On the Controversy of Bloom Filter False Positives - An Information Theoretical Approach to Optimizing Bloom Filter Parameters

***Abstract*—Although Bloom Filters (BF) have been widely used in many networking applications and beyond, the fundamental issue of how to calculate the false positive probability remains elusive. Properly calculating the false positive probability of BF is critical because it is used to calculate the optimal number of hash functions k. Since Bloom gave the false positive formula in 1970, in 2008, Bose *et al.* pointed out that Bloom's formula for calculating the false positive probability is flawed and gave a new false positive formula; and in 2010, Christensen *et al.* further pointed out that Bose's formula is also flawed and gave another new false positive formula. Although Christensen's formula is perfectly accurate, it is time-consuming to calculate the false positive probability, and it is impossible to calculate the derivation of the optimal value of k. While the conventional wisdom is to derive the optimal value of k from a false positive formula, in this paper, we propose the first approach to calculating the optimal k without any false positive formula. Our approach is based on the following observation: for a BF with m bits and n elements, if and only if its entropy is the largest, its false positive probability is the smallest, according to information entropy theory. Furthermore, we propose a new and more accurate upper bound for the false positive probability. When the size of a Bloom filter becomes infinitely large, our upper bound turns equal to the lower bound, which becomes Bloom's formula. This deepens our understanding of Bloom's formula: it is perfectly accurate when m is infinitely large, and it is practically accurate when m is sufficiently large. Besides, we derive the bounds of correct rate of counting Bloom filters (one widely used variant of BFs for estimating item frequencies) by applying our proposed formulas about BFs to them. We release our source code of Bloom Filters in [1] without any identity information.**

I. Introduction

A. Motivation

A Bloom Filter (BF) is a compact data structure used for quickly checking whether an element belongs to a set or not [2]. Given a set S of n elements, we create a bit array A of length m as follows. First, we initialize each bit of A to 0, then for each element $x \in \mathcal{S}$, we use k hash functions to compute k hash values: $h_1(x), h_2(x), ..., h_k(x)$ where each hash value is in the range $[1, m]$. Second, for each $1 \leqslant i \leqslant k$, we let $A[h_i(x)] = 1$. The resulting bit array A is called the BF for set S. To query whether $y \in S$, we first use the same k hash functions to compute k hash values: $h_1(y), h_2(y), ..., h_k(y)$. Second, we check whether the corresponding k bits in A are all $1s$ (*i.e.*, whether $A[h_1(y)] \wedge A[h_2(y)] \wedge \cdots, A[h_k(y)] = 1$ holds); if yes, then $y \in S$ may probably hold and we can further check whether $y \in S$; if no, then $y \in S$ definitely does not hold. The cases that the BF shows that $y \in S$ may hold but actually $y \notin S$ are called false positives (FP). The FP probability f can be calculated from n, k, and m. Thus, given a set of n elements and the required FP probability f, we can calculate the relationship between k and m. Based on the calculated relationship between k and m, we can properly trade off between space and speed: smaller m means smaller space, and smaller k means the smaller number of hash function calculations. In typical BF applications, as m is determined by the memory budget for the BF, with known values of n and f, we calculate the optimal value for the only unknown parameter k.

As set membership query is a fundamental operation in many networking applications, and BFs have the advantages of small memory consumption, fast query speed, and no false negatives, BFs have been widely used in a wide variety of networking applications, such as web caching [3], [4], IP lookup [5]–[8], packet classification [9], [10], regular expression matching [11], [12], multicast [13], [14], content distribution networks [15], [16], routing [17], [18], P2P networks [19], [20], overlay networks [21], [22], name based networks [23], [24], queue management [25], [26], Internet measurement [27], [28], IP traceback [29], [30], sensor networks [31], [32], data center networks [33]–[35], cloud computing [36], [37], cellular networks [38], [39], and more [40], [41]. Most applications with set membership query can potentially be optimized using BFs.

Although BFs have been widely used in many applications, the fundamental issue of how to calculate FP probability remains elusive. Properly calculating the FP probability of BF is critical because it is used to calculate the optimal value of the important parameter k, the number of hash functions. In [2], Bloom gave a formula for calculating the FP probability with known parameters n, k, and m. Based on Bloom's formula, we can also easily compute the optimal value of the parameter k when m and n are known. This formula has been believed to be correct until 2008 when Prosenjit Bose *et al.* pointed out that Bloom's formula is flawed and gave a new FP formula [42]. Interestingly, two years later, Ken Christensen *et al.* pointed out that Bose's formula is also flawed and gave a new FP formula [43]. So far, it is believed that Christensen's formula is perfectly accurate. However, Both Bose's and Christensen's FP formulas are too complicated to calculate the optimal value of k from given values of n and m.

B. Main Contributions

While the conventional wisdom is to derive the optimal value of BF parameter k from the FP probability, in this paper, we propose the first approach to calculating the optimal k without any FP formula. We first observe that for a BF with m bits and n elements, if and only if its entropy is the largest, its false positive probability is the smallest, according to information entropy theory. Based on this observation, our approach is to derive a formula for calculating the optimal k by letting the entropy equal to 1. We also propose another method to calculate FP probability by deriving the left and right limit expressions of FP probability. We prove that when m goes to infinity, the left and right limits are the same, which is essentially the FP probability. Interestingly, our derived FP formula is the same as Bloom's formula in [2]. This deepens our understanding of Bloom's formula: it is perfectly accurate when m is infinitely large, and it is practically accurate when m is sufficiently large.

In summary, we make three key contributions in this paper. First, we propose an information theoretical approach to calculating the optimal value of BF parameter k without calculating FP probability. Second, we propose a new upper bound which is much more accurate than state-of-the-art. When m is infinitely large, our upper bound becomes the same as the lower bound. This result formally proves that Bloom's formula is practically accurate when m is sufficiently large. Third, we conducted experiments to validate our findings. In particular, we show that the error of Bloom's formula is negligibly small when m is large. Furthermore, we release our source code of Bloom Filters in [1] without any identity information.

The rest of this paper proceeds as follows. In Section II, we introduce the controversy on FP probability. In Section III, we show the derivation of the optimal number of hash functions using the information entropy theory. In Section IV, we present a new upper bound of the false positive probability of Bloom Filters. In Section V, we derive the bounds of correct rate of counting Bloom filters through our proposed formulas about Bloom Filters. In Section VI, we conduct experiments to evaluate the error of Bloom Filters. We conclude the paper in Section VII.

II. Prior Art on BF False Positives

In this section, we review the prior art on calculating the false positive probability of Bloom filters. Table I summarizes the notations used in this paper.

