A New Algorithm for Long Flows Statistics—MGCBF[1]

ZHOU Mingzhong[1,2], GONG Jian[1,2], DING Wei[1,2], CHENG Guang[1,2]

[1] Department of Computer Science, Southeast Univ., Jiangsu, Nanjing 210096 China
[2]Jiangsu Province Key Laboratory of Computer Networking Technology
{mzzhou, jgong, wding, gcheng}@njnet.edu.cn

Abstract. As a major work of network traffic measurement, long flows' identification and characteristics analysis become more and more important because long flows take main traffic payload of network. This paper presents a novel long flows' counting and information maintenance algorithm called Multi-granularity Counting Bloom Filter (MGCBF) based on the analysis of long flows' distribution and characteristics in the Internet. Using a little fix memory, the MGCBF maintains the counters for all incoming flows with small error probability, and keeps information of long flows whose length is bigger than an optional threshold than users can set by a expanding data structure. Based on this, this paper builds up the model of long flows' information statistics. This paper also analyzes the space used, calculation complexity and error probability of this model. The experiment uses this model on the CERNET TRACEs, which indicates that the MGCBF algorithm can reduce the resource usage in counting flows and flows information maintenance dramatically with losing little measurement's accuracy.

1 Introduction

Flow-based measurement is wildly used in network usages just like accounting, bandwidth measurement and network security, etc. A flow is defined as a stream of packets subject to flow specification and timeout. The flow definition can be changed according to its usage, but recent studies show that a very small percentage of flows carry the majority of the packets and bytes [1][2][3][8] regardless of its definition. It is very important for improving the network performance that finding out these heavy-hitter flows (called long flows). Some network usages can also take advantage of it.

In the following of this section, we will introduce the recent related works on long flows identification and statistics, and indicate the main contributions of the Multi-granularity Counting Bloom Filter (MGCBF) algorithm in flow identification briefly. In the next section, the algorithm is presented in detail, and its performance and error

[1] This research is partially support by the National Basic Research Program (called 973 Program), No. 2003CB314803; Jiangsu Province Key Laboratory of Network and Information Security BM2003201 and the Key Project of Chinese Ministry of Education under Grant No.105084

probability are anatomized. And then the optimizations are also expressed. It is proved that the error probability can be controlled through parameters adjusting. In Section 3, some experiments are employed to illustrate and evaluate the performance and error probability of MGCBF with traditional hash method and CBF method based on the TRACEs from CERNET. At last, we discuss the usages of MGCBF in other domains and present the future works.

1.1 Related works

As the increasing needs of flow-based traffic measurement, new methods satisfying variant applications are emerging in endlessly. Long flows identification and counting are also wildly studied as one of its main branch [2][3][4][7]. There are several differences when long flows are dealt with for the different applications.

As most widely used way of flow identification, the sampling flow method presented by IETF RTFM group can gather total or parts of flow information that transmitting by the router. But sampling is widely used in these supports because of the resource restriction of routers in flow identification and exporting according to the RTFM criterions. Sampling satisfies the needs of performance by losing the precision. A. Shaikh, J. Rexford and K. G. Shin[3] applied a method to realize the load balance by keeping the information of every flow, and judged the flow belonging to long or not by the packet number it arrived in a fixed time unit. This is a direct but not efficient way. Smitha, I. Kim, and A. L. N.Reddy.[2] provided an algorithm called Least Recent Used (LRU). This algorithm can be used to identify long existing and heavy-hitter flows for load balance, but it can only maintain flows information in a short time and must refresh frequently, and so long flows with large duration will not be kept. C. Estan and G. Varghese [4] used two methods to find out long flows: *sample and hold* and *multistage filters*. And both of them resolve the problem of getting and keeping flows information at packet sampling efficiently, but they are only used to find out and keep the information of very large flows in the network, which takes a very large ratio of total traffic (i.e. the flow volume is larger than 0.1% of total traffic). A. Kumar, J. Xu, et al.[7] provide a algorithm called SCBF for flow counting, which use limited resource to store all flows' length information with a little errors by sampling. This algorithm applied maximum likelihood estimation (MLE) and mean value estimation (MVE) to estimate the length of every flow. C. Estan, G. Varghese and M. Fisk [11] described bitmap algorithm to take count of the flows with different length using little storage resource. A,Kumar, M.Sung. et al. [9] depicted a method for flow counting and estimation in high-speed links using SRAM and hash structure applied in hardware. But all those algorithms and methods do not keep the detail information for flows, which can't satisfy the needs of special applications (i.e. load balance, traffic accounting).

