TELKOMNIKA, Vol.14, No.3, September 2016, pp. 1142~1149
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v14i3.3648

Flow Fair Sampling Based on Multistage Bloom Filters

Liu Yuanzhen*, Huang Shurong, Liu Jianzhao
Yancheng Institute of Technology, Yancheng 224051, Jiangsu, China
*Corresponding author, e-mail: liuyuanzhen_2004@126.com

Abstract

Network traffic distribution is heavy-tailed. Most of network flows are short and carry very few packets, and the number of large flows is small. Traditional random sampling tends to sample more large flows than short ones. However, many applications depend on per-flow traffic other than just large flows. A flow fair sampling based on multistage Bloom filters is proposed. The total measurement interval is divided into n child time intervals. In each child time interval, employ multistage Bloom filters to query the incoming packet's flow whether exists in flow information table or not. If exists, sample the packet with static sampling rate which is inversely proportional to the estimation flow traffic up to the previous time interval; if not, that is it's a new flow's first packet, create the flow information, insert it into the multistage Bloom filters and sample the packet with 100% probability. The results show that the proposed algorithm is accurate especially for short flows and easy to extend.

Keywords: *network traffic, short flow, multistage bloom filters, flow fair sampling*

1. Introduction

Traffic measurement is essential for network management, monitoring and scheduling, and many studies are carried out for different applications [1-2]. For high speed network, routers cannot manage to generate complete data for every packet. They have to employ restricted sampling which is effective to reduce data and resource consumption. In the field of network traffic measurement, according to the sampling strategy, there are mainly three kinds of sampling methods: systematic sampling, stratified random sampling and random sampling. Early traffic sampling mainly concentrated on packet sampling, and later on flow sampling. Flow is a collection of packets with the same properties within a time period. There are many methods to distinguish a flow. The common used is the five tuple method. The five tuple refers to source/destination IP address, source/destination port and protocol type. If any packet of a flow is sampled, then the flow is sampled.

Network traffic is self-similar and the distribution is heavy-tailed. Most of network flows are short and carry very few packets, and the number of large flows that carry large amount of packets is small. A study shows 9% of the total flows occupy about 90% of the traffic [3]. The traditional sampling methods tend to sample more large flows than short ones. And some algorithms focusing on large flow sampling [4-5] are proposed to solve problems in which omit large flow will bring great loss, such as traffic accounting. However there are many other applications which depend on per-flow information rather than large flows, such as attacks detection [6-8]. The purpose of traffic measurement determines what kind of sampling method to be adopted [9].

No matter what kind of sampling, as long as it employs 1 in N packets strategy, the sampling results tend to sample more packets of large flows and omit lots of short ones. Sampling that samples more packets from short flows at expense of low sampling rate of large flows, is called fair sampling. In the field of fair sampling, several researches have been carried out [10-12]. A sketch guide sampling algorithm is proposed [13], make the probability with which an incoming packet is sampled a decreasing sampling function of the size of the flow the packet belongs to, but the short flow estimation error is great, and the space efficiency is low. Zhang J, et al., [14] propose space efficient packet fair sampling, traffic is assumed to follow Zipf distribution with parameter z, and designs the multi resolution sampling statistics, but computing complexity is higher.

Received March 29, 2016; Revised May 17, 2016; Accepted June 5, 2016

In order to meet the needs of those applications that need per-flow information, a novel flow fair sampling method based on multistage Bloom filters (MBF) is proposed to capture more information especially from short flows. The whole measurement interval is divided into numbers of child time intervals. During a child interval, sampling rate for one flow remains unchanged. For an incoming packet, multistage Bloom filters are employed to find the corresponding flow information table, and sample the packet belongs to the flow with a certain sampling rate. The sampling rate is inversely proportional to the estimation traffic of the flow up to the previous child interval. For new arrival flow, there is no flow information, so a full sampling rate is taken, that is sampling the packet with 100% probability. The algorithm is simple to implement, time interval setting is flexible, and the results show that it is accurate especially for short flows.

The rest of this paper is organized as follows. Section 2 provides a brief survey of Bloom filter and multistage Bloom filters. Section 3 presents a detailed description of the proposed flow fair sampling. Section 4 gives theoretical analysis and section 5 gives experimental analysis. We conclude in Section 6.