A. Bloom's False Positive Formula

In 1970, Bloom calculated the false positive probability of a Bloom filter as follows [2]. Given a set $\mathcal{S}$ of elements, let n be the number of elements in $\mathcal{S}$, k be the number of hash functions, and m be the number of bits in the Bloom filter A constructed from set $\mathcal{S}$. In querying an element x, the false positive happens when the Bloom filter reports that $x \in \mathcal{S}$ (*i.e.*, $A[h_i(x)] = 1$ holds for each $1 \leqslant i \leqslant k$), but actually $x \notin \mathcal{S}$. Consider an arbitrary bit $A[b]$ in A. For any element

TABLE I: Notations and abbreviation used in this paper

Symbol	Description
$\mathcal{S}$	Set of elements
m	BF size
n	Number of elements in $\mathcal{S}$
k	Number of hash functions
k^*	Optimal number of hash functions
f	False positive probability
f_{bloom}	False positive probability calculated by Bloom
f_{bose}	False positive probability calculated by Bose
f_{christ}	False positive probability calculated by Christensen
f_{true}	True false positive probability of BF
FP	false positive
BF	Bloom filter

in $\mathcal{S}$ and any hash function h_i ($1 \leqslant i \leqslant k$), the probability that this element is not hashed to bit $A[b]$ by h_i is $1 - 1/m$. As $\mathcal{S}$ has n elements and each element is hashed k times, the probability of $A[b] = 0$ is p':

$$p' = \left(1 - \frac{1}{m}\right)^{kn} \tag{1}$$

Thus, the probability of $A[b] = 1$ is $1 - (1 - 1/m)^{kn}$. For any element $x \notin \mathcal{S}$, the probability that the false positive happens for x, *i.e.*, the probability of $A[h_1(y)] \wedge A[h_2(y)] \wedge \cdots, A[h_k(y)] = 1$, is calculated as follows:

$$f_{bloom} = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \tag{2}$$

This formula can be approximated by $(1 - e^{\frac{-nk}{m}})^k$, based on which we can trade off between space indicated by m and time indicated by k.

B. Bose's Derivation

In 2008, Bose *et al.* pointed out that the last step in Bloom's derivation is flawed because for any element $x \notin \mathcal{S}$, which is hashed into k bits $A[h_1(x)], A[h_2(x)], \cdots, A[h_k(x)]$, the k events $A[h_1(x)] = 1, A[h_2(x)] = 1, \cdots, A[h_k(x)] = 1$ are not actually independent [42]. Although for each bit $A[h_i(x)]$ ($1 \leqslant i \leqslant k$), after inserting n elements into array A, the probability of $A[h_i(x)] = 1$ is $1 - (1 - 1/m)^{kn}$, for the probability of $A[h_1(y)] \wedge A[h_2(y)] \wedge \cdots, A[h_k(y)] = 1$ to be $(1 - (1 - 1/m)^{kn})^k$, the k events $A[h_1(x)] = 1, A[h_2(x)] = 1, \cdots, A[h_k(x)] = 1$ need to be independent. We now analyze the reason that the k events $A[h_1(x)] = 1, A[h_2(x)] = 1, \cdots, A[h_k(x)] = 1$ are not independent. Let us first consider the probability of $A[h_1(x)] = 1$, which is $1 - (1 - 1/m)^{kn}$. Second, we consider the probability of $A[h_2(x)] = 1$ based on the condition that $A[h_1(x)] = 1$. There are two possibilities for $h_2(x)$: (1) $h_2(x) \neq h_1(x)$, and (2) $h_2(x) = h_1(x)$. For the first case, the probability of $A[h_2(x)] = 1$ is $1-(1-1/(m-1))^{kn}$. For the second case, the probability of $A[h_2(x)] = 1$ is 1. Similarly, we can analyze the $A[h_3(x)] = 1$ based on the condition that $A[h_2(x)] = 1$ and $A[h_1(x)] = 1$, etc. Observing the dependency of the k events $A[h_1(x)] = 1, A[h_2(x)] = 1, \cdots, A[h_k(x)] = 1$, Bose *et al.* derive the following false positive formula:

$$f_{bose} = \frac{1}{m^{k(n+1)}} \sum_{i=1}^{m} i^k i! \binom{i}{m} \left(\frac{1}{i!} \sum_{j=0}^{i} (-1)^j \binom{i}{j} j^{kn} \right) \quad (3)$$

Bose *et al.* derived asymptotically closed forms for the upper and lower bounds of the above formula:

$$f_{bloom} < f_{bose} \leqslant f_{bloom} \times \left(1 + \mathcal{O}\left(\frac{k}{p} \sqrt{\frac{\ln m - k \ln p}{m}} \right) \right) \quad (4)$$

where $p = 1 - p' = 1 - \left(1 - \frac{1}{m}\right)^{kn}$. These bounds hold under the condition that

$$\frac{k}{p} \sqrt{\frac{\ln m - k \ln p}{m}} \leqslant c \quad (5)$$

for some constant $c < 1$. Bose *et al.* further showed that for $k \geqslant 2$, f_{bose} is strictly larger than f_{bloom}, and the lower bound converges to f_{bloom} when m becomes infinitely large.

C. Christensen's Derivation

In 2010, Christensen *et al.* pointed out that Bose's formula has a mistake that the term $(-1)^j$ should be $(-1)^{i-j}$, but the lower and upper bounds in Eq. 4 are correct. Christensen *et al.* derived the finally correct false positive formula for Bloom filters as follows:

$$f_{christ} = \frac{m!}{m^{k(n+1)}} \sum_{i=1}^{m} \sum_{j=0}^{i} (-1)^{i-j} \frac{j^{kn} i^k}{(m-i)!j!(i-j)!} \quad (6)$$

Although Christensen's formula is perfectly accurate, it is not much useful in practice. First, given the Bloom filter parameters n, m, and f, it is difficult to calculate the optimal k value as Christensen's formula does not give a closed form expression for calculating the optimal k. Second, given the Bloom filter parameters n, m, and k, the algorithm by Christensen *et al.* takes $\mathcal{O}(knm)$ time to calculate the false positive probability f, which is time-consuming.

III. Computing the Optimal k

Traditionally, the optimal number of hash functions is derived through finding the extrema of the asymptotic formula of f_{bloom} given in Eq. 2 as follows:

$$k^*_{bloom} = \frac{m}{n} \ln 2 \quad (7)$$

However, as we have discussed, the underlying formula for FP probability is not fully correct. In this section, we first present the important theorems that correlate information entropy and the false positive rate of Bloom filters. Then we propose a method of deriving the optimal value of k by minimizing the FP probability given the value of BF size m and number of inserted elements n.

