TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 04/10 pp157-167
Volume 21, Number 2, April 2016

# Comparing Set Reconciliation Methods Based on Bloom Filters and Their Variants

Zhiyao Hu, Xiaoqiang Teng, Deke Guo*, Bangbang Ren, Pin Lv, and Zhong Liu

**Abstract:** Set reconciliation between two nodes is widely used in network applications. The basic idea is that each member of a node pair has an object set and seeks to deliver its unique objects to the other member. The Standard Bloom Filter (SBF) and its variants, such as the Invertible Bloom Filter (IBF), are effective approaches to solving the set reconciliation problem. The SBF-based method requires each node to represent its objects using an SBF, which is exchanged with the other node. A receiving node queries the received SBF against its local objects to identify the unique objects. Finally, each node exchanges its unique objects with the other node in the node pair. For the IBF-based method, each node represents its objects using an IBF, which is then exchanged. A receiving node subtracts the received IBF from its local IBF so as to decode the different objects between the two sets. Intuitively, it would seem that the IBF-based method, with only one round of communication, entails less communication overhead than the SBF-based method, which incurs two rounds of communication. Our research results, however, indicate that neither of these two methods has an absolute advantages over the others. In this paper, we aim to provide an in-depth understanding of the two methods, by evaluating and comparing their communication overhead. We find that the best method depends on parameter settings. We demonstrate that the SBF-based method outperforms the IBF-based method in most cases. But when the number of different objects in the two sets is below a certain threshold, the IBF-based method outperforms the SBF-based method.

**Key words:** set reconciliation; bloom filter; communication overheads

## 1 Introduction

Set reconciliation is a fundamental task in most distributed applications, where two or more nodes wish to compute the union of the sets, such as file systems[1, 2], mobile databases[3, 4], gossip protocols[5], and resource location systems[6–9]. It is realized by synchronizing the unique objects of the two sets. The Standard Bloom Filter (SBF)[10] and its variant, the Invertible Bloom Filter (IBF)[11], are two mainstream methods for set reconciliation in distributed applications. The two methods, however, vary in the communication overhead they entail. This usually depends on set cardinality, the number of different objects in the two sets, and the storage space occupied by each object. To date, the communication overheads of such set reconciliation methods have not been explored. Consequently, users lack knowledge about which set reconciliation method will cause less communication overhead in given application scenarios.

This problem becomes particularly challenging in practice due to the uncertainty of the set cardinality

• Zhiyao Hu, Xiaoqiang Teng, Deke Guo, Bangbang Ren, and Zhong Liu are with the College of Information System and Management, National University of Defense Technology, Changsha 410073, China. E-mail: guodeke@gmail.com.
• Pin Lv is with School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China.
∗ To whom correspondence should be addressed.
Manuscript received: 2016-01-12; accepted: 2016-02-22

and the number of different objects in two sets. For example, for two sets $S_A$ and $S_B$, the set cardinality is $n_A$ and $n_B$, respectively. We assume that $n_A$ and $n_B$ equal 10 000 and 50 000, respectively, while the number of different objects in sets $S_A$ and $S_B$ is 2000. In this setting, it is difficult to decide which method results in less communication overhead. Less communication overhead is very important for many distributed applications, especially when many nodes synchronize with each other.

In this paper, we propose communication overhead models for two representative set reconciliation methods, SBF-based and IBF-based. In these settings, communication overhead depends on set cardinality, the number of different objects in the two sets, and the storage space occupied by each object. In many applications, all objects are generated via a single hash function; hence, they occupy the same amount of storage. Such hash functions, including MD5 with 128 bit, SHA-1 with 160 bit, SHA-256 with 256 bit, SHA-512 with 512 bit, etc., generate fixed length output for any input. By varying the settings of any pair of sets, the proposed communication overhead models demonstrate that the SBF-based method outperforms the IBF-based method in most cases. The reverse conclusion appears only when the number of different objects in two sets is below a certain threshold. We find that the IBF-based set reconciliation method is most suitable to the scenario where a pair of sets have few different objects, while the SBF-based set reconciliation method performs better with two sets with many different objects. Using our models, applications can adaptively select the best method to solve the set reconciliation problem.