2. Bloom Filter and Multistage Bloom Filters

A Bloom filter is a space-efficient data structure and is used to test whether an element is a member of a set. It supports member queries, random storage and has constant hash lookup time. You can add an element and query for an element with Bloom filter. Since proposed by Bloom Burton in 1970 [15], it has been used in database applications, and some improved algorithms are proposed [16-17]. In recent years, it has been widely concerned in the field of network [18-19].

An empty Bloom filter is a bit array of *m* bits, all set to 0. There must also be *k* different hash functions defined, with a uniform random distribution. Each of hash functions hashes some set element to one of the *m* array positions, and the corresponding positions are set to 1, as shown in Figure 1.

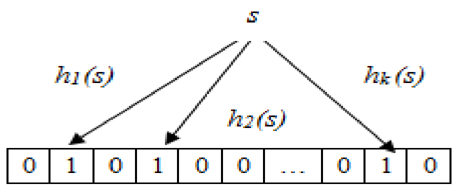

Figure 1. Standard Bloom filter

If an element belongs to a set, then the corresponding hash positions must be set to 1. False negative matches are impossible. However, due to the same position may be set to 1 for many times, other elements' hashing setting may lead to some element does not belong to the set and its corresponding hash positions are set to 1, thus the element is erroneously assumed to belong to the set. The probability is called false positive error, or in short false positive. False positive matches are possible. The probability of false positive will be analyzed below.

Assume that a hash function selects each array position with equal probability. Let *n* be the number of the set elements, *m* be the number of bits in the array, and *k* be the number of hash functions. The probability that a certain bit is set to 1 by a certain hash function is $1/m$. The probability that is not set to 1 is:

$$1-\frac{1}{m}$$

For there are *k* hash functions, so the probability that the bit is not set to 1 by any of the *k* functions during one insertion is:

$$\left(1-\frac{1}{m}\right)^k$$

After insert *n* elements, the probability (denoted as p_0) that certain position still is set to 0 is:

$$p_0 = \left(1-\frac{1}{m}\right)^{kn}$$

The false positive (denoted as p_{err}) is:

$$\begin{aligned} p_{err} &= (1-p_0)^k = \left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^k \\ &\approx \left(1-e^{-kn/m}\right)^k \\ &= e^{k\ln\left(1-e^{kn/m}\right)} \end{aligned}$$

While $k=(m/n)\ln 2$, that is $p_0 = 1/2$, p_{err} takes the minimum value:

$$p_{err-\min} = \left(1-\frac{1}{2}\right)^k = \left(\frac{1}{2}\right)^{m/n\ln 2} = 0.6185^{m/n}$$

Multistage Bloom filters is a parallel structure of several Bloom filters, as shown in Figure 2. Each Bloom filter employs different hash function clusters. To add an element, every Bloom filter should make an add operation. To test an element, only every Bloom filter indicates the element is in the set. Although the inherent false positive error of Bloom filter exists, the different hash function clusters can greatly reduce the error. Let *p* is the false positive error of a single Bloom filter, *n* is the number of stages, assume each Bloom filter has equal false positive error, then the false positive error of a multistage Bloom filters with *n* stages is p^n .

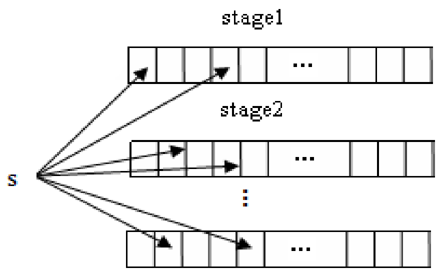

Figure 2. Multistage Bloom filters

3. Flow Fair Sampling Method

Let $f_k\,(k=1,2...)$ be the flow during the measurement interval. Divide the whole measurement interval into *n* child time intervals. we carry out flow sampling via packet sampling, that is if any packet of a flow is sampled, then the flow is sampled, and statistics are based on flows not packets. Collect sampled packets every child time interval and sampling rate for the flow is unchanged during the child time interval. We employ flow information table to maintain sampled flow information. Multistage Bloom filters are used to add or query for corresponding flow record in the flow information table.

For new arrival packet, look for it in the multistage Bloom filters, if any of the bits at the hash positions is 0, which means the corresponding flow record does not exist, then add the packet into multistage Bloom filters and create the flow information record. To add an element, all Bloom filters' corresponding positions should be set to 1. If the record exists, that is all Bloom filters indicate the corresponding positions are set to 1, sample the packets belong to the flow with static sample rate p_k^t and update the flow information, as shown in Figure 3.