A. Information Entropy Basis

Information Entropy: In information theory, *information entropy* is used to measure the uncertainty of a random variable. In this paper, we use *Shannon entropy* [44], which measures the value of the information contained in a variable. Entropy is typically measured in bits, nats, or bans [45]. For a variable with s events with the probabilities of p_1, p_2, ..., p_s. The information entropy E is defined as:

$$E = \sum_{i=1}^{s} p_i log_2 \frac{1}{p_i} \quad (8)$$

Property of Information Entropy: For any variable or message, if its information entropy is not at the maximum, it can be compressed without information losses. For a random variable, the larger the uncertainty is, the bigger the information entropy is. Suppose an m-bit string variable is compressed to m'-bit string variable, the entropy becomes E' from E. Then we have $mE = m'E'$. According to the information entropy formula (Eq. 8), we can obtain the information entropy E of the m-bit variable of a BF as follows, where p' is defined in Equation 1.

$$E = -\left(p' \log_2 p' + (1 - p') \log_2 (1 - p')\right) \quad (9)$$

We illustrate the relationship between entropy E and p' in Figure 1. We can observe that the entropy of an m-bit sequence reaches the maximum when the probability that each bit in the BF is 1 (or 0) is 50%.

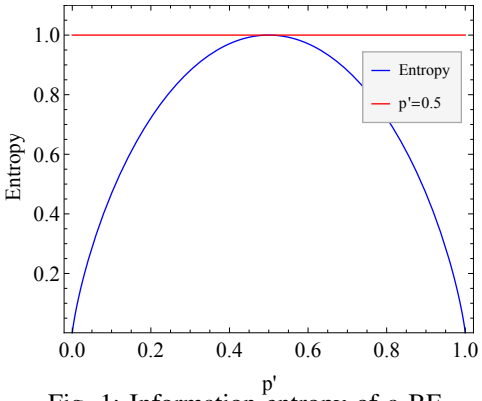

Fig. 1: Information entropy of a BF

B. Relation Between False Positive Probability and Information Entropy

We now show the relation between the false positive probability and information entropy. We first present a lemma and some definitions.

Lemma I: Given a BF, suppose n and k keep unchanged, when m becomes larger, the FP probability gets smaller.

Proof. When n and k are fixed, for larger m, the probability of each bit being 1 in the BF becomes smaller, *i.e.*, p' becomes smaller; thus, the FP probability gets smaller. □

Definition I: *Bloom filter variable.* The false positive rate of Bloom filters is determined by m, n, and k. For different n

elements, the m-bit string varies. Thus, when the values of m, n, k are given, the m-bit string is a *random variable*. When the n elements are given, the m-bit string is a random variable instance. Therefore, when the values of m, n, k are given, we call it a Bloom Filter Variable (BFR). Since it is a random variable, we can compute its information entropy.

Definition II: *Equivalent Bloom filter variables.* Given two Bloom filters variables v_1 and v_2, for the same n elements, there are a pair of BFR instances. Given a set with n elements, if these pairs of BFR instances always report the same result: true or false for any input element, we say v_1 and v_2 are equivalent.

Theorem I: Given a Bloom filter *variable* v_1 with parameters m, n, and k, if its information entropy is not at the maximum, there must exist a smaller equivalent Bloom filter variable v_2 with parameters m', n and k, where $m' < m$.

Proof. For v_1, the parameters are m, n, k. Suppose its k hash functions are $h_1(\cdot), h_2(\cdot), \ldots, h_k(\cdot)$. Since the assumption is that the information entropy of v_1 is not at the maximum, according to the property of information entropy, v_1 can be compressed *without information loss*. After compression, suppose the new random variable has a length of $m'(m' < m)$, we name it v_2. Note that during the compression, the information value $m * E$ keeps unchanged, while $E's$ maximum value is 1; thus, the length of the compressed message has a minimum value. Here v_1 and v_2 are two variables consisting of bits, and we can treat them as two integer variables. We use $In(v_1)$ and $In(v_2)$ to represent the integer value of v_1 and v_2. Furthermore, we use $|In(v_1)|$ represents the length of v_1, then we have $|In(v_1)| = m$, $|In(v_2)| = m'$. Because we compress v_1 and get v_2, this can be regarded as a function $g(\cdot)$. In other words, $g(In(v_1)) = In(v_2)$. We can also obtain v_1 by equation $In(v_1) = g^{-1}(In(v_2))$.

At this stage, we consider the new Bloom filter variable v_2, the parameters are m', n, and k. Note that we use k `different` hash functions, and the k hash functions are

$$\begin{aligned} &g^{-1}(In(v_2)) \ll h_1(y) \gg (|g^{-1}(In(v_2))| - h_1(y) - 1),\\ &g^{-1}(In(v_2)) \ll h_2(y) \gg (|g^{-1}(In(v_2))| - h_2(y) - 1),\\ &\ldots\\ &g^{-1}(In(v_2)) \ll h_k(y) \gg (|g^{-1}(In(v_2))| - h_k(y) - 1) \end{aligned} \tag{10}$$

Where '$\ll$' means left shift and '$\gg$' means right shift.

Given an input element y, we can compute the above k values only using v_2 and $h_i(\cdot)$ without v_1. Then we need to prove that for any incoming element y, v_2 reports the same k-bit value. With formula 10, we use equation $v_1 = g^{-1}(In(v_2))$ and $m = |g^{-1}(In(v_2))|$. These k hash functions are simplified as

$$\begin{aligned} &v_1 \ll h_1(y) \gg (m - h_1(y) - 1),\\ &v_1 \ll h_2(y) \gg (m - h_2(y) - 1),\\ &\ldots\\ &v_1 \ll h_k(y) \gg (m - h_k(y) - 1) \end{aligned} \tag{11}$$

Here $1 \leqslant i \leqslant k$ and $0 \leqslant h_i(y) \leqslant m - 1$. Here $v_1 \ll h_i(y) \gg (m - h_i(y) - 1)$ actually means the value of $h_i(y)$-th bit of v_1. This is the same as the k hash functions as v_1. Therefore, v_1 and v_2 are equivalent. □

Theorem II: Given a Bloom filter variable, *if and only if* its information entropy is at the maximum, its FP probability is at the minimum.

Proof. First, we prove that if the FP probability is at the minimum, then its information entropy must be at the maximum. Given a Bloom filter variable v_1 with parameters m, n, k. Since the assumption is that the information entropy of v_1 is not at the maximum, according to Theorem I, there exists a smaller Bloom filter variable v_2 with parameters m', n, k. Since v_2 and v_1 are equivalent, their FP probabilities are the same, we name it f. At this stage, we enlarge the size of v_2 a little from m' to m'', where $m' < m'' < m$. According to Lemma I, we know the FP probability of v_2 becomes smaller than f. This means that for v_1, there exists a BF variable with a smaller size and a smaller FP probability. Therefore, the FP probability of v_1 is not at the minimum. This means that if its information entropy is not at the maximum, then the FP probability is definitely not at the minimum. The contrapositive is that if the FP probability is at the minimum, then its information entropy must be at the maximum.