The remainder of this paper is organized as follows. We briefly describe the preliminaries in Section 2. We model and analyze the communication overhead of SBF-based and IBF-based set reconciliation methods in Section 3. We comprehensively evaluate the performance of the two methods in Section 4, and conclude this work in Section 5.

## 2 Preliminaries

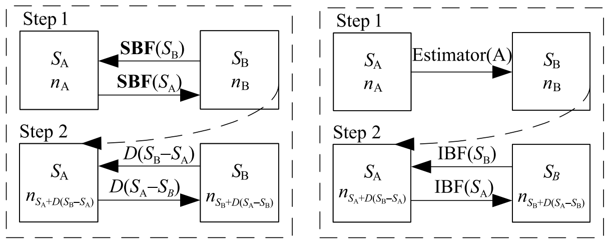
Given two nodes, A and B, let each node maintain a set, i.e., $S_A$ and $S_B$, respectively. Let $n_A$ and $n_B$ be the cardinality of $S_A$ and $S_B$, respectively. The goal of set reconciliation is for A and B to compute $S_A \cup S_B$ with minimal communication overhead. The classical approach to set reconciliation is wholesale transfer, in which one party sends all its elements to the other. The methods described herein are attempts to reduce this overhead.

### 2.1 SBF-based set reconciliation method

An SBF consists of $m$ cells and utilizes a binary vector to represent a set $S=\{s_1, s_2, \ldots, s_n\}$ with $n$ objects. Each cell is a bit with an initial value of 0. A number of $k$ hash functions, denoted as $\langle h_1, h_2, \ldots, h_k \rangle$, are employed to map each object in the set to $k$ random positions in the vector. In this way, we can answer a membership query for any object $s_i$, and a user can check whether all bits $h_j(s_i)$ are set to 1 for $1 \leqslant j \leqslant k$. If not, SBF returns "false", i.e., $s_i$ is not a member of $S$. Otherwise, we assume that $s_i$ is a member of $S$. Due to hash collisions, an SBF may yield a false positive if it wrongly identifies an object $s_i$ as belonging to $S$. The cause is that all bits at SBF$[h_j(s_i)]$ for $1 \leqslant j \leqslant k$ have been set to one by other objects in $S$[12]. The false positive probability can be theoretically derived as follows[13, 14]:

$$f = (1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - \mathrm{e}^{-k \times n/m})^k \qquad (1)$$

The SBF-based set reconciliation method includes two interactions, as shown in Fig. 1a. In the first step, the resulting binary vector of $S_A$, represented as **SBF**$(S_A)$, is sent to node B. By querying the objects in **SBF**$(S_A)$, node B can identify the objects, denoted as $D(S_B - S_A)$, which do not exist in $S_A$. Similarly, node A identifies objects of $D(S_A - S_B)$ after receiving **SBF**$(S_B)$ from node B. In the second step, the two nodes exchange their unique objects and solve the set reconciliation problem. A challenging issue is that the SBF-based approach may yield a false positive and thus an object may be misidentified as common to both sets. The false positive rate depends on the number of bits used for each object and the number of hash functions. Therefore, for a fixed false positive rate, the space used



(a) SBF-based method (b) IBF-based method

**Fig. 1 Two types of set reconciliation.**

by an SBF is proportional to the set cardinality, not the set difference.

### 2.2 IBF-based set reconciliation method

An IBF[11] employs an array of cells, $S[1], \ldots, S[m]$. A cell consists of three fields, called idSum, hashSum, and count, which denoted by $S[i]$.idSum, $S[i]$.hashSum, and $S[i]$.count.

- The idSum field records the XOR results of all objects that are hashed into that cell.
- The hashSum field stores the sum of hash values of all objects.
- The count field records the number of objects mapped into the cell.