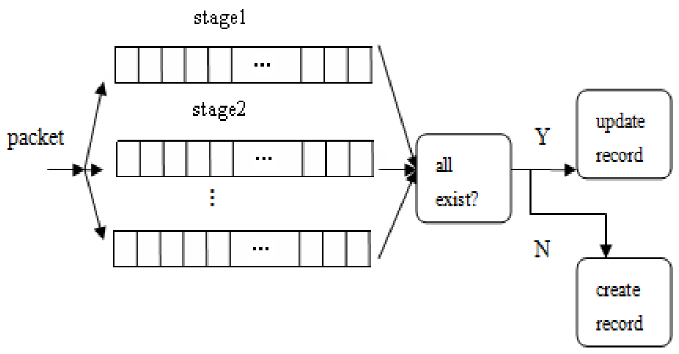

Figure 3. sampling process with multistage Bloom filters

The sampling rate p_k^t is unchanged during the child time interval. It is inversely proportional to the estimate traffic $\tilde{N}_k^{t-1}$ up to the previous child time interval. If the flow is *t* time interval new arrival flow, then the $\tilde{N}_k^{t-1}$ equals to 0, then sampling rate p_k^t equals to 1. As time goes by, the current estimate traffic $\tilde{N}_k^{t-1}$ grows, if a flow is not over, the sampling rate for the flow decreases with time as a step function. Sampling rate p_k^t is as follows:

$$p_k^t = \frac{1}{1+\frac{1}{2}\varepsilon^2 \tilde{N}_k^{t-1}} \tag{1}$$

Where ε is a constant. Let N_k^i be the number of packets sampled from flow f_k during child time interval *t*, and then the estimate traffic of flow f_k after time interval *t* is:

$$\tilde{N}_k^t = \sum_{i=1}^{t} N_k^i / p_k^t \tag{2}$$

Flow fair sampling process as follows:

```
divide the measurement into $n$ child time intervals;
  initialize $\tilde{N}_k^0$ to 0  // initialize traffic is 0
  initialize all $p_k^1$ to 1  //sampling rate for a new flow during a time interval is 1
  while (time interval $t$)  //during the child time interval $t$, $t=1...n$
  x=arrival packet;
    if(Bloom filter stage_i( $h_1(x), h_2(x), ..., h_k(x)$ ==1)
          // that is $x$ exists in flow info table
      sample the packet with static rate  $p_k^t$
      update flow information
    else
      set Bloom filter stage_i( $h_1(x), h_2(x), ..., h_k(x)$ =1 //insert the element into MBF
            sample the packet
            create flow record
    if time interval t is out
          compute $p_k^{t+1}$    //according to equation(1)
          compute $\hat{N}_k^t$    //according to equation(2)
     if it's time to clear Multistage Bloom filters
          clear Multistage Bloom filters, all bits set to 0  //every threshold time T
   // threshold $T = k\Delta t$ , where k is an integer, $\Delta t$ is the duration of child time interval
```

For the duration time of short flow is short and large flow is long, so sampling new arrival flow fully can get more short flows' information. As the number of large flows is small, the consumed resources are limited during child time interval. In order to reduce the pressure of resources caused by sampling, we can divide the measurement interval into shorter child time intervals. Then the resources consumption for large flows during the first arrival time interval is reduced, and during other time intervals sample large flows with certain sampling rate.

At the end of the time interval, the flow records which have not been updated are assumed inactive flows and removed from the flow information table. Clear up multistage Bloom filters every threshold time *T* to reduce conflict of hash collision, and at the same time it will break the flow information and bring error especially for large flows whose duration time go across several time intervals. Threshold *T* is constant times of duration Δt of child time interval:

$$T = k\Delta t \tag{3}$$

Where *k* is an integer.

4. Theoretical Analysis

4.1. Value Estimation

The entire measurement time is divided into *n* child time intervals. Let N_k^i be the packets number and X_k^i be bytes number sampled from flow f_k during child time interval *t*. Then the total packets number estimation value $\hat{N}_k$ is:

$$\hat{N}_k = \sum_{i=1}^{n} N_k^i / p_k^t \tag{4}$$

The total bytes number estimation value $\hat{X}_k$ is:

$$\hat{X}_k = \sum_{i=1}^{n} X_k^i / p_k^t \tag{5}$$

4.2. Space Consumption

The space consumption of the proposed algorithm mainly comes from two parts, one is the multistage Bloom filters taking up space, and the second is flow information table maintenance. A Bloom filter is a space-efficient hash data structure and gets its space efficacy at the cost of false positive error. The stages of multistage Bloom filters can be determined by the balance of resources and false positive error. For flow information table maintenance, the worst case is that the space consumption is constant times of the number of flows during the total measurement time. However, for the number of short flows is large and the duration time is short, at the end of the child time interval, the flows have not been updated are removed from flow information table to do further analysis, thus release much space, so the space needed turns to constant times of flows during child time interval.

