A Double-Sampling and Hold Based Approach for Accurate and Efficient Network Flow Monitoring

Guang Cheng¹, Yongning Tang², and Wei Ding¹

¹ College of Computer Science & Engineering,
Southeast University, Nanjing, P.R. China, 210096

² School of Computer Science, Telecommunications and Information Systems,
DePaul University Chicago IL USA 60604
gcheng@njnet.edu.cn

Abstract. One crucial challenge in network flow monitoring is how to accurately and efficiently monitor the large volume of network flows. Several approaches proposed to address this challenge either lack flexibility adapting to greatly varying network traffic (e.g. sNetFlow), or require intensive computing resources (e.g. ANF). In this paper, we propose a novel double-sampling and hold approach for network flow monitoring to tackle this challenge. We take a coarse-grained packet sampling to initially reduce the total number of monitored packets; then, an enhanced fine-grained sample and hold algorithm (ESHA) is adopted to selectively add packets into flow cache. By optimally adjusting the ESHA sampling rate and taking Early Removal flow cache management scheme, the flow information can be maximized with given limited system resources. Extensive simulation and experiment studies show that our approach can significantly improve both the accuracy and efficiency in network flow monitoring than other methods.

Keywords: Sampling Methods, Internet Traffic, Measurement.

1 Introduction

Network traffic monitoring and analysis is crucial for many network applications, such as network planning, network management, and network security applications. Network packets passing through the monitoring system can be classified into flows based on their header 7-tuple information [1], which can be further analyzed to present more significant implications. NetFlow [2], implemented in Cisco routers, can generate and output flow records, and keep a flow cache into memory as flow records to describe the passing traffic. Current flow monitoring approaches, which require recording flow records into memory to keep the flow status, usually run into problems if the number of flows too huge to hold in the memory. Several approaches have been proposed to address this challenge. However they are either lacking of flexibility adapting to greatly varying network traffic (e.g. sNetFlow), or requiring intensive computing resources (e.g. ANF).

In this paper, we propose a novel approach to tackle this issue by using a double-sampling and hold scheme to make the monitoring system self-adjustable to the

varying monitored traffic. In this approach, firstly, we take a coarse-grained packet sampling to initially reduce the total number of monitored packets to satisfying CPU processing capability in the monitoring system; then, an enhanced fine-grained sampling and hold algorithm (ESHA) is adopted to selectively add packets into flow cache. By optimally adjusting the ESHA sampling rate and taking Early Removal flow cache management scheme, the flow information can be maximized with given limited system resources. The paper is organized as follows: in Section 2, we discussed the related work. In Section 3, we elaborate our approach with detailed discussion on the corresponding algorithms. Comprehensive experiments are conducted and the results are discussed in Section 4. Section 5 concludes this work.

2 Related Work

With the increasing demands from various areas such as network security, network flow monitoring has gained more and more attentions by research community. The IETF Packet Sampling working group (PSAMP) [6] is chartered to define a standard set of capabilities for network elements to sample subsets of packets statistically. Furthermore, sampling techniques have been proposed [3,4,5] and employed in network products [2]. Kumar [10] proposed a novel SCBF that performs per-flow counting without maintaining per-flow state in and an algorithm for estimation of flow size distribution [11]. This is followed by [13] in which the flow distribution is inferred from the sampled statistics. Duffield [7, 12] studied the statistical properties of packet-level sampling using real-world Internet traffic traces, and developed a simple model to predict both the export rate of flow packet-sampled flow statistics and the number of active flows. The measured numbers of flows and the distribution of their lengths can be used to evaluate gains in deployment of web proxies [8].

However Sampled NetFlow suffers from two problems: (1) the number of flow cached in system memory could be significantly increasing if the monitored network hit by certain burst traffic such as DDoS; (2) pre-selected sampling rate cannot be adapted to the varying network traffic. Estan [1, 9] has proposed a family of bitmap algorithms for counting active flows and an adaptive NetFlow (ANF) was proposed. The adaptive NetFlow algorithm keeps resource within fixed limits, and uses renormalization to reduce the number of NetFlow records after a decrease in sampling rate. This algorithm divides traffic stream into fixed time interval bins. However, the algorithm has the following limitations: (1) it consumes a lot of CPU resource during renormalization because it needs repeatedly to analyze previously collected flow records that have already been processed and stored in system memory; (2) it inevitably loses monitoring accuracy due to static sampling rate; (3) it has to frequently adjust its sampling rate to adapt to varying traffic flow statistics.