1.2 Main contribution

This paper propose a long flows counting and identification algorithm called Multi-granularity Counting Bloom Filter (MGCBF) based on standard bloom filter according to the study of flow distribution and characteristics in detail. This algorithm can store the selected flows information satisfying the needs of users, and relax the conflicting of performance and precision in flow measurement. Minimizing the system resource usage with little precision reduction. The fitness and advantages of this algorithm are described as following:

(1) This algorithm counts and maintains flows information whose flow length is bigger than a threshold, and resolves the problem of most long flows statistical algorithm that can't maintain flows information. It has the features of adaptive capacity and expansibility because the threshold can be set according to different usages.
(2) MGCBF takes advantage of the heavy-tailed distribution of flow length in Internet backbone, and uses a new structure to store flows information which can save the resource dramatically;
(3) This algorithm is surpassed by few in performance because it can't keep the flows information whose length less than threshold which can reduce the resource usage to its best. Its resource usage is much less than prevalent algorithms [1][15] that keep the flow information.

2 Flow identification and statistics based on MGCBF

This Section provides the prototype of MGCBF based on introducing the standard bloom filter. And it is introduced following that the improving model of MGCBF can decrease its complexity and increase its precision. In the end of this section we will analyze the performance and error rate of this algorithm.

2.1 MGCBF prototype

The MGCBF employs a serial of CBFs (MGCBF=$\{cbf_0, cbf_1, \ldots, cbf_{h-1}\}$) which use different count spaces ($C=\{1, c_1, c_2, \ldots, c_{h-1}\}$) to count the frequency of different items in the set. This algorithm is mainly fit to "iceberg query" expressed in [14], which is often desirable to identify from a very long sequence of symbols coming from a large alphabet those symbols whose frequency is above a given threshold. Such analysis is sometimes called *hot list analysis* also. The time and space complexity are main problem wanted to solve because of the long length of the sequence [4][8][10][14]. The MGCBF provided by this paper operates iceberg query the sequence which items count follows heavy-tailed distribution. The prototype of this algorithm is introduced as following:

1) When an item *x* wanted to add into MGCBF, the counters at positions $h_1^0(x)$, $h_2^0(x)$, …, $h_{k0}^0(x)$ in vector V_0 increase 1. (V_0 is the vector of MGCBF's first

CBF, cbf_0. h_1, h_2, …, h_{k0} are the hash functions in cbf_0. Without the loss of generality, we suppose $h_1^0(x) \leqslant h_2^0(x) \leqslant \ldots \leqslant h_{k0}^0(x)$)

2) Then check the value $h_1^0(x)$. If $h_1^0(x)=c_1$, the counters at positions $h_1^0(x)$, $h_2^0(x)$, …, $h_{k0}^0(x)$ decrease c_1, then values in the counters change to 0, $h_2^0(x)$-c_1, …, $h_{k0}^0(x)$-c_1; the counters at positions $h_1^1(x)$,$h_2^1(x)$, …, $h_{k1}^1(x)$ in V_1 which is the vector of cbf_1, that means $h_1^1(x)+1$, $h_2^1(x)+1$, …, $h_{k1}^1(x)+1$.
3) And then check the value $h_1^1(x)$. If $h_1^1(x)=c_2$, we operate the same action as 2) in cbf_1 and cbf_2. the check is done until cbf_h. If we suppose the set of counters' minimum value which is set by an item x is $M(x)=\{min_0(x),min_1(x), \ldots, min_{h-1}(x)\}$, then the frequency of x in S is:

$$Counter(x) = min_0(x)+min_1(x)*c_1+ \ldots + min_{h-1}(x)* \prod_{i=1}^{h-1} c_i \circ \quad (1)$$