4.3. Error Analysis

The error is composed of sampling error, false positive error of multistage Bloom filters and error caused by regularly clear up multistage Bloom filters. The false positive of multistage Bloom filters is exponentially decreased refer to single Bloom filter. Clear up multistage Bloom filters regularly is to reduce hash collision and brings error especially for flows whose duration time go across the time intervals. The threshold time is integer times of the time interval, and better longer than short flow duration time, which can reduce error to short flows. Comparatively, the main error is sampling error. Sampling is an effective method for data reduction, however, sampling itself cannot avoid inherent statistical error. In our algorithm, sampling error is more for large flows whose sampling rate is inversely proportional to its traffic, and have accurate information estimation for short flows.

5. Experimental Analysis

In this section, we use the actual data of the Internet to carry out experimental analysis. Data comes from the U.S. National Laboratory for Application Network Research (NLANR) publicly available TRACE.

Flow fair sampling rate changes with time interval. Figure 4 shows sampling rate comparison of our flow fair sampling and random sampling with sampling rate equals to 0.3 for the same flow, and the time interval is 1 millisecond (ms). Sampling rate of our flow sampling method decreases as a step function every child time interval. While 1 out *N* random sampling's sampling rate is set artificial or according to a certain rule and remains static.

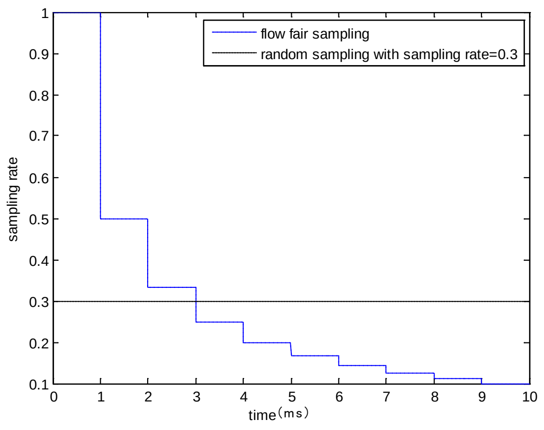

Figure 4. Sampling rate comparison

Under the same packet sampling rate, our flow fair sampling method turns to sample more flow. Figure 5 shows under the same packet sampling rate, flow fair sampling and random sampling's flow ratio sampled. Random sampling samples 1 packet from *N*, for network traffic distribution is heavy-tailed, majority of packets sampled are from large flows. While sampling rate is close to 1, all flow information can be obtained in theory.

The false positive error of multistage Bloom filters decreases as the number of stages increase. Figure 6 shows the relation of the stages number of multistage Bloom filters (MBF) and maximum of false positive error. While stages number equals to 1, multistage Bloom filters turns to a single stage Bloom filter. The false positive rate decreases exponentially with the increase of the stages.

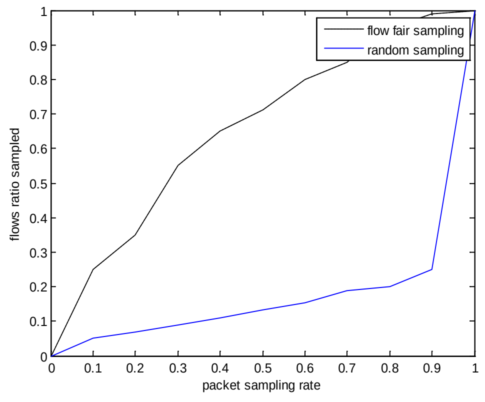

Figure 5. Flow ratio sampled comparison

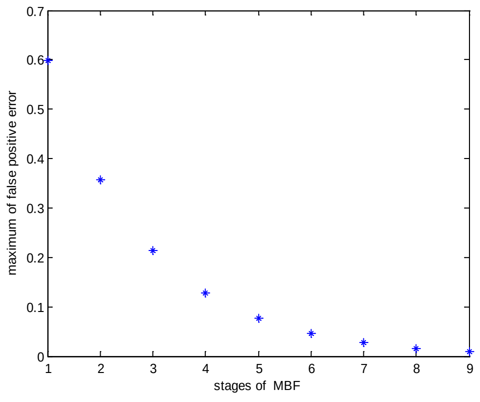

Figure 6. Relation between stages of multistage Bloom filters (MBF) and maximum of false positive error

