A Fingerprint-based Bloom Filter with Deletion Capabilities

Minseok Kwon, Vijay Shankar, Salvatore Pontarelli and Pedro Reviriego

***Abstract*—One drawback of Bloom filters is the inability to delete items due to hash collisions. Counting Bloom filters address this drawback using counters at the expense of increased filter size. Other alternatives like the Deletable Bloom Filter (DlBF) or the Ternary Bloom Filter (TBF) have been proposed to maintain a small filter size while supporting deletions of elements with some probability. These structures offer different trade-offs between deletability, false positive rate and filter size. In this paper, we propose a new filter called the D-FP (Deletable Fingerprint) Bloom filter that stores two-bit fingerprints, instead of a single bit or a counter, that can indicate whether the item can be deleted. By using the fingerprints, the D-FP Bloom filter provides lower false positive rates than the TBF at low filter occupancies and better deletability than the DlBF thus providing more options to desginers that want to trade-off both parameters. In addition to presenting the D-FP Bloom filter, this paper also presents an analysis of the effects of deletions on the performance of the DlBF and TBF. Finally, as a result of the simulations, we observe that the original analytical estimate for the deletable probability of DlBF tends to overestimate the probability.**

***Index Terms*—Bloom filters, counting Bloom filters, item deletion, DlBF, TBF, QBF.**

I. Introduction

Bloom filters (BF) are widely used in Networking applications [1], [2] to speed up processing and they also play an important role in the packet processing in Software Defined Networks. BFs provide probabilistic membership testing allowing false positives in a memory-efficient manner for a wide range of applications that require searching and matching [3]. Two main drawbacks of Bloom filters are possible false positives and no support for item deletion. A counting Bloom filter (CBF) [4] addresses the deletion issue by using counters instead of 0/1 bits; however, this obviously increases its size making it less memory-efficient. The Deletable Bloom Filter (DlBF) [5] allows deletion with some probability but still using only 0/1 bits, not counters. Their idea is to divide the filter into regions and mark each region whether or not it can be removed based on bit collisions. DlBF, however, needs extra space to keep track of information for regions being deletable or not. The Ternary Bloom Filter (TBF) [6] also supports deletion with some probability, but each position is managed independently and saturated when more than one element is mapped to it.

M. Kwon and V. Shankar are with Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, USA, e-mail: mxkvcs@rit.edu.

Salvatore Ponratelli is with Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), Italy, e-mail: salvatore.pontarelli@uniroma2.it.

Pedro Reviriego is with Universidad Carlos III de Madrid, Spain, e-mail: revirieg@it.uc3m.es.

In this paper, we investigate an alternative algorithm called the D-FP Bloom filter that also supports probabilistic deletion of elements. The D-FP Bloom filter stores two bits as a compressed fingerprint in each cell to indicate the insertion of an item instead of a 0/1 single bit. One of the four possible values of these two bits is used to indicate whether the cell can be deleted, not an entire region of multiple cells. This makes the filter deletable probability less dependent on the number of elements inserted than in the DlBF. At the same time, the use of a fingerprint reduces the false positive rate compared to the TBF when the load of the filter is low. Therefore, it provides an intermediate solution between the DlBF and the TBF in terms of deletability and false positive probability.

The rest of the paper is organized as follows. In Section II, we give an overview of the DlBF, the TBF and of the Fingerprint Counting Bloom Filter (FP-CBF), a variant of the CBF. In Section III, we describe the main algorithm of our D-FP Bloom filter. We evaluate their performance in Section IV. Finally, we summarize the conclusion in Section V.

