Computer Networks 54 (2010) 165–174

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Receiver-oriented design of Bloom filters for data-centric routing

Deke Guo [a,*], Yuan He [b], Panlong Yang [c]

[a] Key Laboratory of C4 ISR Technology, National University of Defense Technology, Changsha Hunan 410073, China
[b] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China
[c] Institute of Communication Engineering, P.L.A University of Science and Technology, Nanjing Jiangsu 210000, China

ARTICLE INFO

Article history:
Received 8 June 2009
Accepted 4 October 2009
Available online 13 October 2009
Responsible Editor: Minglu Li

Keywords:
Bloom filters
Data-centric routing
Receiver-oriented design

ABSTRACT

Bloom filter (BF) is a space-efficient data structure that represents a large set of items and supports efficient membership queries. It has been widely proposed to employ Bloom filters in the routing entries so as to facilitate data-centric routing in network applications. The existing designs of Bloom filters, however, cannot effectively support in-network queries. Given a query for a data item at a node in the network, the *noise* in unrelated routing entries very likely equals to the useful information of the item in the right routing entries. Consequently, the majority of queries are routed towards many wrong nodes besides those destinations, wasting large quantities of network traffic. To address this issue, we classified the existing designs as CUBF (Cumulative Bloom filters) and ABF (Aggregated Bloom filters), and then evaluate their performance in routing queries under the noisy environments. Based on the evaluation results, we propose a receiver-oriented design of Bloom filters to sufficiently restrict the probability of a wrong routing decision. Moreover, we significantly decrease the delay of a routing decision in the case of CUBF by using the bit slice approach, and reduce the transmission size of each BF in the case of ABF by using the compression approach. Both the theoretical analysis and experimental results demonstrate that our receiver-oriented design of Bloom filters apparently outperforms the existing approaches in terms of the success probability of routing and network traffic cost.

1. Introduction

Bloom filter (BF) [1] is a space-efficient data structure that represents a large set of items and supports efficient membership queries. BF outperforms other data structures such as binary search trees and tries, as the time needed to insert an item or check whether an item belongs to the filtering set is constant, irrespective of the cardinality of the set. Hence BF has been widely adopted in database and networking applications [1,2], such as web cache sharing [3] and routing [4–6]. Moreover, BF has great potential in memory management, such as summarizing streaming data in memory [7], storing the states of a large number of flows in the on-chip memory of the routers [8], and speeding up the Bayesian filters [9].

The space efficiency of BF, however, is achieved at the cost of false positive judgments. False positive judgment is a unnegligible drawback of BF, which refers to the case that an item does not belong to a set but the BF makes the contrary judgment. In many applications, the savings in storage and computational costs brought by BF outweigh such a drawback, on condition that the false positive probability is sufficiently low. Many efforts have been made to reduce the probability of false positive in standalone and distributed systems [10–13] during the past years.

In the last few years, it has been proposed to employ BF in supporting data-centric routing in overlay networks [4,5,14–16], wireless sensor networks [17], ad hoc networks [18,19], and mesh networks [20]. The common idea

* Corresponding author. Tel.: +86 731 84576603.
E-mail addresses: guodeke@gmail.com, guodeke@ieee.org (D. Guo), heyuan@cse.ust.hk (Y. He), veron_yang@sina.com (P. Yang).

1389-1286/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2009.10.002

of those proposals is that every node uses a BF to represent the information of all its data items and broadcasts the BF to the nodes in its *propagation range*, i.e. the nodes within d hops from the node. Correspondingly, a node receives a number of BFs via each link of it. The link, associated with the BFs received via it or the union of them, is then maintained as a routing entry. The union of BFs is defined as the logic *or* operation among their bit vectors [21]. For any intermediate node routing a query, it forwards the query through the link whose corresponding routing entry satisfies the query. Ideally, a query will be propelled towards its destination once it enters the propagation range of the destination. For example, a query at node E for a data item at node A is routed to the right destination node along a single path $E \rightarrow C \rightarrow B \rightarrow A$, as shown in Fig. 1a.

False positive judgments exist in data-centric routing with BFs. The necessary condition for the BF-based routing schemes outperforming the blind routing schemes is that the false positive probability in the routing entries is sufficiently low. Otherwise, given any query for a data item x at a node in the network, the *noise* in unrelated routing entries very likely equals to the useful information of the item in the right routing entries. The *noise* in a unrelated routing entry is defined as the amount of membership information of x in it, if the node does not receive a BF from the destination of x via the corresponding link. Being misled, the majority of queries are routed towards many wrong nodes besides the right destinations, and result in huge amount of redundant but useless queries in the network. For example, a query at node E for a data item at node A is routed to both the right destination node and other two nodes G and H which do not hold the desired data item, as shown in Fig. 1b. The network in turn presents poor efficiency of query processing and suffers scalability problem.

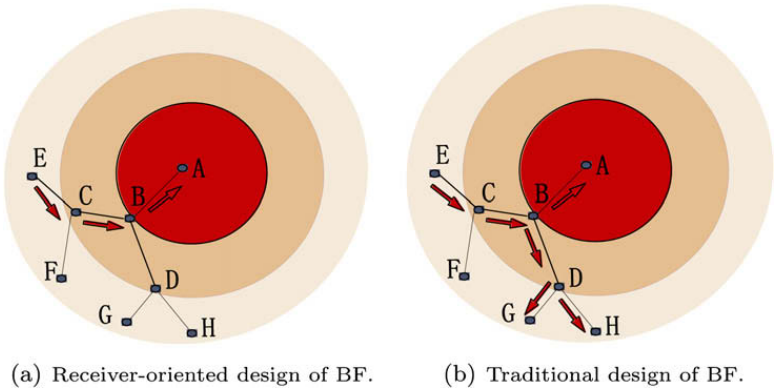
In this paper, we reveal through theoretical analysis that the existing designs of BF cannot satisfy the aforementioned necessary condition. Specifically, we classified the existing BF-based routing schemes as CUBF (Cumulative Bloom filters) and ABF (Aggregated Bloom filters), and then evaluate their performance in routing queries under the noisy environments. Based on the evaluation results, we propose a receiver-oriented design of Bloom filters, with which the false positive probability of any routing entry is low enough so that a node can correctly distinguish the right out-going link from the others. We further conduct extensive simulations to evaluate the performance of the proposed scheme. Both the theoretical analysis and experimental results demonstrate that our receiver-oriented design of BF apparently outperforms the existing designs in terms of success probability of routing and network traffic cost, as shown in Fig. 1.

The rest of this paper is organized as follows. In Section 2, we briefly introduce BF and its traditional design. In Section 3, we summarize the state-of-arts designs of BF-based data-centric routing schemes, and then propose the receiver-oriented design of BF. In Section 4, we further optimize the transmission and storage strategies of BF, and present our study on how to ensure that the false positive probability of any routing entry does not exceed an upper bound in practice. Section 5 presents the performance evaluation results. We concludes this work in Section 6.

