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

Paper #517, 9 pages

***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 its entropy is the largest, then 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.**

## I. Introduction

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: $h_1(x)\%m, h_2(x)\%m, ..., h_k(x)\%m$. Second, for each $1 \leqslant i \leqslant k$, we let $A[h_i(x)\%m] = 1$. 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: $h_1(y)\%m, h_2(y)\%m, ..., h_k(y)\%m$. Second, we check whether the corresponding $k$ bits in $A$ are all $1s$ (*i.e.*, whether $A[h_1(y)\%m] \wedge A[h_2(y)\%m] \wedge \cdots, A[h_k(y)\%m] = 1$ 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 speed and accuracy: smaller $k$ results in faster BF queries; too small $k$ results in large FP probability. 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], etc. Most applications with set membership querying 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 $n$, $f$, and $m$. Based on Bloom's formula, we can easily compute the optimal value of the unknown parameter $k$. 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 [40]. Interestingly, two years later, Ken Christensen *et al.* pointed out that Bose's formula is also flawed and gave a new FP formula [41]. 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 $n$, $f$, and $m$.

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 its entropy is the largest, then 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 entropy of a BF and then use it to calculate the optimal $k$ by letting the entropy equal

to 1. First, we derive the left and right limit expressions of FP probability. Second, 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.

*Paper organization:* 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 conduct experiments to evaluate the error of Bloom Filters. We conclude the paper in Section VI.

## II. Prior Art on BF False Positives

In this section, we review prior art on calculating the false positive probability of Bloom filters, and show the flaw of Bloom's and Bose's formulas. Table I summarizes the notations frequently used in this paper.

TABLE I
SYMBOLS USED IN THIS PAPER

| Symbol | Description |
|---|---|
| BF | Bloom filters |
| $\mathcal{S}$ | the set of elements in the Bloom filter |
| $m$ | the size of filter |
| $n$ | the number of elements |
| $k$ | the number of hash functions |
| $k^*$ | optimal number of hash functions |
| $f$ | false positive probability |
| FP | false positive |
| $p'$ | the probability that one bit of the array is still 0 after inserting $n$ elements in the BF |
| $f_{bloom}$ | the FP probability formula of BF derived by Bloom |
| $f_{partitioned}$ | the FP probability formula of partitioned BF |
| $f_{true}$ | the precise FP probability formula of BF |

### A. Bloom's False Positive Formula

An element not belonging to a set may be claimed to be by the Bloom filter is called a false positive. Bloom caculated the false positive rate as follows. Let $n$ be the number of elements in a given set $\mathcal{S}$, let $k$ be the number of hash functions, and let $m$ be the size of the Bloom filter. Two assumptions were made: 1) $kn < m$; 2) all hash functions are completely independent, *i.e.*, any two different hash functions map a value $x$ to two independent and random positions. The key value to compute here is $p'$, *the probability that one bit of the array is still 0 after inserting $n$ elements into the BF*. For a single hash function, the probability of hashing $x$ to a specific bit of the array is $\frac{1}{m}$. Therefore, the probability that a specific bit in the array is not set by one of the hash functions becomes $(1-\frac{1}{m})$. As each usage of a hash function is independent from the previous ones and that hash functions are used $kn$ times in order to insert $n$ elements in the BF, the probability $p'$ is derived as :

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

Now, Bloom makes in his derivation a second independence assumption: when an element $x \notin \mathcal{S}$ is presented to the BF, the probability that the application of any hash functions points to a bit of the array that is already set is independent from each other, resulting in a FP probability equal to $(1-p')^k$. Asymptotically for large BF vector size $m$, we have

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

that is the formulation widely used for analyzing the performance of BFs in terms of FP probability.