II. Background

A. Deletable Bloom Filter

The deletable Bloom filter (DlBF) [5] supports item deletion (with some probability), which is impossible in Bloom filters, but with less memory use than counting Bloom filters [4]. DlBF also lowers false positive rates, but yet maintains no false negatives, unlike some variants of Bloom filters that enable less false positives at the expense of allowing false negatives [7]. In DlBF, the filter is divided into several regions, and each region is marked to indicate that it can either be deleted or not. A region can be deleted if it has no bit collision, and cannot otherwise. A bit collision arises when a bit in the filter is set by more than one item insertion, that is, at least two hash values are mapped to this particular bit. The bit cannot be reset when an item mapped to it is removed because no information is available on the number of items that are mapped to the bit. Hence, part of the filter in DlBF called the collision bitmap is reserved to keep track of information on whether a region is collision-free or not, and of course, this consumes extra space.

A challenge lies with the choice of the number and size of regions because a trade-off exists between the ability to delete and the size of regions. Specifically, the larger the size of regions, the more items cannot be deleted, since each region spans more bits. Perhaps the size of regions should change dynamically depending on the number of items, but this requires adjustable bitmap size and the re-computation of the filter for the new size.

B. Ternary Bloom Filter

The Ternary Bloom filter (TBF) [6] has been proposed as an alternative to the deletable Bloom filter. Each position can take one of 0, 1 or x, where 0 and 1 mean no or one element mapped to the position, respectively. The meaning of x is similar to the marking of a region when a collision occurs in the DlBF. That is, x indicates that more than one element was mapped at some point in the past, but we no longer know the exact number of elements mapped (there can be even no element mapped if all the elements were deleted). A drawback of using two bits for the three values in a position is to leave one combination unused wasting memory space. To mitigate this, some compression can be used to pack several positions for a large block of bits. An alternative is to use two bits allowing 0, 1, 2, and x, so that we can keep track of positions to which two elements are mapped. This variant is named the Quaternary Bloom filter (QBF) [6], and improves the deletability probability at the cost of a higher false positive rate.

C. Fingerprint Counting Bloom Filter

The Fingerprint Counting Bloom Filter (FP-CBF) [8] is a variant of counting Bloom filters that enables item insertion and removal with no limit other than counter saturation and can achieve lower false positive rates. Each element in the filter has a counter and a fingerprint. When an item is inserted, the k hashes of the item are computed, and the counters of the elements found by these hashes are incremented by one (similar to counting Bloom filters). Also, their fingerprints are updated by performing the *xor* of the fingerprint of the new item and the current fingerprint. To remove an item, all the elements computed again by the same k hashes are found, and their counters are decremented by one. The fingerprints of those elements are updated by performing the *xor* of the fingerprint of the item to be removed and the current fingerprint. For lookup, all the elements pointed by the hash values are checked for the counters and fingerprints. If the counter of any element is zero, the lookup fails. If all the counters are non-zero, then we compare the fingerprint of the item only with the fingerprints whose counter is 1. If they are all equal, the lookup is successful. For counters that are non-zero, but not 1, fingerprints cannot be used for the checking, and thus they always are considered a match. FP-CBF reduces false positive rate significantly compared to counting Bloom filters, without much memory increase using the *xor* bitwise operation.

III. The D-FP Bloom Filter

A. Algorithm

The primary benefit of Bloom filters is its compactness made possible by a single bit used at each cell, which is an element in the filter, mapped by the hash values. One drawback, however, that comes from this setup is information loss due to the limited size of the single bit, ultimately causing false positives. Moreover, this single bit setup hides the number of items inserted into a particular cell (in case of collisions), and makes the deletion process hard. This motivates counting Bloom filters at the expense of increased memory size. To tackle this drawback while not compromising the benefit significantly, we use two bits per cell containing more information with not much burden in terms of memory size (see Figure 1). We use the two bits as the fingerprint of an item stored in the cell. We also use one of the four possible values of the two bits, namely 11, for collisions that represent more than one item is hashed into the cell. This way, an item can easily be determined whether it can be deleted like counting Bloom filters, i.e., the item that has at least one cell that is not '11'. In DlBF, however, an entire region of multiple cells may not be deleted if at least one cell in the region cannot be deleted. Compared to the TBF, the use of fingerprints allows us to reduce the false positive rate when the position has only one element mapped.

