Identifying elephant flows has also been discussed by Estan [6]. The main idea of their approach is to focus on elephant flows and neglect numerous mice flows. They proposed two novel techniques, referred to as sample-and-hold and multistage filters, respectively. Both techniques improve the process of extracting statistics of elephant flows in high-speed links, while still keeping the memory consumption reasonably low. The main difference of their approach from ours is that sample and hold algorithm can't control the memory consumption and the multistage filters can't maintain the estimated accuracy.

In this paper, we propose a novel approach to tackle this issue by using an identifying Elephant flows based on a Scalable Non-uniform Sampling algorithm (ESNS) to make the monitoring system self-adjustable to the varying monitored traffic. By caching estimating values rather than actual measured value, our ESNS system could adapt to the flow sampling in varying sampling rate.

ESNS records the estimated flow value rather than the measured value during a certain period, thus ESNS can use multiple sampling rates to adapt the traffic change, and we can also keep the measured accuracy of each sampling rate and don't waste the measured resource by a high sampling rate. We also propose one non-uniform flow management policies that can control the flow cache size and keep the estimated accuracy at the same time. A sample & hold module is adopted to selectively add packets into flow cache. By optimally adjusting the memory sampling rate and taking removal flow cache management scheme, the elephant flow information can be maximized with given limited system resources.

The paper is organized as follows: we discussed the related work, and provide an introduction to the related works in the identify elephant flows in Section 2. In Section 3, we elaborate our approach with detailed discussion on the corresponding

[1] This work has been supported by the National Grand Fundamental Research 973 program of China under Grant 2009CB320505, the Excellent Youth Teacher of Southeast University Program under Grant 4009001018, the Free Research Program of Key Lab of Computer Network in Guangdong Province under Grant No. CCNL 200706, and the Jiangsu Natural Science Foundation 2008.

**978-1-4244-2251-7/08/$25.00 ©2008 IEEE**

algorithms, and give a flow removal algorithm which removes some small flow according to a dependent-size sampling method. Comprehensive experiments are conducted and the results are discussed in Section 4. Section 5 concludes this work.

## II. Related Works

With the increasing demands from various areas such as network security, network flow monitoring has gained more and more attentions by some research communities. The IETF Packet Sampling working group (PSAMP) [7] is chartered to define a standard set of capabilities for network elements to sample subsets of packets statistically. Chaudhuri [8] has proved that sampling can be compensated for the estimated traffic in packets or bytes. Kumar [9] proposed a novel SCBF that performs per-flow counting without maintaining per-flow state and an algorithm for estimation of flow size distribution [10]. This is followed by [11] in which the flow distribution is inferred from the sampled statistics. Duffield [12] studied the statistical properties of packet-level sampling method using real-world Internet traffic traces, and developed a simple model to predict both flow statistics and the number of active flows.

There are some solutions to improve estimated precision of the measured flow. Estan [13] proposed a sample and hold algorithms for identifying the large flows, and they also described the optimization of early removal further improve the accuracy. Ashwin Lall [14] also uses a similar sample and hold algorithm that after one item is sampled, an exact count is maintained for it. Raspall [15] present a Shared-State Sampling algorithm to detect a large flows in the high-speed networks, which is a generation of sample and hold algorithm.

Recent work has shown that non-uniform sampling is necessary in order to control estimation variance arising from the observed heavy-tailed distribution of flow lengths [16]. Duffield [17] proposed a correlated sampling strategy that is able to select an arbitrarily small number of the “best” representatives of a set of flows.

Cisco’s NetFlow is an open but proprietary network protocol developed by Cisco Systems to run on Cisco IOS-enabled equipment for collecting IP traffic information. In the case of NetFlow, Cisco uses the 5-tuple definition, where a flow is defined as a unidirectional sequence of packets all sharing all of the following 5 values: Source IP, Destination IP, source port, destination port, and protocol. Maintaining NetFlow data can be computationally expensive for the router and burden the router's CPU to the point where it runs out of capacity. To avoid problems caused by router CPU exhaustion, Cisco provides "Sampled NetFlow". Rather than looking at every packet to maintain NetFlow records, the router looks at every nth packet, where n can be configured or it is a randomly selecting interval. When using sampled Netflow records, we can estimate the result by multiplying n, but it will bring into some error for the sampling rate.

All of these sampling techniques are valuable to study our algorithm. Obviously, there is a trade-off between monitoring accuracy and limited system resources (e.g. memory size). How to select an optimal sampling rate to achieve satisfactory accuracy with given system resource is a significant challenge. In this paper, we will try to tackle this issue.

