Algorithm Based on LL_CBF for Large Flows Identification

Lei Bai[1,2,3](✉), Jianshe Zhou[1,2], and Yaning Zhang[1,2]

[1] School of Literature, Capital Normal University, Beijing 100048, China
[2] Research Center for Language Intelligence of China, Beijing 100048, China
[3] North China Institute of Science and Technology, Langfang 065201, Hebei, China

Abstract. In order to manage large-scale network, it is very important to measure and monitor the network traffic accurately. Identifying large flows timely and accurately provide data support for network management and network security, which has important meaning. Aiming at the deficiency of high false negative rate by using traditional algorithm to detect large flows, a novel scheme called LL_CBF is presented, which uses the policies of "separation of large flow filtering and large flow identification" to improve the accuracy of traffic measurement. The algorithm is improved from four aspects: large flows handled firstly, using counting bloom filter to filtrate most small flows, using least recent used mechanism to filter small and medium flows and pre-protect large flows, and using least elimination strategy to identify large flows. The theoretical analysis and the simulation result indicates that compared with the standard LRU algorithm and LRU_BF algorithm, our algorithm can identify the large flow in the network timely and accurately, and reduce the computing resource requirements effectively.

Keywords: Traffic measurement · Large flow · Least recent used · Least elimination strategy

1 Introduction

Accurately measuring and monitoring network traffic is the basis for managing large-scale networks. However, along with the rapid expansion of the Internet and the continual emergence of new applications, network traffic presents the characteristics of high speed, large scale and complexity, and the obvious feature is that the large amount of data generated and the high frequency of data packet arrival. This requires shorter data processing time than before, which brings great challenges to the storage capacity, processing capacity and transmission capacity of the network measurement system. Fortunately the flow-based measurement method opens up a new way for flow monitoring. By merging packets into the flow, the data volume is greatly compressed, making the storage, processing and transmission of network data easier. In network traffic measurement, a series of data packets that satisfy certain specifications are abstracted into a flow. According to this specification, some attributes of the packets are mapped into the flow to represent a unique identifier for the flow (flow ID). The most commonly used flow specification is the 5-tuple of the packet header (source/destination IP addresses, source/destination port numbers, and transport protocol).

Many studies show that the statistics of network flow present a strong heavy tailed distribution. Heavy tailed distribution is a probability distribution model, which means that in the statistical set, a small part of the elements have a very high frequency of appearance, occupying the vast majority of the set, and most of the elements appear at a very low frequency. This characteristic is called “the elephant and mice phenomenon”, which means that most flows only have a small number of packets, while a small number of flows have a large number of packets. A notable feature of large flows is that they only account for a small part of the total traffic but generate the vast majority of the total traffic. So, in practical applications, in most cases, we only need to master large flow information to have an overall understanding of all network flows passing through the link which is convenient to manage and monitor the network traffic, and plays an important role in network traffic accounting, security detection, traffic control and other engineering applications. Therefore, how to use limited hardware resources to realize large flow identification has become a research hotspot in high-speed network measurement.

Traditional methods to achieve large flow identification needs to collect all packets in the network, and then extract their flow statistics, just as many previous studies have indicated. However, for the reason of system hardware computing speed and storage capacity is limited, and the network traffic data scale is huge, and the message arrival speed is extremely high, e.g. on the OC-768 (40 Gbps) backbone link, the average packet processing time is 8 ns. So, traditional methods have some flaws. In this paper, we propose and implement a new method called LL-CBF algorithm which use least obsolete (LEAST) & least recent used (LRU) & count bloom filter (CBF) algorithm to realize large flows identification.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the traditional method of identifying large flows. Section 4 states the strategies used in our algorithm. In Sect. 5, we discuss how to identifying large flows by LL-CBF algorithm. Section 6 analyzes its performance theoretically. Experimental results on trace are presented in Sect. 7. Finally, Sect. 8 concludes the whole paper.