### B. Bose's Derivation

In 2008, Bose et al. [40] explained that the second independence assumption needed to derive $f_{Bloom}$ is too strong and does not hold in general, resulting in an underestimation of the FP probability. More precisely, although the $k$ hash functions are independent, one cannot deduce that the event $\{$*The previous $i-1$ verified hash positions pointed to set bits* $\}$ and the event $\{$*the $i^{th}$ hash position points to a set bit*$\}$ are independent. This can be seen by observing that there are two cases for each $k$ hash functions. Or each hash function points to a bit different from other hash functions, and in this case the independence assumption holds, or two or more hashes point to the same bit in the BF array and the independence assumption does not hold any more. This is the second case that was missed by Bloom in its derivation. Bose *et al.* derived a new formula claimed to be exact that was based on balls-and-urns model:

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

Moreover, Bose *et al.* derived asymptotic closed form upper and lower bounds for their formula as:

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

These bounds hold under the condition that

$$\frac{k}{p}\sqrt{\frac{\ln m - k \ln p}{m}} \leqslant c \tag{5}$$

for some constant $c < 1$. As stated in [40], for $k \geqslant 2$, $f_{bose}$ is strictly larger than $f_{Bloom}$, and the upper and lower bound converges to $f_{bloom}$, meaning that for small BFs (for example, of size 32 bits) $f_{bloom}$ formula underestimates FP probability and for larger BFs the relative error decreases.

### C. Christensen's Derivation

Later, Christensen *et al.* [41] claimed that Bose's formula is incorrect: in the formula 3, $(-1)^j$ should be $(-1)^{i-j}$. Furthermore, the authors used a balls-and-bins model to derive a recursive expression for the probability of $P(N, M, K)$ of $K$ bins being occupied in an $M$ bins and $N$ balls model:

$$\begin{aligned} P(N,M,K) =& P(N-1,M,K)\frac{K}{M} \\ &+ P(N-1,M,K-1)\frac{M-K+1}{M} \end{aligned} \tag{6}$$

Where $K = k$, $M = m$, and $N = k * n$. By solving the above recursive equation 6, Christensen *et al.* derived a correct formula for FP probability as

$$f_{christ} = \frac{m!}{m^{k(n+1)}} \sum_{i=1}^{m} \sum_{j=0}^{i} (-1)^{i-j} \frac{j^{kn} i^k}{(m-i)!j!(i-j)!} \tag{7}$$

In order to efficiently compute $f_{christ}$, the authors of [41] proposed an iterative table-based algorithm that can be calculated in $\mathcal{O}(knm)$, which is still hard to compute for large set. What is worse, the optimal value of $k^*$ cannot be derived by this formula.

In practice, if Christensen's formula is used for Bloom filter, then there will be two problems. First, when the set $\mathcal{S}$ is large, *i.e.*, $n$ is large, and the false positive is required to be very low, then the value of $m$ must be larger – tens of times of $n$, and $k$ is always bigger than 1. In this case, calculating the false positive probability is a time-consuming issue, as the proposed iterative algorithm has a complexity of $\mathcal{O}(knm)$. Second, even if the false positive probability can be computed, the optimal value of $k^*$ is unknown and cannot be determined. The only way is to try different value of $k$ and then compute the corresponding false positive probability. In the next section, we will derive this optimal value $k^*$ through information entropy theory.

## III. Computing the Optimal k

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

$$k^*_{bloom} = \frac{m}{n} \ln 2 \tag{8}$$

However, as we said, the underlying formula for FP probability is not fully correct. In this section, we first show the theory of information entropy, then we propose the important theorems which links information entropy and the false positive rate of Bloom filters. Third, we propose a novel method of deriving the optimal value of $k$, minimizing the FP probability given the value of BF size $m$ and number of inserted elements $n$.

### A. Introduction of Information Entropy

**Information Entropy**: In information theory, *information entropy* is used to measure the uncertainty of a random variable. In this paper, it refers to the *Shannon entropy* [42], which measures the value of the information contained in a variable. Entropy is typically measured in bits, nats, or bans [43]. For a variable with $s$ events with the probabilities of $p_1$, $p_2$, ..., $p_s$. The information entropy $E$ is defined as:

$$E = \sum_{i=1}^{s} p_i log_2 \frac{1}{p_i} \tag{9}$$

**Property of information entropy:** For any variable or message, if its information entropy is not at the maximum, it can be compressed without information loss. Suppose a $m$-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. 9), we can obtain the information entropy $E$ of the $m$-bit variable of a BF:

