ITC 2/49

Information Technology and Control
Vol. 49 / No. 2 / 2020
pp. 197-205
DOI 10.5755/j01.itc.49.2.24633

Multiclass Membership Determination Integrating an ID-Bound Method with Bloom Filter

Received 2019/11/17 | Accepted after revision 2020/03/18

http://dx.doi.org/10.5755/j01.itc.49.2.24633

HOW TO CITE: Chen, L.-I., Ma, H. (2020). Multiclass Membership Determination Integrating an ID-Bound Method with Bloom Filter. *Information Technology and Control*, 49(2), 197-205. https://doi.org/10.5755/j01.itc.49.2.24633

Multiclass Membership Determination Integrating an ID-Bound Method with Bloom Filter

Lu-I Chen

Ph.D. Program of Technology Management, Chung Hua University; 707, Sec.2, WuFu Rd., Hsinchu, 30012 Taiwan, R.O.C.; phone: +886 3 5186082; fax: +886 3 5186575; e-mail: d10303010@g.chu.edu.tw

Heng Ma

Department of Industrial Management; Chung Hua University; 707, Sec.2, WuFu Rd., Hsinchu, 30012 Taiwan, R.O.C.; phone: +886 3 5186082; fax: +886 3 5186575; e-mail: hengma@g.chu.edu.tw

Corresponding author: hengma@g.chu.edu.tw

Membership determination with classification is an essential function for a variety of applications, such as network routing and packet inspection. We aim to constitute a mechanism capable of performing such a service with the considerations of efficiency and error ratio issues. The proposed method employs an array structure in which each cell composes of multiple bits of memory. The array houses a circular bound representation and a Bloom filter, where the former encodes the members and makes judgments of queries, while the latter acts as a secondary means for programming cases indeterministic by the former. Experimental results show that the error ratios were competitive and consistent with different levels of classification, as well as the memory space utilized. The lookup time decreased as the memory space increased. We depict specific characteristics for tackling the problem, including simplicity of data structure for encoding the members, avoidance of checking all classification IDs in the lookup process, and adoption of an auxiliary means for improving the error ratios. The ID-bound method is novel, and with the Bloom filter as the assistant, the proposed approach is robust in performances of both efficiency and error ratio.

KEYWORDS: Membership determination, Classification, Bloom filter, Data structure, Distributed systems, Network services.

1. Introduction

Generally, a mechanism of membership determination comprises of two operational modules, the encoding, and the checking one, with a specifically-designed data structure. The former is responsible for inserting known members to the data structure, while the latter judges a query's status accordingly. The checking process must perform as quickly as possible because it often works at the network transmission speed [1]. The Bloom filter (BF) has been notably capable in dealing with such a problem because it incorporates a hashing scheme and a bit array, allowing a nearly constant speed for determining membership. False-positive error is inevitable due to the hashing scheme and partly tight memory constraint [2]. Even so, the BF and its variants have successfully applied to a variety of problem issues because of its simple encoding and checking procedures with the easiness of operating the bit array [16].

The BF, however, could only provide the membership information, true or false, of a query, not competent in combining a classification ID with a recognized member if it belongs to one of many disjoint sets [4]. The determining or lookup time mainly relates to the number of hash functions because it represents the number of memory accesses [9]. Geravanda and Ahmadib [6] indicated that each retrieval with only one bit is wasteful because multibit accesses within the system capacity do not affect the overall performance and could associate a member with more information of interest. Based on this concept, we aim to encode the members with the classification ID into a single data structure with only a small number of retrievals to the memory space. Our goal is to expedite the lookup process with the least error ratios, including the false-positive (FP) and the true-negative (TN) ones.

We formalize the problem as given g sets and their classification or set IDs (SID) starting from one to g. The size of each group could vary to some extent, and g is ordinarily huge, e.g., several thousand. When processing a query Q, the mechanism must respond to the membership status and its designated SID as quickly as possible with low error ratios.

The performance metrics include the two error ratios, TN and FP, and the online processing time of the checking module. A feasible mechanism must be robust on these metrics. For example, the average lookup time should not significantly increase as g grows, and the overall error ratio does not deteriorate when the number of total members becomes relatively large. Among the performance metrics, the lookup time is essential because it symbolizes the efficiency of the mechanism in real-time, which accounts for the number of memory accesses and the computational complexity of the checking procedure [14].