2 Related Work

Tatsuya Mori [1] described how to identify large flows through counting periodically sampled packets. Their key idea is to find the threshold of per-flow packets in sampled packets which can reliably indicate whether or not a flow is actually a large flow in not sampled packets and that such a threshold can be obtained based on Bayes’ theorem. Kumar et al. [2] proposed a new technique referred to as space-code Bloom filter (SCBF) for extracting per-flow statistics of traffic in high-speed networks. Smitha Kim I [3] proposed LRU algorithm for identifying long-term high-rate flows at a router. Zhen Zhang [4] Used Bloom filter and LRU algorithm to realize long stream information statistics. ANTUNES N [5] estimate the tails of flow duration and size distributions under various sampling methods. Qingjun Xiao [6] proposed cardinality estimation solution mechanism to estimate large flows, which allocate each flow with a virtual estimator, and these virtual estimators share a common memory space. Aiping Zhou [7] design a new sketch data structure to detect network-wide persistent flow.

3 Classical Least Recent Used Algorithm

Least recent used (LRU) algorithm has a wide application in the computer field, such as database cache management, page management, disk cache management, etc. The fundamental of LRU is that always keeps the new element in the top of the cache and keep the least recent used element in the bottom. The LRU algorithm uses information about the pages accessed in recent past to predict the near future.

In an LRU cache every new entry is placed at the topmost position in the cache. The entry that was the least recently used is at the bottom. This is chosen to be replaced when a new entry has to be added and there is not enough space in the cache. This mechanism ensures that the recently used entries remain in the cache. The objective is to store state information for only large flows in the LRU cache. Smitha Kim I is the first to use LRU algorithm in traffic measurement. With a cache of limited size, a flow has to arrive at the router frequently enough to remain in the cache. Small flows are likely to be replaced by other flows fairly soon. These flows do not pump packets fast enough to keep their entries at the top of LRU list and hence become candidates for replacement. Large flows are expected to retain their entries in the LRU cache for long periods of time. However, when there are too many small flows or short-lived flows arrive at the cache suddenly, the large flows will be replaced by the small flows. This will decrease the accuracy of the measurement. To solve this problem we propose a new method.

4 Strategies Used in Our Algorithm

In order to solve the shortcomings of LRU algorithm that the small flows will replace large flows frequently, we improve the LRU algorithm from four aspects to improve the measurement accuracy.

4.1 Large Flows Handled Firstly

In order to identify large flows at the router, LRU algorithm must employ a cache which is of a fixed pre-determined size. For the reason of the number of memory we can afford is significantly smaller than the number of flows, when many small flows arrive at the cache suddenly, the large flows including the large flows that already identified will be replaced by the small flows.

To avoid this situation, we propose large flows handled firstly, which means that when a packet arrives, algorithm will validate whether this packet is belonged to a large flow which has been identified. If it is, then update the large flow information, if it is not, then packet will be handled later by other module. By this way, large flows which identified before will not be replaced by small one. By doing this can improve the measurement accuracy.

4.2 Using Counting Bloom Filter to Filter Most Small Flow Packets

Now that flow statistics has a characteristic which most small flows have a small number of packets, while a very few flows have a large number of packets. If we can filter out the packets of most small flows before they enter LRU cache, this will be reduce the number of flows which LRU algorithm handled, depress the probability of large flows replaced, and then will improve the accuracy of LRU algorithm.

Counting Bloom filter for representing a set S = $\{x_1, x_2......x_n\}$ of n elements is described by an array of m bits, initially all set to 0. It uses k independent hash functions with range {1,….,m}. CBF extend each unit in the standard Bloom Filter from a bit to a counter, so that it can add and delete elements. When adding an element, CBF uses k hash functions to map into the corresponding storage space and add 1 to the value of its mapped position; when deleting the element, the values of the corresponding k positions are reduced by 1.

The structure of CBF is shown in Fig. 1.