## III. Identifying Elephant Flows Algorithm based on non-uniform sampling

Sampling measurement must solve two problems: improving estimated precision and reducing consumption of the measured resources. The number of flow decides the size of flow memory, so we use a sampling module to sample arriving packets to control the number of short flow to save the size of flow cache. We call this sampling module to memory sampling module, and name the sampling rate as memory sampling rate. If a flow is sampled into the flow memory, then its following packets are sampled but the size of flow cache isn’t increased, so it can improve the estimated accuracy of the measured flow. The memory sampling module is a sample and hold module.

The updated flow module updates flow information by considering new received packets. The module adopts a sample & hold mechanism, which records all packets information which belong to a existed flows, and then updates the corresponding flow entries. Otherwise, the packets are sampled with a probability and new entries will be created accordingly. In this manner, the heavy-tailed flows will be measured with a higher priority, which also contain more packet information. Thus, this approach can maximize to record flow information and estimate accuracy with a given flow cache size.

The Flow Removal Module adopts a removal policy to remove some flows (e.g. small flows) from the flow cache to accommodate memory space for more informative flows. In this paper, we provide a sampling dependent-size removal algorithm. The algorithm uses dependent-size sample to remove some small flows from the recorded flow cache.

We show the architecture of the ESNS system in the figure 1. The architecture consists of two modules: Sample & Hold Module, and Flow Removal Module.

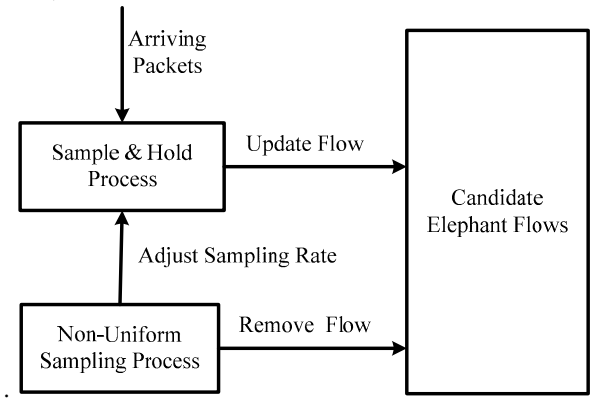


Figure 1. The ESNS Algorithm

In this architecture, all packets are processed by the sample & hold module. If there is a flow entry in the flow memory, then the flow entry is updated, otherwise the packet is sampled using one out of m to decide whether a new flow is created. If the number of flows in the flow memory is larger than a

threshold R, then a flow removal process will be started up to remove some flow entries from the flow cache.

We use a sample & hold module to update the flow cache. After a flow is recorded, all its follow packets are holden. The memory sampling ratio is 1 in N, that is p = 1/N. Suppose a flow f be sampled, there are x packets which have been missed, before the first packet in the flow f is sampled. This missed packets obey geometric probability distribution, and its probability distribution is $p(1-p)^x$ , so E(x)=1/p-1, D(x)=(1-p)/p2. When the first packet in flow f is collected, 1/p packets have passed, so we record 1/p as the initial value of flow f rather than 1, and its standard deviation SD(x)=sqrt(1-p)/p. As soon as the first packet in flow f is collected, all its following packets will be recorded in the flow entry. Let its following packets is c, then the value in flow f entry is equal to c+1/p, so its standard deviation is SD(x)=sqrt(1-p)/p.

The removal algorithm uses a size-dependent sampling scheme, which keeps long-size flows by a high sampling rate and small-size flows using a low sampling rate. One long-size flow can remember more packets information than one small-size flow using one flow entry in the recorded flow cache, so we use different sampling rate to different size flows. Let a threshold M, and the size of a flow be k. If k is larger than M, then the flow is sampled by a probability 100%, and if k is less than M, then the sampling rate of the flow is set to p=k/M. We replace uniform random sampling with the size-dependent probability. In this case, all flows larger than the threshold M will always be sampled 100%, so no removal sampling error is introduced into these flows. On the other hand, we use sampling rate k/M for small flows; this can vacate some spaces from the recorded flow cache. Because the underlying flows are small, so the error involved is small. If a flow is sampling with sampling rate p=k/M, then we compensate k/p for this flow. If k>=M, p=100%, k/p=k, these large flow don't need any compensation, but if k<M, p=k/M, then k/p=M. So a flow larger than M will keep the same value after the removal process, and the small flows which are sampled will obtain compensation according to their sampling rate, and all sampled small flows are M after they obtain compensation.