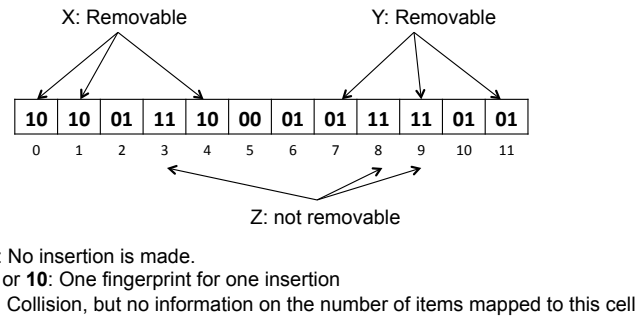

Fig. 1. Illustration of the D-FP Bloom filter.

Suppose that the filter has $\frac{m}{2}$ cells of two bits. An item x is inserted as follows. First, we compute the two-bit fingerprint of x, $fp(x)$, that is either 01 or 10. Then we find the locations of the target cells by computing k hash functions on x. If a cell has 00, then insert $fp(x)$ into the cell; otherwise, insert 11 there. To search for an item y, we first perform the bitwise *and* of $fp(y)$ and the fingerprint stored at each cell mapped by the k hash functions, and then *or* the resulting two bits. For example, if $fp(y) = 10$ and the cell has 01, the result is 0, because the *and* of $fp(y)$ and the cell value is 00, and the *or* of those two bits is 0. If the cell has 11, the result is 1. In general, if the fingerprint is 11, the search always returns 1; if the fingerprint is 10 or 01, the search returns either 0 or 1 with a 50% chance. If all the k values are 1, then the search is successful; if not, the search fails.

An item can be deleted if at least one fingerprint found in its k hash-mapped positions is either 01 or 10 because this indicates only one item is inserted into that cell. After the item is removed, the fingerprint of 10 or 01 is updated to 00, while the fingerprint of 11 remains the same since the number of items inserted there is unknown.

Figure 2 shows some examples of insertion and deletions on the D-FP-BF.

B. Analysis

We seek to estimate the deletable probability and false positive rates of the D-FP Bloom filter theoretically. As defined

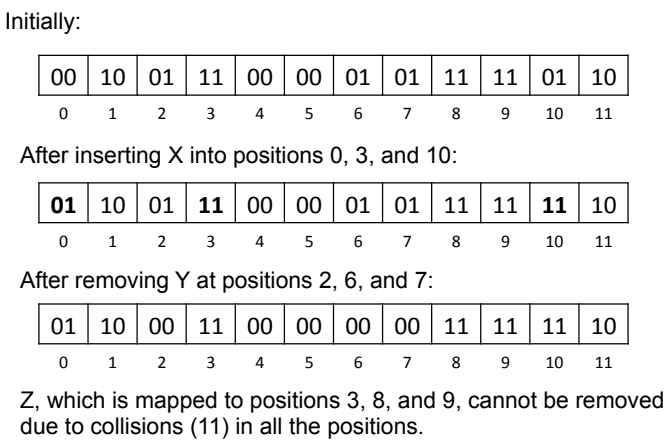

Fig. 2. Examples of the insertion and deletion algorithms for the D-FP Bloom filter.

above, let the filter be again an array of size $\frac{m}{2}$ of two bits. As in Bloom filters, we use k hash functions for item insertion, and insert a total of n items. Given these, the probability p_i that a cell in the filter contains i items mapped to itself can be approximated as:

$$p_i \cong \frac{(\frac{2kn}{m})^i}{i!} e^{-\frac{2kn}{m}}. \quad (1)$$

This follows the well-known fact that the probability that each cell in a Bloom filter has i items mapped to itself can be approximated as a Poisson distribution when the filter size is sufficiently large, i.e., more than a few hundred positions [9]. The approximation of p_i in Eq. 1 is derived from this property.

