International Journal of Future Generation Communication and Networking
Vol. 9, No. 5 (2016), pp. 75-82
http://dx.doi.org/10.14257/ijfgcn.2016.9.5.08

Frequent Items Mining Algorithm Over High Speed Network Flows Based on Double Hash Method

Lei Bai and Chao Chen

Computer Department, North China Institute of Science and Technology
leib_cn@163.com

Abstract

In the high-speed backbone network, with the increasing speed of network link, the number of network flows increase rapidly. Meanwhile, with restrictions on hardware computing and storage resources, so, how to identify and measure large flows timely and accurately in massive data become a hot issue in high speed network flow measurement area. In this paper, we propose a new algorithm based on double hash algorithm to realize large flow frequent items identification, according to the defect of MF algorithm which produces false positive easily and frequent updates to bring the huge pressure to the system. The complexity and false positive rate of the algorithm was analyzed. The effect of large flow frequent items statistical accuracy and discard rate for parameter configuration was analyzed through simulation. The theoretical analysis and the simulation result indicate that compare to MF algorithm, our algorithm can identify large flow frequent items more accurately, and satisfies the need of actual measurement.

Keywords: *network measurement; massive data; data mining; frequent item; hash method*

1. Introduction

For the past few years, with the rapid development of the internet and the emergence of new network applications, the trend of computer network is developing towards high speed, large scale and complexity [1]. The notable feature is that the large amount of data generates, and the data packet arrival rate becomes higher. This will result in data packet processing unit time is getting shorter, and present a great challenge for storage capacity, processing power and transmission capability of the system. Currently on the high-speed backbone network, the average time to process each packet must be completed in the nanosecond [2]. For example, the average time to process each packet in OC-192 link is 32ns approximately, and it will be down to 8ns in OC-768 link. So, for mass data, how to mining frequent items in the stream network timely and accurately become a hot issue in network flow measurement area. However, the measurement method based on network flow has opened up a new way for high speed network traffic measurement. By merging the data packets to the corresponding flow, it reduced the amount of data greatly, and made the storage, processing and transmission of network data more easily.

In fact, not all applications need to know the information of each flow. Studies have shown that the heavy-tailed distribution of network traffic is an important feature of flow distribution, which means that a few flows have numerous packets account for most traffic, while the rest of the traffic consists of a large number of smaller flows which have few packets [3]. Usually, the flows which make larger contribution to the network traffic referred to as large flow. Most of the actual applications need only to know the information of large flows. Large flow identification is an important foundation of network monitoring, network management and network accounting. By identifying these large traffic flows, network managers can clearly grasp the distribution of network traffic,

ISSN: 2233-7857 IJFGCN

find denial of service attacks timely, control the traffic growth trend and allocate network resources and the link capacity reasonably. Therefore, using the limited hardware resources to focus on large flows becomes a better choice. So, large flow identification problem in high speed network monitoring area is equivalent to the frequent item mining problem in data stream which means that monitoring network flows and finding the frequency of a packet that contains some features is higher than the given threshold.

At present, there are three kinds of mining algorithms for network traffic: sampling, hash and counting [4]. Sampling method is to select the part of large data sets from data entry to approximate statistical frequent item in the entire dataset. Hashing method is to map the large range of data sets to a small range of the target data set, the data set used to generate frequent target item as an approximate solution for the entire data set. Counting method is to use a finite number of counting unit to carry out the cooperative count to maintain the entire data set frequent items.

2. Definition of Network Flow and Frequent Items

Flow can be defined as a call or connection of logic correspondence. Property values associated with the flow having a polymeric nature which reflects the events occurred between the start and stop time, such as source/destination address, packet count, byte count *etc.*

Frequent item is the flow which length exceeds a predefined threshold T for a period of time.

Network flow frequent items mining algorithm constraints: (1) Limited storage space. The storage space required to maintain summary statistics is much smaller than the size of the complete data set, usually only store small amounts of data entry summary information. (2) Real time processing. The processing time for each data item is very short, small and easy to operate. (3) Linear scan at a time. The order of arrival of each data item is completely random, therefore should read the data steam from first to last sequentially [5].