Obviously, there is a trade-off between monitoring accuracy and limited system resources (e.g. memory size, CPU speed). 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.

3 Double-Sampling and Hold Based Flow Monitoring

3.1 System Overview

The double-sampling and hold based flow monitoring system consists of two components: Flow Information Updating (FIU) and Flow Cache Management (FCM). FIU updates flow information by considering new received packets, and FCM adopts an Early Removal policy to remove the number of least significant flows (e.g. small flows) from the flow cache to accommodate memory space for more informative flows. FIU adopts a double-sampling and hold mechanism, which consists of a coarse-grained sampling module, an enhanced fine-grained sampling and hold module. The coarse-grained sampling module is to protect the monitoring system from overwhelmed by effectively controlling the number of measured packets to decrease the load of CPU, processing the large volume of incoming traffic. Currently it simply samples one of every N packets. N is a system parameter configured based on CPU capacity and average traffic statistics. Fine-grained Sampling and hold module [4] records all packets information which belong to the 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 given higher priority, which also contain more information. Thus, this approach can maximize flow information and estimation accuracy with given flow cache size. FCM takes Early Removal policy remove small flows to control the total number of flows in the flow cache. Figure 1 show the architecture of the system.

Fig. 1. This algorithm architecture. In this figure, all packets are sample by one out of n^{th} , and all sampled packets are processed by hold function. If there is a flow entry in the flow memory, then the flow entry is updated, otherwise the packet is sampled again using one out of m^{th} 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 removal algorithm will be started up to remove some flow entries.

3.2 Algorithm Analysis

In the following, we first define some notations: $N1$ and $N2$ are the sampling rates used in coarse-grained and fine-grained samplings respectively; S is a collection of flows in the flow cache; $T1$ and $T2$ are the thresholds to trigger and stop Early Removal processing for the flow cache; $F = |S|$; m_1 and m_2 are the number of packets sampled by coarse-grained and fine-grained samplings respectively; m is the total number of incoming packets.

Network flows (called original flows) based on 7-tuple header information, which are, can be further aggregated to form new flows (called generated flows) based on a subset of 7-tuple header information. Assuming an aggregated flow x contains t original flows, and the number of flows in the flow cache S is T , the number of packets that are selected by coarse-grained sampling but dropped by fine-grained sampling is $m_1 - m_2$. Apparently, based on our sampling and removing mechanism, these dropped $m_1 - m_2$ packets must belong to small flows. Because of the applied hold algorithm, the packet number in flow cache is equal to the packet number of the sampled flows. Since all dropped flows and packets belong to small flows, which have the same random distribution, we have that the ratio of the flow number in the aggregated flow x to the number of flows in the flow cache is equal to the ratio between dropped packets of the aggregated flow x and the original packets, that is $t/T = x/(m_1 - m_2)$, x is the unsampled packets. According to the ratio, we can estimate $x = (m_1 - m_2)T/t$. Thus we can add up the two parts to estimate the sample result $\hat{x} = m_x + (m_1 - m_2) \cdot T/t$. If we consider the first sample information, the equation (1) is the estimated result of the aggregation flow x .

$$\hat{x} = (m_x + (m_1 - m_2) \cdot T/t) / p_1 \quad (1)$$

Let the ratio between the number of packets of aggregation flows and the total number of packets be f , so the packet number of aggregation flows is $f \cdot m$. The distribution of sampled aggregation has a binomial distribution with mean $p \cdot f \cdot m$, and variance $p(1-p)f \cdot m$. Since the estimate for the number of packets in the aggregation stream can be computed by multiplying the number of sampled packets by $1/p$, and the estimated variance is $(1/p^2)p(1-p)f \cdot m = (1-p)f \cdot m/p$. The standard deviation of this estimate is $SD(x) = \sqrt{(1-p)f \cdot m/p}$, $p = m_x/(m_2 N_1)$, and its relative error $RE(x)$ is equation (2).

$$RE(x) = \frac{\sqrt{(1-p)f \cdot m/p}}{f \cdot m} = \sqrt{\frac{1-p}{f \cdot m \cdot p}} = \sqrt{\frac{1-m_x/m_2 N_1}{f \cdot m \cdot m_x/m_2 N_1}} = \sqrt{\frac{m_2 N_1 - m_x}{f \cdot m \cdot m_x}} \quad (2)$$

3.3 Early Removal

The goal of Early Removal is to adapt fine-grained sampling rate and further control the total number of entries in the flow cache. The adaptive algorithm reduces the fine-grained sampling rate whenever the number of flow entries exceeds a predefined threshold. In this paper, we remove all flow entries whose size is less than a threshold d . If the number of the removed entries cannot satisfy the system requirement on the flow cache, then the threshold d increased by Δ and removing process will be conducted recursively until the requirement is satisfied.

