Journal of Applied Science and Engineering Innovation, Vol.2 No.8, pp.334-337
ISSN (Print): 2331-9062
ISSN (Online): 2331-9070

Counting Data Stream Based on Double Hash Algorithm

Lei Bai

Computer College, North China Institute of Science and Technology, Yanjiao , China

Abstract: In order to solve the problem of high false positives probability of Bloom Filter (BF), this paper proposed a novel mechanism based on double hash algorithm for counting data stream in high speed network. The algorithm improves the performance by using Time Bloom Filter to sample the packets of flows, and using Hierarchy Bloom Filter to identify flows. In this way, the algorithm can separate the process of fliteration from the recognition. The mechanism could not only record more flows, but also adapt dynamically when the number of flows rapidly rising. The experimental simulation results shown that compared with Bloom Filter, the algorithm can significantly improve the accuracy of recording flows information.

Keywords Traffic measurement, Network flow, Hash, Bloom filter

INTRODUCTION

With the development of high-speed network technology, In particular gigabit and higher rate network technology appearance, traditional statistical techniques for recording flow information become no longer useful gradually[Cheng *et al.*, 2013]. At present, the research on statistical flows information mainly concentrated in the packet sampling techniques and packet filtering technology aspects. Packet sampling technique is to sample packets from all elements. It can effectively reduce the storage space required during the process of statistical flows information. But it can also cause some loss of information, even though some studies reduce this loss by using inference methods such as EM algorithm and so on, however, the calculation of those methods is too big, and is not suitable for online processing[Domenico *et al.*, 2008]. Packet filtering technology is starting from the application of Bloom Filter in the network [Estan *et al.*, 2001]. As a kind of simple information representation scheme, Bloom Filter can meet the demand of high interaction and search requirement in the development of high speed network. It not only supports the collection element query, but filtrates out the elements which does not belong to this collection. Because of the cost of time and storage space is few the BF algorithm has very good practical value, so it is widely applied to the network traffic statistics [Yie *et al.*, 2010]. However, when more elements need to be handled, BF will generate false positive, which means an element is considered in a collection even though it is not. This will result in a significant impact for network traffic statistics. In this paper we proposed a novel mechanism based on double hash strategy for counting network flows statistic.

STANDARD BLOOM FILTER

A Bloom filter is a data structure which is described by an array of m bits for representing a set S= { $x_1, x_2 \ldots\ldots x_n$ } with n elements, initially set all of m bits to 0 [Kun *et al.*, 2009]. It uses k independent hash functions $H_1, H_2, \ldots H_k$ with range {1,….,m}. Bloom Filter will use the functions to map all of the elements of the set into {1,….,m} vector space [Andrei *et al.*, 2003]. For each element $x \in S$, the bits $H_i(x)$ are set to 1 for $1 \le i \le m$. A location can be set to 1 multiple times, but only the first change has an effect. To check if an item y is in S, we check whether all $H_i(y)$ are set to 1. If not, then clearly y is not a member of S. Figure 1 shows an example of BF with 3 functions.

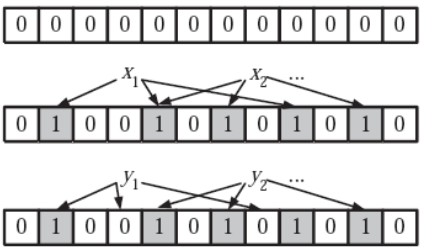

Figure 1. An example of a original bloom filter

As shown above, the filter begins as an array of all 0. Each item of set is hashed 3 times, with each hash yielding a bit location; those bits are set to 1. To check if an element y is in the set, hash it 3 times and check the corresponding bits. The element y_1 can not be in the set, since a 0 is found at one of the bits. Even though the corresponding bits of y_2 is 1, but it is either in the set or it has yielded a false positive. A

Corresponding Author: Lei Bai, Computer College, North China Institute of Science and Technology, Yanjiao 101601, China

false positive means that an element is considered in a collection even though it is not.

DOUBLE HASH ALGORITHM

Time Bloom Filter

Time Bloom Filter(TBF) is similar to BF except that it does not have m bits, but m buckets instead, each of which contains a timestamp. The *m* timestamps are denoted as *t*[0], *t*[1], …,*t*[*m*-1]. Besides, it has a bucket time-out value t_0. TBF compare the difference between two packets timestamp which hash to the corresponding k buckets in order whether is greater than a pre-set timeout t_0, then will update the timestamp of k buckets to the current timestamp of arriving packet. If any of the k timestamps, the ith for example, $t[h_i(c_j)]-t[h_i(c_{j-1})]>t_0$, the packet is discarded[Shao *et al.*, 2006].

Small flows have different 5-tuple identity (source/destination IP addresses, source/destination port numbers, and protocol identifier), will hash into different position of TBF. If any of the position timeout, the packets will be discarded. Packets belong to a large flow have less interarrival time and will hash into same position, then will be recorded. Extremely, when $t_0 = N_{\max}$, TBF algorithm will become invalid.