Given p_i, the probability of an item being deletable (a.k.a. deletability) is:

$$deletability = 1 - (1 - p_0)^k = 1 - (1 - e^{-\frac{2kn}{m}})^k \quad (2)$$

where p_0 denotes the probability that a cell has no other item mapped to it (apart from the one being deleted) after n items are inserted. The probability represents an item with at least one such cell, in which no other items are hashed into, so that the item can be deleted.

The false positive rate fpr of the D-FP Bloom filter is estimated as follows:

$$fpr \cong (1 - p_0 - \frac{p_1}{2})^k = (1 - e^{-\frac{2kn}{m}} - \frac{kn}{m} \cdot e^{-\frac{2kn}{m}})^k. \quad (3)$$

A false positive may occur only when a cell contains 11 or a half of the cases 01 and 10, and thus the complement of when the cell contains 00 or the other half of the cases 01 and 10.

C. Impact of Deletion on Performance

Item deletion causes performance degradation for all the filter data structures considered in this paper including the D-FP-BF, DlBF, TBF, and QBF. This is why these data structures are often used for applications in which the number of deletions is limited. Saturated positions after item deletion is the main reason behind this adverse impact of deletion on the performance. A saturated position occurs when more than one item is mapped to a same position, so that deleting an item mapped to the position does not change the value of the position. This saturated position arises when two or more items are mapped to a position in the D-FP-BF and TBF, and three or more in the QBF. Suppose that we insert $n + r$ elements in the filter, and subsequently remove r of them. Ideally, the performance like deletability and false positive rates should be the same as that of inserting only the n elements. Saturated positions, however, adversely affect, and both the deletability and the false positive rate in this case worsen.

The deletability probability will remain unchanged even after removal of r elements because elements that are not removable (saturated) remain so even with those removals. This assumes that the r elements are not selected for removal discriminately subject to whether the elements are deletable or not. These removals do not thus affect the deletability of the the n elements that remain in the filter after the removals. This means that we can compare deletability of the $n + r$ and n elements in the filter to evaluate the impact of item removals. This reasoning can be applied to all the data structures considered: D-FP-BF, DlBF, TBF and QBF. The impact of removal on the false positive rate is not so easy to evaluate again due to saturated positions or regions. In a TBF, for example, if two elements are mapped to one position and then removed, we cannot decrement the value in the position on deletion as the position is saturated.

IV. Evaluation

We first validate the deletable probability and false positive rate equations obtained for the D-FP Bloom filter by comparing them with simulation results. We then perform the same validation for DlBF [5] to show that the equation for deletable probability in their work is not accurate. Finally, we compare the performance of the D-FP Bloom filter with the DlBF, the TBF and the QBF, both with and without deletions.

The D-FP Bloom filter is implemented and the DlBF, TBF and QBF are also implemented and tested. We simulate with different values of k, n and m, and compare the deletable probability and false positive rate with those estimated by Eqs. 2 and 3. Specifically, we use k = 2, 3, 4, 5, m = 65,536, 131,072 and 262,144, and n varying from $\frac{m}{64}$ to $\frac{m}{k}$ in $\frac{m}{64}$ increments.

For all the configurations, the simulation results match the analytical estimates well except for the DlBF. In DlBF, the results for false positive rates are well-matched to the estimates of Eq. 2 in [5]. In contrast, the results for deletable probability show significant differences with those predicted by Eq. 1 [1] in [5]. This can be explained as the analysis of DlBF in [5] assumes that an element is mapped with the same probability to all the regions. This is not true as regions with collisions are more likely to be selected as they on average have more elements mapped to them. In the following comparison, we take this into account, and use simulation results, not based

[1]Note that there is a typo in Eq. 1 in [5] that should indeed be $1 - p_d$, instead of p_d, as it denotes the probability that all k bits fall in regions with collisions. The typo-corrected formula is used in the comparison.

on the theoretical models, for DlBF. As an example, Figure 3 illustrates the deletable probabilities for $k = 4$, $m = 262,144$ and the region sizes of 4, 8, and 16. It can be seen that the differences between the model and the simulations are non-negligible.