The IBF allows the following operations:

- The operation Insert $(x, y)$ adds a key-value to the IBF, i.e., $S[h_i(x)]$.idSum $= S[h_i(x)]$.idSum $\oplus\, x$, $S[h_i(x)]$.hashSum $= S[h_i(x)]$.hashSum $\oplus\, y$, and $S[h_i(x)]$.count $= S[h_i(x)]$.count+1.
- The operation Delete$(x,\ y)$ removes a key-value pair $(x,\ y)$ from the IBF. It subtracts $x$ and $y$ from the idSum and hashSum fields and decrements the count field.
- The operation Get$(x)$ retrieves the value associated with a key $x$ from the IBF. If $S[h_i(x)]$.count $= 1$, then it returns $S[h_i(x)]$.hashSum. Otherwise, it returns "not found".
- The operation ListEntries() recovers all the objects stored in the IBF by sequentially removing the objects whose counter value equals one. It faces the listing failure issue in this process. See Ref. [15] for more details.

Each object in any set is mapped to at most $k$ cells in the IBF via a set of hash functions. It is the idSum field in each cell that encodes all objects in it via the *XOR* operation. After representing two sets with the IBF and exchanging the two IBFs, the proposed subtract operation between IBFs can eliminate common elements of sets $S_{\mathrm{A}}$ and $S_{\mathrm{B}}$. Furthermore, the different objects can be recovered by using the *extraction* operation iteratively with a given probability.

As shown in Fig. 1b, nodes A and B generate IBF($S_{\mathrm{A}}$) and IBF($S_{\mathrm{B}}$). The two nodes then exchange their local IBFs with each other, and subtract the received remote IBF from each local IBF. Then the two nodes try to recover the objects that are different between the two related sets.

### 2.3 CBF-based set reconciliation method

Recall that in the SBF-based set reconciliation method, the deletion of any object enables all cells in SBF$[h_j(s_i)]$ for $1 \leqslant j \leqslant k$ to be reset to 0. Other objects in $S$ that hash to one or more cells at SBF$[h_j(s_i)]$ for $1 \leqslant j \leqslant k$ will no longer be correctly identified. To address this problem, Fan et al.[16] proposed the Counting Bloom Filter (CBF). A CBF provides a way to implement a delete operation on an SBF without regenerating the filter. In a CBF, each item of its bit vector is extended from being a single bit to being an $m \times l$-bit counter, and $l = 4$ usually suffices[17, 18].

The CBF-based set reconciliation method requires each node to represent its objects using a CBF, which is exchanged with the other node. A receiving node subtracts the received CBF from its local CBF so as to query the objects not common to the two sets. However, in this paper, we focus on two mainstream methods for set reconciliation, i.e., SBF-based and IBF-based methods, to understand the set reconciliation problem.

In this paper, a single capital letter like $S$ denotes a set with cardinality $n$. Other notations and symbols are summarized in Table 1.

## 3 Modeling and Analysis

A metric is derived to evaluate the communication overhead of the SBF-based and the IBF-based set reconciliation methods. For the SBF-based method, due to false positives, some objects that only appear in one set may not be identified. The IBF-based method avoids

**Table 1 Symbols and notations.**

| Term | Definition |
|---|---|
| $n$ | Set cardinality |
| $d$ | Total size of the set differences |
| $u$ | Universe set |
| $c_k + \lambda$ | Coefficient of $d$ in IBF |
| $f$ | False positive probability of an SBF |
| $\rho$ | Ratio of $d$ to $n$ |
| $M$ | Storage space size of an object in the set |
| $R$ | Number of cells in each IBF partition |
| $L$ | Number of strata tried in difference estimation |
| $C$(ES) | Communication overhead of exchanging SBFs |
| $C$(ED) | Communication overhead of exchanging unique objects from another node for SBF |
| $C$(EP) | Communication overhead of estimating the number of different objects for IBF |
| $C$(EI) | Communication overhead of exchanging IBFs between the two nodes |
| $C$(SBF) | Communication overhead of the SBF-based method |
| $C$(IBF) | Communication overhead of the IBF-based method |

this problem, but faces the vital risk that some objects of the two sets cannot be recovered successfully (i.e., listing failure probability). If any of the different objects cannot be decoded, each node has to allocate more cells to its IBF. In the worst case, all of the different objects are not decoded by the IBF-based method. In order to fairly compare the performance of the two methods, the missed different objects at both nodes should be less than or equal a threshold, e.g., at most 1% of the set differences for both methods.

## 3.1 Communication overhead of the SBF-based method

The communication overhead of the SBF-based method is determined in two phases. The first phase is determining the cardinality of each set as a result of the SBF exchange process. The second phase involves determining the number of different objects and the storage requirements of each object.