Second, we prove that if its information entropy is at the maximum, the FP probability is at the minimum. Given a Bloom filter variable v_1 with parameters m, n, k. Since the assumption is that the FP probability of v_1 is not at the maximum, there exists an optimal Bloom filter variable v_0 with parameters m, n, k', where $k' \neq k$. According to $\leftarrow$, the information entropy of v_0 is at the maximum. According to Eq. 9 and Figure 1, the p' of v_0 is 0.5 whereas the p' of v_1 is not because they have different value of k. Therefore, the information entropy of BF_1 is not at the maximum. This means if the FP probability is not at the minimum, its information entropy must be not at the maximum. The contrapositive is that if its information entropy is at the maximum, the FP probability is at the minimum. □

C. Computing the Optimal k

According to Theorem II, when the information entropy of the Bloom filter variable is at the maximum, the FP probability is at the minimum. Recalling the definition of p' in Eq. 1, one can use this interpretation to find k^*, *i.e.*, the optimal number of hash functions. From Figure 1, we know that when p' is 0.5, E reaches the maximum value 1. By setting the value of p' to 0.5, we have

$$p' = \left(1 - \frac{1}{m}\right)^{k^* n} = 0.5 \tag{12}$$

Further, we have:

$$k^* = -\frac{\ln 2}{n} / \ln\left(1 - \frac{1}{m}\right) \tag{13}$$

This formula is very close to the formula of k^* obtained by Bloom. When x is very small, $\ln(1 + x) \approx x$, and therefore $-1/\ln(1 - \frac{1}{m}) \approx m$, resulting the same term as in Eq. 7.

Theorem III: Given any BF variable, when m and n are fixed, the FP probability f is a function of k, we represent it $f(k)$. Then $f(k)$ is a *convex function*, which has only one minimum value.

Proof. Given a Bloom filter variable v_1 with parameters m, n, k_1, its entropy is E_1. Given another Bloom filter variable v_2 with parameters m, n, k_2, its entropy is E_2. (1) For any $k_1 < k_2 \leqslant k^*$, according to Eq. 1, Eq. 9 and Figure 1, we known $p'_1 < p'_2 \leqslant 0.5$. We compress v_1 to v_3 with parameters m_3, n, k_1. To make v_3's entropy equal to E_2, m_3 should be mE_1/E_2. In this case, the entropy of v_3 is equal to that of v_2. When the entropy of BF variables is less than 0.5, the same entropy leads to the same p'. In other words, $p'_3 = p'_2$. Because v_2 has more hash functions ($k_2 > k_1$), the FP probability of v_2 is smaller than that of v_3. While v_3 and v_1 have the same FP probability, therefore, the FP probability of v_2 is smaller than that of v_1. In other words, for any $k(k < k^*)$ increasing, the FP probability of BFs decreases. (2) For any $k^* \leqslant k_1 < k_2$, according to Eq. 1, Eq. 9 and Figure 1, we known $p'_1 > p'_2 \geqslant 0.5$. Using the similar derivation, we can derive that the FP probability of v_2 is larger than that of v_1. According to the above two cases, we know that given any BF variable, when m and n are fixed, the FP probability f is a function of k, we represent it $f(k)$. So $f(k)$ is a *convex function*. □

IV. Asymptotic Form of the FP Probability

In this section, we derive a new approach to computing the asymptotic form for the FP probability of BFs. The new derivation is based on *partitioned Bloom Filters* (pBF) that are used frequently to carry out parallel queries. Its underpinning principle is simple: the BF is divided into k even partitions, and each hash function only acts on one of the partitions, respectively. The probability that one bit of the BF array remains 0 after inserting n elements in the BF becomes the following as now each hash maps into $\frac{m}{k}$ separate bits.

$$p'_{partition} = \left(1 - \frac{k}{m}\right)^n \tag{14}$$

It is intuitive that the FP probability of partitioned BF is a little bigger than that of BF. Unfortunately, there is no strict proof. Here we show one proof method, which is based on the following Lemma.

Lemma II: For $m > 1\ , k > 1,\ \ n > 1,\ m > k$,

$$\left(1 - \frac{k}{m}\right)^n < \left(1 - \frac{1}{m}\right)^{kn} \tag{15}$$

Proof. Note that when m is large, the left expression approximates to the right one. Below we give the derivation details. Let $g(k) = (1 - \frac{1}{m})^k - (1 - \frac{k}{m})$. For $m > 1$ and $k > 1$, $g(k)$ is a continuous and derivable function, and we can obtain the following inequality in terms of its derivative:

$$g'(k) > \left(1 - \frac{1}{m}\right) \ln\left(1 - \frac{1}{m}\right) + \frac{1}{m} \tag{16}$$

Let $f(m) = \left(1 - \frac{1}{m}\right) \ln\left(1 - \frac{1}{m}\right) + \frac{1}{m}$. Then $f'(m) = \frac{1}{m^2} \ln\left(1 - \frac{1}{m}\right) < 0$, which means that the function $f(m)$ is strictly decreasing. When m goes to infinity, we have $\lim_{m\to\infty} f(m) = 0$. Therefore, we know that $f(m) \geqslant 0$, and $g'(k) > f(m) \geqslant 0$, which means that the function $g(k)$ is strictly increasing. Thus, we have

$$\left(1 - \frac{k}{m}\right)^n < \left(1 - \frac{1}{m}\right)^{kn} \tag{17}$$

□

The above lemma shows that $p'_{partition} < p'_{true}$ or equivalently $1 - p'_{partition} > 1 - p'_{true}$. Thus, we know that with the same parameters, the FP probability of the pBF will be larger than that of the standard BF f_{true}, *i.e.*, $f_{partition} > f_{true}$. In addition, Boses bounds in Eq. 4 state that the precise value of FP probability for a BF f_{true} is larger than f_{bloom}, *i.e.*, $f_{true} > f_{bloom}$. Therefore, we have the following upper and lower bounds:

$$f_{partition} > f_{true} > f_{bloom} \tag{18}$$

For a partitioned BF, the probability that one bit of the array is still 0 p' is shown in Eq. 14. Different from standard Bloom Filters, for a partitioned Bloom Filter, the event $E(h_1 = 1), E(h_2 = 1), E(h_3 = 1), ..., E(h_{i-1} = 1)$ is independent of the event $E(h_{i-1} = 1)$, where $E(h_{i-1} = 1)$ means that the event that the position of $h_{i-1}(x)$ is 1 because each hash function is responsible for one partition, and has no impact on each other. Therefore, we have

$$f_{partition} = (1 - p'_{partition})^k = \left(1 - \left(1 - \frac{k}{m}\right)^n\right)^k \tag{19}$$