The MGCBF algorithm is based on the supposition that frequencies of different items in the set need to be tested, ***S,*** follow heavy-tailed distribution. Heavy-tailed distribution means most of the items frequencies are low while a few items frequencies are very high for their total number take a large percents in the set ***S***. The MGCBF applies multi-stages to deal with the items in the set, which can put low frequency items into lower CBFs. And the main count payload of high frequency items is taken by the higher CBFs in the MGCBF. The vector spaces of CBFs in MGCBF decrease exponentially as the stages increase because of the heavy-tailed distribution of items frequency. This new structure can save space dramatically comparing with single CBF structure. But it introduces counting complexity and error rates. We will analyze the space usage; compute complexity and error rates of this algorithm in the next section.

The flow length follows heavy-tailed distribution in Internet [2][3][4], that means very few long flows taking most network traffic. And so it is more efficiency that we use the MGCBF algorithm to replace the traditional counting methods.

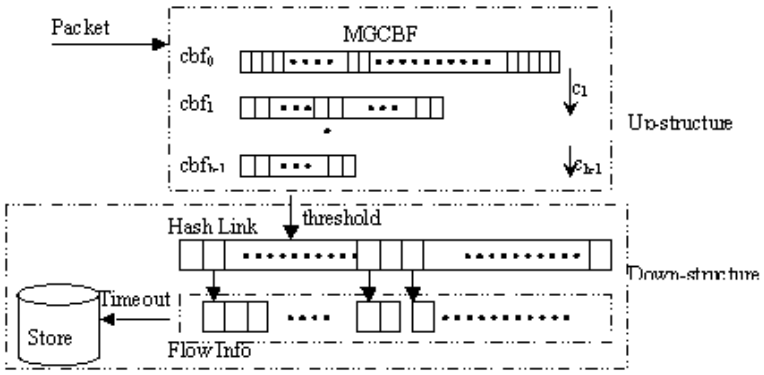

Fig. 1. Long flow information statistics model based on MGCBF

Fig. 1 illustrates the long flow information statistics model based on MGCBF. The data structure of MGCBF in the upside of this figure is used to maintain the packet number of every active flow, and the downside of this figure describes the data struc-

ture used to keep the flow information of needing to keep by a hash table and a link table. The working flows of this figure can be expressed as following:

1) When a packet incomes, we use MGCBF to maintain it, and we can get the packet number of flow which the packet is belonged to.
2) When some flow's packet number is equal to threshold, we first decrease the value of threshold in the flow's counting positions of the MGCBF structure, and then use a hash function to store the flow information to a structure made up of with a hash table and a link table. If the flow information is maintained in this structure, then it will be refreshed, else new flow information will be created.
3) The flow information in the down structure is maintained by a timeout, and the ended flows are driven out from this structure by periodical scanning. Those ended flows will be stored in permanent storage instruments for long time storage or other usages.

Using a prearranged ***threshold*** for flow length, this algorithm can maintain all kinds of flows information, whose length is bigger than 1 packet, for different usages. With reducing of the threshold value, the stages of MGCBF are reducing, and this will reduce the spending of the algorithm. When the threshold is reduced to a value small enough (i.e. threshold$\leqslant$10), the MGCBF will degrade to a CBF. And so the MGCBF has good expansibility.

We depict them with two threads in the pseudo-code. The related parameters used in this pseudo-code are explained as following:

$x_1,\ldots,x_N$: the incoming packets sequence

$cbf_0, \ldots, cbf_{h-1}$: the CBFs which construct the MGCBF, h is number of stages;

$C=\{c_0,c_1,c_2, \ldots, c_{h-1}\}$: the space of counting unit in every stage;

$M(x_i)=\{min_0(x_i),min_1(x_i), \ldots, min_{h-1}(x_i)\}$: the serial made up of minimum count of the flow x_i in every stage;

Count(x_i): the packet number of xi in the MGCBF;

HashLink(x_i): information of the flow x_i which is stored by the hash table and the link table;

Thd: the threshold of flow length;

To: the timeout of flows.