The proposed approach uses a single array A of m cells, each of which is composed of b bits, where b relies on the number of disjoint sets g. Supposing g=500, an SID requires at least 9 bits for representing the IDs from 1 to 511. The array includes the bound and support fields, where the former consists of two bounds for describing the scope of SIDs involved, and the later contains two additional bits for each cell, establishing a couple of bitstreams on the array for assisting the bound-field encoding and judgment actions. Table 1 lists all the notations and their descriptions used throughout this paper.

Table 1
Notation and description

Notation	Description
A	The proposed array of multi-bit cells
g	The number of disjoint sets
m	The size of A
n	The total number of all elements in g sets
k	The number of hash functions
e	A member element
S_e	The SID of e
Q	A query in the checking mode
C_e	The cells of A addressed by e using the hashing scheme
C_q	The cells of A directed by Q using the hashing scheme

2. Related Work

Since a single BF could no longer handle the addressed problem, some researchers alter the bit array of BF to other forms, while others adhere to the original one and employ additional hash functions for resolving the issue. Yu and Mahapatra [22] proposed a multi-tier structure for saving power in dealing with packet classification on the Internet. Each tier is composed of several BFs for representing many classes of packets. On the bottom level, each BF denotes only one category to obtain the SID information when a query reaches this tier. The purpose of such a structure is to recognize undetermined packets at higher levels so it could minimize the number of memory accesses. However, it is challenging to decide the number of layers and the sizes for the BFs for proper error ratios. Xiao & Hua [17] proposed three structures, including the ones named PBF, PBF-HT, and PBF-BF, where PBF is responsible for storing multiple attributes of a set element, and the other two reduce the false-positive errors and remove unnecessary memory space. The Bloom tree [21] deploys many BFs on a hierarchy for determining the SID information of a query. The sizes of the BFs on each level could be troublesome to specify because the dimensions of the disjoint sets could vary to some extent. The Invertible Bloom Lookup Table [7] stores the element and the attached information as key-value pairs in the same data structure. Each cell contains three fields, including the element count, the sum of the keys, and the values of all the items at the same cell. When the lookup process finds a cell whose count is one, it returns the attached content; however, it could answer "not found" for a member if all the counts of the hash-map cells are greater than one, which accounts for a significant portion of the true-negative errors.

Hao et al. [8] proposed the combinatorial of Bloom filters (COMB), which employs a single Bloom filter of an enormous length with the combinations of hash functions to achieve distinguishing the classification. The method could downgrade the overall lookup efficiency when there is a tremendous number of sets because all the combinations must proceed through the checking procedure. The Bloom multi-filter (BMF) [18] is an integer-number array, which employs bitwise operations for both the insertion and deletion of the set elements. The process, however, could elevate the false-positive ratios with many sets involved. Dai et al. [3] also used an array composed of multi-bit cells, the noisy Bloom filter (NBF), where the first bitstream is the standard BF for encoding members, and the rest is to record the SIDs. The ID Bloom filter [10] employs bitwise operations, which designates the SIDs with decimal numbers for saving memory accesses at the lookup phase. The bitwise operations could be difficult to encode a considerable amount of sets into the data structure.

The shifting framework [20] stores membership and additional information using a BF, and retrieves a chunk of memory from the bit array with an offset following $\boldsymbol{C_e}$. The framework encodes each (e, S_e) pair to a single bit array, which could become extremely long, and thus complicated to maneuver. The Difference Bloom Filter [18] applied two designs for providing the desired information, where they considered the memory type as a cost-efficient strategy. The OMASS scheme [12] used a block of BFs, which is subsequently divided into several subblocks and mapped by the same set of hash functions for obtaining the SID information. Einziger & Friedman [5] pointed out that the BF is not memory efficient; therefore, the authors proposed several designs for saving memory space with low false-positive ratios using an array of words as the data structure. Additional access to memory could take place as several data structures involved. A neural network approach [11] based on the cerebellar model articulation controller uses the S_e of each e as the target for mapping e during the encoding process. All elements of the same set establish boundaries for each SID, so all the members fall within the enclosed limits in the lookup phase to recognize as a member with that SID.