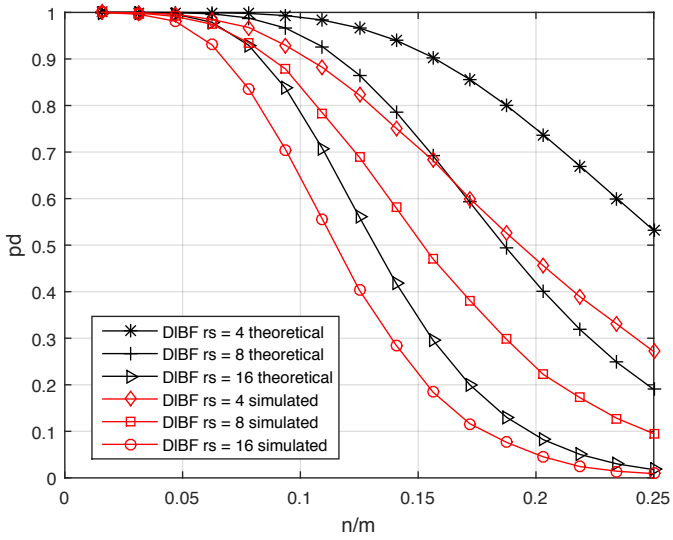

Fig. 3. Comparison of the theoretical and simulated deletable probability for DlBF.

The parameter values that we use for our comparison are as follows: Bloom filter size m = 262,144; number of hash functions k = 2, 3, 4, region Size (DlBf) rs = 4, 8, 16, and a fraction of elements deleted r = 0, 0.1, 0.2, 0.3 and 0.4. The results for deletable probability are depicted in Figure 4 and for the false positive rate in Figure 5 when $r = 0$, both for $k = 4$. Finally, the results for $k = 4$ and $r = 0.2$,0.3 are shown in Figures 6, 7.

The results show that our D-FP-BF provides higher deletable probabilities than the DlBF, but lower than the TBF or QBF. Also, as shown in Figure 5 the D-FP-BF has lower false positive rates than both of the QBF and TBF at low loads when no deletion occurs. For example, the reduction on the false positive rate is larger than two for the lowest load compared to the TBF. For the DlBF, large region sizes are preferably used, but this leads to poor deletability (see Figure 4).

As discussed in Section III-C, the effect on deletability of removing r elements after having inserted $n + r$ can be estimated as the probability that an element is deletable when $n + r$ elements are in the filter. This deletability probability is shown in Figure 4 using the load $n + r$ instead of n. For example, if $r = 0.2 \cdot n$ and $n = 0.10$, the difference between 0.12 and 0.10 is the degradation on deletability. The false positive rates are also measured when items are deleted as r and a fraction of n vary. The results show that the degradation is acceptable only when the number of deletions is small as shown in Figures 6, 7 for $r = 0.2, 0.3$. Note that the D-FP Bloom filter outperforms the TBF at low loads after item removals. For example, again for the lowest load considered the gain is larger than a factor of two.

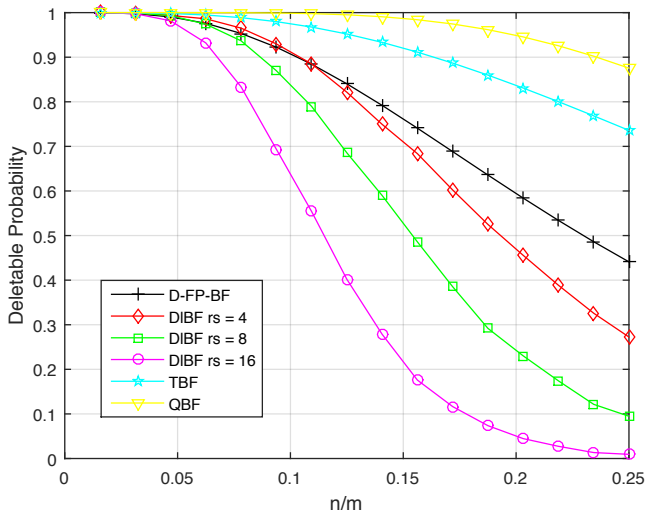

Fig. 4. Comparison of the deletable probability p_d for the D-FP Bloom filter versus DlBF when $k = 4$