The insertion algorithm for MGCBF is show as following:

Thread1: MGCBF maintenance
Initiate(MGCBF); //set every counter of each stage CBF to zero;
for i:=1 to N, do
  cbf_0.add(x_i, 1); //add the x_i into the stage 0 CBF
  for j:=1 to h, do
    if($min_{j-1}(x_i)==c_j$)
      cbf_{j-1}.remove(x_i, c_j);
      cbf_1.add(x_i, 1);
    end;
    count(x_i):= $min_0(x_i)+min_1(x_i)*c_1+ \ldots + min_{h-1}(x_i)*\prod_{k=1}^{h-1} c_k$;
    if(count(x_i)>Thd)
      MGCBF.**refresh**(Flow(x_i), count(x_i));
      Hashlink.**refresh**(Flow(x_i), count(x_i));

```
    end;
  end.

Thread2: Hashlink.scan
for i:=1 to H, do
  if(Hashlink[i]!=NULL)
    tmp=Hashlink (i)
    while(tmp !=NULL)
      if(tmp .eg. TO)
        tmp.store;
        tmp := tmp->next;
      end;
    end;
  end;
end.                ■
```

2.3 Optimizations

Because the confliction of hash function, the error rates of MGCBF inevitable as the number of different items in the set is very large while the volumes of MGCBF's vectors are small relatively. This section will introduce two methods to improve the performance and reduce the error rates of this algorithm.

(1) Periodical refreshing

When the items number of a set are very large (i.e. the packets number measured in some point in 24 hours), the volume of the set space V is too large to be stored in the memory of a measurement system as using MGCBF to analyzing this set. Even though the measurement system can maintain this huge structure, it can't be applied by the too low value of cost performance.

The set $\boldsymbol{S}$ is divided into several subsets using a method called periodical refreshing, $S=\{S_1,S_2,\ldots,S_\gamma\}$, and that means the original set space is divided γ equal subspace. For every subset S_i, we use the MGCBF to calculate and statistical analysis. When the first subset is finished, we initialize the MGCBF structure for next subset but unchanged the down structure in Fig. 1. Because in flow measurement the packets define every subset in fixed period gotten from the network, this method is called ***periodical refreshing***. The cost of this method is splitting the relationship between S_i and S_{i+1}, which may introduce errors to the measurement and increase the costs of calculation. The error analysis and calculation cost are illustrated in Sect. 2.4.

(2) Recurring minimum

The error of MGCBF is mainly from two aspects: 1) the ***original errors*** coming from Bloom Filter, denoted by $\boldsymbol{E_b}$; 2) ***the progressive counting errors,*** denoted by $\boldsymbol{E_c}$.

About the original errors coming from Bloom Filter, B.Bloom illustrated them in detail as he introduced his algorithm in [5]. It used k hash functions to insert n keys from the set S at random to the array of size m, the probability error that Bloom Filter in some situation is $E = (1 - e^{-kn/m})^k$. When $k=\ln 2 \times m/n$, in which case the error

rate is $(1/2)^k = (0.6185)^{m/n}$, the right-hand expression is minimized. Thus, we can control the error rate by estimating the value m and then adjusting the value of k and n. The following is the probability error of Bloom Filter when the input and the parameters of Bloom Filter are fixed.

m/n=6 k=4 E=0.0561
m/n=8 k=6 E=0.0215
m/n=12 k=8 E=0.00314
m/n=16 k=11 E=0.000458.

The progressive counting errors can be illustrated by the following example. Firstly, we set the item q has a serial of counting positions in CBF's vector, $C_1=\{c_1(q), c_2(q),\ldots, c_k(q)\}$; Secondly, we suppose that there are several items $X=\{x_1\ x_2, \ldots, x_p\}$ whose counting positions comprise the set C_2; Then we can detect the changing of C_1 even though it is not changed by item q when $C_1 \subset C_2$ and the items in set X all changed. That is progressive counting errors. A method called recurring minimum is introduced by [6] to reduce the probability of this errors, it is also fit for MGCBF.