Sun et al. [15] proposed a magic cube Bloom filter approach that employs a second set of hash functions for obtaining offsets in coding parallel Bloom filters. Such an approach could bring down the TN ratio; however, it still needs to go through all the possible SID arrays for a specific answer, which could become less efficient when g is large. Qiao et al. [13] employed two data structures, including an index filter and a set-ID table. The index filter is a BF, while the set-ID table is responsible for storing the S_e for each e. The checksum of e functions as a secondary confirmation if the result the BF is positive. An empty cell must be available in $\boldsymbol{C_e}$ for storing the S_e and the checksum. In the checking procedure, the method concatenates each cell address of $\boldsymbol{C_q}$ on the set-ID table with Q to consult with the index filter for the existence of a member. When the membership is positive, and the checksum matches that of Q, it is then a member with the SID stored at the address of the set-ID table. This approach is efficient in terms of processing a lookup because it only needs to examine k hash addresses. The checksum could significantly reduce the FP error ratio; however, it must find an empty cell to store the S_e in the encoding process, which could cause additional TN errors when the memory space is compact.

There are three profiles that we draw from the previous investigations for tackling the addressed problem. First, the data structure and the hashing scheme must be simple for performing efficient lookups. Second, the checking module should not scrutinize all the SIDs for reaching a decision. Third, a secondary mechanism could involve in the process to slim down the error ratios. Thus, we adopt the work of Qiao et al. [13] as the benchmark because it fits all the profiles.

3. The Proposed Approach

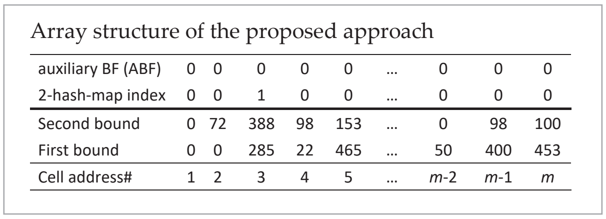
The proposed approach encodes n pairs of (e, S_e) to an array structure, while the checking procedure determines the status of Q according to the encoded consequence.

3.1. Array Structure

In the proposed array structure, each cell consists of a bound field, including the first and second bounds, and a support field composed of two additional bits, as shown in Figure 1. The memory requirement for the former depends on g, while the latter is constant. For example, it requires 9 bits to designate g=1~511, while 10 bits could have g up to 1,023. Thus, when g=500, each cell in A claims 20 bits (9*2+2) of the memory space. And if each member consumes c bits, we have $n*c$ bits as the working memory capacity, and m would be $n*c/20$.

Figure 1

Array structure of the proposed approach



Array structure of the proposed approach

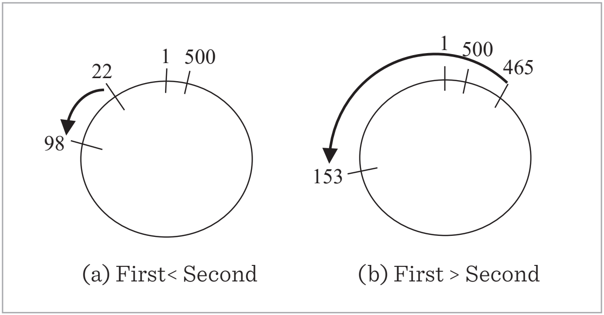
auxiliary BF (ABF)	0	0	0	0	0	…	0	0	0
2-hash-map index	0	0	1	0	0	…	0	0	0
Second bound	0	72	388	98	153	…	0	98	100
First bound	0	0	285	22	465	…	50	400	453
Cell address#	1	2	3	4	5	…	*m*-2	*m*-1	*m*

Figure 1 shows a fragment of the results after the encoding process is complete. There are four types of outcomes for each cell, including the ones receive 0, 1, 2, and 3-or-more (3plus-) hash-maps by all the n members (also referred to as the payload). We note that each member has k hash-maps to A. To elaborate this, the 0-hash-map cells are those that the payload never encounters with, and both bounds remain 0, such as cell #1. The 1-hash-map cells contain only one bound with a SID, e.g., cell #2 and #m-2. The location of the SID, first or second, relies on the order the hash functions in our approach, for facilitating an additional tool to detect false-positive errors in the lookup phase. The hashing scheme is a sequence of hash functions, each of which transforms e to an address of A. In our approach, we store the SID in the first bound when an odd-number hash function reaches the cell, while the counterpart is the opposite. Explicitly, an even-number hash function encounters cell #2, while an odd-number one runs into cell #m-2.