$$E = -\left(p' \log_2 p' + (1-p') \log_2 (1-p')\right) \tag{10}$$

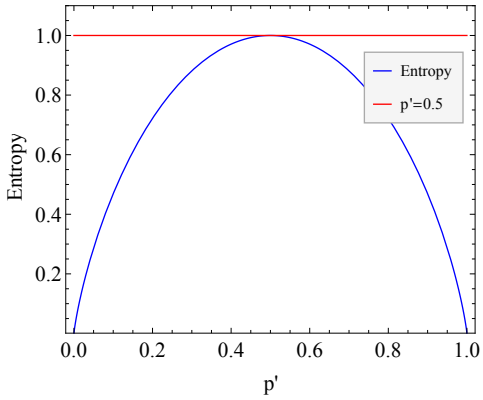



Fig. 1. Information entropy of a BF

To be more vivid, we draw the value of $E$ with the increase of $p'$ in Figure 1. Obviously, the maximum value of entropy of an $m$-bit sequence is obtained when the probability that each bit in the BF is 1 (or 0) is 0.5.

### B. Proposed Theorems about BFs

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

*Proof.* This lemma is obviously correct. 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 are 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 pair of BFR instances always report the same result: true or false for any inputting 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 $m'(m' < m)$[1], we name it $v_2$.

$v_1$ and $v_2$ are two variables consisting of bits, and we can regard them as two integer variable. 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

$$\begin{aligned} &g^{-1}(In(v_2)) \ll h_1(y) \gg (|g^{-1}(In(v_2))| - h_1(y) - 1),\\ &g^{-1}(In(v_2)) \ll h_2(y) \gg (|g^{-1}(In(v_2))| - h_2(y) - 1),\\ &\ldots\\ &g^{-1}(In(v_2)) \ll h_k(y) \gg (|g^{-1}(In(v_2))| - h_k(y) - 1) \end{aligned} \tag{11}$$

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

Given an inputting element $y$, we can compute 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. Let focus on formula 11, we use equation $v_1 = g^{-1}(In(v_2))$ and $m = |g^{-1}(In(v_2))|$, these $k$ hash functions are simplified as

$$\begin{aligned} &v_1 \ll h_1(y) \gg (m - h_1(y) - 1),\\ &v_1 \ll h_2(y) \gg (m - h_2(y) - 1),\\ &\ldots\\ &v_1 \ll h_k(y) \gg (m - h_k(y) - 1) \end{aligned} \tag{12}$$

[1]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 $1 \leqslant i \leqslant k$ and $0 \leqslant h_i(y) \leqslant m - 1$. $v_1 \ll h_i(y) \gg (m - h_i(y) - 1)$ 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.

We show both directions. First, we prove "⇐".

*Proof.* 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 for $v_1$, there exists a BF variable with smaller size, but with 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: if the FP probability is at the minimum, then its information entropy must be at the maximum. □

Second, we prove "⇒".

*Proof.* 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. 10 and Figure 1, the $p'$ of $v_0$ is 0.5, while 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 if its information entropy is at the maximum, the FP probability is at the minimum. □

According to above "⇒" and "⇐", we can prove that Theorem II is correct.

### 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:

$$p' = \left(1 - \frac{1}{m}\right)^{k^* n} = 0.5 \tag{13}$$

which is derived previously with the first independence assumption that is not disputed, we obtain:

$$k^* = -\frac{\ln 2}{n} / \ln\left(1 - \frac{1}{m}\right) \tag{14}$$

This formula is very close to the formula of $k^*$ obtained by Bloom. As we have, as for small values of $x$, the approximation $\ln(1+x) \approx x$, and therefore $-1/\ln(1-\frac{1}{m}) \approx m$ yielding the same term as in Eq. 8.

**Theorem III:** Given any BF variable, when $m$ and $n$ are fixed, the FP probability $f$ is a function of $k$, we represent it $f(k)$. We claim that $f(k)$ is a *convex function*.