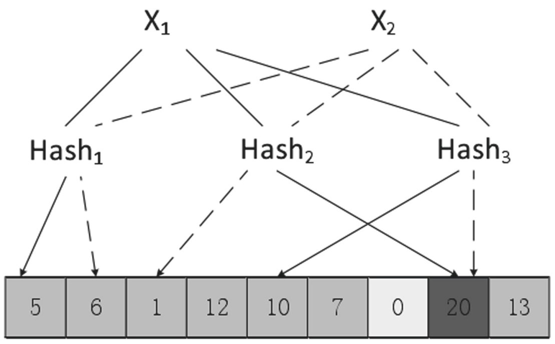

Fig. 1. CBF structure diagram

We use the characteristics of CBF to filter most small flows. When a packet arrives k hash functions are used to hash the flow ID of the packet which will be map into different positions of CBF. Since all packets belonging to the same flow will be mapped into the same location, the value of the large flow mapping location will be larger; for most small stream packets, the value stored in the mapping location is relatively small because of the different spatial locations of the mapping. By this way, most small flows can be filtered through the preset threshold n1, that is, when the value of the mapping position of the message is greater than or equal to n1, the message will be processed by the subsequent module. Most of the small flows are filtered out by CBF because they contain less packets and the value of mapping position is small. Due to the huge amount of data, there are also many small flows mapped to the same location, which leads to the small flows being submitted to the subsequent module for processing. Therefore, further processing of packets passing through the CBF is required.

4.3 Using the LRU Mechanism to Filter Small and Medium Flows and Pre-protect Large Flows

The LRU module is used to filter the middle and small flows passing through the CBF module, reducing the number of flows entering the LEAST module, thereby achieving pre-protection for large flows. The LRU maintains a cache and always keeps the flow which records of the latest arriving packets at the top of the LRU cache, while the longest unreachable flows are stored at the bottom of the cache queue. When a new flow reaches the LRU module through the CBF module, the LRU will create a new flow record and place it at the top of the cache. If the cache is full, the bottom flow will be replaced. Due to the short duration and the low arrival rate of the small flows, they are always possible to be replaced; while large flows tend to be stored at the top of the LRU cache because of their long duration and frequent access to the cache. The LRU is implemented as a doubly linked list. Each node contains an entry for flow id and the packet count. If the flow exists in the LRU linked list, put the flow at the top of the linked list; if a new flow arrives and the linked list is full, the bottom flow is eliminated and the new flow is placed at the top of the linked list. In order to make the search into the linked list easy, it is indexed by a hash table. Therefore, in addition to a small amount of additional operations due to hash conflicts, most objects only need to perform a hash operation to locate the flow record to which the message belongs.

However, due to the small storage space of LRU, when a large number of burst flows arrive or no new packets arrive in a short period of time, large flow objects may be eliminated by medium and small flows. Therefore, it is necessary to further use LEAST module to identify large stream objects.

4.4 Using LEAST Elimination Mechanism to Identify Large Flows

The LEAST module always eliminates the smallest flow and realizes long flow identification. If a packet arrives and its corresponding flow record is found in the LEAST table, then the size of the packet is added to this record; if the flow record corresponding to the packet is not in the LEAST table, then it will be mapped by the hash function To the corresponding items of CBF, and processed by CBF and LRU modules respectively. If the flow object passes the threshold set by CBF and LRU, a new entry will be added to the LEAST table to record the information of the flow. When the length of the LEAST table (the number of items stored) reaches the maximum, the smallest flow in the LEAST table will be eliminated in order to free up storage space for the newly arrived flow.

5 Identifying Large Flows by LL-CBF Algorithm

The large flow identification algorithm based on LL-CBF is composed of flow filtering function and flow identification function. The function of flow filtering is to filter out most of the small and medium flows information in network traffic, which is divided into two modules: CBF filtering and LRU filtering; the role of the LEAST module is to realize large flow detection. The pseudocode of the algorithm is described in Fig. 2 below.

```
initialize (CBF,LRU,LEAST)          //Initialize CBF,LRU,LEAST
While a packet x arrives
calculate H=h(1),h(2),…,h(k)        // Calculate k hash function values
if( isLargeFlows() ){       // If the large flow ID has already been identified
      update(count);                // Number of packets in the flow plus 1
}else{                              // Unrecognized flow ID
      CBF();                        // Call CBF function
}
CBF(){
      if  (any location of CBF[$h_i(x)]_{i=1..k}<n_1$){
                                //The k positions of CBF are all less than $n_1$
        add((CBF[hi(x)]i=1..k))     // CBF count plus 1
    }
    else        //The k positions of CBF are all greater than or equal to $n_1$
        LRU(x);                 // Call LRU function
}
LRU(x){
    if(find(x) or not full){
          // If already in the LRU linked list, or the linked list is not full
        update(count);              // Number of packets in the flow plus 1
        setTop();                   // Put the flow node on top
        if (number(count)>=$n_2$)
          LEAST(x)
    }
    else                    // Not in the linked list, and the linked list is full
      eliminate (last);             // Eliminate the last flow node
}
LEAST(x) {
    if(find(x) or not full){
        update (count);             // Number of packets in the flow plus 1
    }
    else                    // Not in the linked list, and the linked list is full
        obsolete (minimum);         // Eliminate the smallest stream
}
```