When no zero is in the bound field, e.g., cell #3, #4, #5, and #m, it could be a 2- or 3plus- hash-map case. We allocate a bit in the support field, referred to as the 2-hash-map index, for distinguishing these two instances. For example, cell #3 has the bounds of 285 and 388, and the 2-hash-map index is one, indicating that only those members with these two SIDs fall in the cell. The 2-hash-map index is initially zero and turns to one when exactly two SIDs engage in the same cell. It returns to zero when more than two SIDs occupy the same compartment. For the 3plus-hash-map cells, we introduce a circular representation in the counter-clockwise direction for minimizing the acceptance range (AR), as shown in Figure 2. The AR composes of all the SIDs that seize the cell. With such a representation, the FP ratio has the largest chance to shrink because it covers the minimal number of SIDs and the opportunity that a non-member drops in the AR decrease. Figure 2 (a) and (b) demonstrate the ARs of cell #4 and #5, respectively.

Figure 2

Circular representation of SIDs for 3plus-hash-map cells



3.2. Encoding Process

There are two stages in the encoding process, including (1) program the payload to A according to Table 2, and (2) configure the auxiliary BF (ABF) for resolving the uncertain cases of the bound field. In the first stage, each (e, S_e) pair sequentially inserts to A, where e hashes to $\boldsymbol{C_e}$ and S_e merge into $\boldsymbol{C_e}$ using Table 2. The second stage collects the TN and FP errors that the bound field fails to discriminate and configures those to the ABF for a confirmed judgment of Q.

Table 2

Operations of inserting a member pair to different types of cell

Type of cell	Operations
0-hash-map	Store S_e to the correct bound
1-hash-map	(1) Store S_e to the 0 location (2) Switch the 2-hash-map index to 1
2-hash-map	(1) Set the bounds for the minimal AR (2) Switch the 2-hash-map indicator to 0
3plus-hash-map	Set the bounds for the minimal AR

We now investigate the probabilities that a random cell receives r encounters by the payload, P_r, as in (1)~(5).

$$P_r = \frac{\binom{nk}{r}(m-1)^{nk-r}}{m^{nk}}, \tag{1}$$

$$P_0 = \left(\frac{m-1}{m}\right)^{nk}, \tag{2}$$

$$P_1 = nk\frac{(m-1)^{nk-1}}{m^{nk}} = \frac{nk}{m-1}P_0, \tag{3}$$

$$P_2 = \frac{nk(nk-1)}{2}\frac{(m-1)^{nk-2}}{m^{nk}} = \frac{nk(nk-1)}{2(m-1)^2}P_0, \tag{4}$$

$$P_{3+} = 1 - P_0 - P_1 - P_2. \tag{5}$$

We derive (1) from the birthday problem, which is a general expression of the probability that a cell collects r appointments by n members and (2)~(5) according to (1). Therefore, if k=5, n=2,000,000, and m=4,000,000 using 40 bits of memory space for each member, we have P_0=0.0821, P_1=0.2053, P_2=0.2563 and P_{3+}=0.4563. When k=3, the portions are P_0=0.2231, P_1=0.33475, P_2=0.251 and P_{3+}=0.1912.

With the probability distribution of various types of cells, we establish the membership determining rules based on Table 2 as: (1) A first or second bound is an "eligible" SID for Q when it is within all the ARs of $\boldsymbol{C_q}$, (2) If there is no eligible bound, Q is a non-member, (3) A single eligible bound is the SID of Q, and (4) When there are multiple eligible bounds, the one with the most appearances is the winner. For example, if an e, with k=3, whose mapping addresses are cell #4, #5, and #m-1 in Figure 1, both 98 and 22 are eligible. But since there are two bound appearances of 98, and 22 has only one, the former is the winner as the S_e.

On the other hand, when an e hashes to cell #4, #5, and #m, although both 98 and 22 are eligible, they both have only one bound appearance. In such a tied situation, also a type of the TN error, the process initiates to configure the ABF by adopting the shifting strategy of [20]. For example, if the second eligible bound 22 is the S_e, we offset $\boldsymbol{C_e}$ by two cells for encoding the ABF, i.e., zeros to ones. Conversely, if 98 is the S_e, the process configures the ABF on the $\boldsymbol{C_e}$+1 addresses. Such a method could accommodate the situation when there is a tie of several bounds. For the checking procedure, therefore, we examine these tied bounds, and the one with the ABF confirmation (all ones) is the winner. On the other hand, Q is a non-member if there is no endorsement of the ABF.