*Note that in this paper a convex function means that it 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. 10 and Figure 1, we known $p'_1 < p'_2 \leqslant 0.5$. We compress $v_1$ to $v_3$ with parameters $m_3$, $n$, $k_1$. In order 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, same entropy leads to same $p'$. In other words, $p'_3 = p'_2$. Because $v_2$ has more hash functions ($k_2 > k_1$), thus 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. 10 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 varbile, when $m$ and $n$ are fixed, the FP probability $f$ is a function of $k$, we represent it $f(k)$. $f(k)$ is a *convex function*. □

## IV. Asymptotic Form of the FP Probability

We will derive a new and very simple approach to computing 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 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

$$p'_{partition} = \left(1 - \frac{k}{m}\right)^n \tag{15}$$

as now each hash maps into $\frac{m}{k}$ separate bits.

It is intuitive that the FP probability of partitioned BF is a littler 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 $m > 1\ , k > 1,\ \ n > 1,\ m > k$,

$$\left(1 - \frac{k}{m}\right)^n < \left(1 - \frac{1}{m}\right)^{kn} \tag{16}$$

A similar but different inequality is shown in page 3 of the well known BF survey [44]. The survey did not give the proof, thus we give one here.

*Proof.* 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:

$$g'(k) > \left(1 - \frac{1}{m}\right) \ln\left(1 - \frac{1}{m}\right) + \frac{1}{m} \tag{17}$$

Let $f(m) = \left(1 - \frac{1}{m}\right) \ln\left(1 - \frac{1}{m}\right) + \frac{1}{m}$, we can find $f'(m) = \frac{1}{m^2} \ln\left(1 - \frac{1}{m}\right) < 0$, meaning 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 $g'(k) > f(m) \geqslant 0$, meaning that the function $g(k)$ is strictly increasing. We have therefore

$$\left(1 - \frac{k}{m}\right)^n < \left(1 - \frac{1}{m}\right)^{kn} \tag{18}$$

□

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, Christensens 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:

$$f_{partition} > f_{true} > f_{bloom} \tag{19}$$

For partitioned BF, the probability that one bit of the array is still 0 $p'$ is shown in Eq. 15. Different from standard Bloom Filter, for partitioned Bloom Filter, the event "$E(h_1 = 1), E(h_2 = 1), E(h_3 = 1), ..., E(h_{i-1} = 1)$" is independent with the event "$E(h_{i-1} = 1)$", where $E(h_{i-1} = 1)$ means the event that the position of $h_{i-1}(x)$ is 1, because each hash function is responsible for one partition, and has no effect with each other. Therefore,

$$f_{partition} = (1 - p'_{partition})^k = \left(1 - \left(1 - \frac{k}{m}\right)^n\right)^k \tag{20}$$

Then the formula 19 becomes

$$\left(1 - \left(1 - \frac{k}{m}\right)^n\right)^k > f_{true} > \left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^k \tag{21}$$

Then we use the well known limit formula:

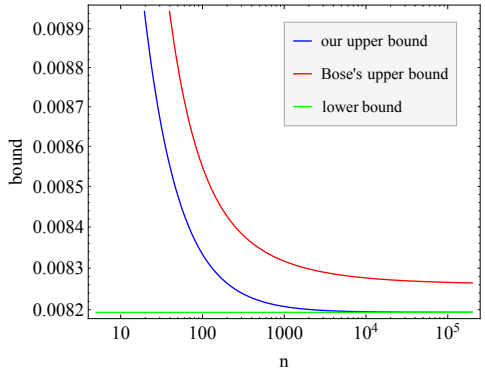



Fig. 2. Upper and lower bound for $f_{true}$ for $k = 7$ and $m = 10n$.