3. Multistage Filters Algorithm

Multistage Filters (MF) algorithm is put forward by Eestan[6]. The basic principle is that the algorithm uses a series of counting bloom filters and filter with multiple different hash spaces and hash functions, so as to be able to filter out all the frequent items. The executing procedure of MF algorithm is shown in Figure 1.

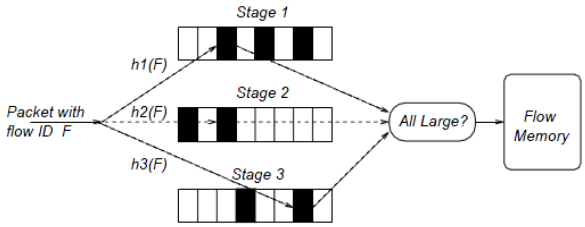

Figure 1. The Executing Procedure of MF Identify Frequent Items

MF algorithm is divided into multistage sequential filtration and parallel multistage filtration. In multistage sequential filtration mode independent hash functions are stored at each filter stage which is used to calculate the hash value for each packet corresponding to the flow tag, then add it to the corresponding space. All packets of the same flow are hashed to the same address space. Each address space has a counter to count the total length of the flow which hashed into this position and initialized to 0. The storage position which all counters are greater than T/d considered as a large space. Frequent items will be mapped to the large spaces. Parallel multistage filtration is the use of

multiple stages in parallel. Each stage has an independent hash function. When a packet which belongs to a flow identified as F arrives, each stage calculates its hash value, and find out the corresponding position in the hash table. The counter for this position is initialized to 0. This counter is incremented every time the hash packet length size. For the reason of all packets of the same flow are hashed to the same location, thus if the total packet length size of flow F exceeds the threshold of T, then the counter of each stage must also exceed the threshold T, At this moment can identify the large flow F.

However, there are some restrictions on the use of MF algorithm in large scale network links. Firstly, for the reason of traffic burstiness and massiveness, the number of packets will fill the whole hash space in a very short period of time, thereby resulting in false positive which means mistake multiple small flows for a frequent item. If take the method updated filters frequently and regularly, not only will lose the previous stage measurement information, but also bring great pressure to the system hardware. Secondly, MF algorithm can only identify the information of several overlarge flows. For the length of the relatively small flows, identification accuracy is not high. Thirdly, MF algorithm uses multiple filters which lead to the algorithm takes up more space.

4. Double Hash Algorithm

Because of the number of small flows in network account for the total flow volume of 80%-90% [7], so how to filter out most of the small stream becomes the key to identify the frequent item in the network. In this paper, we propose a double hash algorithm which composed by the Time-out Bloom filter (TBF) and Counting Bloom Filter (CBF). By using TBF to filter small streams and using CBF to Statistics and identify frequent items, our algorithm can identify large flow frequent items more accurately.

4.1. Time Bloom Filter

Time Bloom Filter (TBF) is modified by BF. Different from BF is that TBF maintains the packet's timestamp instead of 0 or 1. When a packet arrives, using k hash functions to calculate the packet tag and hash to the corresponding k units. TBF will determine whether the difference between the two timestamps of the k corresponding units is greater than the predetermined timeout period t_0. If there is any unit which time difference is larger than t_0, the packet will be discarded. Regardless of whether or not the packet is discarded, TBF will update the timestamp of k units to the current timestamp of arriving packet. Thus TBF update operation only updates timestamp, without the need to reset as frequently as BF. Figure 2 describes the process.

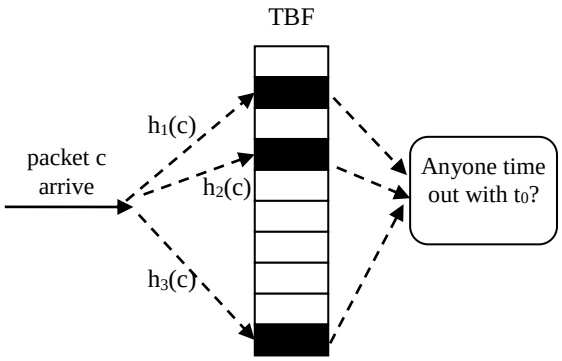

Figure 2. A TBF with 3 Hash Function

The researches show that the longer the flow length, the higher the frequency of the frequent items, and the average packet arrival time interval is smaller [8, 9].