Recurring minimum method uses an affiliated CBF called CBF_t to reduce the progressing counting errors. When a new item x_i incomes, if it has recurring minimum among its counts, let it input into the original CBF called CBF_p; Else if it has only one minimum, we put it into the CBF_t. If let $P(R_x)$ as the fraction of cases with recurring minimum, $P(E_x|R_x)$ as fraction of estimation errors in those cases, E^s as the calculated error for the CBF_t. The next column shows the expected error ratio which is calculated by

$$E_{RM} = P(R_x)P(E_x \mid R_x) + (1 - P(R_x))E^s$$

An example presented in [6] shows that $E_{RM}<E/18$ when the parameters are set as following: k=5, n=1000, m/n= k*ln(2)=0.7k, $m^s=m/2$ (m^s is the vector space of the CBF_t) It is to say that the error probability can be reduce to 1/18 by using the recurring minimum method, which only increase 1/2 storage space and 1- $P(R_x)$ calculating costing.

In realization of Long flow information statistics model using the MGCBF algorithm, only the high-stage CBF using recurring minimum is the balancing the counting error influence and the compute complexity. There are three reasons about this choice. Firstly, the high-stage CBF error reducing the correctness of countering result more heavy; And secondly, the vector space of high-stage CBF is much smaller than the low-stage which can reduce the cost of maintaining; Thirdly, the low-stage CBF has big volume of vector space while algorithm is little sensitive with its counter errors.

2.4 Performance analysis and error estimation of MGCBF

In this section, we evaluate the algorithm efficiency and estimate the probability errors of the long flow information statistical model by introducing the MGCBF algorithm used in this model. Because the MGCBF is used for long flow counting in this paper, the performance analysis and error rate estimation are all based on the long flow information statistical model. The maximal counting value in MGCBF is the

threshold denoted long flow because when the counting value of some flow is added to the threshold, it will be submitted to other structure to maintain which is illustrated in Fig. 1. down-structure. But it can be proved that this above analysis will not lose its generality if only the items sequence in the set needed checking follows heavy-tailed distribution.

(1) Performance

Firstly, we suppose that flows information whose packet number bigger than 1000 (***Threshold***=1000) is wanted to gather. Considering the performance and costing, we let the MGCBF two stages (h=2), and the second stage CBF uses recurring minimum method to reduce the error probability. It is proved that the flow length follows heavy-tailed distribution in [15]. The flows whose length is smaller than 16 packets take above 90 percents of total flows. And so if the first stage CBF used the counter whose maximum value is 16 (c_1=16), the second stage CBF vector V_2 can be as small as 1/10 of the first stage CBF vector V_1 (because $n_2 \leqslant n_1/10$, we can set $m_2=m_1/10$) without introducing more errors. Referring to § 2.2 equation (1) and the parameters Threshold=1000, c_1=16, we can calculate the value of c_2, $MAX(min_1)=(Threshold-min_0)/c_1 = (1000-16)/16=61.5<64=2^6$, that means we can set c_2=64. Referring to the suggestion L. Fan, P. Cao, et al. proposed in [12], when a serial of unrepeated items insert into one CBF, if the counter volume of this CBF takes the value 16 that means its length is 4 bit ($16=2^4$), the possibility of counter overflowing by adding can be ignored.

$$Pr(max(c) \geqslant 16) \leqslant 1.37 \times 10^{-15} \times m$$

the left-side of inequation means the possibility of counter overflowing, m is the space of vector. And then we can define the counter volume of first-stage CBF is log(16)+4=8 bit, and the counter volume of second-stage CBF is $\log(2^6)+4=10$ bit, then we can calculate the space of MGCBF structure in the next equation.

$$M_{MGCBF}=8 \times m_1 + 10 \times m_2 + 1/2 \times (10 \times m_2)=9.5m_1$$

While using traditional CBF to store the flow number information, the space needed can be calculated as following:

$$M_{CBF} = (\log(1000)+4)m_1=14m_1$$

When we set threshold 1000, MGCBF can save 1/3 storage space than CBF; and when the threshold changes to 65536, the space saving ratio will change to 1/2; when the threshold value keeps increasing, the space saving ratio keeps decreasing with introducing little compute complexity and error ratio. The model in this paper can only maintain the information of flows whose length is equal with or bigger than the threshold. The storage space is reducing rapidly as the threshold increasing because of the heavy-tailed distribution of flow length.