2. Preliminaries

A BF for representing a set X of n items is described by a vector of m bits, initially all set to 0. A BF uses k independent hash functions $h_1, \ldots, h_k$ to map each item of X to a random number over a range $\{1, \ldots, m\}$ [1] uniformly. For each item x of X, we define its BF address as $bfaddress(x)$ consisted of $h_i(x)$ for $1 \leqslant i \leqslant k$, and the bits belonging to $bfaddress(x)$ are set to 1 when inserting x. Once the set X is represented as a BF, to judge whether an element x belongs to X, one just needs to check whether all the $h_i(x)$ bits are set to 1. If so, x is a member of X (note that there is a probability that this could be wrong). Otherwise, we assume that x is not a member of X. It is clear that a BF may yield a false positive due to hash collisions, for which it suggests that an element x is in X even though it is not. The reason is that all indexed bits were previously set to 1 by other items [1].

The probability of a false positive for an element not in the set can be calculated in a straightforward fashion, given the assumption that hash functions are perfectly random. Let p_0 denote the probability that a random bit of the BF is 0, and let n be the number of items that have been added to the BF. Then $p_0 = (1 - 1/m)^{k \cdot n} \approx e^{-k \cdot n/m}$ as $n \cdot k$ bits are randomly selected, with probability $1/m$ in the process of adding each item. We use $f^{BF}_{m,k,n}$ to denote the



(a) Receiver-oriented design of BF. (b) Traditional design of BF.

Fig. 1. Illustrative examples of BF-based routing for a query for an item at node A.

false positive probability caused by the $(n+1)$th insertion, and we have the expression

$$f_{BF} = (1 - p_0)^k \approx (1 - e^{-k \cdot n/m})^k. \quad (1)$$

Given the false positive probability and the set cardinality, we can calculate the filter size of BF and the number of hash functions according to Formula (1). From [1], we know that the minimum value of f_{BF} is $0.6185^{m/n}$ when $k = (m/n)\ln 2$. In practice, k must be integer, and a smaller k is preferred so as to reduce the amount of computation required.

3. Receiver-oriented bloom filters

In this section, we first introduce the state-of-arts designs of BF-based data-centric routing schemes, including CUBF and ABF. We then propose a receiver-oriented BF, which is superior to the existing designs with respect to the false positive probability.

3.1. Background

We start with a brief introduction to the network model used in this paper. Let n be the average number of data items at a node in the network. We assume every node propagates its BF to the nodes within d hops. For any arbitrary node, s denotes the average number of BFs received via any one of its links. Therefore, starting from any link of a node, let X denote the set consisted of all reachable data items within d hops in the network. We may calculate the cardinality of set X as $N \simeq s \cdot n$.

In the case of data-centric routing, a link and the s BFs received through it may be stored as a single routing entry. We call a set consisted of the s independent BFs as a CUBF (Cumulative Bloom filters), as shown in Fig. 2. Alternatively, a link may be stored with the union of the s BFs received through it as a routing entry. We call the union of the s independent BFs as an ABF (Aggregated Bloom filters), as shown in Fig. 3.

When forwarding a query through a certain link, either CUBF or ABF may be used to track the data items reachable within d hops in the network. For example, node A receives

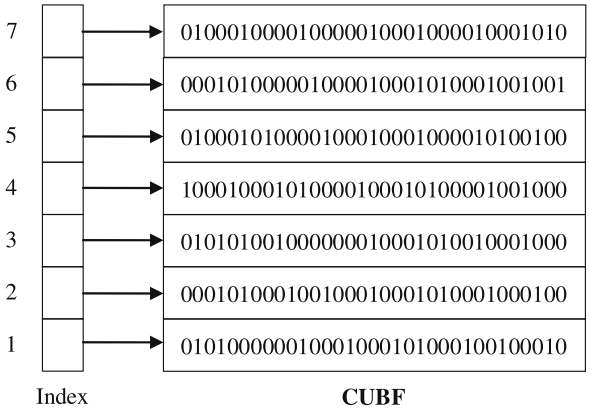


Fig. 2. An example of a CUBF consisted of seven BFs, where $m = 33$, $n = 3$, $k = 3$, and $f_{BF} = 0.0136$. Note that the false positive probability of the resulting CUBF is $f_{CUBF} = 0.0914$.

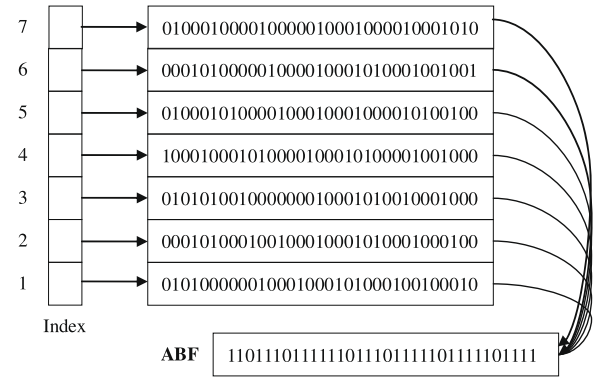


Fig. 3. An example of an ABF based on seven BFs, where $m = 33$, $n = 3$, $k = 3$, and $f_{BF} = 0.0136$. Note that the false positive probability of the resulting ABF is $f_{ABF} = 0.618$.

the BF of node B through link $B \to A$, and BFs of nodes C, D, E, F, G, and H through link $B \to A$ as well, as shown in Fig. 1 where $d = 3$. Node A may directly use the combination of the seven independent BFs received through link $B \to A$ as a routing entry. Alternatively, it may use the union of those BFs as a routing entry. In this paper we assume homogeneous routing entries in the network, which is the general case in practice. In other words, we do not consider that some routing entries use CUBF while the others adopt ABF.

Rhea and Kubiatowicz proposed a variant of data-centric routing based on BF [5]. They have each node keep an array of BFs for every link in the overlay topology. In the array of BFs for each link, there is a BF for each distance $1 \leqslant i \leqslant d$, so that the ith BF in the array keeps track of files reachable via i hops through the overlay network along that link. They call this array of BFs an attenuated Bloom filter. An attenuated Bloom filter usually finds files within at most d hops along a single path, but it is likely to find the shortest path to a file replica if many such paths exist. It is worth noticing that an attenuated Bloom filter is similar to a CUBF, however, every component of an attenuated Bloom filter is an ABF while that of a CUBF is a BF.

To ensure the inter-operability between individual BFs in many distributed applications, all nodes are required to adopt an identical configuration of parameters m, k, and the hash functions. In the case of CUBF, for a query of an item x, all individual BFs in a CUBF share the same Bloom filter address of the item x. Thus it is not necessary to calculate $bfaddress(x)$ for the query in each individual BF, hence reducing the computational overhead of determining the right out-going link of the query and in turn increasing the routing efficiency. In the case of ABF, the union operation of the BFs require each BF adopt an identical configuration, too.

3.2. Receiver-oriented design of Bloom filters

In this subsection, we first present theoretical analysis on the probability of false positive judgments using CUBF and ABF. Based on the analytical results, we propose the receiver-oriented design of Bloom filters.