### 3.1.1 Communication overhead of exchanging SBFs

In the process of exchanging SBFs, the communication overhead depends on the length of the SBF vector (i.e., $m$). Broder and Mitzenmacher[19] found the lower bound of $m$. When the optimal value of $k$ is set to $\frac{m}{n}\ln 2$, the false positive rate of an SBF is given by $f=\left(\frac{1}{2}\right)^k \geqslant \left(\frac{1}{2}\right)^{\frac{m\ln 2}{n}}=0.6185^{\frac{m}{n}}$ [17]. We can derive that

$$m \geqslant \frac{n\times\log(f)}{\log(0.6185)} \tag{2}$$

In what follows, all communication overheads are computed based on the minimal value of $m$. Recall that the numbers of objects in $S_{\rm A}$ and $S_{\rm B}$ are denoted by $n_{\rm A}$ and $n_{\rm B}$. Since both nodes have to transmit their SBFs, we derive that the communication overhead $C(\text{ES})$ on both sides of any node pair is

$$C(\text{ES}) = m_{\rm A} + m_{\rm B} = \frac{n_{\rm A}\times\log(f_{\rm A})}{\log(0.6185)} + \frac{n_{\rm B}\times\log(f_{\rm B})}{\log(0.6185)} \tag{3}$$

where $f_{\rm A}$ and $f_{\rm B}$ denote the false positive rates of $S_{\rm A}$ and $S_{\rm B}$.

As mentioned, every object in $S_{\rm A}$ may be undiscovered as being common to both nodes with probability $(1-{\rm e}^{-k\times|S_{\rm B}|/m})^k$ in node A. Similarly, every object in $S_{\rm B}$ may be undiscovered as common to both nodes with probability $(1-{\rm e}^{-k\times|S_{\rm A}|/m})^k$ in node B. Let $f_{\rm B}=(1-{\rm e}^{-k\times|S_{\rm B}|/m})^k$ and $f_{\rm A}=(1-{\rm e}^{-k\times|S_{\rm A}|/m})^k$. Since the number of undiscovered different objects should be less than or equal to 1% of the set differences, there is a constraint:

$$0 \leqslant n_{\rm A}\times f_{\rm B} + n_{\rm B}\times f_{\rm A} \leqslant \lceil 0.01\times d\rceil \tag{4}$$

Based on the cardinality of each set, the false positive probability of each SBF can be calculated. For example, we assume that $n_{\rm A}=n_{\rm B}=10\,000$ and $d=1000$, then $f_{\rm A}+f_{\rm B}\leqslant 0.001$. In this paper, $f_{\rm A}$ and $f_{\rm B}$ have the same value. Hence, $f_{\rm A}=f_{\rm B}=0.0005$ if the maximal communication overhead is considered.

### 3.1.2 Communication overhead of delivering different objects

As shown in Fig. 1a, unique objects are exchanged in the second step of the SBF-based method. Node A sends $D(S_{\rm A}-S_{\rm B})$ (i.e., the unique set of $S_{\rm A}-S_{\rm B}$) to node B, while node B sends $D(S_{\rm B}-S_{\rm A})$ to A. In this process, the communication overheads are determined by the cardinalities of $D(S_{\rm A}-S_{\rm B})$ and $D(S_{\rm B}-S_{\rm A})$, and the size of each object ($M$) in these two sets, where $D(S_{\rm A}-S_{\rm B})+D(S_{\rm B}-S_{\rm A})=d$. Due to false positives, the number of different objects that are not transferred is $n_{\rm A}\times f_{\rm B}+n_{\rm B}\times f_{\rm A}$. The communication overhead of exchanging different objects $C(\text{ED})$ is

$$C(\text{ED}) = M\times d - M\times n_{\rm A}\times f_{\rm B} - M\times n_{\rm B}\times f_{\rm A} \tag{5}$$

### 3.1.3 Total communication overheads

As a result, the communication overhead of the SBF-based set reconciliation method $C(\text{SBF})$ is

$$\begin{aligned} C(\text{SBF}) = C(\text{ES}) + C(\text{ED}) = \\ \frac{n_{\rm A}\times\log(f_{\rm A})}{\log(0.6185)} + \frac{n_{\rm B}\times\log(f_{\rm B})}{\log(0.6185)} + \\ M\times d - M\times n_{\rm A}\times f_{\rm B} - M\times n_{\rm B}\times f_{\rm A} \end{aligned} \tag{6}$$

It is clear that the communication overhead of the SBF-based set reconciliation method is dominated by six parameters, $M$, $d$, $n_{\rm A}$, $n_{\rm B}$, $f_{\rm A}$, and $f_{\rm B}$.