Then the formula 18 becomes

$$\left(1 - \left(1 - \frac{k}{m}\right)^n\right)^k > f_{true} > \left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^k \tag{20}$$

Then we use the well known limit formula:

$$\lim_{x\to\infty} \left(1 - \frac{1}{x}\right)^{-x} = e \tag{21}$$

Asymptotically, when m becomes large, we already know that f_{bloom} converges to the term in Eq. 2. Nevertheless, the upper bound has also an asymptotic behaviour as the following, which is the same term as the lower bound limit.

$$\lim_{m\to\infty} \left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^k = \left(1 - e^{-nk/m}\right)^k \tag{22}$$

Through the Sandwich Theorem (also known as squeeze theorem) we obtain the following equation, which is similarly to Christensen and Bose:

$$\lim_{m\to\infty} f_{true} = \left(1 - e^{-nk/m}\right)^k \tag{23}$$

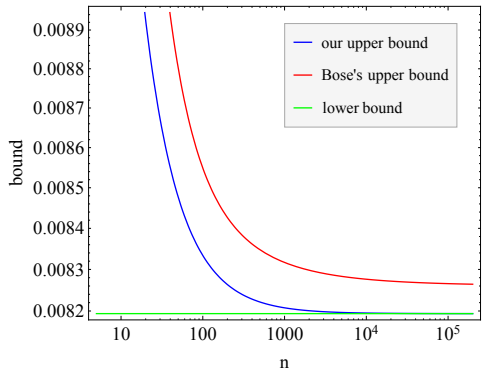

Fig. 2: Upper and lower bound for f_{true} for $k = 7$ and $m = 10n$.

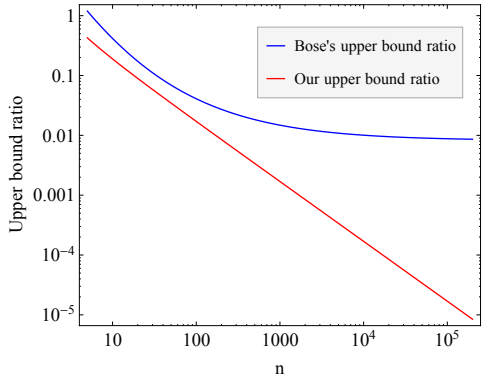

Fig. 3: Bounds error ratio for Bose's bound and the bound derived in this paper for $k = 7$ and $m = 10n$.

This means that when m is large, the Bloom's formula can be used with negligible error. However, we still need to evaluate what means m *being large*. We will do this by comparing the two bounds we have in hand: the one from Bose and the one we derived in this paper. We show in Figure 2 that the two upper bounds along with the lower bound obtained for $k = 7$ and $m = 10n$ as a function of n, the number of elements inserted in the BF. As can be seen, the upper bound derived in this paper and the lower bound f_{bloom} converge relatively fast for $n = 9$, while the upper bound derived by Bose has a much slower convergence. We can see this better by looking at the behavior of the bounds error ratio β, defined as $\beta = \frac{upper\ bound - lower\ bound}{lower\ bound}$, for the two bounds in Figure 3.

As can be seen, the gap between our derived upper bound and f_{bloom} is decreasing polynomially at a constant speed, while Bose's bound has a lower speed of convergence. In order to extend this observation, we show in Figure 4 the evolution of the bounds error ratio for a BF with $m = 10000$, $n = 1000$ and varying k.

As expected, error involved with using f_{bloom} increases with the number of hash functions k increases. However, it can be seen that the convergence behavior of the bounds derived in this paper is much better than the one obtained by Bose.

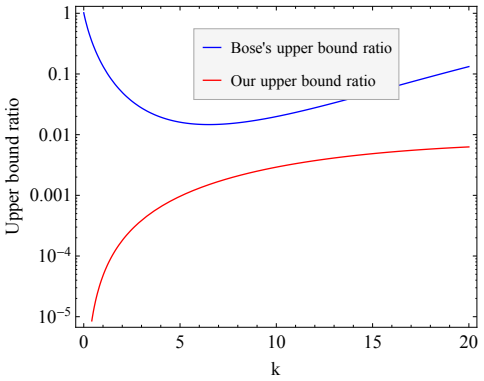

Fig. 4: Bounds error ratio for Bose's bound and the bound derived in this paper for $m = 10000$, $n = 1000$ and varying k.

V. Correct Rate of Counting Bloom Filters

The counting Bloom filter (CBF) [46], one widely used variant of standard Bloom filter, replaces each bit with one counter, supporting estimating the frequency of each element in a multiset. Specifically, given a multiset $\mathcal{S}$ of n distinct elements with their corresponding frequencies, we create a counter array A of length m as follows. First, we initialize each counter of A to 0, then for each element $x \in \mathcal{S}$, we use k hash functions to compute k hash values: $h_1(x), h_2(x), ..., h_k(x)$ where each hash value is in the range $[1, m]$. Second, for each $1 \leqslant i \leqslant k$, we let $A[h_i(x)] = A[h_i(x)] + 1$. Let f_x be the frequency of element x in multiset $\mathcal{S}$. Therefore, the step of $A[h_i(x)] = A[h_i(x)] + 1$ for each $1 \leqslant i \leqslant k$ will occur f_x times. The resulting counter array A is called the CBF for multiset $\mathcal{S}$. To query the frequency of an element y in multiset $\mathcal{S}$, we first use the same k hash functions to compute k hash values: $h_1(y), h_2(y), ..., h_k(y)$. Second, we report the minimum value of the k counters: $A[h_1(x)], A[h_2(x)], ..., A[h_k(x)]$ as the estimated frequency of this element. Obviously, the estimated frequency reported by the CBF is always larger than or equal to the real frequency for any element in multiset $\mathcal{S}$. The case that the estimated frequency from the CBF is equal to the real frequency for one element is called the correct case. The probability of such case happening is called the correct rate of the CBF ($\mathcal{C}_r$).