A CUBF might yield a false positive for a data item x_1 which is unreachable within d hops from the current node

in the network. The reason is that all bits of $bfaddress(x_1)$ in at least one BF of the CUBF have been set to 1 by items of X. The false positive probability of the CUBF can be calculated as follows. The false positive probability of each BF of the CUBF is given by Formula (1). Thus the probability that not all the bits of $bfaddress(x)$ in each BF of the CUBF are set to 1 is $(1-f_{BF})^s$. The probability that all the bits of $bfaddress(x)$ in at least one BF of the CUBF are set to 1 is

$$f_{CUBF} = 1-(1-f_{BF})^s = 1-\left(1-(1-e^{-k\cdot n/m})^k\right)^s. \quad (2)$$

It is also possible for an ABF to yield a false positive for an item $x_1 \notin X$ when all bits of $bfaddress(x_1)$ in the ABF are set to 1 by items of X. When calculating the false positive probability, we first consider an analogous process that $s\cdot k\cdot n$ balls are dropped into m empty bins randomly. The location of each ball is independently and uniformly chosen from the m possibilities. The union operation of s BFs is equivalent to the ball dropping process. Hence the probability that a random bit in the ABF is set to 1 is equivalent to the probability that a random bin becomes non-empty. It can be calculated as follows:

$$p_1 = 1-(1-1/m)^{s\cdot k\cdot n} \approx 1-e^{-s\cdot k\cdot n/m}.$$

Thus, the probability that all the bits of $bfaddress(x_1)$ in the ABF are set to 1 is

$$f_{ABF} = p_1^k = (1-e^{-s\cdot k\cdot n/m})^k. \quad (3)$$

According to Formulas (2) and (3), we can see that s is another important parameter besides those well-known ones and the false positive probabilities of CUBF and ABF are monotonic increasing functions of s. Even if the false positive probability of each single BF is low, the false positive probability of the resulting CUBF or ABF will very likely increase to an unacceptable level as the value of s increases. For example, node A in Fig. 1 receives seven BFs, yielding a false positive judgment with very low probability of 0.0136. The false positive probabilities of the resulting CUBF and ABF, however, are 0.0914 and 0.618, as shown in Figs. 2 and 3, respectively. Thus, given a query for an item at an arbitrary node in the network, false positives very likely arise in unrelated routing entries. In turn the query is forwarded towards not only the correct destination nodes but also some wrong nodes with high probability. None of the existing designs of BF is able to tackle such a challenging issue, no matter it is based on CUBF or ABF (see Table 1).

Table 1
Summary of main notations.

Term	Definition
m	Number of bits of a BF
n	Average number of data items at a node
k	Number of hash functions used by a BF
d	Propagation range of a BF
s	Average number of BFs received via any link
p_0	Probability of a random bit in a BF to be 0
p_1	Probability of a random bit in a BF to be 1
f_{BF}	False positive probability of a BF
f_{CUBF}	False positive probability of a CUBF
f_{ABF}	False positive probability of an ABF
σ	Upper bound of the false positive probability

Let σ denote an application-specific upper bound of the false positive probability of a CUBF or an ABF. It is already known that a traditional BF is optimal when $k=(m/n)\ln 2$, with respect to the false positive probability. Such an optimal setting, however, cannot be applied to the case of CUBF or ABF.

To address this issue, we propose a receiver-oriented design of Bloom filters. The basic idea is to derive the parameter configuration of homogeneous BFs for a CUBF or an ABF such that the value of Formula (2) or (3) cannot be greater than σ. That is,

$$1-(1-f_{BF})^s \leqslant \sigma \quad \text{and} \quad (1-e^{-s\cdot k\cdot n/m})^k \leqslant \sigma.$$

Given σ, n and s, we wish to optimize the number of hash functions and the sizes of homogeneous BFs. In the case of CUBF, the false positive probability of each BF is $f_{BF} \leqslant 1-(1-\sigma)^{1/s}$. According to [1], the optimal number of hash functions that minimize f_{BF} is $k=(m/n)\ln 2$, and the minimum value of f_{BF} is $0.6185^{m/n}$. Thus, we derive the lower bound on the optimal filer size as

$$m \geqslant n\cdot \log_{0.6185}(1-(1-\sigma)^{1/s}).$$

The lower bound on the optimal number of hash functions is

$$k \geqslant \log_{0.5}(1-(1-\sigma)^{1/s}.$$

In practice, k and m must be integers. Smaller k and m are preferred so as to reduce the computational and storage overhead. Thus, the optimal size of filters and the optimal number of hash functions are as follows:

$$m = \left\lceil n\cdot \log_{0.6185}(1-(1-\sigma)^{1/s})\right\rceil, \quad (4)$$

$$k = \lceil \log_{0.5}(1-(1-\sigma)^{1/s}\rceil. \quad (5)$$

In the case of ABF, the optimal number of hash functions that minimizes f_{ABF} can be easily obtained by taking the derivative. Note that

$$f_{ABF} = \exp(k\ln(1-e^{-s\cdot k\cdot n/m})).$$

Let $g = k\ln(1-e^{-s\cdot k\cdot n/m})$. Minimizing the false positive probability f_{ABF} is equivalent to minimizing g with respect to k. Thus we have

$$\frac{\varphi g}{\varphi k} = \ln(1-e^{-s\cdot k\cdot n/m}) + \frac{s\cdot k\cdot n}{m}\frac{e^{-s\cdot k\cdot n/m}}{1-e^{-s\cdot k\cdot n/m}}.$$

Clearly the derivative is 0 when $k=(\frac{m}{s\cdot n})\cdot \ln 2$. Alternatively, using $p_0 = e^{-s\cdot k\cdot n/m}$, we get

$$g = -\frac{m}{s\cdot n}\ln(p_0)\ln(1-p_0),$$

where g is minimized when $p_0 = 1/2$, namely $k=(\frac{m}{s\cdot n})\cdot \ln 2$. Correspondingly, the false positive probability of ABF is $(1/2)^k \approx (0.6185)^{m/s\cdot n}$. Thus, we derive the lower bound of the optimal filter size as

$$m \geqslant s\cdot n\cdot \frac{-\log_2\sigma}{\ln 2}.$$

The lower bound of the optimal number of hash functions is

$$k \geqslant -\log_2\sigma.$$

In practice, m and k must be integers. Smaller m and k are preferred so as to reduce the computational and storage overhead. Thus, the optimal size of filters and the optimal number of hash functions are as follows:

$$m = \left\lceil s \cdot n \cdot \frac{-\log_2 \sigma}{\ln 2} \right\rceil, \quad (6)$$

$$k = \lceil -\log_2 \sigma \rceil. \quad (7)$$

Due to the similar reason mentioned in [1], the fact that $p_0 = 1/2$ when the false positive probability of ABF is minimized, does not depend on the asymptotic approximation that $(1 - 1/m)^{s \cdot k \cdot n} \approx e^{-s \cdot k \cdot n/m}$.