Besides the enhancement in reducing the TN error using the ABF, we also explore the FP one by using an adequate amount of non-members to test an encoded A. Consequently, the most FP scenario lies in that there is a single eligible bound with only one bound appearance. Therefore, we consider such a situation is a non-member unless there is a confirmation from the ABF for indicating that it is a member. So, in the second stage of the encoding process, we deal with such a matter by encoding the ABF at the $\boldsymbol{C_e}$ address to 1 for such a condition. For example, if e addresses cell #5, #m-1, and #m in Figure 1, only the bound 465 is eligible with only one appearance. The process, therefore, encodes ones to the ABF at $\boldsymbol{C_e}$ for the confirmation as a member in the lookup phase.

The ABF plays a supporting role in discriminating uncertain cases of the bound-field determination, so we concern about its inherent error of FP because it is, after all, a BF. As a standard BF, the larger the ratio m/n, the less the FP errors. Fortunately, the ABF only engages in a relatively small portion of the TN, and FP errors escaped from the bound-field filtration (n), and the size of it (m) is adequate so that we could ignore the FP ratio of the ABF [2]. Therefore, we modify rule (3) of the membership determining to "A single eligible bound whose bound appearance is one must consult with the ABF for being a member.", and (4) to "When there are multiple eligible bounds, the one with the most appearances is the winner, but the process must conform with the ABF when there is a tie".

3.3. Lookup Process

The lookup process determines Q's status, i.e., if there is a single eligible bound, Q is a member, whose ID is that bound; otherwise, Q is a non-member. Figure 3 shows the pseudo-codes of the checking procedure.

The purpose of holding such a checking sequence is to determine the status of Q as quickly as possible. Our approach has a couple of advantages for such use. First, since there is a single array structure involved, it could quicken the procedure in real-time since the number of memory retrievals lessens. Second, the method could be more efficient since there are two sites in the bound field, representing the limits of the SIDs at the same address, and that could result in practical maneuvers for fast filtering the non-members. For example, when there is a 1-hash-map cell in $\boldsymbol{C_q}$, the procedure checks if the SID's location, first or second, is correct, and it is immediately a non-member when it is on the wrong site.

Figure 3
Pseudo-codes of the checking procedure

```
procedure CheckingStatus(Q)
1:  for i=1 to k do if Cq(i) ∈ {0-hash-encounter} then
    return (0);
2:  for i=1 to k do if Cq(i) ∈ {1-hash-encounter} then
3:    if Cq(i).FirstBound=0 and i mod 2=1 return (0);
4:      if Eligible(Cq(i).SecondBound) then
5:        return (Cq(i).SecondBound);
6:      end if
7:    end if
8:    if Cq(i).SecondBound=0 and i mod 2=0 return (0);
9:      if Eligible(Cq(i).FirstBound) then
10:       return (Cq(i).FirstBound);
11:     end if
12:   end if
13: end for
14: EligibleList←∅
15: NumberEligible←0;
16: for i=1 to k do
17:   if Eligible(Cq(i).FirstBound) or Eligible(Cq(i).
    SecondBound) then
18:       Add Cq(i).FirstBound or Cq(i).SecondBound
    to EligibleList;
19:       NumberEligible←NumberEligible+1;
20:   end if
21: end for
22: if NumberEligible=1 and Appear(EligibleList(1))=1
    then
23:   if ABF(Cq)=1 then return (EligibleList(1))
24:   return (0);
25: end if
26: if NumberEligible>1 then
27    NumberEligibleAppearTied=FindTied(EligibleList)
28:   for i=1 to NumberEligibleAppearTied do
29:     if ABF(Cq+i) =1 then return (EligibleList(i));
30:     return (0);
31:   end for
32: end if
end CheckingStatus
```

Furthermore, we could check the eligibility of Q for concluding its status. The ABF only participates in the situations when the bound field could not ascertain the result of Q. Two scenarios constitute the most cases, including (1) potential TN error with multiple eligible SIDs, and (2) possible FP error of a sole suitable SID with a single bound appearance. We deal with these two situations by taking advantage of programming the ABF, and the error ratios significantly reduce, as shown in the next section.