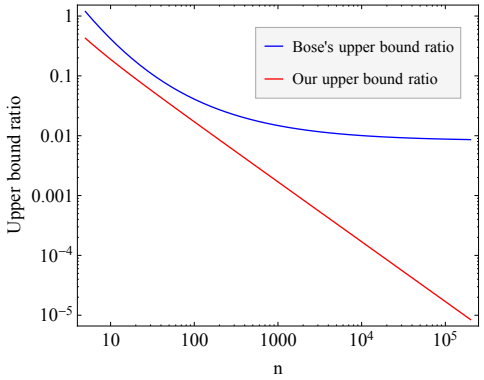



Fig. 3. Bounds error ratio for Bose's bound and the bound derived in this paper for $k = 7$ and $m = 10n$.

$$\lim_{x\to\infty}\left(1-\frac{1}{x}\right)^{-x} = e \tag{22}$$

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 :

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

that is the same term as the lower bound limit. Through the Sandwich Theorem (also known as squeeze theorem) we get, similarly to Christensen and to Bose, that :

$$\lim_{m\to\infty} f_{true} = \left(1-e^{-nk/m}\right)^{k} \tag{24}$$

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 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 $\beta$, defined as $\beta = \frac{upperbound-lowerbound}{lowerbound}$, for the two bounds in Figure 3. As can be seen, the gap between the our derived upper bound and $f_{bloom}$ is decreasing polynomially at a constant speed, while Bose 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.

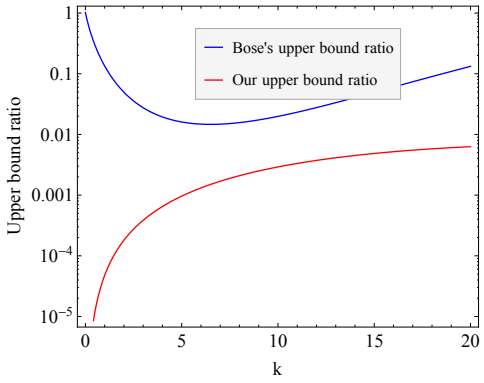



Fig. 4. Bounds error ratio for Bose's bound and the bound derived in this paper for $m = 10000$, $n = 1000$ and varying $k$.

## V. Evaluation

### A. Experimental Setup

In order to evaluate experimentally the FP probability of a set of entries, we wrote a C++ program to generate a large set of unique entries with 20 characters each. It takes around 1 hour on an Intel(R) Core(TM) i7-920 with 4 cores computer working at 2.67GHz to generate 100M unique entries and more than 3 hours to get 300M entries which occupy more than 6GB of space. These two sets are used to evaluate empirically FP probability and optimal values of $k$.

### B. False Positive Probability Test

*1) Basic Test:* Our goal in this paper is to evaluate the quality of our proposed bound and methodology. In place of trying to find low complexity hashes we rather use hashes known to be uniform. For this purpose, we used a universal hashing function based on 26 basic hash functions and we build the needed hashes from this universal hash function. Thereafter the 100M-entries file described above is used for the experiments. Each round of experiments is done with a given value of $k$ chosen as an even number from 2 to 16. We insert the first 5000 ($n$ = 5000) elements into the Bloom Filter, and then calculate the optimal $m$ according to the formula 14,

and then get:

$$m = \frac{1}{1 - 2^{-\frac{1}{kn}}} \tag{25}$$

In order to evaluate the FP probability we query 100,000,000 elements from the BF we buit, and we show the query results in Table II. We show experimental results of FP probability of BF as well as the FP rate using the formula of $f_{bloom}$ It can be seen that as $k$ increases from 2 to 16, the number of false positives decreases from 24814826 to 1528 and the error ratio is 2.41% at most. The error ratio shows that the experimental results of fp probabilitiy follow the predictions of $f_{bloom}$ very well.

TABLE II
FALSE POSITIVE PROBABILITY IN THEORY AND SIMULATION.

| k | # of FP | FP from $f_{bloom}$ | FP of simulation | Error ratio (%) |
|---|---|---|---|---|
| 2 | 24814826 | 0.25003480 | 0.24814830 | -0.7545290 |
| 4 | 6263312 | 0.06250841 | 0.06263312 | 0.1995030 |
| 6 | 1565056 | 0.01562703 | 0.01565056 | 0.1505790 |
| 8 | 392198 | 0.00390674 | 0.00392198 | 0.3901310 |
| 10 | 95314 | 0.00097668 | 0.00095314 | -2.4102060 |
| 12 | 24946 | 0.00024417 | 0.00024946 | 2.1670100 |
| 14 | 6166 | 0.00006104 | 0.00006166 | 1.0125530 |
| 16 | 1528 | 0.00001526 | 0.00001528 | 0.1283920 |