Since all packets of the same flow are hashed to the same address space, for frequent items belong to a large flow, the time interval is less than the threshold value t_0, therefore as long as the hash space is not occupied by other flows within the time interval of two packets, this part of the packets will not be discarded. For a large number of small flow packets, because of the packets of different flows are hashed to different spaces, it will inevitably lead to the timestamp difference of some unit is larger than interval t_0. This part of the packets will be filtered. However, in extreme cases, when the hash spaces of the frequent items are frequently occupied by other flows, TBF will discard a large number of packets belong to some frequent items. This will reduce the recognition accuracy of frequent items. So we need to use another filtering technology to reduce the number of discarded packets.

4.2. Counting Bloom Filter

Counting Bloom filter is modified by standard BF. Unlike the BF, the m unit of CBF maintains a counter rather than a bit. In this way, CBF not only has the add operation and query operation just as BF, but also has the delete operation. By using *k* independent hash functions, the add operation is to put the counter plus 1 for *k* corresponding storage space. Delete operation is to subtract a value from the counter for *k* units.

4.3. Double Hash Algorithm

Double hash algorithm is composed by the TBF and CBF. The basic structure is shown in Figure 3. Algorithm allocates a memory size of M unit of storage space A1 and A2 respectively for TBF and CBF module. Each unit of TBF module maintains packets arrive timestamp, and each unit of CBF module maintains a counter. The initial value of all units is set to 0. TBF and CBF use different hash functions of hash function set $H_1, H_2 \ldots\ldots H_k$ to function for the arrival of packets, making mapped into memory space A1 and A2.

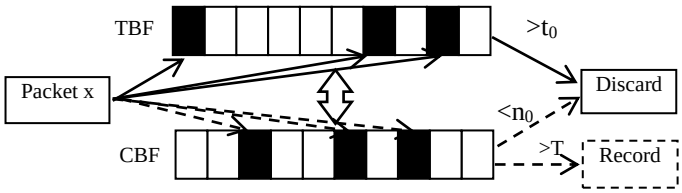

Figure 3. Double Hash Algorithm Process Schematics

Figure 4 shows the basic process of the double hash algorithm: When a packet arrives, handled by TBF firstly. TBF determine whether the difference of timestamp between the two of the *k* corresponding to units is greater than the predetermined timeout period t_0. If there is a unit of time difference greater than t_0, and the hash position in the CBF module is less than n_0, then the packet will be discarded, otherwise will be updated by CBF module. The timestamp of the *k* unit in the TBF will be updated regardless of whether the time interval is greater than t_0. CBF module adds 1 to the counter of the *k* units. If all of the counters of the *k* unit are greater than the preset threshold value n_0, the stream is a large flow frequent item. If any cell count is less than n_0, the stream may be a small stream, or it may be the first part of the large current frequent item. Meanwhile, if the flow in the TBF timeout, it will be discarded also. In order to reduce the number of small streams to the same location, CBF counter minus one at corresponding position while discard the packet data item. When the counters of *k* hash positions of CBF module are greater than a

preset threshold T, algorithm will establish a new large flow frequent item records, and the calculation of the *k* location minus T.

```
x=Packet Label;
initialize (TBF,CBF)            //initialize TBF and CBF
compute(h(x),g(x))          //TBF and CBF hash function value
if (any location of TBF[hi(x)]i=1..k > t0) { //if  TBF time out
      add(min(CBF[hi(x)]i=1..k));            //update CBF
       if  (any location of CBF[hi(x)]i=1..k<n0){
       //if any  counter of CBF unit is less than n0
            discard(x);                       //discard this packet
            sub((CBF[hi(x)]i=1..k))           //CBF count minus 1
        }
        else
             // all counters of CBF are greater than n0
            add(min(CBF[hi(x)]i=1..k));
             //update CBF
else{
             //if TBF no time out
            add(min(CBF[hi(x)]i=1..k));
             // update CBF
}
update timestamp (TBF[hi(x)]i=1..k);
// updated TBF timestamps regardless whether discard or not
```