4. Discussion

To further improve our approach, we address two key issues, namely to optimize the transmission size of ABF and storage strategy of CUBF. We then discuss how to ensure that the false positive probability of any routing entry does not exceed an upper bound in practice.

4.1. Optimization of transmission size for ABF

In the case of ABF, Formula (6) denotes the optimal size of BF implemented by any node to ensure that the false positive probability of each resulting ABF does not exceed the threshold σ with high probability. In fact, however, the size of BF is too large. The storage space appears to be underutilized. The fraction of bits "1" in each BF is significantly less than 1/2. Note that a BF created by each node must be delivered to other nodes as a series of messages in order to establish routing entries. The transmission size of each BF become a critical factor, which significantly affects the amount of network traffic. To deal with such a problem, we propose to compress the BFs before transmission.

The authors in [22] show that the uncompressed BF, which is traditionally optimized by $k = (m/n)\ln 2$ cannot achieve any gain by compressing it. The reason is that under good random hash functions, each bit of BF is 0 or 1 independently with probability 1/2. On the contrary, the BF which is optimized according to Formulas (6) and (7) in the case of ABF can achieve considerable gain with compression. The reason is that under the same assumption about hash functions, the probability of a bit in a BF to be 1 is lower than 1/2, namely $p_1 \ll 1/2$. In theory, a m bits BF can be compressed down to only $m \cdot H(p_1)$ bits, where

$$H(p_1) = -p_1 \log_2 p_1 - (1 - p_1)\log_2(1 - p_1)$$

is the entropy function. In the case of ABF, a BF with $m = \lceil s \cdot n \cdot \log_{0.6185} \sigma \rceil$ bits can be compressed down to only $m \cdot H(p_1)$ bits, where $p_1 = 1 - e^{k \cdot n/m} = 1 - (1/2)^{1/s}$.

It is worth noticing that the arithmetic coding is a near-optimal compressor which requires $m(H(p_1) + \varepsilon)$ bits for any $\varepsilon > 0$. Such theoretical analysis demonstrates that compression is a viable method that significantly reduces the transmission size of BFs and saves network traffic cost in the case of ABF. The compression and decompression of each BF can be implemented by using simple arithmetic coding at each node. Thus their computational cost is very low and can thus be neglected.

4.2. Storage optimization for CUBF

Since a CUBF does not aggregate the original BFs, those routing schemes based on CUBF cost large amount of storage space for storing BFs at each node. When a query is given at a node, a full scan of all its CUBFs is required. For any CUBF, it appears to be inefficient to scan all its BFs for seeking a routing decision at the worst case, especially when the number of its BFs is large. As expect, the long delay of each routing decision lowers down the concurrency of transmissions and degrades the network throughput. Such a problem becomes more serious when the average node degree is high and the *propagation range* of BFs is large, especially in the resource-constrained networks such as wireless sensor networks [23–25]. Consequently, CUBF-based schemes probably suffer poor efficiency of routing and scalability problem due to the inefficient storage of BFs.

There are two ways for storing a CUBF consisted of s BFs, namely bit string and bit slice [26].

In the bit string method, the bits matrix in Fig. 2 is stored in a set of files in a row-wise manner, as shown in Fig. 4a. Note that the s BFs are sequentially stored, which is easy to implement. When a query is to be resolved with such a CUBF, however, a full scan of the s BFs is required. Therefore, it is usually slow in retrieving the queried result.

As for the bit slice method, the bits matrix is vertically partitioned into m bit slices. Each bit slice, represented by a column in Fig. 4b, stores one bit per BF for all the BFs in the CUBF. When resolving a query, instead of loading the full BFs in the CUBF for scanning, the bit slice method only needs to load a few bit slices from the CUBF that are associated with the query. In this way, the scanning cost is significantly reduced. The delay for making a routing decision at a node is decreased as well. In words, the bit slice method greatly improves the efficiency of routing and the scalability of CUBF-based routing schemes.

4.3. Practical issues in data-centric routing

The parameter σ is dependent on the applications, and can be determined according to the specific requirement,

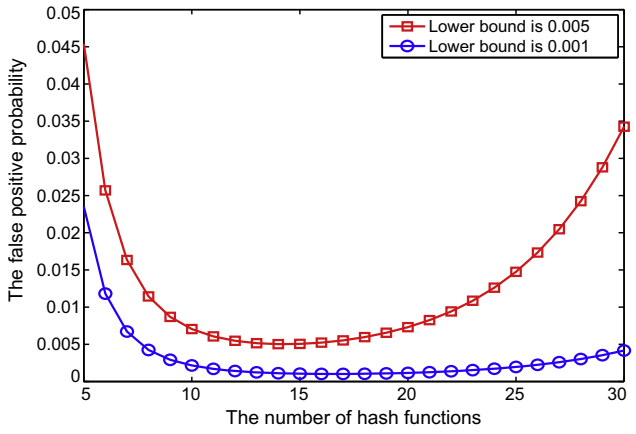


Fig. 4. Illustration of bit-string and bit-slice storage methods of a CUBF consisted of 7 BFs, where $m = 12$.

e.g. the constraint on the error probability in a routing decision. The parameters n and s should be assigned with appropriate values based on the topological properties, data and query distributions among the nodes, popularity of queried data items, and the propagation range of BFs. Many efforts have been made to measure the topological properties (including the number of nodes, the distribution of node degree, and the network diameter), and to investigate the data distribution and query distribution. For example, bubbleStorm includes a protocol for measuring global system state [27]. The propagation range should be assigned an appropriate value limited by the network diameter. Thus, it is reasonable to assume that we are given n and s, and the design approaches of CUBF and ABF can be put into practice based on Formulas (4)–(7).

In reality, the following practical issues directly dominate the final effect of CUBF and ABF even if each node implements its local BF for CUBF or ABF. In the case of CUBF, the number of reachable nodes within d hops in the network along any link usually does not equal to s for the majority of links. Therefore, each node has to record the number of received BFs along each link, and denies those BFs behind the first s BFs such that f_{CUBF} of each routing entry always does not exceed σ. In the case of ABF, the distribution of node degree and the distribution of number of received BFs along one link are usually not uniform. Thus, for the majority of links, the number of all reachable data items within d hops in the network along a link from an arbitrary node usually does not equal to $n \cdot s$. Therefore, each node should monitor the fraction of bits "1" in each ABF, and denies other BFs received through a link once the fraction of 1 bits in a corresponding ABF reaches 1/2 such that f_{ABF} always does not exceed σ.

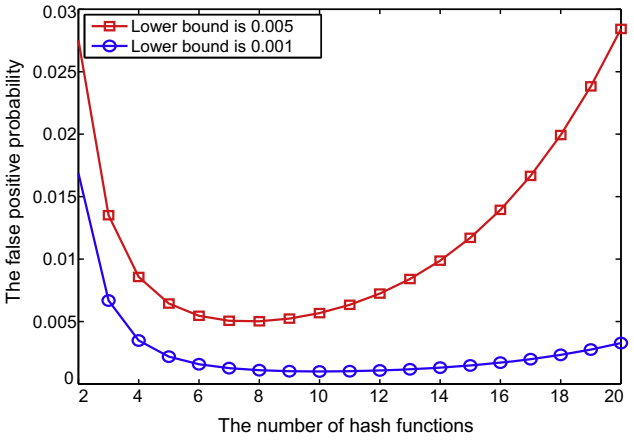


Fig. 5. The false positive probability of CUBF as a function of the number of hash functions used, where $s = 100$ and $n = 30$.