After a measurement epoch is over, let the smallest flow sampling rate of the sample & hold module be 1/m. The size of a flow is the equation (1)

(c-1)+1/p (1)

Where, p>=1/m, and its variance be $1/p^2$, then the removal sampling rate is q=(c+1/p)/m. Its unbiased estimator is (c+1/p)/q=m, and its variance is Equation (2).

$$\frac{c}{q^2}+\frac{1}{q^2p^2}=\frac{(c\cdot p^2+1)m^2}{(c\cdot p+1)^2}\le\frac{m^2}{c\cdot p+1}<m^2 \qquad (2)$$

If a flow is larger than threshold M during the removal process, then it's sampling error only comes from the sample and hold module. Let the sampling rate of sample and hold be p, p>=1/m, so its variance is $\frac{1}{p^2}\le m^2$. We can get conclusion that the variance of all flows in the recorded flow cache is less than $m^2$.

## IV. EXPERIMENT ANALYSIS

We use packet header traces in this paper gathered from CERNET[2], which were collected on an OC48 backbone link of CERNET on the November 10, 2005. Table 1 gives some sketch information of these traces. The first column in the table 1 is the names of the two traces. The second column is the total number of the source IP and destination IP pair. The third column is the packet number in the trace. The forth column is the elephant number using 1‰ threshold of the total packet number. The fifth column is the elephant number using 0.1‰ threshold of the total packet number.

Table1. the sketch information in the traces

| Traces | # Flows | # Packets | # 1‰ Elephant | # 0.1‰ Elephant |
|---|---|---|---|---|
| CERNET1 | 314081 | 19068897 | 8 | 1044 |
| CERNET2 | 454936 | 37936657 | 9 | 981 |
| CERNET3 | 564006 | 57364410 | 10 | 959 |
| CERNET4 | 662698 | 75947778 | 9 | 965 |

In the flow definitions, flow label is the two tuples <source IP, destination IP >. The elephant flows are the flows whose flows exceed a predefined threshold. In the experiment, the threshold is defined 1‰ and 0.1‰ of the total packet number.

Before we begin to examine the measured accuracy of different algorithms, two error metrics are defined. The $error_i$ metrics in the equation (3) evaluates the estimated error of the $i^{th}$ elephant flow, where $X_i$ is the actual packet numbers of the $i^{th}$ elephant flow, and $\hat{X}_i$ is the estimated value of the $i^{th}$ elephant flow. The avg_error metrics in the equation (4) is an average estimated error of all n estimated elephant flows, where n is the number of elephant flows.

$$error_i=\frac{(X_i-\hat{X}_i)}{X_i}\times 100\% \qquad (3)$$

$$avg\_error=\sum_{i=1}^{n} error_i/n \qquad (4)$$

We will compare the measured accuracy of the sampled algorithm, the SH algorithm, and the ESNS algorithm with the same size of the flow memory. In the experiment, we set the size of the flow memory to 10000.

The measured error of all application packets, which is larger than 0.1% total packets, is compared among the three algorithms in the figure 2, and the measured error of all elephant packets, which is larger than 0.01%, and less than 0.1% total packets, is showed in the figure 3 using the traces in the second trace in the table 1. This figure 4 is the average error of the elephant flows which are larger than 0.1% total packets. This figure shows that the accuracy of the ESNS algorithm is better than PSH and sampled algorithm.

The main advantages of this ESNS algorithm are as follows.

1. Record the Estimated Value. Traditional sampling algorithm records the measured value to flow memory entries, and estimates the expected number using the measured value

and sampling rate when the measured period is over, so the adaptive algorithms can only allow one sampling rate in a measurement period, and the sampling rate is the last one, and the lowest one. The most important character of our method is that the estimated result of measured value is recorded rather than the measured value itself, so ESNS can use multiple sampling rates in a measurement period.