*2) Error ratio vs. Querying number:* We build the same SBF as described previously and use the same 100M-traffic file. We run experiments when $k$ is 4, 5, and 6, respectively. $n$ is still 5000, and $m$ is calculated to be the optimal number according to formula 25. We query 1,000,000 up to 100,000,000 elements and calculate the error ratio when every next 1,000,000 elements are queried.

The results are shown in Figure 5 that, at the very beginning, the error ratio is much bigger and also with drastic fluctuation. The more elements we query, the stabilizer the error ratio will be. When the number of false positives in a test is small, the test result can hardly be reliable. In our test, the smallest number of FPs is 15,820, which is the first spot in Figure 5c, when $k$ is 6 and query number is 1,000,000.

All in all, the error ratio is actually so small that it can be ignored. They are ranging from -0.533% to +0.356% when $k = 4$, +0.188% to +0.699% when $k = 5$, and -0.138% to +1.235% when $k = 6$.

To give a more intuitive result, we draw Figure 6. The blue line is the theoretical FP probability, while the other curve is the evaluated FP probability (defined as FP number/total query number). That error ratio stabilizes with the increase of the querying number still holds.

To give a whole picture, we carry out another experiment. In this experiment, we query the first $n$ elements and sum up the FP number. And then, we pick up the latter $n$ elements and also get the FP number. These two groups of elements are totally different without duplicate elements across the groups. Thus, we query up to 100*n elements and get 100 spots. The 100M-element traffic file is not large enough for this experiment, so we use the 300M-element traffic file which contains more than 300M elements we described in the experimental setup section.

As shown in Figure 7, the smallest number of FP is 1,075, which is still in the first spot in Figure 7c, when $k$ is 6 and this time query number is 68,712. Although it is smaller than the previous accumulated experiment, it is still bigger than 1,000, so the result is reasonable. By the way, the 100th query number is $100 * 68712 = 6,871,200$ and the number of FP is 109,713, which is large enough.

It can be seen from Figure 7, the error ratios fluctuate drastically at the beginning and quickly converge to a small range. And also, the error ratios are very small ranging from -0.812% to +1.500% when $k = 4$, -3.559% to +1.284% when $k = 5$, and +0.115% to +5.581% when $k = 6$.

### C. Optimal $k$ Formula Validation

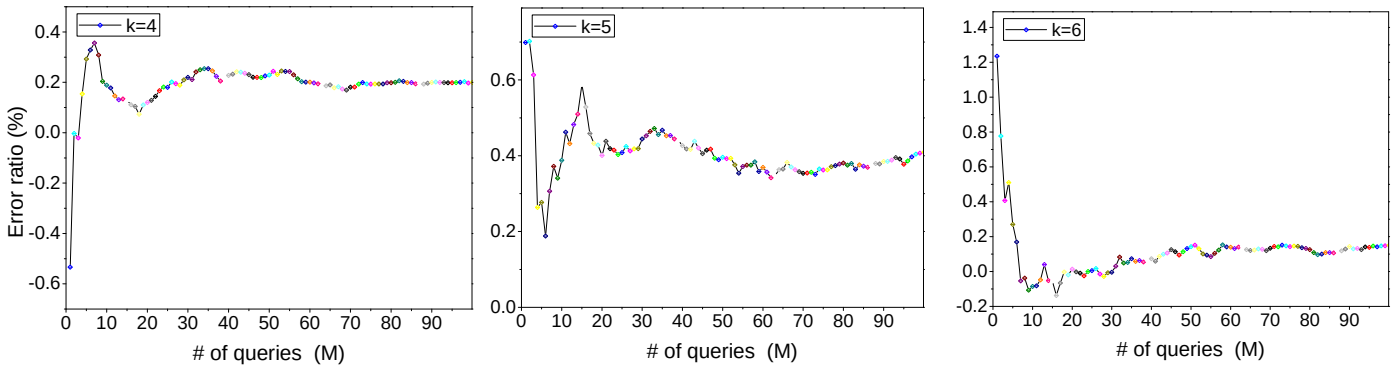
In this experiment, we validate that the formula Eq. 14 derived for $k^*$, gives a value minimizing the fp probability. For this purpose, we create large BFs with $m = 100,000$ and with different $k$ ranging from 2 to 16, and then insert $n = 8000, 10000, 12000$ entries into it. We use the same 100M-entries to evaluate the fp probability. We show the results in Figure 8. We also show the value of fp probability derive using the formula $f_{bloom}$. It can be seen in the figure 8 for the given value of $m$, $n$, and $k$, the experimental results and the theoretical values of fp probability well match. Moreover, when using formula in Eq. 14, on the third scenario that is $n = 12,000$, one can derive an optimal value of $k^* = 6$. Looking at the Figure 8, it can be seen that this is indeed the value minimizing the fp probability with a FP probability of 1.816%. Similar observation are made for $n = 8000$ and 10,000.