6. Conclusion

This paper proposes a fair flow sampling algorithm based on multistage Bloom filters. The total measurement interval is divided into n child time intervals. During the same time interval, the sampling rate for certain flow keep static. The sampling rate is inversely proportional to the flow's estimate traffic up to the previous child time interval. It's simple to evaluate the total traffic according to sampled packets. The number of stages of multistage Bloom filters is adjustable according to the resources and the false positive error demand. The results show that the proposed algorithm can sample more short flows. And the algorithm can be applied in many applications such as attacks detection, traffic engineering.

References

[1] Zhao Y, Xie X, Jiang M. Hierarchical Real-time Network Traffic Classification Based on ECOC. *TELKOMNIKA Indonesian Journal of Electrical Engineering.* 2014; 12(2): 1551-1560.

[2] Zhou D, Liu W, Zhou W, et al. Research on Traffic Identification Based on Multi Layer Perceptron. *Telkomnika*. 2014; 12(1): 201-208.

[3] Fang W, Peterson L. *Inter-AS Traffic Patterns and Their Implications*. Proceedings of IEEE GLOBECOM. Rio, Brazil. 1999: 1859-1868.

[4] Zhang Z, Wang B, Lan J. Identifying elephant flows in internet backbone traffic with bloom filters and LRU. *Computer Communications*. 2015; 61: 70-78.

[5] Zhou AP, Cheng G, Guo XJ, et al. Parallel detection algorithm on long duration data streaming. *Journal of Communications.* 2015; 36(11): 156-166.

[6] Pontarelli S, Reviriego P, Maestro JA. Efficient Flow Sampling With Back-Annotated Cuckoo Hashing. *IEEE Communications Letters.* 2014; 18(10): 1695-1698.

[7] Bartos K, Rehak M. IFS: Intelligent flow sampling for network security–an adaptive approach. *International Journal of Network Management*. 2015; 25(5): 263-282.

[8] Joseph N. *Using Sub-optimal Kalman Filtering for Anomaly Detection in Networks*. Proceeding of International Conference on Electrical Engineering, Computer Science and Informatics (EECSI 2014). Yogyakarta, Indonesia. 2014; 1: 408-414.

[9] Zhou AP, Cheng G, Guo XJ. High-Speed network traffic measurement method. *Journal of Software.* 2014; 25(1): 135-153.

[10] Wang YX, Chen SQ, Zhang Z. *A fair sampling algorithm of network flow based on dynamic count filter*. IEEE 4th International Conference on Software Engineering and Service Science (ICSESS). 2013: 811-815.

[11] Yi P, Qian K, Huang WW. Adaptive flow sampling algorithm based on sampled packets and force sampling threshold S towards anomaly detection. *Journal of Electronics & Information Techanology.* 2015; 37(7): 1606-1611.

[12] Duffield N. *Fair Sampling Across Network Flow Measurements*. SIGMETRICS'12. London, England, UK. ACM. 2012: 367-378.

[13] Kumar A, Xu J. *Sketch guided sampling-Using on-line estimates of flow size for adaptive data collection*. In Proc. of IEEE Infocom 2006. Washington. 2006: 1-11.

[14] Zhang J, Wu JX, Niu XN. Space-Efficient fair packet sampling algorithm. *Journal of software*. 2010; 21(10): 2642-2655.
[15] Burton B. Space/time tradeoffs in hash coding with allowable errors. *Communications ACM.* 1970; 13(7): 422-426.
[16] Pontarelli S, Reviriego P, Maestro JA. Improving counting Bloom filter performance with fingerprints. *Information Processing Letters.* 2015; 116(4): 304-309.
[17] J Aguilar-Saborit, P Trancoso, V Muntes-Mulero. Dynamic Count Filters. *ACM SIGMOD Record*. 2006; 35(1): 26-32.
[18] Broder A, Mitzenmacher M. Network Applications of Bloom Filters: A Survey. *Internet Math.* 2003; (4): 485-509.
[19] Liu W, Qu W, Gong J, et al. Detection of Superpoints Using a Vector Bloom Filter. *IEEE Transactions on Information Forensics & Security.* 2015; 11(3): 514-527.