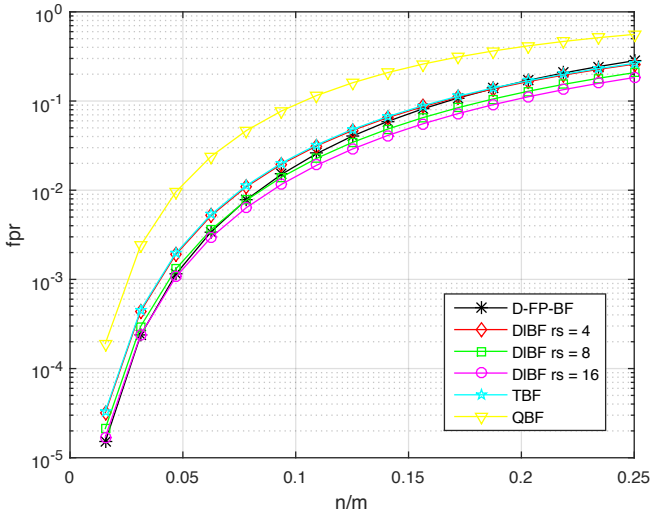

Fig. 5. Comparison of the false positive rate fpr for the D-FP Bloom filter versus DlBF when $k = 4$ and $r = 0$

In summary, the D-FP Bloom filter enables higher deletability than the DlBF for practical region sizes, and lower false positive rate than the TBF and QBF for low filter loads. Hence, it provides designers with another option being useful to trade-off deletability and false positive rate.

V. Conclusion

In this paper, we have presented the D-FP Bloom filter, a data structure that provides approximate membership check with probabilistic deletion. The proposed structure provides a deletability that is worse than that of the TBF but better than that of the DlBF. Conversely, its false positive rate is better than that of the TBF at low loads but can be worse than that of the DlBF depending on the region size. Therefore, it provides an intermediate option between the DlBF and the TBF that can be useful in some applications.

In addition to presenting the D-FP Bloom filter, we have also discussed the effects of deletions on deletability and false

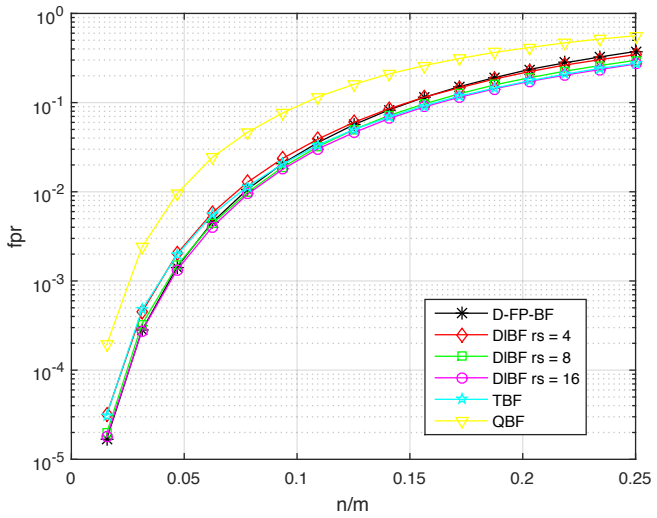

Fig. 6. Comparison of the false positive rate fpr for the D-FP Bloom filter versus DlBF when $k = 4$ and $r = 0.2$