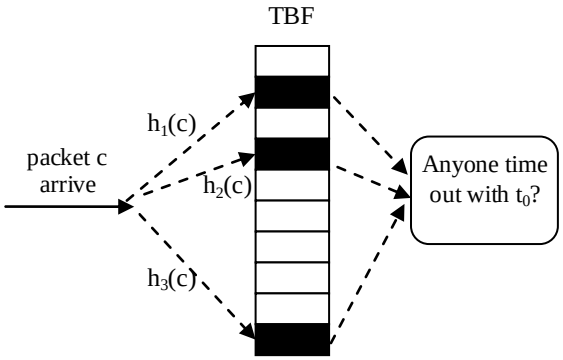

Figure 2. A TBF with 3 hash function

HIERARCHY BLOOM FILTER

A Hierarchy Bloom Filter aims at the problem of having big false positive rate when the set elements arbitrary growth in single Bloom Filter. A HBF is composed of a series of Bloom Filters that increase the length gradually. Each Bloom Filter in HBF is created with a predefined false positives probability. When the element of some Bloom Filter exceeds limited number, the HBF will increase a new BF which length multiplied. For example, when the number of elements of some BF , HBF will create a new BF with 2 times the length of the original BF.

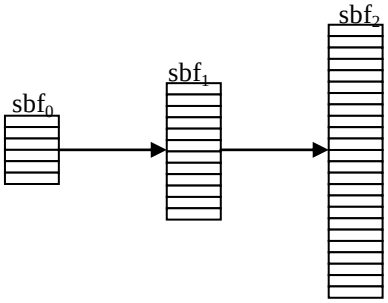

Figure 3. An example of the process of HBF with 3 BF

The Procedure Of Counting Flows By Double Hash Algorithm

According to the analysis result shown in the previous subsection, the procedure for recording flows by double hash algorithm is described as follows.

1. First, when a packet arrives, k independent hash functions will map a given key to one of the buckets according to the 5-tuple identity (source/destination IP addresses, source/destination port numbers, and protocol identifier)
2. Secondly, TBF compare the difference between two packets timestamp which hash to the corresponding buckets in order whether is greater than a pre-set timeout t_0, and update the timestamp of buckets to the current timestamp of arriving packet. If any of the timestamp is greater than t_0, the packet will be sampled. Then, the sampled packet will handled by HBF.
3. HBF will lookup whether the packet has presented before, If can find, then discard, else record the corresponding information of the flows, and add this packet to HBF.
4. This process is continued until to the end of the measurement period. At last, output the flows which we recorded.

Figure 4 illustrate the whole process.

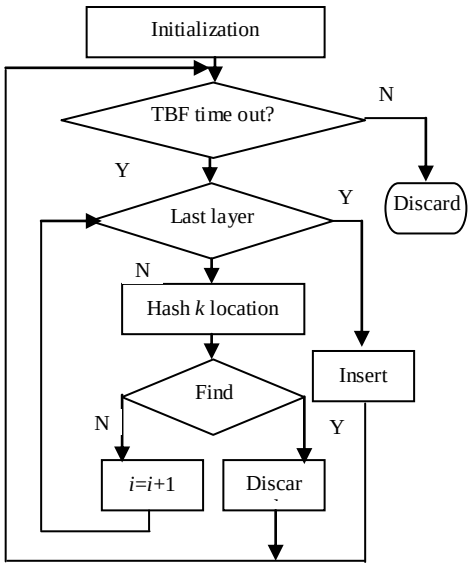

Figure 4. Flow chat of our algorithm record flows

Performance Evaluation

Just as BF, our algorithm also will bring a false positive error probability which is caused by TBF and HBF. Giving a set S={ $x_1, x_2 \ldots\ldots x_n$ } of n elements, the probability of hash position is empty in TBF is $p' = (1-1/m))^k \approx e^{-k/m}$, let $p = e^{-k/m}$, then $f_{TBF} = (1-p')^k \approx (1-p)^k = (1-e^{-k/m})^k$.

At a given times, we will have l filters in HBF with false positive error probabilities $P_0, P_1, P_2, , , , P_{l-1}$. For every BF the probabilities $P_i = (1-e^{-kn/m_i})^k$, m_i is the capacity of ith BF. Let $P_i = P_0 r^{i-1}$, r is a ratio with $0 < r < 1$, then $P = 1 - \prod_{i=0}^{l} (1 - P_0 r^i)$,now that $1 - \prod_i (1 - P_i) \le \sum_i P_i$, so $P \le \lim_{1 \to \infty} \sum_{i=0}^{l-1} P_0 r^i$, and therefore $P \le P_0 \frac{1}{1-r}$.