The calculation of the correct rate of CBFs can benefit from our derivation of the false positive probability of standard Bloom filter. In querying an element x, the correct case happens when there exists at least one hashed counter (among $A[h_1(x)], A[h_2(x)], ..., A[h_k(x)]$) that is not hashed by any elements in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$. The contrapositive is that the correct case does not happen when all the k hashed counters are also hashed by some elements in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$. Consider an arbitrary counter $A[b]$ in A. For any distinct element in $\mathcal{S} \setminus \{x \cdot f_x\}$ and any hash function h_i ($1 \leqslant i \leqslant k$), the probability that this element is not hashed to counter $A[b]$ by h_i is $1 - 1/m$. As $\mathcal{S} \setminus \{x \cdot f_x\}$ has $n - 1$ distinct elements and each distinct element is hashed k times, the probability

that $A[b]$ is not hashed by any element in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$ is p_c:

$$p_c = \left(1 - \frac{1}{m}\right)^{k(n-1)} \tag{24}$$

Thus, the probability that $A[b]$ is hashed by some elements in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$ is $1-(1-1/m)^{k(n-1)}$. The probability that all the k hashed counters are also hashed by some elements in multiset $\mathcal{S} \setminus \{x \cdot f_x\}$ is answered by f_{true} with element number of $n-1$. We denote $f_{true}|_{n-1}$ as f_{true} with element number of $n-1$, and get:

$$1 - \mathcal{C}_r = f_{true}|_{n-1} \Rightarrow \mathcal{C}_r = 1 - f_{true}|_{n-1} \tag{25}$$

Applying Eq. 20, we can get the upper and lower bounds of the $\mathcal{C}_r$ of CBFs:

$$1 - \left(1 - \left(1 - \frac{k}{m}\right)^{n-1}\right)^k < \mathcal{C}_r < 1 - \left(1 - \left(1 - \frac{1}{m}\right)^{(n-1)k}\right)^k \tag{26}$$

VI. Experimental Results

In this section, we first validate our proposed formula of optimal k (Eq. 13). Second, we compare our proposed upper bound of the FP probability of BFs (Eq. 20) with Bose's upper bound (Eq. 4). Third, we validate our proposed upper and lower bounds of the correct rate of CBFs (Eq. 26).

A. Experimental Setup

Datasets: We use the anonymized IP trace collected in 2016 from CAIDA [47]. Each flow is identified by its source IP address (sip) and destination IP address (dip). For BF, each distinct sip-dip pair from this trace functions as an element in the aforementioned set. We use (part of) the first 100M distinct sip-dip pairs to construct BFs, and query the next 300M distinct sip-dip pairs to get the empirical FP probabilities of these BFs. For CBF, each distinct sip-dip pair and its occurrence number from this trace function as a distinct element and the corresponding frequency in the aforementioned multiset, respectively. We use (part of) the first 100M distinct sip-dip pairs and their occurrence numbers in the current trace to construct CBFs, and query their frequencies to get the empirical correct rates of these CBFs.

Implementation: We have implemented the standard Bloom filter in C++. We use the Bob Hash (obtained from the open source website [48]) with different initial seeds to implement the hash functions in BFs as recommended by literature [49]. All the implementation source code is made publicly available anonymously at GitHub [1].

B. Optimal k Formula Validation

1) Optimal k vs. n: Figure 5 plots the empirically and theoretically optimal k with different n increasing from 10M to 100M with a step of 10M for $m = 500$M. *Our results show that the optimal k calculated from our new formula follows the empirically optimal k very well, regardless of the values of n.* We observe that the optimal k calculated from our new formula is very close to the one calculated from the formula obtained by Bloom. The reason is that the above two formulas about the optimal k have the same asymptotic form when m is large enough (see Section III-C).

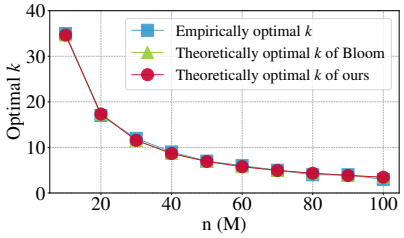

Fig. 5: Optimal k vs. n for $m = 500$M.

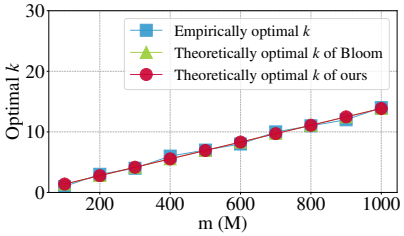

Fig. 6: Optimal k vs. m for $n = 50$M.

2) Optimal k vs. m: Figure 6 plots the empirically and theoretically optimal k with different m increasing from 100M to 1000M with a step of 100M for $n = 50$M. *Our results show that the optimal k calculated from our new formula follows the empirically optimal k very well, regardless of the values of m.* We observe that the optimal k calculated from our new formula is very close to the one calculated from the formula obtained by Bloom, especially when m becomes larger.

C. Upper Bound Comparison

1) Upper Bound vs. n: Figure 7 plots the empirical results, Bloom's theoretical results, Bose's upper bounds, and our upper bounds of FP probability with different n increasing from 10M to 100M with a step of 10M for $m = 500$M and $k = 6$. *Our results show that our upper bounds of FP probability follows the empirical FP probability very well, regardless of the values of n.* We find that all above four results almost coincide with each other, which demonstrates the tightness of bounds in Eq. 4 and 20.

To compare upper bound of Bose and ours more intuitively, in Figure 10, we plot the bounds error ratios β, defined as $\beta = \frac{upper\ bound - lower\ bound}{lower\ bound}$, of these two upper bounds with different n increasing from 10M to 100M with a step of 10M for $m = 500$M and $k = 6$. *Our results show that the bounds error ratio of our upper bound is* $26821.2 \sim 125492.5$ *lower than that of Bose's upper bound.* We find that our upper bound almost coincides with the lower bound, which demonstrates the superiority of our upper bound.

2) Upper Bound vs. m: Figure 8 plots the empirical results, Bloom's theoretical results, Bose's upper bounds, and our upper bounds of FP probability with different m increasing from 100M to 1000M with a step of 100M for $n = 50$M and $k = 6$. *Our results show that our upper bounds of FP*

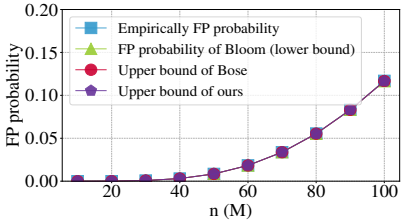

Fig. 7: FP probability vs. n for $m = 500$M and $k = 6$.

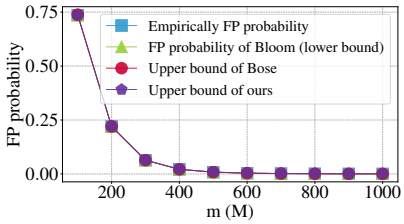

Fig. 8: FP probability vs. m for $n = 50$M and $k = 6$.