5. Performance evaluation

In this section, we first evaluate the performance of CUBF and ABF in terms of the optimal number of hash functions, the false positive probability, and the transmission size. We then conduct experiments to examine the false positive probability of CUBF and ABF in practice.

5.1. Optimal number of hash functions

Formulas (2) and (3) demonstrate that the false positive probability is a function of the number of hash functions. We conduct simulations to evaluate f_{CUBF} and f_{ABF}. As shown in Fig. 5, f_{CUBF} is minimized to 0.005 for 15 hash functions, and is minimized to 0.001 for 17 hash functions. On the other hand, according to Formula (5), we can verify that the optimal number of hash functions should be 15 and 17 when σ is 0.005 and 0.001, respectively. As shown in Fig. 6, using 8 and 10 hash functions can minimize f_{ABF} to 0.005 and 0.001, respectively. According to Formula (7), we find that the optimal number of hash functions should be 8 and 10 when σ is 0.005 and 0.001, respectively. Thus, the simulation results match well with our theoretical analysis. It is worth noticing that the optimal number of hash functions is a monotonic decreasing function of σ, in both CUBF or ABF, as shown in Formulas (6) and (7) and Figs. 5 and 6.

5.2. False positive probability in theory

We first evaluate the false positive probability of CUBF and ABF with the traditional designs of BF. In such cases, each node constructs its local BF with $k = (m/n)\ln 2$ and $m = n \cdot \log_{0.6185}\sigma$, where σ is a given upper bound of false positive probability. Each node constructs a CUBF or an ABF as a routing entry for each link based on all BFs received through that link. We conduct simulations to evaluate f_{CUBF} and f_{ABF} of any routing entry at an arbitrary node. Fig. 7 plots the simulation results. We can see that f_{CUBF} and f_{ABF} increase as the number of received BFs increases,

7 bit-string files

BF1	0	1	0	0	0	1	0	0	0	1	0
BF2	0	0	0	1	0	1	0	0	0	0	1
BF3	0	1	0	0	0	1	0	1	0	0	0
BF4	1	0	0	0	1	0	0	0	1	0	1
BF5	0	1	0	1	0	1	0	0	1	0	0
BF6	0	0	0	1	0	1	0	0	0	1	0
BF7	0	1	0	1	0	0	0	0	0	1	0

(a) Bit string method.

12 bit-slice files

0	1	0	0	0	1	0	0	0	0	1	0
0	0	0	1	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0	1	0	0	0	0
1	0	0	0	1	0	0	0	1	0	1	0
0	1	0	1	0	1	0	0	1	0	0	0
0	0	0	1	0	1	0	0	0	1	0	0
0	1	0	1	0	0	0	0	0	0	1	0

(b) Bit slice method.

Fig. 6. The false positive probability of ABF as a function of the number of hash functions used, where $s = 100$ and $n = 30$.

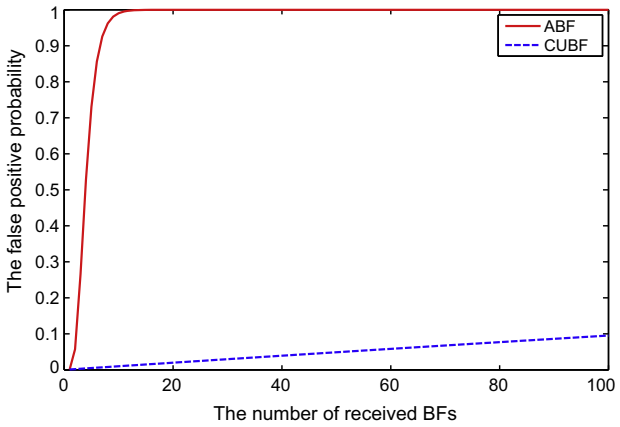


Fig. 7. f_{CUBF} and f_{ABF} are functions of the number of received BFs, where $f_{BF} = 0.001$, $m/n = \log_{0.6185} 0.001$, and $k = (m/n)\ln 2$.

especially that f_{ABF} quickly reaches to almost one even a small number of BFs are received. Such a result indicates that the traditional design of BF cannot ensure that the false positive probability is sufficiently low. Consequently, as we reiterate in this paper, CUBF and ABF with traditional designs are not applicable with those data-centric routing schemes in practice.

We then demonstrate that the proposed receiver-oriented design of BFs in this paper can ensure that f_{CUBF} and f_{ABF} always do not exceed any given upper bound σ. Here, each node designs its local BF with parameters m and k configured according to Formulas (4)–(7), respectively. The simulation results of f_{CUBF} and f_{ABF} are shown in Fig. 8. As the number of received BFs increases, f_{CUBF} almost increases linearly while f_{ABF} increases exponentially. They, however, do not exceed the upper bound $\sigma = 0.001$ when the number of received BFs does not exceed a given upper bound $s = 100$. On the other hand, f_{ABF} is always less than f_{CUBF} when the number of received BFs ranges from 1 to 100. In words, ABF outperforms CUBF in terms of false positive probability. Furthermore, the storage cost of ABF is much lower than that of CUBF. Thus ABF is more suitable than CUBF in the data-centric routing schemes.

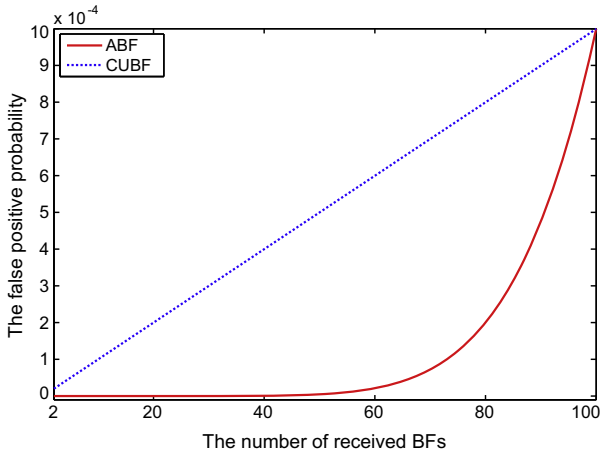


Fig. 8. The false positive probability of CUBF and ABF as functions of the number of BFs, where $\sigma = 0.001$, $n = 30$, $s = 100$, and the parameters m and k are optimized by Formulas (4)–(7).