Now that, double hash algorithm handle the packets by using hash functions, so the cost of computing once time by hash algorithm is $o(n)$. Because the packets belonged to the same flow have the same flow identity, and the flow identity is unique, so will always map to the same position. Furthermore, the hash result is associate with store address, we can find the location of the flow in $o(n + \frac{n}{cm})$, c is a constant.

EXPERIMENTAL SIMULATE

We simulate and verify our algorithm validity by actual experimental data, and compare the measurement accuracy between the standard Bloom Filter and our algorithm. The experimental data we used is come from The Cooperative Association for Internet Data Analysis (CAIDA) [Zhang *et al.*, 2013]. There are a total of 6187376 packets and 68367original flows. Figure5 displays the distribution of original flows.

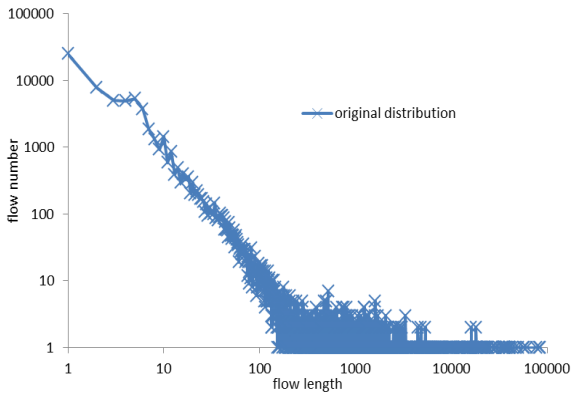

Figure 5. Original flows distribution

As shown in Figure 5, the flows distribution of network traffic meet the heavy-tailed distribution which means that most flows only have a small number of packets, while a very few flows have a large number of packets.

Figure 6 displays TBF sample rate when timeout interval t_0 changed. When t_0 =0, all packets will be sampled, t_0 =+∞, none packet will be sampled. So the value of the threshold t_0 can determine the number of packets sampled.

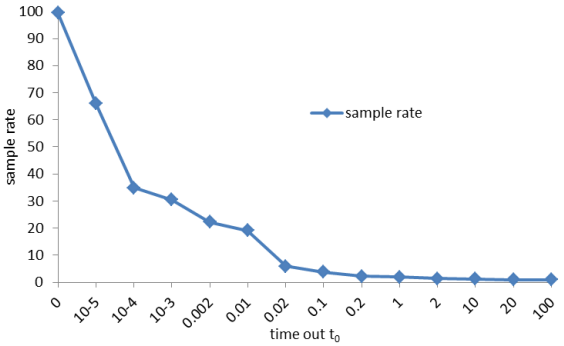

Figure 6. TBF sample rate when t_0 changed

Considering the sampling rate and the statistical error of the algorithm, we let $t_0 = 0.2s$, table 1 shows the difference between our algorithm and standard bloom filter in sampling rate and statistical error.

Table 1. Our algorithm compared with the BF

Algorithm	Sample rate	statistical error
Our algorithm	2.484	0.427
BF	0.962	12.976

As upwards table showed, the sample rate of our algorithm is slightly higher than the BF, but the statistical error is much smaller than the BF. For the reason of increasing the number of packets, BF will be full filled, and the error probability grows exponentially. For our algorithm, by using TBF to sample mice flows and using HBF which restricted by dynamic multilayer scalable mechanism, can reduce the measurement error significantly, and record more flows information statistic.

ACKNOWLEDGMENT

This work is supported by the Fundamental Research Funds for the Central Universities (No. 3142014085, No. 3142014100).

REFERENCES

Cheng G, Tang YN, 2013, "Estimation algorithms of the flow number from sampled packets on approximate approaches", Journal of Software, vol.24, No2,pp255-265.

Domenico Ficara, Stefano Giordano, 2008, "MultiLayer Compressed Counting Bloom Filters" , In Proceedings of the 27th IEEE INFOCOM, Washington,pp.933-941

Estan.C, Varghese.G, 2001, "New Directions in Traffic Measurement and Accounting", ACM SIGCOMM 2001, San Francisco, pp.75-80.

Ye W, Jian G, 2010, "Algorithm Based on Double Counter Bloom Filter for Large Flows Identification" ,Journal of Software. vol.21, No.5, pp.1115−1126.

Kun xie, 2009, "Bloom Filter Query Algorithm", Journal of Software, vol.20, No.1, pp.96−108.

Andrei Broder,Michael Mitzenmacher, 2003, "Network Applications of Bloom Filters:A Survey", Internet Math,vol.4,pp485～509

Xiaoxin Shao, Tao He, Shijin Kong, 2006, "Time-Driven vs Packet-Driven: A Deep Study on Traffic Sampling", Lecture Notes in Computer Science. pp610-619.

Zhang Z, Wang BQ, 2013, "Traffic measurement algorithm based onleast recent used and Bloom filter", Journal on Communications, vol.34, No. 1,pp111-120.