Fig. 2. The pseudocode of LL-CBF algorithm

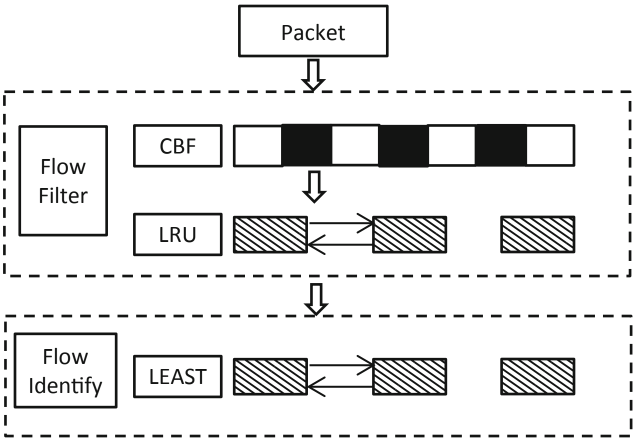

Fig. 3. The structure diagram of the LL-CBF algorithm

Figure 3 shows the structure diagram of the LL-CBF algorithm.

The procedure for identifying large flows by LL-CBF algorithm is enumerated as follows.

Step 1: First, when a packet arrives, k independent hash functions will compute k keys according to the 5-tuple identity (source/destination IP addresses, source/destination port numbers, and protocol).

Step 2: Secondly, validate whether this packet is belonged to a large flow. If it is, then the counter of large flow add one, if it is not, then packet will be handled by CBF.

Step 3: CBF validate whether the k positions of CBF are all less than n_1. If any of position is less than n_1, the counter of each position plus 1. Otherwise, the packet will be handled by LRU algorithm.

Step 4: LRU algorithm will search the cache to check if that flow's entry exists in the cache. On a miss, the flow is added to the cache if there is space in the cache. If there is no space in the cache, it replaces the least recently seen entry. It adds this entry in the topmost position in the cache. On a hit, update the entry in the cache. When the count of this flow exceeds the threshold n_2, this flow will be submitted to LEAST module.

Step 5: In LEAST module, LEAST algorithm will search the cache to check if that flow's entry exists in the cache. If find, then the size of the packet is added to this record; if not, it will determine whether the leap cache is full, if reaches the maximum size, the smallest flow in the LEAST will be eliminated, if not, then a new entry will be added to the LEAST table to record the information of the flow.

Step 6: At last, output the flows whose length is greater than the specified threshold TH which we recorded in LEAST, and those flows are large flows.

6 Performance Evaluation

The algorithm may have false positive during the large flow recognition process. False positive ratio (FPR) is the probability of identifying non large flows as large flows. In

CBF, the length of the counter array is m, the total number of detected packets is N, and all packets belong to n flows. When an element is inserted into the CBF, the probability that a certain position is mapped to is $1/m$, $n * k$ mappings were performed in total. The probability that the hash position is empty is P'

$$P' = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{\frac{nk}{m}} \tag{1}$$

False positive ratio P

$$P = \left(1 - P'\right)^{k} \approx \left(1 - e^{\frac{nk}{m}}\right)^{k} = \exp(kln\left(1 - e^{\frac{nk}{m}}\right) \tag{2}$$

Let

$$f(k) = k\ln\left(1 - e^{\frac{nk}{m}}\right)$$

When f takes the minimum value

$$\frac{\partial f(k)}{\partial k} = 0$$

Then

$$k = \ln 2\left(\frac{m}{n}\right)$$

So, the minimum value of CBF FPR is

$$P_{min} = \left(\frac{1}{2}\right)^{k} \tag{3}$$

In any measurement time period, the flow rate obeys the Pareto distribution with position parameter 1. Assuming that the total number of packets in the measurement period is M, the LRU creates a new flow identifier every N packet on average, and eliminate a flow at the bottom of the linked list. Suppose the size of a large flow F is exactly equal to the threshold TH, then the probability of no large flow F in N consecutive messages obeys the hypergeometric distribution.