## 3.2 Communication overhead of IBF-based method

The communication overhead of the IBF-based method is determined in two phases. The first phase is the estimator process, while the second phase involves exchanging IBFs between two nodes.

### 3.2.1 Communication overhead of estimator

A precondition of IBF-based set reconciliation is to estimate the number of necessary cells in each IBF. When the number of allocated cells is underestimated, the probability that the IBF cannot be decoded to recover the different objects from the two sets will significantly increase. In the case of overestimation, the set reconciliation incurs additional communication overhead. Actually, the number of employed cells heavily depends on the number of different objects in each pair of sets. To ensure that all different objects in the two sets can be decoded via IBF with high probability, it is essential to estimate the number of different objects in the two sets in advance. Note that the number of required cells is determined by the number of different objects in the two sets[11]. The problem becomes estimating the number of different objects.

Eppstein et al.[11] designed the *Strata Estimator* to tackle this challenging issue. The universe set $u$ is separated into $L = \log_2(u) - 1$ partitions, such that the $i$-th partition covers a fraction that is $1/2^{i+1}$ of $u$. Then each partition is packed into an array of fixed size $T$ (Eppstein et al.[11] chose an 80-cell array). This layered structure of $\log_2(u)$ IBFs is called the strata estimator. The estimator sends its output to the remote peer. The peer then tries to do an IBF subtract on all the strata starting from layer $L$. Each time recovering succeeds, the number of recovered objects is added to a counter. If different objects in level $k$ are not successfully recovered, then the estimator returns $2^{k+1} \times$ count, where count is the number of objects recovered. The approximate value is treated as the number of different objects.

A node (e.g., A) initiates the estimating process by sending its own estimator to another node (e.g., B). After B receives the IBFs from A, it estimates the set difference and replies with an IBF. Therefore, the estimating process results in one communications round to compute the set difference. Let $R$ denote the cell number of each IBF in the Strata Estimator. Then the communication overhead of estimator $C(\text{EP})$ is

$$C(\text{EP}) = L \times R \times i \tag{7}$$

where $i$ denotes the size of each cell.

### 3.2.2 Communication overhead of exchanging IBFs

After the Strata Estimator returns the size of the set difference, $d$, we allocate the IBFs for A and B, which consist of tables with $m = \alpha d$ cells. We will describe the setting of the parameter $\alpha$ in Section 3.2.4. Each cell of the table contains three fields (i.e., idSum, hashSum, and count).

Let $M$ denote the size of idSum. For the size of the hashSum field, Eppstein and Goodrich[20] proved that it should be at least $(2 \times \log(n) + \log(d))$ bits. It can store $d$ integers in the range $[0, n^2]$. Let $C$ denote the size of the count field. Generally, the expectation of count field is 16 bits, which is sufficient to represent 65 536 objects for each cell. Thus, the size of each cell can be calculated by the following equation:

$$i \approx M + 2 \times \log(n) + \log(d) + C \tag{8}$$

Consider that the exchange of IBFs between two nodes is bidirectional. Its communication overhead can be calculated as

$$C(\text{EI}) = 2 \times \alpha \times d \times i \tag{9}$$

### 3.2.3 Total communication overhead

The total communication overhead of the IBF-based set reconciliation method $C(\text{IBF})$ can thus be calculated as

$$\begin{gathered} C(\text{IBF}) = C(\text{EP}) + C(\text{EI}) = \\ L \times R \times i + 2 \times \alpha \times d \times i \approx \\ (L \times R + 2 \times \alpha \times d) \times (M + 2 \times \log(n) + \log(d) + C) \end{gathered} \tag{10}$$

### 3.2.4 Listing failure probability

The IBF-based set reconciliation method will generally return the object and a null value if queried for an object not in the set. However, an IBF may yield a listing failure, whereby it returns a “not found” value for objects in the set and objects not in the set. The cause is that any empty cell could not be found in the IBF table. The listing failure probability depends on the definition of the probabilistic model for objects and hash functions.