It can be seen from Figure 8 that the optimal $k$s are totally identical between the theoretical and practical results. For example, when $m = 100,000$, $n = 12,000$ and the best $k$ should be 5.78 obtained from our formula 14. Since $k$ is an integer, we set $k = 6$. In this experiment, when $k$ is 6, the FP probability is the lowest, which is 1.816% in the 15 spots ranging from 2 to 16. The same logical also applied to other cases when $n$ is 8,000 and 10,000.

## VI. CONCLUSION

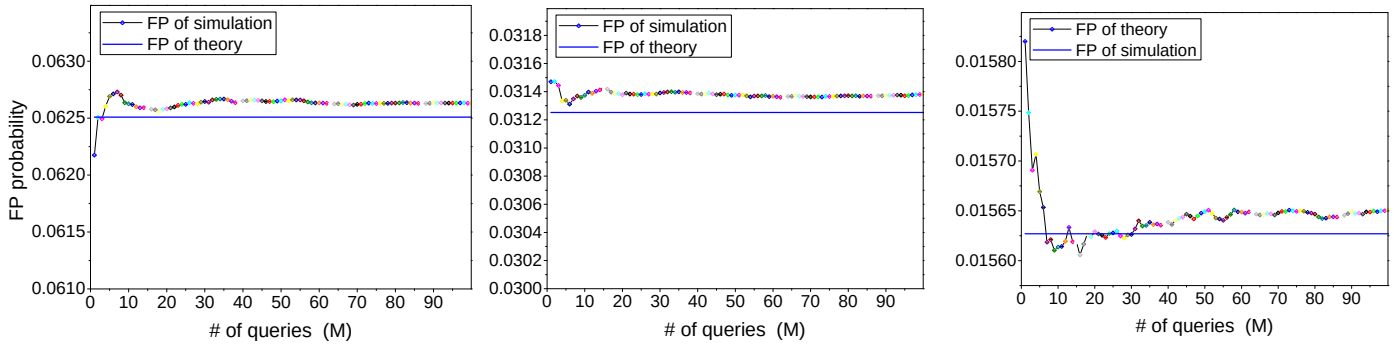
In this paper, we discuss the controversy of the formula of Bloom Filter. Three formulas of false positive probability are deduced: Bloom's formula, Bose's formula, and Christensen's formula. There is an error in the deduction process of Bloom's formula, and a minor error in Bose's formula. Christensen's formula is correct, and can compute the false positive using an iterative table-based algorithm in $O(knm)$. However, 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 prove a limit form of false positive probability using left and right limit, and find out that it is same as Bloom's. 3) We also show that the error is negligible when $m$ is not large using two simple examples.