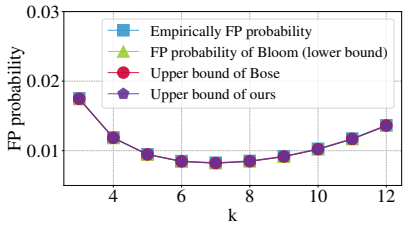

Fig. 9: FP probability vs. k for $n = 50$M and $m = 500$M.

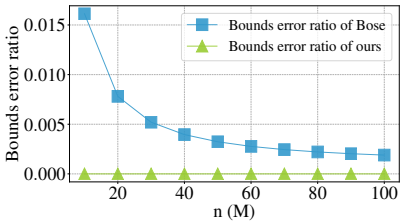

Fig. 10: Bounds error ratio vs. n for $m = 500$M and $k = 6$.

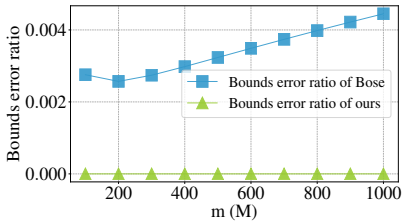

Fig. 11: Bounds error ratio vs. m for $n = 50$M and $k = 6$.

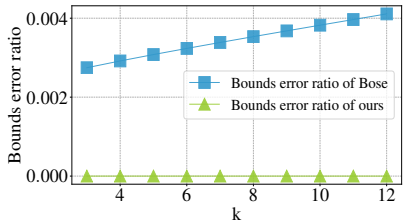

Fig. 12: Bounds error ratio vs. k for $n = 50$M and $m = 500$M.

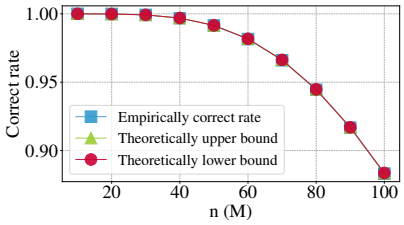

Fig. 13: Correct rate vs. n for $m = 500$M and $k = 6$.

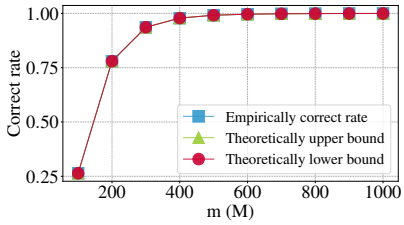

Fig. 14: Correct rate vs. m for $n = 50$M and $k = 6$.

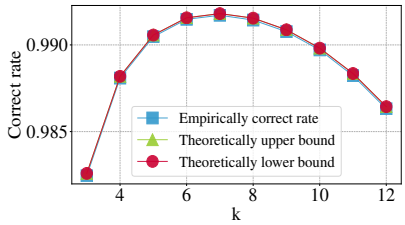

Fig. 15: Correct rate vs. k for $n = 50$M and $m = 500$M.

probability follows the empirical FP probability very well, regardless of the values of m.

Figure 11 plots the bounds error ratios of these two upper bounds with different m increasing from 100M to 1000M with a step of 100M for $n = 50$M and $k = 6$. *Our results show that the bounds error ratio of our upper bound is* $21885.4 \sim 150182.6$ *lower than that of Bose's upper bound.* We find that our upper bound almost coincides with the lower bound, regardless of the values of m.

3) Upper Bound vs. k: Figure 9 plots the empirical results, Bloom's theoretical results, Bose's upper bounds, and our upper bounds of FP probability with different k increasing from 3 to 12 with a step of 1 for $n = 50$M and $m = 500$M. *Our results show that our upper bounds of FP probability follows the empirical FP probability very well, regardless of the values of k.*

Figure 12 plots the bounds error ratios of these two upper bounds with different k increasing from 3 to 12 with a step of 1 for $n = 50$M and $m = 500$M. *Our results show that the bounds error ratio of our upper bound is* $39483.8 \sim 64668.5$ *lower than that of Bose's upper bound.*

D. C_r Formula Validation

Figure 13, 14, and 15 plot the correct rates of CFBs with different values of n, m, and k, respectively. *Our results show that our lower and upper bounds of the correct rate of CBFs follow the empirical correct rates very well, regardless of the values of n, m, and k.*

VII. Conclusion

In this paper, we discuss the controversy of the formula of Bloom Filter. Three formulas of false positive probability are presented: Bloom's formula, Bose's formula, and Christensen's formula. There is an error in the deduction process of Bloom's formula, and a minor error in Bose's formula. Christensen's formula is correct, but the false positive must be caculated using an iterative table-based algorithm with a time complexity of $O(knm)$. What is worse, it cannot deduce the optimal value of k.

To compute the optimal value of k, we use information and entropy theory to deduce the exact formula of k. To compute the false positive probability, 1) For small m, Christensen's formula can be used; 2) For large m, we propose a new upper bound which is much more accurate than state-of-the-art. Fortunately, when m is infinitely large, our upper bound becomes the same as the lower bound, which is Bloom's formula. Besides, we derive the bounds of correct rate of counting Bloom filters through our proposed formulas about Bloom filters. All the implementation source code is made publicly available anonymously at GitHub [1].

REFERENCES

[1] "Open souce website [on line]," Available:https://github.com/bfcodeopensource/bfcode.

[2] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[3] 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 (TON)*, vol. 8, no. 3, pp. 281–293, 2000.

[4] M. Mitzenmacher, "Distributed, compressed bloom filter web cache server," Jul. 19 2005, uS Patent 6,920,477.

[5] S. Dharmapurikar, P. Krishnamurthy, and D. E. Taylor, "Longest prefix matching using bloom filters," in *Proc. ACM SIGCOMM*. ACM, 2003, pp. 201–212.

[6] H. Lim, K. Lim, N. Lee, and K.-H. Park, "On adding bloom filters to longest prefix matching algorithms," *IEEE Transactions on Computers (TC)*, vol. 63, no. 2, pp. 411–423, 2014.

[7] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, "Fast hash table lookup using extended bloom filter: an aid to network processing," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 181–192, 2005.

[8] S. Haoyu, H. Fang, K. Murali, and L. TV, "Ipv6 lookups using distributed and load balanced bloom filters for 100gbps core router line cards," in *Proc. IEEE INFOCOM*, 2009, pp. 2518–2526.

[9] F. Chang, W.-c. Feng, and K. Li, "Approximate caches for packet classification," in *Proc. IEEE INFOCOM*, 2004.

[10] H. Yu and R. Mahapatra, "A memory-efficient hashing by multi-predicate bloom filters for packet classification," in *Proc. IEEE INFOCOM*, 2008.