The listing process is probabilistic and similar to the problem of constructing the 2-core of a random hypergraph[21–23]. Goodrich and Mitzenmacher[15] presented a constant, $c_k > 1$, such that solution can be found with high probability if $m > (c_k + \lambda) \times d$ for any $\lambda > 0$, where $m$ denotes the number of cells in an IBF. As shown in Ref. [15], these values are given by

$$c_k^{-1} = \sup\{\beta\colon 0 \leqslant \beta \leqslant 1; \forall x \in (0,1), 1 - \mathrm{e}^{-k\beta x^{k-1}} \leqslant x\} \tag{11}$$

Numerical values for $k \geqslant 3$ are given in Table 2. Eppstein and Goodrich[20] proved the fail probability of such a probabilistic process.

**Table 2 Thresholds for the 2-core rounded.**

| $k$ | $c_k$ | $k$ | $c_k$ |
|---|---|---|---|
| 3 | 1.222 | 6 | 1.579 |
| 4 | 1.295 | 7 | 1.721 |
| 5 | 1.425 | | |

**Theorem 1**[20] As long as $m$ is chosen such that $m > (c_k + \lambda)d$ for some $\lambda > 0$, decoding different objects fails with probability $O(d^{-k+2})$, whenever $d \leqslant n$.

Due to the number of undiscovered different objects should be less than or equal 1%, we need to meet the constraint that $d \times O(d^{-k}) \leqslant \lceil 0.01 \times d \rceil$.

## 4 Performance Evaluation

In this section, we evaluate the performance of the SBF-based and IBF-based set reconciliation methods in terms of their communication overheads. We point out which set reconciliation method will result in less communication overhead under different application scenarios.

### 4.1 Experimental methodologies

We implement the SBF and IBF methods and extend them to support set reconciliation. The parameters include the set cardinality, $n$, the storage requirement of each object, $M$, the set differences between two sets, $d$, and the false positive rate of SBF. For each setting of the parameters, we report the average results for 200 rounds of experiments. An $u$ of all 32-bit values is used in our evaluation. We have two degrees of randomizing in creating pairs of sets containing keys from $u$, $S_A$, and $S_B$, each at one node: the set cardinality and the number of set differences between the two sets. In our experiments, the two sets have the same cardinality. According to Eq. (4), we have $f_A = f_B = \dfrac{0.01 \times d}{2n}$. We use 4 hash functions, so we take $c_k = 1.295$ and use a table size of $\lceil c_k + \lambda \rceil$ cells[15]. The 4 hash functions are chosen randomly by adopting the ones used in Ref. [17].

$$h_i(x) = (g_1(x) + i \times g_2(x)) \bmod m \qquad (12)$$

where $g_1(x)$ and $g_2(x)$ are two random and independent integers in the universe with a range $\{1, 2, ..., m\}$. The value of $i$ ranges from 0 to $k - 1$. In order to conduct a similar experiment to that of Ref. [11], we take $R = 80$. We perform 200 rounds of experiments to take $L = \lfloor \log_2 d \rfloor$, as shown in Table 3. For example, we report the number of successful estimate of the difference in eighth strata is 176 of 200 experiments, i.e., $\lfloor \log_2(500) \rfloor = 8$, where $n = 5000$ and $d = 500$.

### 4.2 Impact of storage size of each object in sets

To efficiently use the two methods of set reconciliation, we firstly examine how the storage size of each object in a set affects the communication overhead due to the different hash methods. The storage size of each object varies under different hash methods, such as MD5 with 128 bit, SHA-1 with 160 bit, SHA-256 with 256 bit, and SHA-512 with 512 bit. Therefore, we discuss the parameter $M$ in more general settings.