5.3. Transmission size

Considering the fact that the transmission size of BF is a critical design factor in network applications, we now evaluate the transmission sizes of four types of BFs, including the traditional BF, BF for CUBF, BF for ABF, and compressed BF for ABF. For a given value of n, the transmission sizes of the four types of BF are functions of σ and s. In this subsection, we will evaluate the transmission sizes of the filters under two scenarios. Under the first scenario, the value of s is fixed while that of σ is variable. Under the other one, the value of s is variable while that of σ is fixed.

Figs. 9 and 10 plot the evaluation results under the first scenario. We can clearly see that all three curves in Fig. 9 follow a similar trend as the value of σ increases continuously from 0.001 to 0.1. The filter sizes of traditional BF, BF for CUBF, and compressed BF for ABF decrease as the value of σ increases. In addition, we find that the compressed BF for ABF consumes more bits than BF for CUBF and the traditional BF, irrespective of the value of σ. Fig. 10 shows that the BF for ABF occupies almost 100 times of bits as the traditional BF does. The compressed BF for ABF consumes less than 5 times of bits as the traditional BF does. The BF for CUBF uses less than 3 times of bits as the traditional BF does. Such results demonstrate that compression is indeed an efficient way to reduce transmission size of BF in the case of ABF.

Fig. 11 plots the evaluation results under the second scenario. The two curves follow a similar trend as the value of s increases continuously from 2 to 10,000. At the beginning, the two curves go up quickly as the value of s increases, and then slow down ascending. The simulation results of BF for uncompressed ABF are not shown in the figure because it almost consumes s times of bits as the traditional BF does. As s ranges from 2 to 10,000, we find that the compressed BF for ABF consumes at most 7.5 times of bits as the traditional BF does, while the BF for CUBF occupies at most 2.5 times of bits as the traditional BF does. On the other hand, the compressed BF for ABF consumes a bit more bits than the BF for CUBF does, without respect to the value of s. Such results again demonstrate that compression is indeed efficient in reducing transmission size of BF in the case of ABF.

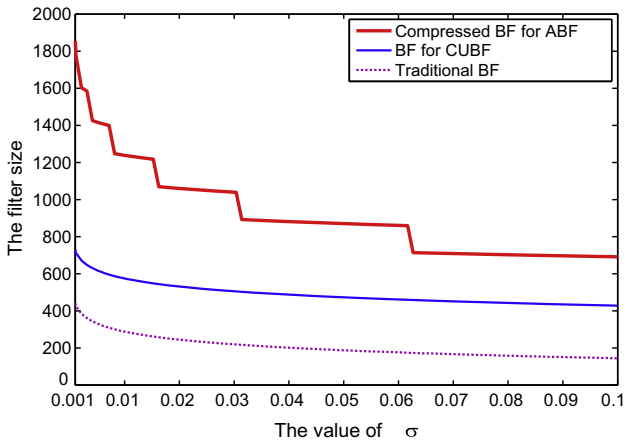


Fig. 9. The sizes of traditional BF, BF for CUBF, and compressed BF for ABF as functions of σ, where $s = 100$ and $n = 30$.

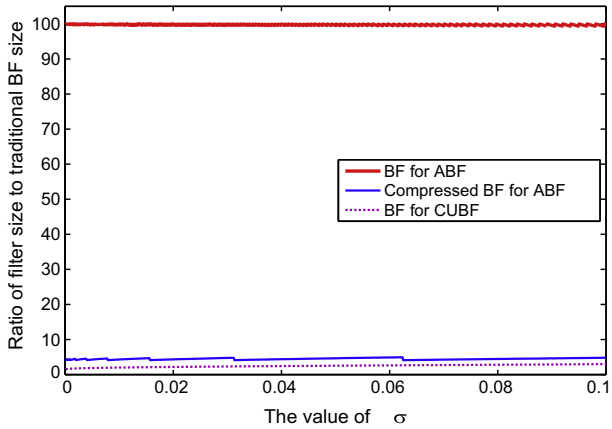


Fig. 10. The ratio of filter size to the traditional BF size as function of σ, where $s = 100$ and $n = 30$.

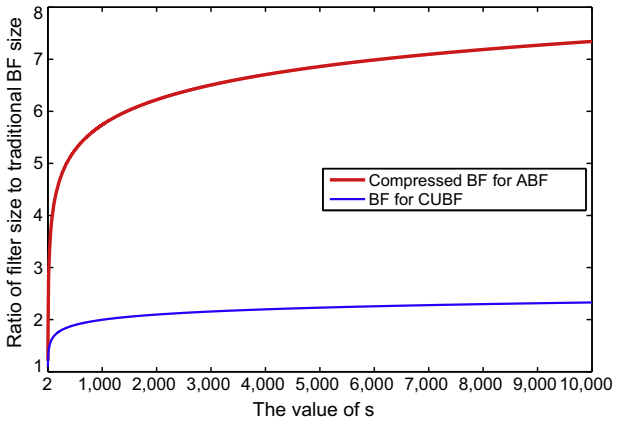


Fig. 11. The ratio of filter size to the traditional BF size as functions of s, where $\sigma = 0.001$ and $n = 30$.

In summary, the compressed BF for ABF and the BF for CUBF consume more bits than the traditional BF. The low false positive probability of CUBF or ABF, however, outweighs the high storage cost since the extra bits can be accommodated by most applications. On the other hand, ABF outperforms CUBF in terms of false positive probability, as shown in Fig. 8, and the delay of each routing decision. Thus, ABF is more suitable to the data-centric routing schemes in network applications, despite that it incurs a bit higher storage cost.

5.4. False positive probability in practice

The hash functions are the key elements that influence the value of f_{BF}, f_{CUBF} and f_{ABF}. Ideally, BFs are assumed to be realized with hash functions that can map each item in the unknown universe to a random number over the range $\{1, \ldots, m\}$ uniformly. In practice, however, such an assumption is too strict to realize. As a result, it is extremely difficult, if not impossible, to implement a CUBF or an ABF which achieves the theoretical results exactly mentioned in Formulas (2) and (3). Thus it is necessary to examine f_{CUBF} and f_{ABF} from a practical aspect, under the scenarios of both the traditional BF and the receiver-oriented design of BF.

As there are no benchmark data sets in the field of BF, our experiments do not seek particular sets but simply use a data set from the DBLP as the data stored on the networking nodes. We use the distinct author names to initialize a data set Y with $n = 30$ items for each peer. Further, we initialize a common data set Z to be used by the tests of false positive judgments at any peer, where $Y \cap Z = \emptyset$ and the cardinality of Z is 10 times larger than that of Y. In each experiment, we sample 100 peers randomly, and let each sampled peer execute a test of false positives for each routing entry after receiving s BFs through each corresponding link. We evaluate f_{CUBF} and f_{ABF} from the theoretical and experimental aspects under two scenarios.