[11] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep packet inspection using parallel bloom filters," in *Proc. IEEE HPI*. IEEE, 2003, pp. 44–51.

[12] J. W. Lockwood, J. Moscola, M. Kulig, D. Reddick, and T. Brooks, "Internet worm and virus protection in dynamically reconfigurable hardware," in *Proceedings of the Military and Aerospace Programmable Logic Device Conference*, 2003.

[13] M. Sarela, C. E. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott, "Forwarding anomalies in bloom filter-based multicast," in *Proc. IEEE INFOCOM*. IEEE, 2011, pp. 2399–2407.

[14] B. GrtinvaU, "Scalable multicast forwarding," *SIGCOMM Computer Communications Review*, vol. 32, no. 1, pp. 68–68, 2002.

[15] P. Kulkarni, P. Shenoy, and W. Gong, "Scalable techniques for memory-efficient cdn simulations," in *Proc. ACM WWW*, 2003.

[16] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed content delivery across adaptive overlay networks," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, 2002.

[17] C. E. Rothenberg, P. Jokela, P. Nikander, M. Sarela, and J. Ylitalo, "Self-routing denial-of-service resistant capabilities using in-packet bloom filters," in *Proc. EC2ND*. IEEE, 2009, pp. 46–51.

[18] T. Wolf, "A credential-based data path architecture for assurable global networking," in *Proc. IEEE MILCOM 2007*. IEEE, 2007, pp. 1–7.

[19] M. E. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo, "Towards collaborative security and p2p intrusion detection," in *Proc. IEEE IAW*. IEEE, 2005, pp. 333–339.

[20] P. Reynolds and A. Vahdat, "Efficient peer-to-peer keyword searching," in *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*. Springer-Verlag New York, Inc., 2003, pp. 21–40.

[21] G. Koloniari, Y. Petrakis, and E. Pitoura, "Content-based overlay networks for xml peers based on multi-level bloom filters," in *Databases, Information Systems, and Peer-to-Peer Computing*. Springer, 2004.

[22] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *Selected Areas in Communications, IEEE Journal on*, vol. 22, no. 1, pp. 41–53, 2004.

[23] Y. Wang, T. Pan, Z. Mi, H. Dai, X. Guo, T. Zhang, B. Liu, and Q. Dong, "Namefilter: Achieving fast name lookup with low memory cost via applying two-stage bloom filters," in *Proc. IEEE INFOCOM*. IEEE, 2013, pp. 95–99.

[24] F. Angius, M. Gerla, and G. Pau, "Bloogo: Bloom filter based gossip algorithm for wireless ndn," in *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications*. ACM, 2012, pp. 25–30.

[25] W.-c. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "Stochastic fair blue: A queue management algorithm for enforcing fairness," in *Proc. IEEE INFOCOM*, vol. 3. IEEE, 2001, pp. 1520–1529.

[26] F. Bonomi, M. Mitzenmacher, R. Panigrah, S. Singh, and G. Varghese, "Beyond bloom filters: from approximate membership checks to approximate state machines," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 315–326, 2006.

[27] C. Estan and G. Varghese, *New directions in traffic measurement and accounting*. ACM, 2002, vol. 32, no. 4.

[28] A. Kumar, J. Xu, and J. Wang, "Space-code bloom filter for efficient per-flow traffic measurement," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 12, pp. 2327–2339, 2006.

[29] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-based ip traceback," in *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4. ACM, 2001, pp. 3–14.

[30] T.-H. Lee, W.-K. Wu, and T.-Y. Huang, "Scalable packet digesting schemes for ip traceback," in *Proc. IEEE ICC*, vol. 2. IEEE, 2004, pp. 1008–1013.

[31] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "Minisec: a secure sensor network communication architecture," in *Proceedings of the 6th international conference on Information processing in sensor networks*. ACM, 2007, pp. 479–488.

[32] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 4, pp. 839–850, 2005.

[33] M. Yu, A. Fabrikant, and J. Rexford, "Buffalo: bloom filter forwarding architecture for large organizations," in *Proc. ACM CoNext*. ACM, 2009, pp. 313–324.

[34] D. Li, H. Cui, Y. Hu, Y. Xia, and X. Wang, "Scalable data center multicast using multi-class bloom filter," in *Proc. IEEE ICNP*. IEEE, 2011, pp. 266–275.

[35] D. Li, J. Yu, J. Yu, and J. Wu, "Exploring efficient and scalable multicast routing in future data center networks," in *Proc. IEEE INFOCOM*. IEEE, 2011, pp. 1368–1376.

[36] A. Papadopoulos and D. Katsaros, "A-tree: Distributed indexing of multidimensional data for cloud computing environments," in *Proc. IEEE CloudCom*. IEEE, 2011, pp. 407–414.

[37] V. Roussev, L. Wang, G. Richard, and L. Marziale, "A cloud computing platform for large-scale forensic computing," in *Advances in Digital Forensics V*. Springer, 2009, pp. 201–214.

[38] F. Sailhan and V. Issarny, "Scalable service discovery for manet," in *Proc. IEEE PerCom*. IEEE, 2005, pp. 235–244.

[39] Y. Zhang, W. Lou, W. Liu, and Y. Fang, "A secure incentive protocol for mobile ad hoc networks," *Wireless Networks*, vol. 13, no. 5, pp. 569–582, 2007.

[40] T. Yang, A. X. Liu, M. Shahzad, Y. Zhong, Q. Fu, Z. Li, G. Xie, and X. Li, "A shifting bloom filter framework for set queries," *Proceedings of the VLDB Endowment*, vol. 9, no. 5, pp. 408–419, 2016.

[41] M. Mitzenmacher, P. Reviriego, and S. Pontarelli, "Omass: One memory access set separation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1940–1943, 2016.

[42] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, and Y. Tang, "On the false-positive rate of bloom filters," *Information Processing Letters*, vol. 108, no. 4, pp. 210–213, 2008.

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

[44] C. E. Shannon, "Bell system technical journal," *A Mathematical Theory of Communication*, vol. 27, no. 3, pp. 379 – 423, 1948.

[45] L. Brillouin, "Science & information theory," *Dover Publications*, p. 293, 2004.

[46] 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 (TON)*, vol. 8, no. 3, pp. 281–293, 2000.

[47] "The caida anonymized 2016 internet traces." http://www.caida.org/data/overview/.

[48] "Hash website," http://burtleburtle.net/bob/hash/evahash.html.

[49] C. Henke, C. Schmoll, and T. Zseby, "Empirical evaluation of hash functions for multipoint measurements," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 39–50, 2008.