We suppose that cbf_1 will refresh every c_1 packets coming at means, but actually the incoming packet number will much bigger than c_1 because most flow lengths are smaller than this value which can induce the refreshing of the cbf_1. We denote τ as the time for inserting and/or extracting in CBF, and then we can deduce that the mean calculating time for every packet in this MGCBF fitting with two-stages is $\tau_{MGCBF} < \tau_0 + 1/c_1 \times \tau_1$. For assuring the precision of CBF in high-stage, we set the parameter $k_1=\alpha k_0 > k_0$; And the calculation complexity introduced by using recurring mini-

mum in cbf_1 is about 20% [6], $\tau_1=1.2\alpha\tau_0$[2]. Then we can deduce the calculation costing of every packet in MGCBF is $\tau_{MGCBF}<(1+1.2\alpha/c_1)\tau_0$. When the parameters are set as following $c_1=16$, $\alpha=4/3$, then can get $\tau_{MGCBF}<1.1\tau_0$, that means that every packet increases only 1/10 calculation costing meanly. It is also can be proved that the increasing scope of calculation costing decreases with the stages of MGCBF increasing.

(2) Error analysis

The errors in flow statistical model used MGCBF can be divided into two types: (1) the error of MGCBF; (2) the error introduced by periodical refreshing.

The error ratio of the MGCBF algorithm can calculate in different stages: cbf_0 set as E_0, cbf_1 set as E_1. It only introduces error ratio E_1 in the cbf_1 of this MGCBF compared with traditional CBF. And because recurring minimum are used in cbf_1, the error probability will reduce to 1/18 according to [18]. Keeping to adjust the parameters m^s and k, we can make the error ratio $E_2<<E_1$. The error ratio increase about 1/288 when we compare the error ratio of this MGCBF with that of CBF in the same scale because the total error ratio of this MGCBF can be denoted as this equation: E_0+1/c_1*E_1. Such small increasing can be omitted when we gauge the precision of MGCBF. And so we can conclude that the MGCBF's first stage (cbf_0) error probability determines MGCBF error probability. The error probability of high stages can be dismissal.

The measurement errors caused by the periodical refreshing are mainly from two parts: (1) the lost of partial packet information in long flows, denoted $\boldsymbol{E_p}$; (2) the truncation of the long flows whose lengths bigger than threshold but not reaching the threshold, denoted as $\boldsymbol{E_t}$. The first kind errors E_p can be eliminated by scanning the whole structure of MGCBF, and finding out the flows rudimental packets counting information in this structure, and refreshing the flows information structure. But this method will not be implemented and this type of errors are tolerated for the balance of error rates and calculating costing. And the second errors E_t may cause some long flows or partial packets of long flows being discarded. If we set the long flows incoming rates is not changing badly, the discarded long flows number is determined by the *threshold*, *flows' rate* and *flows' timeout value.* The long flows whose rate is v takes η percents of total flows, the flows number in time unit is s', flows timeout is To, then we can deduce the probability flows number that is influenced:

$$s'_f = \sum_{i=1}^{n} \int_0^{To} \eta_i(t) s' dt$$

$\eta_i(t)$ is the percents of flows in total flows whose rate is $v_i<$threshold/(To-t). For most long flows, $\eta_i(t)$ is a small value at To-t<<To, and else $\eta_i(t)$ is almost 0 when this condition is changed. And so we can conclude that s'_f <<(s'*To). In Sect. 3, the experiments using different TRACEs prove that the second type errors caused by periodical refreshing cannot be larger than 1%. The periodical refreshing method

[2]The calculation time is as direct proportion as the hash function number of CBF *k* because most calculation costing in every operation in CBF is hash calculation. We can deduce this equation: $k_0/k_1=\tau_0/\tau_1$

proposed by this paper can reduce the storage space dramatically by little compute costing with losing of little flow identification precision.

3 Experiments

The datasets used by the experiments are the TRACEs gathered from the CERNET backbone in different time: CERNET1 and CERNET2.