Figure 4. Double Hash Algorithm

From the above analysis we can know that the algorithm discard packets only when TBF time out and the counter of CBF is less than n_0, which means the algorithm only discard small flow packets or the time out packets of first n_0 packets of large flow frequent item. By doing so, most of the large flow frequent items are counted statistically, so as to identify the frequent items of network flow.

4.4. Performance Evaluation

Similar to BF, our algorithm also will bring a false positive error probability which is caused by TBF and CBF. Giving a set S={ $x_1, x_2 \ldots\ldots x_n$ }, the probability of hash position is empty in TBF is $p' = (1-1/m))^k \approx e^{-k/m}$, then $f_{TBF} = (1-p')^k \approx (1-p)^k = (1-e^{-k/m})^k$ .For single CBF, the probability of m counters is increased n_0 times is $P\{c(i) = n_0\} = \binom{nk}{n_0}\left(\frac{1}{m}\right)^{n_0}\left(1-\frac{1}{m}\right)^{nk-n_0} \le \binom{nk}{n_0}\frac{1}{m^{n_0}} \le \left(\frac{enk}{n_0 m}\right)^{n_0}$ .When the algorithm used *i* CBF, so the false positive error probability of CBF is $f_{CBF} = \left(\frac{enk}{n_0 m}\right)^{n_0 i}$ .Because the condition of our algorithm sample the packets is that satisfy both the requirement of TBF and CBF, so the false positive error probability of our algorithm is $f = f_{TBF} * f_{CBF} = (1-e^{-k/m})^k \left(\frac{enk}{n_0 m}\right)^{n_0 i}$.

Since the algorithm uses the same hash function set, the cost of TBF and CBF visit hash memory is $O(k)$. To update the *k* timestamp of TBF and update the *k* counters of CBF cost $O(k)$. For the reason of flow tag is unique, and the storage location of the flow has a strong correlation with the hash functions, so the cost of our algorithm is $O(3k + \frac{kn}{c})$, in which c is a constant.

5. Experimental Analysis

In order to verify the validity of the proposed algorithm in this paper, we use the trace collected from CAIDA to simulation test [10, 11]. There are a total of 6187376 packets and 68367original flows. The algorithm uses k = 6 hash functions. Storage space m= 2^{16} =65536. The hash space is [0...65535]. Figure 5 shows the distribution of original flows. As can be seen from the figure the distribution of network flow demonstrates heavy-tailed distribution.

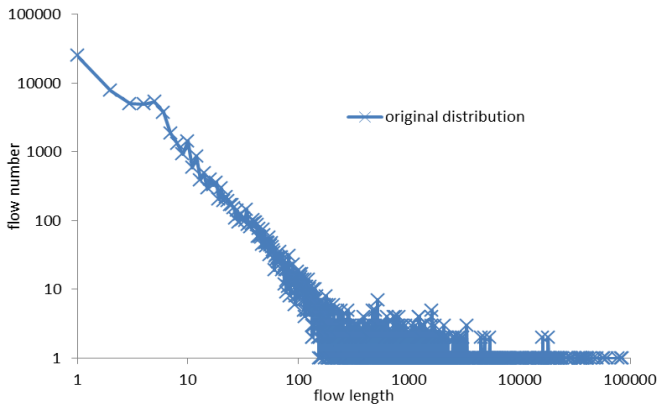

Figure 5. Original Flows Distribution

The parameters which need to configure in double hash algorithm are described as follow: TBF timeout interval t_0, CBF filtering threshold n_0, frequent item packet threshold T. We use absolute error e_r as measurement criteria to verify the measurement accuracy, where $e_r = \frac{|N - N'|}{N} \times 100\%$, which N indicates the actual number of frequent item, and N' denotes the number of frequent item that we identified in the experiment.