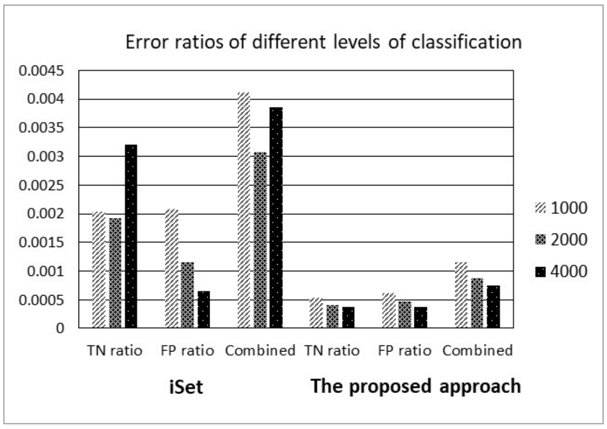
4. Experimental Results

We conducted several experiments with varies of two parameters, including the level of classification and the memory-space utilization, for evaluating the error ratios. The error ratios involve the TN (true-negative) and FP (false-positive) ones. The former is the case that the mechanism misjudges a member as a non-member, while the latter is the counterpart. Besides, we also investigated the lookup efficiency and compared all the results with those of the "iSet" approach by Qiao et al. [13], a state-of-the-art effort in recent years. Since there are no guidelines in setting the configuration of the reference, we used the same size for both the index filter and the set-ID table, and the checksum is of the same number of bits with the SID in the set-ID table.

4.1. Level of Classification

The number of sets represents the level of classification, and there were 1,000, 2,000, and 4,000 ones in the experiment. We used 2,000,000 randomly-generated elements, strings of 32 characters, as the members for encoding both approaches at different levels of classification, where the number of hash functions k=5. The assignments of the SID for each member were random between one and g. The memory-space utilization is 40 bits per member; thus, it required 80M bits of the memory space for this experiment. We also employed another set of 2,000,000 non-member samples with no SID attached for estimating the FP ratios. Figure 4 shows the results.

Figure 4
Error ratios for different levels of classification

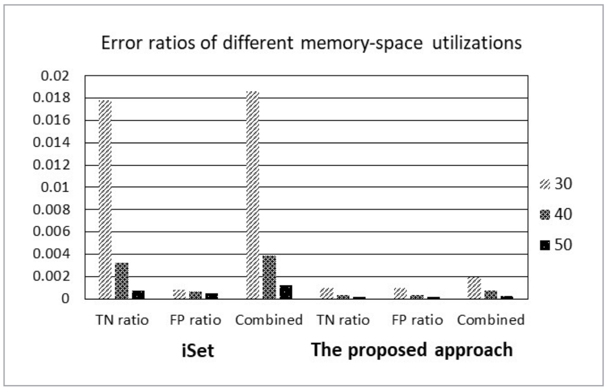


The combined rate is the summation of the TN and FP ones. It appears that the error ratios of our approach are more consistent than the "iSet" method with various levels of classification, and it had approximately 69~88% decreases in the combined rates compared with the reference.

4.2. Memory-Space Utilization

Intuitively, the larger the memory-space usage, the less the error ratios. Therefore, in this subsection, we looked into the sensitivity of different utilizations of memory space to the error ratios for both approaches. We allocated the memory space of 30, 40, and 50 bits per member, and the level of classification was 4,000. Figure 5 shows the results for both means.

Figure 5
Error ratios for different memory-space utilizations



Both approaches achieved lower error ratios as the memory-space utilization increased. The proposed method reported about 77~89% reductions to those in the combined section of the benchmark. The results also show that the TN ratios of our approach were competitive in this category. Such an outcome could benefit from the number of unsuccessfully-encoded members is not significantly high using our scheme due to the circular representation.

4.3. Lookup Time

We performed the lookup procedures of both approaches on an Intel Core i7-7700 at 3.6 GHz with 128M SRAM. Figure 6 shows the elapsed time in seconds for processing 2,000,000 queries of the member and non-member samples with different levels of classification, based on 40 bits of memory space per member. Figure 7 presents the ones with varied memory-space utilizations, where the classification level was 4,000.

Figure 6

Elapsed time for processing 2,000,000 queries of different levels of classification