Figs. 12 and 13 plot the evaluation results of f_{CUBF} and f_{ABF} under the first scenario in which each BF is designed by using the traditional approach. In Fig. 12, the two curves follow a similar trend as the value of s increases continuously from 1 to 50. Although the experimental results do not exactly equal to the theoretical results, the difference between them is small, without respect to the value of s. Such a result confirms the correctness of Formula (2). On the other hand, the two curves in Fig. 13 present similar

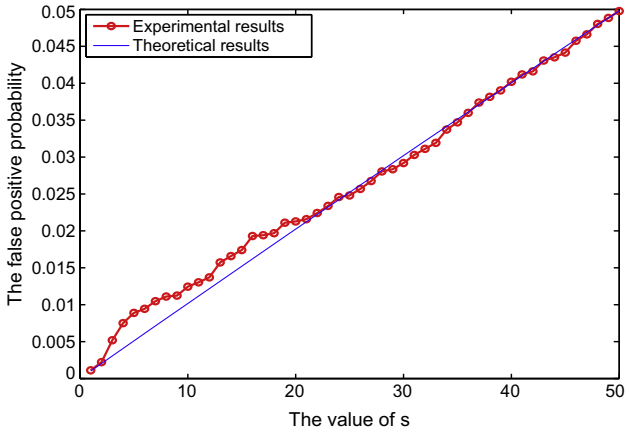


Fig. 12. The theoretical and experimental values of f_{CUBF} as functions of the number of BFs. For each single BF, the false positive probability is 0.001, $m/n = \log_{0.6185} 0.001$, and $k = (m/n) \ln 2$.

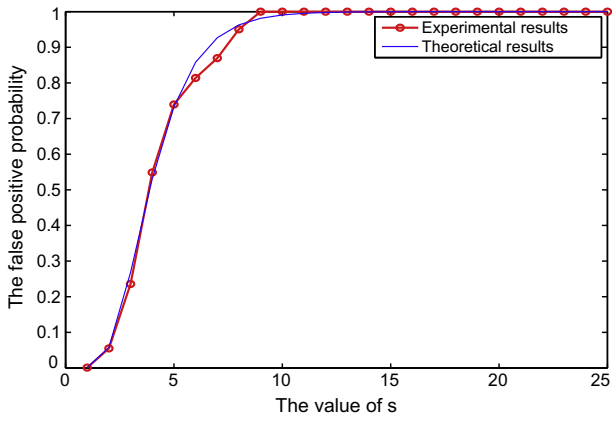


Fig. 13. The theoretical and experimental values of f_{ABF} as functions of the number of BFs. For each single BF, the false positive probability is 0.001, $m/n = \log_{0.6185} 0.001$, and $k = (m/n) \ln 2$.

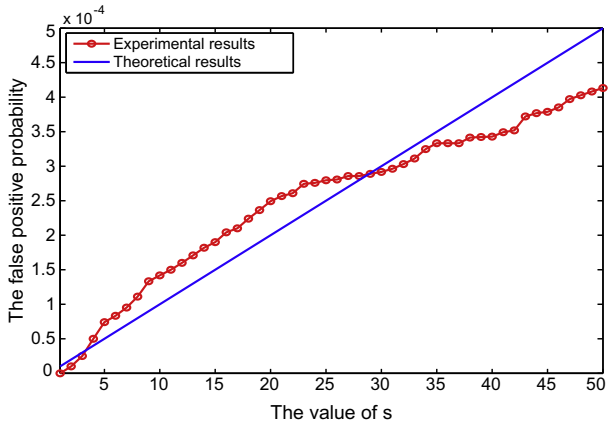


Fig. 14. The theoretical and experimental values of f_{CUBF} as functions of the number of BFs, where $\sigma = 0.001$, $n = 30$, m and k are optimized by Formulas (4)–(7).

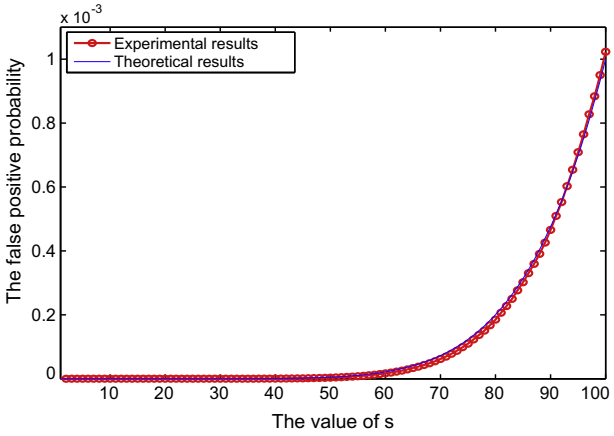


Fig. 15. The theoretical and experimental values of f_{ABF} as functions of the number of BFs, where $\sigma = 0.001$, $n = 30$, and the parameters m and k are optimized by Formulas (4)–(7).

trend and match well as the value of s increases continuously from 1 to 25. Such a result further confirms the correctness of Formula (3).

Figs. 14 and 15 plot the evaluation results of f_{CUBF} and f_{ABF} under the second scenario in which the receiver-oriented design of BF is adopted. The two curves in Fig. 14 do not match completely, however, the difference is very small when s ranges from 1 to 50. The two curves in Fig. 15 match well, without respect to the value of $1 \leqslant s \leqslant 100$. The evaluation results demonstrate that the optimizations of parameters k and m, as shown by Formulas (4)–(7), are correct and effective.

6. Conclusion

BF-based data-centric routing has been widely used and extensively studied in the field of network applications. In this paper, we study the false positive problem in the traditional designs of BFs in data-centric routing schemes. We disclose that previous data-centric routing schemes using Bloom filters cannot facilitate in-network queries correctly, due to the noise in BF-based routing entries. Based on the evaluation results of previous designs, namely CUBF and ABF, we propose the receiver-oriented design of BFs, which guarantees that the probability of making a wrong routing decision at any given node is sufficiently low. Further, we enhance our proposal by significantly decreasing the delay of each routing decision in the case of CUBF, and reducing the transmission size of each BF in the case of ABF.

Following this work, we will extend it in several potential directions. First, we plan to evaluate the impact of the topological properties, data distribution among the nodes, and the popularity of queries on the receiver-oriented designs of CUBF and ABF. Second, weak state routing using decay BFs is another potential research direction [6]. Third, we will study the impact of the topology mismatch problem [28–30] on this work.

Acknowledgments

This work is supported in part by NSF China under Grants Nos. 60903206, 60903225, National Basic Research Program of China (973 Program) under Grants Nos. 2009CB3020402, 61364, and National High Technology Research and Development Program of China (863 Program) under Grant No. 2008AA01Z216.