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

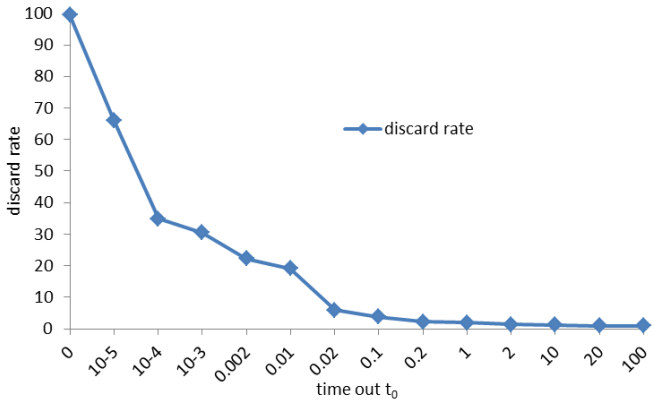

Figure 6. Relationship between Packet Discard Rate and Timeout Interval Threshold T_0 in TBF

Figure 7 displays CBF takes a fixed value threshold n_0, the impact on the measurement accuracy when time out threshold t_0 changed.

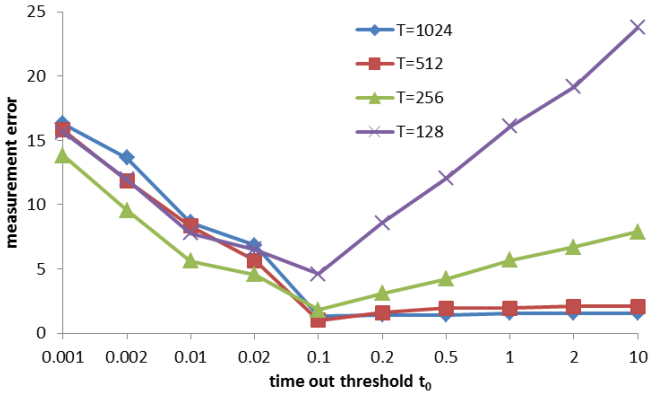

Figure 7. Accuracy Effect between T0 and Large Flow Frequent Items Threshold T

The measurement error decreases with the increase of t_0, as shown in Figure 7, when the threshold is less than 0.1 seconds, the measurement error decreases with the increase of t_0. This is because most of the packets are discarded when t_0 takes a small value. However, when t_0 is bigger than 0.1 seconds, measurement error increases with the increase of t_0. This is because when the t_0 takes a large value, the number of discarded packets is gradually reduced, most of the small streams are not filtered out, so as to increase the false positive rate of the algorithm, and result in error increases.

Figure 8 displays when TBF takes a fixed value, the measurement accuracy and discard packet ratio along with n_0 changed.

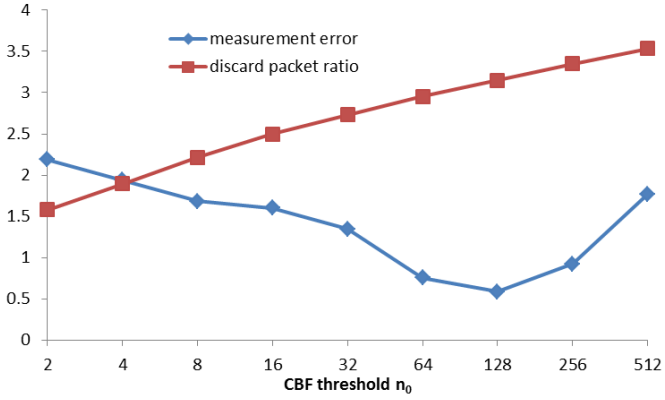

Figure 8. Measurement Accuracy and Discard Packet Ratio Influence of N_0

As Figure 8 shown, with the increase of the value of n_0, the discard packet ratio increases linearly. This is due to when n_0 increases, the packets that belong to the flows which length are less than n_0 will be filtered out. Meanwhile, when n_0 increases to a certain value, measurement error will increase gradually. This is because of part of the packets that belong to large flows are discarded when n_0 takes a larger value. Accordingly increase the measurement error.