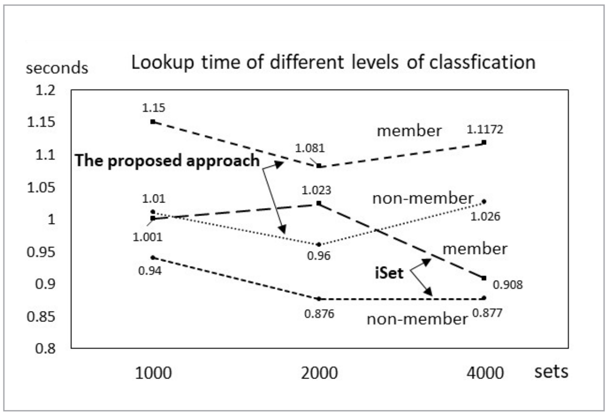
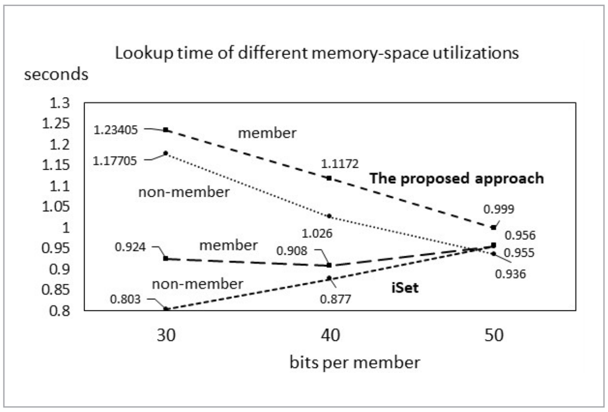


Figure 7

Elapsed time for processing 2,000,000 queries of different utilizations of memory space



In Figure 6, the lookup time for members was longer than those of the non-members in both approaches, and the levels of classification did not significantly affect the lookup efficiencies. It could conceive that the lookup time for a member is more complicated than that of the non-members because many non-members could filter themselves out at the early stage of the procedure, but the member must accompany with a SID to conclude. Our approach required about 5~15% more time than the reference at all levels. However, in Figure 7, it shows that lookup efficiency improves as the memory-space utilization increases for our approach, while that of the benchmark lacks such a tendency.

5. Conclusion

The proposed approach is an effort resulted from recent endeavors in the literature. We realize that membership determination with classification is a difficult task to manage and that there are specific characteristics for tackling it, including (1) a simple data structure and scheme for encoding the members, (2) a checking procedure that does not go through all the classification IDs for a decision, and (3) a secondary means for distinguishing the uncertain cases of the primary scheme. The first one is the cornerstone for the entire mechanism, where we propose a single array with an ID-bound method for meeting such a need. For (2), the checking procedure of our approach could produce a result by only examining the bound SIDs of those cells induced by a query. And for the third, we employ an auxiliary Bloom filter embedded in the array structure as the second net for capturing the members slipped through the primary (ID-bound) filtration. Such a design, as the experimental results show, substantially reduces the error ratios because it only handles a small portion of the indistinguished members, and the performance is remarkable because the size of it is well sufficient for such a task. The proposed mechanism could achieve relatively low error ratios with different levels of classification, as well as the memory-space utilization. Furthermore, it could also promote lookup efficiency as the memory-space utilization increases, which represents the ability to adapt to the environment when necessary with sufficient memory space available.