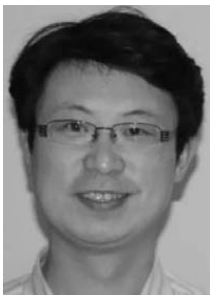
References

[1] A. Broder, M. Mitzenmacher, Network applications of bloom filters: a survey, Internet Mathematics 1 (4) (2005) 485–509.
[2] J. Mullin, Optimal semijoins for distributed database systems, IEEE Transactions on Software Engineering 16 (5) (1990) 558–560.
[3] L. Fan, P. Cao, J. Almeida, A. Broder, Summary cache: a scalable wide-area web cache sharing protocol, IEEE/ACM Transactions on Networking 8 (3) (2000) 281–293.
[4] J. Li, J. Taylor, L. Serban, M. Seltzer, Self-organization in peer-to-peer system, in: Proceedings of the 10th ACM SIGOPS European Workshop, Saint-Emilion, France, 2002.
[5] S. Rhea, J. Kubiatowicz, Probabilistic location and routing, in: Proceedings of the IEEE INFOCOM, New York, USA, 2004, pp. 1248–1257.
[6] A. Kumar, J. Xu, E. Zegura, Efficient and scalable query routing for unstructured peer-to-peer networks, in: Proceedings of the IEEE INFOCOM, Miami, FL, United States, 2005, pp. 1162–1173.
[7] F. Deng, D. Rafiei, Approximately detecting duplicates for streaming data using stable bloom filters, in: Proceedings of the 25th ACM SIGMOD, Chicago, Illinois, USA, 2006, pp. 25–36.
[8] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, G. Varghese, Beyond bloom filters: from approximate membership checks to approximate state machines, in: Proceedings of the ACM SIGCOMM, Pisa, Italy, 2006, pp. 315–326.
[9] K. Li, Z. Zhong, Fast statistical spam filter by approximate classifications, in: Proceedings of the SIGMETRICS/Performance, Saint Malo, France, 2006, pp. 347–358.
[10] M. Jimeno, K. Christensen, A. Roginsky, A power management proxy with a new best-of-n bloom filter design to reduce false positives, in: Proceedings of the 26th IEEE International Performance Computing and Communications Conference (IPCCC), Louisiana, USA, 2007.
[11] F. Hao, M. Kodialam, T.V. Lakshman, Building high accuracy bloom filters using partitioned hashing, in: Proceedings of the SIGMETRICS/Performance, San Diego, CA, USA, 2007, pp. 277–287.
[12] B. Chazelle, J. Kilian, R. Rubinfeld, A. Tal, The bloomier filter: an efficient data structure for static support lookup tables, in: Proceedings of the Fifth SODA, New Orleans, Louisiana, USA, 2004, pp. 30–39.
[13] R.P. Laufer, P.B. Velloso, O.C.M.B. Duarte, Generalized bloom filters, Tech. Rep. Research Report GTA-05-43, University of California, Los Angeles (UCLA), September 2005.
[14] T. Hodes, S. Czerwinski, B. Zhao, An architecture for secure wide-area service discovery, Wireless Networks 8 (2–3) (2002) 213–230.

[15] P. Reynolds, A. Vahdat, Efficient peer-to-peer keyword searching, in: Proceedings of the ACM International Middleware Conference, Rio de Janeiro, Brazil, 2003, pp. 21–40.
[16] D. Bauer, P. Hurley, R. Pletka, M. Waldvogel, Bringing efficient advanced queries to distributed hash tables, in: Proceedings of the IEEE Conference on Local Computer Networks, Tampa, FL, United States, 2004, pp. 6–14.
[17] P. Hebden, A.R. Pearce, Data-centric routing using bloom filters in wireless sensor networks, in: Proceedings of the Fourth International Conference on Intelligent Sensing and Information Processing (ICISIP), Bangalore, India, 2006.
[18] P.-H. Hsiao, Geographical region summary service for geographical routing, in: Proceedings of the Second ACM MobiHoc, Long Beach, CA, USA, 2001, pp. 263–266.
[19] W.H. Yuen, H. Schulzrinne, Improving search efficiency using bloom filters in partially connected ad hoc networks: a node-centric analysis, Computer Communications 30 (16) (2007) 3000–3011.
[20] C.F. Chan, Mole: multi-hop object location in wireless mesh networks, Ph.D. Thesis, Hong Kong University of Science and Technology, August 2008.
[21] D. Guo, J. Wu, H. Chen, X. Luo, Theory and network applications of dynamic bloom filters, in: Proceedings of the 25th IEEE INFOCOM, Barcelona, Spain, 2006.
[22] M. Mitzenmacher, Compressed bloom filters, IEEE/ACM Transactions on Networking 10 (5) (2002) 604–612.
[23] J. Lian, Y. Liu, K. Naik, L. Chen, Virtual surrounding face geocasting with guaranteed message delivery for ad hoc and sensor networks, IEEE/ACM Transactions on Networking 17 (1) (2009) 200–211.
[24] M. Li, Y. Liu, Underground coal mine monitoring with wireless sensor networks, ACM Transactions on Sensor Networks 5 (2) (2009).
[25] M. Li, Y. Liu, L. Chen, Non-threshold based event detection for 3d environment monitoring in sensor networks, IEEE Transactions on Knowledge and Data Engineering 20 (12) (2008) 1699–1711.
[26] Y.J. Chen, Y.B. Chen, On the signature tree construction and analysis, IEEE Transactions on Knowledge and Data Engineering 18 (9) (2006) 1–18.
[27] W.W. Terpstra, J. Kangasharju, C. Leng, A.P. Buchmann, Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search, in: Proceedings of the SIGCOMM, Kyoto, Japan, 2007.
[28] Y. Liu, A two-hop solution to solving topology mismatch, IEEE Transactions on Parallel and Distributed Systems 19 (11) (2008) 1591–1600.
[29] Y. Liu, L. Xiao, L. Ni, Building a scalable bipartite p2p overlay network, IEEE Transactions on Parallel and Distributed Systems 18 (9) (2007) 1296–1306.
[30] Y. Liu, L. Xiao, X. Liu, L. Ni, X. Zhang, Location awareness in unstructured peer-to-peer systems, IEEE Transactions on Parallel and Distributed Systems 16 (2) (2005) 163–174.

Deke Guo received the BE degree in industry engineering from Beijing University of Aeronautic and Astronautic, Beijing, China, in 2001, and the PhD degree in management science and engineering from National University of Defense Technology, Changsha, China, in 2008. He was a visiting scholar in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology from January 2007 to January 2009. Currently, he is an assistant professor of Information System and Management, National University of Defense Technology, Changsha, China. His current interests include peer-to-peer computing, pervasive computing, and wireless multi-hop networks. He is a member of the ACM and the IEEE.

Yuan He received his BE degree in Department of Computer Science and Technology from University of Science and Technology of China in 2003, and his ME degree in Institute of Software, Chinese Academy of Sciences, in 2006. He is now a PhD student in the Department of Computer Science and Engineering at Hong Kong University of Science and Technology, supervised by Dr. Yunhao Liu. His research interests include peer-to-peer computing, sensor networks, and pervasive computing. He is a student member of the IEEE and the IEEE Computer Society.



Panlong Yang received his BS degree, MS degree, and PhD degree in communication and information system from Nanjing Institute of Communication Engineering, China, in 1999, 2002, and 2005, respectively. During November 2006 to March 2009, he was a postdoc fellow in the Department of Computer Science, Nanjing University. He is now an associate professor in the Nanjing Institute of Communication Engineering. His research interests include wireless mesh networks, wireless sensor networks and cognitive radio networks. He is a member of the IEEE Computer Society and ACM SIGMOBILE Society.