The system may take long time to remove a predefined number of flows if we have to go through the entire flow cache multiple times. Here we introduce a quick search algorithm, which only searches the flow cache one time. Firstly, we set a k size dimension K . When the flow cache is updated, the dimension K is also updated. The index in the dimension K is the length of flow in the flow cache. If the length of a

flow s is x , and $x < k$, then the packet which belongs to s will be sampled and s will be updated accordingly. So we can set $K[x]=K[x]-1$, $K[x+1]=K[x+1]+1$. If $x=k$, then $K[x]=K[x]-1$. If a flow size is larger than k , then the flow won't be recorded in the dimension K . If a new flow is created in the flow cache, then $K[1]=K[1]+1$.

So if we want to set H as the removal threshold, according to the dimension K , the length of removal flow can be computed in equation (3). In the equation (3), if we set the removal threshold $d=j$, then the removal threshold H can be obtained by only going through the flow cache one time. If the k is too small, it may be $H > \sum_{i=1}^k K[i]$, then we can set $d = k+1$, and loop the flow cache to remove the threshold H flows. In this experiment, we set k be 100.

$$\sum_{i=1}^{j-1} K[i] < H \leq \sum_{i=1}^j K[i], j \leq k \quad (3)$$

4 Experiment Analysis

We use one team packet header traces gathered at NLNR [5]. This team traces use OC192MON hardware to collect data on August 19, 2004, from 13:40pm to 14:40pm. These traces information is displayed in table 1.

Table 1. Traces used in this experiment

No.	Time	File Size	#packets	#flows
1	2004/08/19 13:40	173MB	8434885	144813
2	2004/08/19 13:50	154MB	6922629	146665
3	2004/08/19 14:00	172MB	8251311	184213
4	2004/08/19 14:10	157MB	7111907	154750
5	2004/08/19 14:20	161MB	7388868	140297
6	2004/08/19 14:30	176MB	8560571	143341
7	2004/08/19 14:40	162MB	7527445	149196

In this experiment, we set the time bin 10 minutes, and the flow cache size is $2^{13}=8192$, $2^{14}=16384$, $2^{15}=32768$, $2^{16}=65536$, $2^{17}=131072$, $2^{18}=262144$, respectively. Firstly, we give a definition of estimated error. Let the size of aggregation flows be X , and its estimated value be $\hat{X}$, then we can get a relative estimated error metrics, $error = (X - \hat{X}) / X \times 100\%$, and a metrics about absolute value sum of relative estimated error, $sum_error = \sum_{i=1}^n (|X - \hat{X}| / X)$. Figure 2 is the relationship between the SPORT aggregation flow and their relative estimated error. Figure 3 shows a comparison based on sum error metrics between double-sampling algorithm and ANF algorithm and figure 4 gives the different percent relative error in the different time bins.

Table 2. The comparison between Estan ANF algorithm and Double-Sampling algorithm (DS). The initial sampling rate are all 1/2 in the two algorithms.

No	Flow memory	Ratio ANF	#packet ANF	#flow ANF	Ratio DS	#packet DS	#flow DS
1	4096	1/1024	9214	2708	1/512	3560390	3277
2	8192	1/256	35765	6790	1/128	3685570	7689
3	16384	1/128	72847	11603	1/64	3792411	12868
4	32768	1/32	260982	30183	1/16	3915760	28561
5	65536	1/16	486643	50297	1/8	4046263	49747
6	131072	1/4	1858527	112875	1/2	4181913	97189
7	262144	1/2	4217442	129862	1	4217442	129862

Figure 2(a)

Figure 2(b)

Fig. 2. Relative estimated error of SPORT aggregation flows. In this experiment, we let flow cache be 8192, and experiment data is the first team data in the table 1, and only analyze the SPORT aggregation flows whose size is larger than 0.1% of the total packets. The figure 2(a) is the relationship between the aggregation size and relative estimated error, and figure 2(b) shows the relationship between the SPORT number and relative estimated error. X axis in the figure 2(a) means the size of sport aggregation flows, and X axis in the figure 2(b) means SPORT number. Y axis in the two figures also means relative estimated error.