$$\binom{TH}{0}\binom{M - TH}{N}/\binom{M}{N} \tag{4}$$

When $M >> N$, formula (1) can be approximated as $\left(1 - \frac{TH}{M}\right)^{N}$. Therefore, the probability that the current F is eliminated is

$$P_{LRU}(F = TH) = P(F = TH)\left(1 - \frac{TH}{M}\right)^{N} \tag{5}$$

Since

$$P(F = TH) = \theta / TH^{\alpha+1}$$

Then

$$P_{LRU}(F = TH) = \frac{\theta}{TH^{\alpha+1}}\left(1 - \frac{TH}{M}\right)^N \tag{6}$$

In which θ is the normalized parameter.

$$\theta = \left(\sum\nolimits_{i=1}^{M} i^{-\alpha-1}\right)^{-1}$$

Because the algorithm search process uses hash function set, the memory cost of one access to hash space is $O(k)$. At the same time, the LRU linked list uses doubly linked list, and the zipper method is used to resolve hash conflicts. The average search length of the algorithm is $O(1 + \beta/2)$, β is the filling factor.

7 Experimental Analysis

In order to verify the effectiveness of the LL_CBF algorithm, we use Trace collected from CAIDA for simulation experiments. There are 6187376 packets and 68367 flows in total. CBF uses $k = 6$ hash functions, and it's hash space is [0.. 65535]. The original distribution of network traffic is shown in Fig. 4. It can be seen from the figure that the distribution statistics of the network flow present a heavy-tailed distribution.

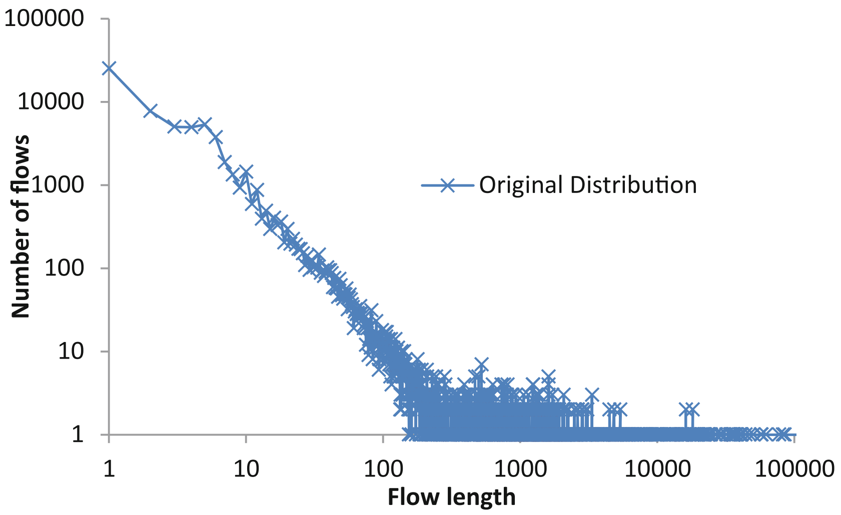

Fig. 4. The original distribution of network traffic

Figure 5 shows the comparison of whether the big flow is processed first. It can be seen from the figure that the measurement accuracy of large flow handled firstly is much higher than that of post handled. This is because the post-processing method will cause

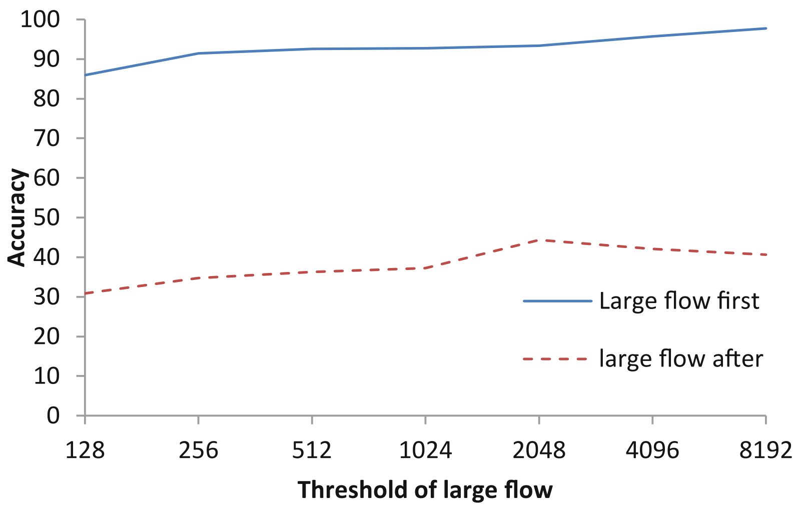

Fig. 5. Comparison of large flow pre-processing and post-processing

the large flows to be eliminated by the small flows, thus the measurement accuracy is reduced.