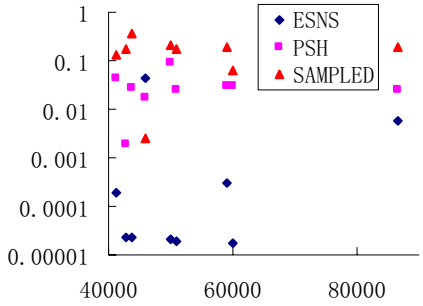


Figure 2. Measured Error of Elephant Flows Larger than 0.1% Total Packets

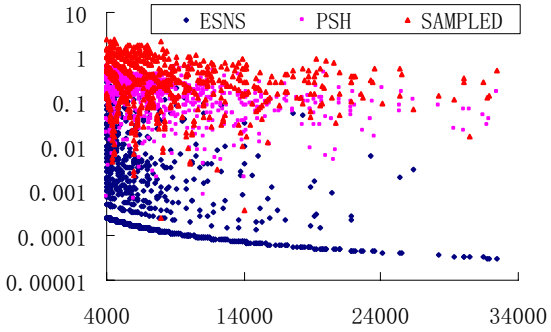


Figure 3. Measured Error of Elephant Flows Larger than 0.01% Total Packets

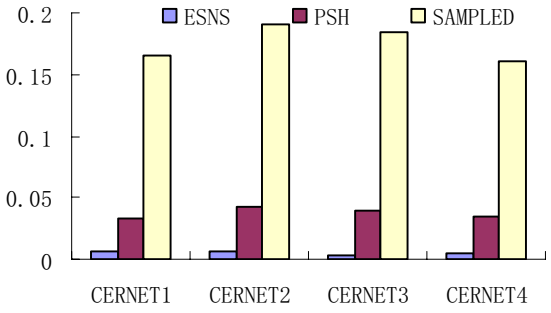


Figure 4. Average Error of the Elephant Flows Larger than 0.1% Total Packets

2. Non-uniform Sampling-Based Flow Removal Module. The flow removal module in the ESNS manages the size of the recorded cache and keeps the long flows into the recorded cache which can record more packets. It only computes again the entries which are moved from the flow memory in probability and long flows aren't processed, so it can assure more accurate and efficient flow monitoring than others methods. Sample and Hold Algorithm can't control the size of the flow cache, and the sampling rate which is configured before measured traffic. ANF NetFlow changes its sampling rate to adapt the traffic burst and the estimated error is equal to that of the smallest sampling rate. The renormalization in ANF NetFlow algorithm computes all NetFlow entries using the new sampling rate, so it consumes a lot of CPU resource to renormalize flow entries and may cause the CPU congestion during the renormalization.

## V. Conclusion

This paper presents an ESNS algorithm to identify elephant flows, which includes a sample & hold module, and a flow removal module. The ESNS can control the flow number in the flow cache by the flow removal module, and sample & hold module can record more packets into the flow cache. The ESNS algorithm has the advantages to adapt the traffic change and improve the estimated precision. We use CERNET traces to compare the performance the ESNS algorithm with the sample and hold algorithm, and sampled algorithm. The experiment shows that the ESNS has better precision than PSH and sampled algorithm under the same flow memory size and configures. There are two reasons for this result.

## References

[1] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," IEEE Network, vol. 11, no. 6, pp. 10–23, November/December 1997.

[2] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the Characteristics and Origins of Internet Flow Rates," In Proceedings of ACM SIGCOMM, pp. 309–322, August 2002.

[3] Fang, W. and Peterson, L. 1999. Inter-as traffic patterns and their implications. In Proceedings of IEEE GLOBECOM.

[4] Cisco IOS NetFlow Introduction, http://www.cisco.com/en/US/products/ps6601/ products_ios_protocol_group_home.html

[5] Sampled NetFlow Documentation, http://www.cisco.com/univercd/cc/td/doc/product/software/ ios120/120newft/120limit/120s/120s11/12s_sanf.htm

[6] C. Estan, Ken Keys, David Moore, George Varghese, Building a Better Netflow, SIGCOMM, August 2004

[7] Packet Sampling (PSAMP), http://www.ietf.org/html.charters/psamp-charter.html, 2007.1.

[8] S. Chaudhuri, R. Motwani, and V. Narasayya. Random Sampling for histogram construction: How much is enough? In Proceedings of the ACM SIGMOD, 1998.

[9] Abhishek Kumar, Jun Xu, Li Li, and Jia Wang, Space Code Bloom Filter for Efficient Traffic Flow Measurement, ACM/USENIX IMC, Miami, FL, October 2003.

[10] Abhishek Kumar, Minho Sung, Jun (Jim) Xu and Jia Wang, Data streaming algorithms for efficient and accurate estimation of flow size distribution, ACM SIGMETRICS 2004.

[11] Duffield, N.G., Lund, C. , Thorup, M.: Estimating Flow Distributions from Sampled Flow Statistics. ACM SIGCOMM . 2003, Karlsruhe,Germany. August 25-29. 325-336.

[12] Duffield, N.G., Lund, C., Thorup, M.: Properties and Prediction of Flow Statistics from Sampled Packet Streams, ACM SIGCOMM IMW 2002, November 6-8, 2002.

[13] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In SIGCOMM, Aug. 2002.

[14] Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, Hui Zhang, Data streaming algorithms for estimating entropy of network traffic, ACM SIGMETRICS Performance Evaluation Review, Volume 34 , Issue 1 June 2006.

[15] Frederic Raspall, Sebastia Sallent, Josep Yufera, Shared State Sampling, in Proceeding of IMC Oct. 2006.

[16] N.G. Duffield, C. Lund, M. Thorup, Learn more, sample less: control of volume and variance in network measurement, IEEE Transactions in Information Theory, vol. 51, no. 5, pp. 1756-1775, 2005.

[17] N.G. Duffield, C. Lund, M. Thorup, Flow Sampling Under Hard Resource Constraints, Sigmetrics 2004.