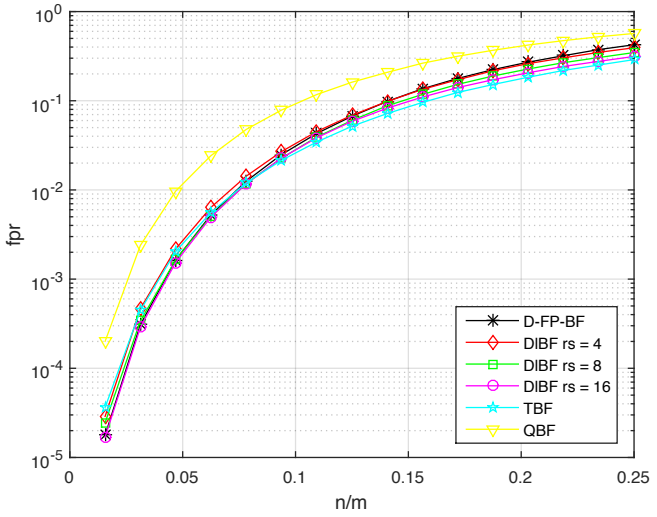

Fig. 7. Comparison of the false positive rate fpr for the D-FP Bloom filter versus DlBF when $k = 4$ and $r = 0.3$

positive rates for all the structures considered. The analysis of deletability yields an interesting result as the degradation can be computed directly from the deletability of an static analysis. This result will be useful to better understand the performance of the DlBF, TBF and QBF. In terms of false positive rate, the effect has been evaluated by simulation showing that the structures tolerate the removal of a small percentage of the elements with a limited degradation. The derivation of an analytical model for the false positive rate when elements are deleted is an interesting topic for future work.

Finally, as part of the comparison with the DlBF, we have discovered that the analytical estimate for the deletable probability presented in the original paper is not accurate. The differences can be significant and therefore revisiting the modeling of the DlBF can be an interesting topic for future work.

ACKNOWLEDGMENTS

Pedro Reviriego would like to acknowledge the support of the TEXEO project TEC2016-80339-R funded by the Spanish Ministry of Economy and Competitivity and of the Madrid Community research project TAPIR-CM grant no. P2018/TCS-4496. Salvatore Pontarelli is partially supported by the European Commission in the frame of the Horizon 2020 project 5G-PICTURE (grant #762057).

REFERENCES

[1] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Mathematics*, vol. 1, pp. 485–509, 2004.

[2] L. Luo, D. Guo, R. T. B. Ma, O. Rottenstreich, and X. Luo, "Optimizing Bloom filter: Challenges, solutions, and comparisons," *IEEE Communications Surveys Tutorials*, 2019.

[3] B. Bloom, "Space/Time Tradeoffs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, pp. 422–426, 1970.

[4] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 281–293, 2000.

[5] C. E. Rothenberg, C. A. B. Macapuna, F. L. Verdi, and M. F. Magalhaes, "The deletable bloom filter: a new member of the bloom family," *IEEE Communications Letters*, vol. 14, no. 6, pp. 557–559, June 2010.

[6] H. Lim, J. Lee, H. Byun, and C. Yim, "Ternary bloom filter replacing counting bloom filter," *IEEE Communications Letters*, vol. 21, no. 2, pp. 278 – 281, 2017.

[7] B. Donnet, B. Baynat, and T. Friedman, "Retouched bloom filters: Allowing networked applications to trade off selected false positives against false negatives," in *ACM CoNEXT*, 2006, pp. 13:1–13:12.

[8] S. Pontarelli, P. Reviriego, and J. A. Maestro, "Improving counting bloom filter performance with fingerprints," *Information Processing Letters*, vol. 116, no. 4, pp. 304 – 309, 2016.

[9] K. Christensen, A. Roginsky, and M. Jimeno, "A new analysis of the false positive rate of a bloom lter," *Information Processing Letters*, vol. 110, no. 21, pp. 944–949, Oct 2010.