Figure 6 shows the proportion of the number of filtered flows when the CBF counter changes. The greater the value of counter, the more flows are filtered. As the counter value increases, most of the small flows are discarded because the value of the mapping position is smaller than the counter.

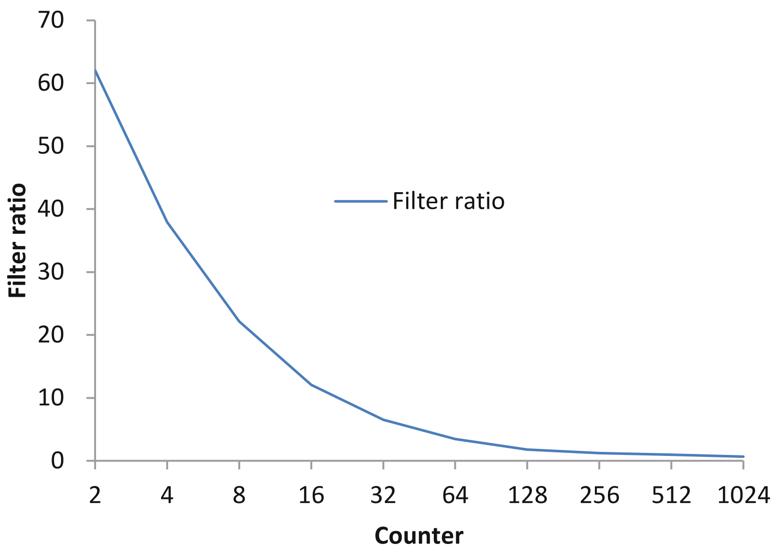

Fig. 6. The proportion of the number of filtered flows when the CBF counter changes

Figure 7 displays the measurement error when LRU cache space changes. The larger the cache space, the smaller the probability of the large flow being replaced and the higher the accuracy.

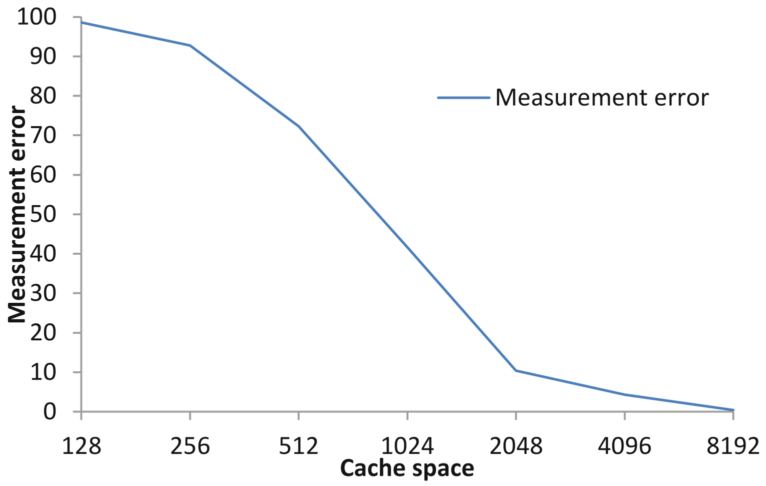

Fig. 7. Measurement error when LRU cache space changes

Figure 8 indicates the comparison of measurement accuracy between LL_CBF algorithm, the LRU_BF algorithm and the standard LRU algorithm, using the same hash function and number and cache space.