Fig. 3. A comparison based on metrics in equation (5) between double-sampling algorithm and Estan algorithm. X axis in this figure is set 4096, 8192, 16384, 32768, 65536, 131073, 262144 respectively. We notice that the error of Estan algorithm is larger than that of double-sampling algorithm, but when the flow cache size is equal to 262144, the two errors are equal. The reason is that the size of flow cache is larger than the flow number of the measured traffic, so the two algorithms will keep all flow information.

Figure 4(a)

Figure 4(b)

Fig. 4. The different percent relative error in the different time bins. Figure 4(a) is the relative error ratio between Estan's algorithm and the double-sampling algorithm in 5%, 25%, 50%, 75%, 95% percentile respectively. Figure 4(b) is the relationship between Estan's algorithm and the double-sampling algorithm in 50% percentile. The two figures show that the relative error of Estan's algorithm is at least 3 times larger than that of the double-sampling algorithm.

The experiments show that the accuracy of the double-sampling algorithm is higher than that of Estan's algorithm with the same system resources. There are two reasons for this result. Firstly, Estan's algorithm loses partial packet information of each flow in the flow cache during renormalization. The Early Removal algorithm in this paper only removes least significant small flow entries, but keeps all information of heavy-tailed flows. Secondly, only sampled packets can be added into the flow cache, but the double-sampling and hold algorithm in this paper can collected all packets after their flows are recorded into the flow cache. Therefore, the double-sampling based approach can keep more packets than Estan's algorithm with the same flow cache size, and thus it can obtain more accurate traffic flow information.

5 Conclusion

This paper proposes an adaptive NetFlow algorithm on double sample and hold, which includes two sample process, one hold process, and one early removal process. We can control the flow number in the flow memory by the hold and early removal process, and hold process can record more packets into the flow memory, so the algorithm has two advantages: firstly, it can control and adapt the flow cache size; secondly, it can improve the estimated precision; thirdly, it can save the load of CPU during early removal. We also use NLANR data to compare the performance between the double sample ANF algorithm in this paper and the Estan ANF algorithm. We analyze the SPORT aggregation flows over 0.1% of the total traffic, and the experiments shows that the double sample algorithm has better precision than the Estan ANF algorithm under the same flow memory size and configure parameters.

Acknowledgments. This work has been supported by the Natural Science Foundation of Jiangsu Province under Grant No. BK2006092, the Key Project of Chinese Ministry of Education under Grant No. 105084, the National Grand Fundamental Research 973 program of China under Grant No. 2003CB314804, and the Excellent Youth Teacher of Southeast University Program under Grant No. 4009001018, the National Natural Science Foundation of China under Grant No. 50609006.

References

1. C. Estan, Ken Keys, David Moore, George Varghese, Building a Better Netflow, SIGCOMM, (August 2004)
2. Cisco IOS NetFlow Introduction, http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html
3. Sampled NetFlow Documentation, http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120s11/12s_sanf.htm
4. C. Estan, G. Varghese.: New directions in traffic measurement and accounting. In SIGCOMM, (Aug. 2002).
5. Abilene-V Trace Data, <http://pma.nlanr.net/Special/ipls5.html>
6. Packet Sampling (psamp), <http://www.ietf.org/html.charters/psamp-charter.html>, (2002).
7. Duffield, N.G., Lund, C., Thorup, M.: Properties and Prediction of Flow Statistics from Sampled Packet Streams, ACM SIGCOMM IMW 2002, (November, 2002).
8. Feldmann, A. , Caceres, R. , Douglis, F. , Glass, G., Rabinovich, M.: Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments, IEEE INFOCOM 99.
9. Estan, C. and Varghese, G., Bitmap algorithms for counting active flows on high speed links, in Proc. ACM SIGCOMM Internet Measurement Conference, (Oct. 2003).
10. 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).
11. Abhishek Kumar, Minh Sung, Jun (Jim) Xu and Jia Wang, Data streaming algorithms for efficient and accurate estimation of flow size distribution, ACM SIGMETRICS 2004.
12. Duffield, N.G., Lund, C. , Thorup, M.: Properties and Prediction of Flow Statistics from Sampled Packet Streams. ACM SIGCOMM IMW 2002, (November, 2002).
13. Duffield, N.G., Lund, C. , Thorup, M.: Estimating Flow Distributions from Sampled Flow Statistics. ACM SIGCOMM . 2003, Karlsruhe, Germany. (August 325-336).