Figure 9 displays comparison between double hash algorithm when, $t_0 = 0.1$, $n_0 = 16$ and MF algorithm in frequent items mining. As can be seen, the measurement accuracy of our algorithm is higher than the MF algorithm both in frequent item measurement and frequent item packets number measurement. Furthermore, when the large flow frequent item T takes a small value, our algorithm has the more obvious advantages. This is due to the double hash algorithm can filter out most of the small flows, so as to reduce the effect of small flow on the large flow mapping space, and reduce the false positive rate of the algorithm.

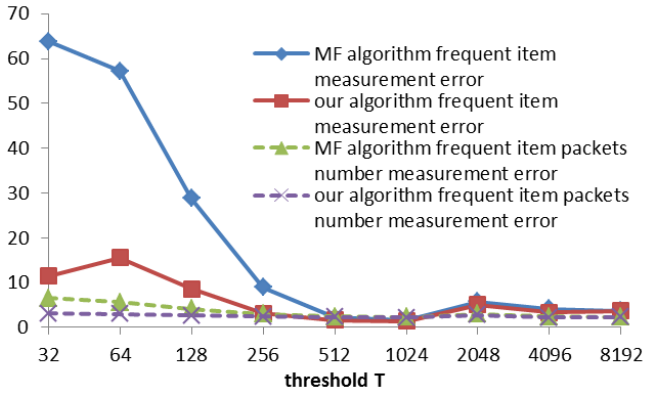

Figure 9. Comparison between our Algorithm and MF Algorithm in Frequent Items Mining

Acknowledgements

The research work was supported by National Natural Science Foundation of China (61472137), and the Fundamental Research Funds for the Central Universities (3142014085, 3142014100, 3142014125, 3142015022, 3142013098, and 3142013070).

References

[1] A. P. Zhou, G. Cheng and X. J. Guo, "High-Speed network traffic measurement method", Journal of Software, vol. 1, no. 25, **(2014)**.
[2] H. Wu, J. Gong and W. Yang, "Algorithm based on double counter bloom filter for large flows identification", Journal of Software, vol. 5, no. 21, **(2010)**.
[3] J. Xia and X. Zhao, "Frequent items mining algorithm over network flows based on time and flow length constraints", Journal of University of Science and Technology of China, vol. 10, no. 43, **(2013)**
[4] M. Tatsuya, U. Masato and K. Ryoichi. "Identifying elephant flows through periodically sampled packets", Proceedings of ACM SIGCOMM, USA, **(2004)**.
[5] N. G. Duffield, C. Lund and M. Thorup, "Estimating flow distributions from sampled flow statistics", Proceedings of the ACM SIGCOMM Conference on Applications, Karlsruhe, Germany, **(2003)**.
[6] C. Estan and G. Varghese, "New directions in traffic measurement and accounting", ACM Transactions on Computer Systems, vol. 3, no. 21, **(2003)**.
[7] G. Cheng and Y. N. Tang, "Estimation algorithms of the flow number from sampled packets on approximate approaches", Journal of Software, vol. 2, no. 24, **(2013)**.
[8] Z. Zhang and B. Q. Wang, "Traffic measurement algorithm based on least recent used and Bloom filter", Journal on Communications, vol. 1, no. 34, **(2013)**.
[9] H. Dong, G. L. Sun and D. D. Li, "Application Layer Traffic Classification Based on Link Homophily. Journal of Harbin University of Science and Technology, vol. 4, no. 18, **(2013)**.
[10] F. Y. Wang and S. Q. Guo, "A method of extracting heavy-hitter flows efficiently", Journal of Computer Research and Development, vol. 4, no. 50, **(2013)**.
[11] H. Wang and Z. H. Gong, "Hits and Holds: Two algorithms for identifying the large flows", Journal of Software, vol. 6, no. 21, **(2010)**.