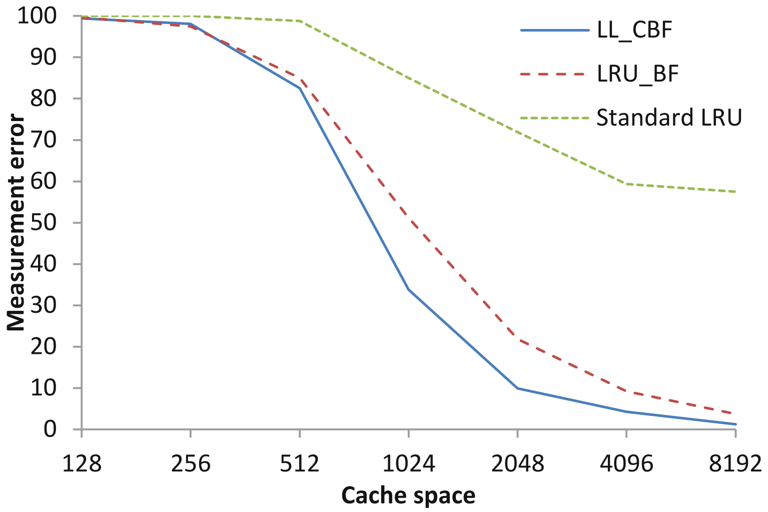

Fig. 8. Comparison of the accuracy of three algorithms for measuring large flow

Experimental results show that compared with the standard LRU algorithm and LRU_BF algorithm, the LL_CBF algorithm proposed in this paper has higher measurement accuracy, especially when the cache space is relatively small the advantage of the LL_CBF algorithm is more obvious. This is because when the cache is small, the standard LRU algorithm and the LRU_BF algorithm, for the cache space is full, a large number of newly arrived small flows will eliminate the unidentified large flows in the cache. However, the LL_CBF algorithm can filter out most of the small flows through the filtering mechanism, while retaining the identified large flow information, reducing the impact of the small flow on the large flow, thereby reducing the measurement error of the algorithm.

8 Conclusion

Due to the rapid and large-scale development of the network, it is becoming more and more difficult to completely measure network flow information online. In this paper, based on the measurement defects of the sudden large number of small flows that lead to the elimination of large flows and the characteristics of network heavy tail distribution, propose a large flow detection algorithm based on LL_CBF, which will filter out most of the small flow through filtering mechanism, and reduce the probability of small flows entering the cache space, realize the strategy of “grasp the big and let go of the small”. The complexity and error rate of the algorithm are analyzed, and the effectiveness of the algorithm is verified by experimental data. The results show that compared with the standard LRU algorithm and LRU_BF algorithm, the new algorithm can identify the large flow in the network timely and accurately under the condition of using less storage space, and meet the actual measurement needs.

Acknowledgment. We would like to express our appreciation for the assistance with data collection we acquire from CAIDA. This research was financially supported by “Research on key technologies and model verification of prose genre oriented text understanding (ZDI135-101)”, “Research and Application of Key Technologies of Intelligent Auxiliary Reading System (ZDI135-79)”, Capacity Building for Sci-Tech Innovation-Fundamental Scientific Research Foundation (20530290082).

References

1. Tatsuya, M., Masato, U., Ryoichi, K.: Identifying elephant flows through periodically sampled packets. In: Proceedings of ACM SIGCOMM/IMC 2004, pp. 115–120. ACM Press, Taormina (2004)
2. Kumar, A., Xu, J., Wang, J., Spatschek, O., Li, L.: Space-code bloom filter for efficient per-flow trafficmeasurement. In: Proceedings of IEEE INFOCOM 2004, Hong Kong, China, (2004)
3. Kim, S.I., Reddy, N.A.L.: Identifying long-term high-bandwidth flows at a router. In: Proceedings of the 8th International Conference on High Performance Computing, Hyderabad, India, pp. 361–371 (2001)
4. Zhang, Z., Wang, B.: Traffic measurement algorithm based on least recent used and Bloom filter. J. Commun. **34**(1), 111–120 (2013)

5. Antunes, N., Pipiras, V.: Estimation of flow distributions from sampled traffic. ACM Trans. Model. Perform. Eval. Comput. Syst. 1(3), 1–28 (2016)
6. Xiao, Q., Chen, S.: Cardinality Estimation for Elephant Flows : A Compact Solution Based on Virtual Register Sharing. IEEE/ACM Trans. Netw. (TON) **25**(6), 3738–3752 (2017)
7. Zhou, A., Zhu, C.: Detection method for network-wide persistent flow based on sketch data structure. Comput. Appl. **39**(08), 2354–2358 (2019)