Table 1. Flow length distribution of the TRACEs

	Total flows number	Long flows number (threshold =1000)	percentage
CERNET1	17164783	30316	0.17%
CERNET2	59987620	80850	0.13%

The statistics result shows that the long flows take 0.1-0.2 percentage of total flow (the former is 0.17 percentage, and the later is 0.13 percentage), and so they both follow the distribution of heavy-tailed.

Firstly we should determine the parameters of MGCBF by estimating the active flows number in a fixed period in network. In general, the mean of flow length in a network is a stable value at normal. The measurement result of us shows that its value is about 20 in CERNET applying the 5-tuples and 64-second timeout flow definition. And so we can determinate the space of MGCBF's vector $\boldsymbol{V}$ by incoming packets number. The level of MGCBF is 2, and the other parameters are following: threshold=1000, $m_1=16*10^6$, $m_2=16*10^5$, $k_1=k_2=6$. The periodical refreshing applied in MGCBF, we set the periodical refreshing interval as 3.3 minters when the pps (packets per second) is about $200*10^3$, that means the packet number is $n=m_1/8*20=40*10^6$. At the second-stage of MGCBF, we applied recurring minimum to reduce the measurement errors further which space is set as $m^s=m_2/2$.

Then we use three methods: 1) classical flow information (hash table + link table), 2) CBF, and 3) MGCBF based measurement to analyze and study the long flows whose packet number than 1000 in two TRACEs as shown in Fig. 2. And the CBF method is simplified from MGCBF, use the first-stage CBF but its vector space is expanded to $m=128*10^6$ which can store the two hours flows with the 2% error probability. The counter' unit length is expanded to 14 bits to store the flows long enough without losing precision.

The experiment results show, the error probability difference between MGCBF and CBF in the long flow information maintenance and flow number storage is no more than 1% (CERNET1 is 0.26%, and CERNET2 is 0.81%). It is periodical refreshing and second-stage CBF cause this difference, for the latter we can estimate it by calculating, and then we can get the error caused by periodical refreshing. The error probability difference between MGCBF and classical flow information method is about 2% (CERNET1 is 1.6%, and CERNET2 is 1.3%), and then we can estimate the error probability caused by the first-stage CBF in MGCBF.

Intuitively, the classical flow maintenance method will take largest system resource. But actually, It is method used CBF takes more space in vector $\boldsymbol{V}$ for main-

taining the flow information and it will increase with the increasing of incoming packet number. The active flows number in every period is about 700,000, mean space for maintaining every flow is about 300 bits. Using classical flow maintenance method, it needs at least $29.05*10^6$ bytes to store the flow information; Using CBF method, the space comprises two parts: CBF vector space and flow information maintenance space, and it needs $224.03*10^6$ bytes; While using MGCBF method, it only needs $19.32*10^6$ for storing. The difference in structure between MGCBF and CBF makes MGCBF save only 34% space.[3] The large difference in space between the methods we applied mainly caused by the periodical refreshing, which decomposes the dataset into several subsets and tackles them separately. Because calculation complexity of hashing is much less than that of numerical comparison, CBF and MGCBF are better than traditional flow maintenance method in calculation complexity. And the latter's advantage will be outstanding when the distribution of dataset follows heavy-tailed.

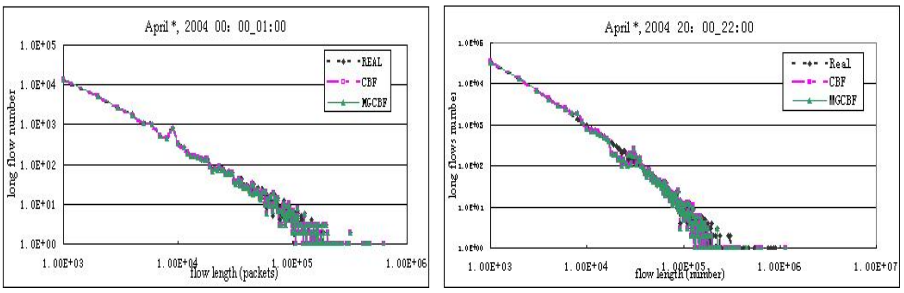

Fig. 2. The contraction of long flows (threshold=1000) distribution using different measurement

The experiment results illustrate that the long flows measurement based on MGCBF can improve the efficiency of flow information maintenance and save the storage space with little flow number measurement errors.