References

1. Chen, H., Jin, H., Chen, L., Liu, Y., Ni, L. Optimizing Bloom Filter Settings in Peer-to-Peer Multikeyword Searching. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(4), 692-706.https://doi.org/10.1109/TKDE.2011.14
2. Christensen, K., Roginsky, A., Jimeno, M. A New Analysis of the False Positive Rate of a Bloom Filter. Information Processing Letters, 2010, 110(24), 944-949. https://doi.org/10.1016/j.ipl.2010.07.024
3. Dai, H., Zhong, Y., Liu, A. X., Wang, W., Li, M. Noisy Bloom Filters for Multi-Set Membership Testing. The ACM SIGMETRICS, 2016, Juan-les-Pins, France, 139-151. https://doi.org/10.1145/2896377.2901451
4. Ding, N., Bi, X., Zhang, D. An Efficient Hybrid Hierarchical Trie Packet Classification Algorithm Based on No Prefix Relationship. Journal of Computational Information Systems, 2013, 9(22), 9193-9202. DOI: 10.12733/jcis8315
5. Einziger G., Friedman R. Tinyset - an Access Efficient Self Adjusting Bloom Filter Construction. IEEE/ACM Transactions on Networking, 2017, 25(4), 2295-2307. https://doi.org/10.1109/TNET.2017.2685530
6. Geravanda, S., Ahmadib, M. Bloom Filter Applications in Network Security: A State-of-the-art Survey. Computer Networks, 2013, 57(18), 4047-4064. https://doi.org/10.1016/j.comnet.2013.09.003
7. Goodrich, M. T., Mitzenmacher, M. Invertible Bloom Lookup Tables. The 49th Allerton Conference on Communication, Control and Computing, 2015, Monticello, IL, USA, 792-799. arXiv.org > cs > arXiv:1101.2245
8. Hao, F., Kodialam, M., Lakshman, T. V., Song, H. Fast Dynamic Multiple-Set Membership Testing Using Combinatorial Bloom Filters. IEEE/ACM Transactions on Networking, 2012, 21(1), 295-304. https://doi.org/10.1109/TNET.2011.2173351
9. Lim, H., Lee, N., Lee, J., Yim, C. Reducing False Positives of a Bloom Filter using Cross-Checking Bloom Filter. Applied Mathematics & Information Sciences, 2014, 8(4), 1865-1877. https://doi.org/10.12785/amis/080445
10. Liu, P., Wang, H., Gao, S., Yang, T., Zou, L., Uden, L., Li, X. ID Bloom Filter: Achieving Faster Multi-Set Membership Query in Network Applications. IEEE International Conference on Communications, 2018, Kansas City, MO, USA, 1-6. https://doi.org/10.1109/ICC.2018.8422627
11. Ma, H., Tseng, Y. C., Chen, L. I. A CMAC-Based Scheme for Determining Membership with Classification of Text Strings. Neural Computing with Applications, 2016, 27(7), 1959-1967. https://doi.org/10.1007/s00521-015-1989-6
12. Mitzenmacher, M., Reviriego, P., Pontarelli, S. OMASS: One Memory Access Set Separation. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(7), 1940-1943. https://doi.org/10.1109/TKDE.2016.2535286
13. Qiao, Y., Chen, S., Mo, Z., Yoon, M. When Bloom Filters Are No Longer Compact: Multi-Set Membership Lookup for Network Applications. IEEE/ACM Transactions on Networking, 2016, 24(6), 3326-3339. https://doi.org/10.1109/TNET.2016.2536618
14. Rottenstreich, O., Keslassy, I. The Bloom Paradox: When Not to Use a Bloom Filter. IEEE/ACM Transactions on Networking, 2015, 23(3), 703-716. https://doi.org/10.1109/INFCOM.2012.6195533
15. Sun, Z., Gao, S., Liu, B., Wang, Y., Yang, T., Cui, B. Magic Cube Bloom Filter: Answering Membership Queries for Multiple Sets. IEEE International Conference on Big Data and Smart Computing, 2019, Kyoto, Japan, 32-39. https://doi.org/10.1109/BIGCOMP.2019.8679119
16. Tarkoma, S., Rothenberg, C.E., Lagerspetz, E. Theory and Practice of Bloom Filters for Distributed Systems. IEEE Communications Surveys & Tutorials, 2012, 14(1), 131-155. https://doi.org/10.1109/SURV.2011.031611.00024
17. Xiao, B., Hua, Y. Using Parallel Bloom Filters for Multi-attribute Representation on Network Services. IEEE Transaction on Parallel and Distributed Systems, 2010, 21(2), 20-32. https://doi.org/10.1109/TPDS.2009.39
18. Xu, C., Liu, Q., Rao, W. BMF: An Indexing Structure to Support Multi-element Check. In Cui, B., Zhang, N., Xu, J., Lian, X., Liu, D. (Eds) Web-Age Information Management, Lecture Notes in Computer Science, 2016, 9658, 441-453. https://doi.org/10.1007/978-3-319-39937-9_34
19. Yang, D., Tian, D., Gong, J., Gao, S., Yang, .T, Li, X. Difference Bloom Filter: A Probabilistic Structure for Multi-Set Membership Query. IEEE International Conference on Communications, 2017, 1-6. https://doi.org/10.1109/ICC.2017.7996678
20. Yang, T., Liu, A. X., Shahzad, M., Yang, D., Fu, Q., Xie, G., Li, X. A Shifting Framework for Set Queries. IEEE/ACM Transactions on Networking, 2016, 25(5), 3116-3131. https://doi.org/10.1109/TNET.2017.2730227
21. Yoon, M. K., Son, J. W., Shin, S. H. Bloom Tree: A Search Tree Based on Bloom Filters for Multiple-Set Membership Testing. IEEE International Conference on Computer Communications, 2014, Toronto, Canada, 1429-1437. https://doi.org/10.1109/INFOCOM.2014.6848077
22. Yu, H., Mahapatra, R. N. A Power and Throughput-Efficient Packet Classifier with n Bloom Filters. IEEE Transactions on Computers, 2011, 60(8), 1182-1193. https://doi.org/10.1109/TC.2010.213