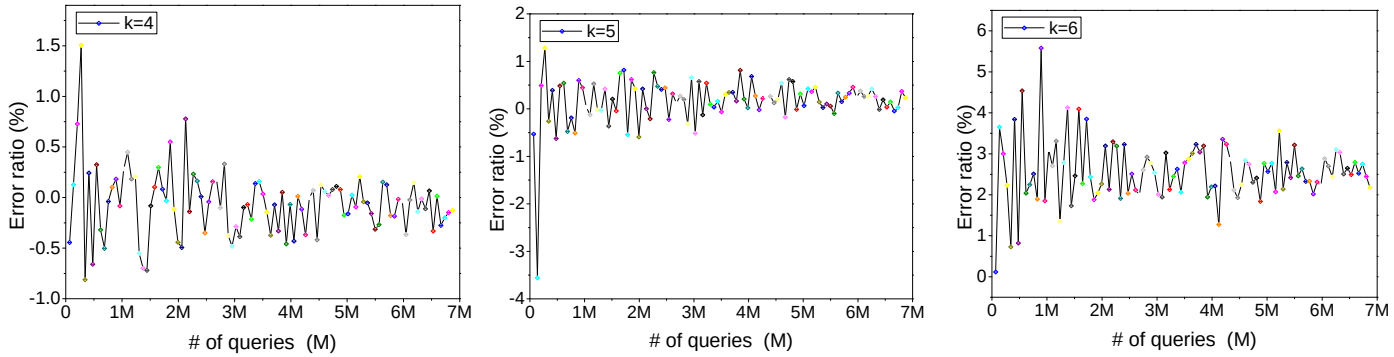

(a) k=4 (b) k=5 (c) k=6

Fig. 5. FP probability error ratio vs. # of queries with different $k$.




(a) k=4 (b) k=5 (c) k=6

Fig. 6. FP probability error vs. # of queries with different $k$.




(a) k=4 (b) k=5 (c) k=6

Fig. 7. FP probability error vs. # of queries with independent queries.

## REFERENCES


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

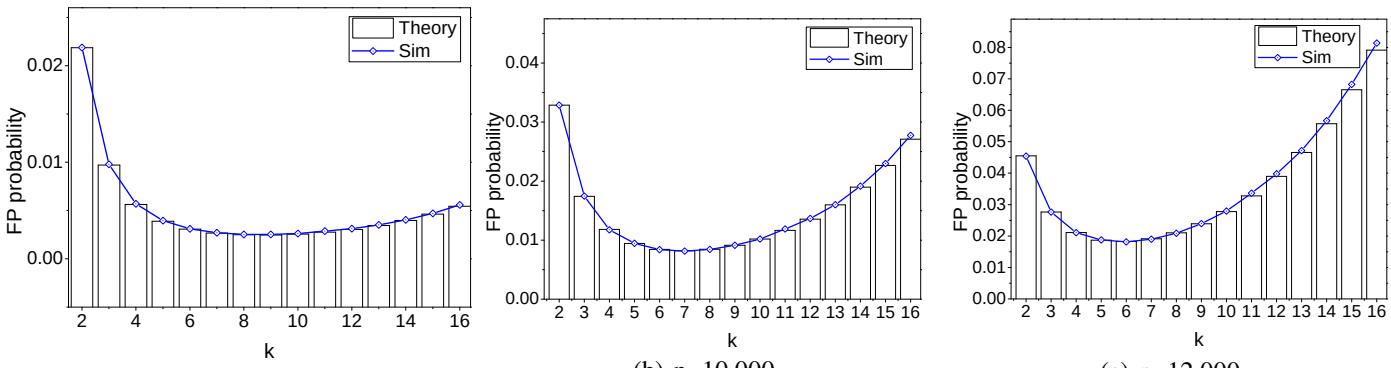
[6] H. Lim, K. Lim, N. Lee, and K.-H. Park, "On adding bloom filters to longest prefix matching algorithms," *IEEE Transactions on Computers (TC)*, vol. 63, no. 2, pp. 411–423, 2014.

[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.




(a) n=8,000 (b) n=10,000 (c) n=12,000

Fig. 8. Variation of fp probability as a function of the number of hash functions $k$ for $m = 100,000$ and different number of inserted entries $n$


[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] 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.

[41] 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.

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

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

[44] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.