4 Conclusion and future work

This paper presents a long flow statistical and maintaining model based on MGCBF algorithm according to characteristics of flow length's heavy-tailed distribution in networks. This model can reduce the resource dramatically with little calculating cost, and maintain the flow information that is not existed in other flow length distribution and estimation algorithms without losing the integrality of long flow information. The model this paper provided has excellent expansibility to maintain all flow information whose length is longer than 2 packets. And this model can also be widely used in other related domains if the frequency of items in the datasets follows heavy-tailed distribution.

[3] The value can be calculated in the following equation：(14-(8＋11×1.0))/14=0.34。

The error rate provided by MGCBF is mainly coming from fixed confliction in CBF, while the other is coming from the long slow flows identification. Periodical refreshing in this paper used for flow number estimation is based on the estimation of the mean flow length. And it will induce more errors when network traffic becomes abnormal which causes the changing of the mean flow length. The future work of this paper is to improve the precision of long flow identification without losing its efficiency.

Reference

[1] N. Brownlee, C. Mills and G. Ruth. Traffic Flow Measurement: Architecture. RFC 2722. October, 1999.

[2] Smitha, I. Kim, and A. L. N. Reddy. Identifying Long Term High Rate Flows At A Router. In *Proceedings of High Performance Computing*. December, 2001.

[3] A. Shaikh, J. Rexford and K. G. Shin. Load-Sensitive Routing of Long-Lived IP Flows. In *Proceedings of the ACM SIGCOMM 1999*.Cambridge, MA, USA. August, 1999.

[4] C. Estan and G. Varghese, New Directions in Traffic Measurement and Accounting, In *Proceedings of the ACM SIGCOMM 2002*. Pittsburgh, PA. August 19–23, 2002.

[5] B. Bloom, Space/Time trade-offs in hash coding with allowable errors. In *Commun.ACM*. Vol. 13, no.7, pp. 422-426, July 1970.

[6] S. Cohen, Y. Matias. Spectral Bloom Filters. In *Proceedings of the ACM SIGMOD 2003*. San Diego, California, USA. June, 2003.

[7] A. Kumar, J. Xu, et al. Space-Code Bloom Filter for Efficient Per-Flow Traffic Measurement. In *IEEE Infocom 2004*, Hongkong, China. March, 2004.

[8] R. M. Karp, S. Shenker, and C. H. Papadimitriou, A Simple Algorithm For Finding Frequent Elements In Streams And Bags, In *ACM Transactions on Database Systems (TODS)*. vol. 28, pp. 51–55, 2003.

[9] A,Kumar, M.Sung. et al. Data Streaming Algorithms for Efficient and Accurate Estimation of Flow Size Distribution. In *Proceedings of the ACM SIGMETRICS 2004*. New York, NY, USA. June 10-14, 2004.

[10] M. Charikar, K. Chen, and Farach-Colton, Finding Frequent Items In Data Streams, In *ICALP. Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, Germany*. 2002.

[11] C. Estan, G. Varghese and M. Fisk. Bitmap Algorithms For Counting Active Flows On High Speed Links. In *Proceedings of the ACM IMW*, 2003.

[12] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking*, 8(3):281-293, 2000.

[13] A. C. Snoeren, C. Partridge, et al. Hash-Based IP Traceback. In *Proceedings of the ACM SIGCOMM 2001*, San Diego, California, USA. August 2001.

[14] M. Fang, N. Shivakumar, et al. Computing iceberg queries efficiently. In *Proceedings of the 24th International Conference on Very Large Data Bases, VLDB*, pp299–310. Morgan-Kaufmann, San Mateo, Calif, USA. 1998.

[15] K.C.Claffy, H.W.Braun, G.C.Polyzos. A Parameterizable Methodology for Internet Traffic Flow Profiling. In IEEE Journal on Selected Areas In Communications, Vol.12, No.8,Oct. 1995.pages: 1481-1494.