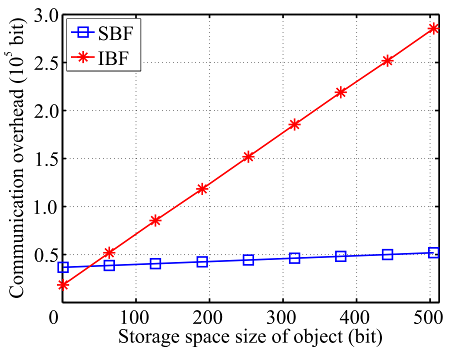
**Table 3 The number of tried layers for Estimator.**

| $n$ | $d$ | Layers | | | | | |
|---|---|---|---|---|---|---|---|
| | | 5 | 6 | 7 | 8 | 9 | 10 |
| 1000 | 100 | 11 | $\underline{177}$ | 12 | – | – | – |
| 5000 | 500 | – | – | 5 | $\underline{176}$ | 19 | – |
| 10 000 | 1000 | – | – | – | 13 | $\underline{174}$ | 13 |
| 20 000 | 2000 | – | – | – | – | 15 | $\underline{185}$ |

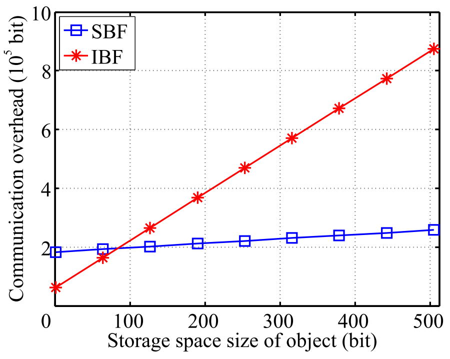
In this case, $M$ changes from 1 bit to 512 bit. Figure 2 shows the communication overheads of SBF-based and IBF-based methods for various sizes of $M$, where $d = 0.03n$ for different values of $n$. We can see that the two lines of SBF- and IBF-based set reconciliation methods, under different settings of $M$, follow similar trends of linear increase, as $M$ increases from 1 bit to 512 bit. The IBF-based method is sensitive to $M$. That is because that IBF-based set reconciliation method has two communication rounds for the idSum field in IBF between two nodes, including the difference estimating process and exchanging the IBFs. The SBF-based set reconciliation method has only one communication round for the objects in BF between two nodes, i.e., the process of exchanging unique objects. In addition, we observe that neither of those two methods has absolute advantages under different settings of $M$. For example, as shown in Fig. 2c, the two lines of SBF- and IBF-based set reconciliations intersect at $M =$ 110 bit. The two set reconciliation methods achieve equal communication overhead when $M = 110$ bit, $n = 10\,000$, and $d = 0.03n$. We also see that the IBF-based method outperforms the IBF-based method when $M$ is less than 110 bit. Only when $M$ is more than 110 bit, does the SBF-based method outperform the IBF-based method.

### 4.3 Impact of set cardinality

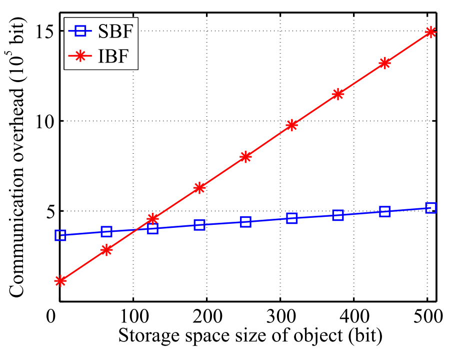
We examine the influence of the set cardinality ($n$) on the communication overheads of SBF- and IBF-based set reconciliation methods. Let $n$ vary from 100 to 20 000. We have $d = 100$ and $M = 128$ bit. In Fig. 3a, we report experimental results. We see that the two curves of SBF- and IBF-based set reconciliation methods, under different settings of $n$, follow similar



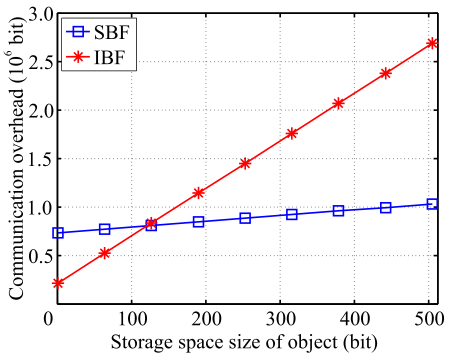
(a) $n$=1000



(b) $n$=5000

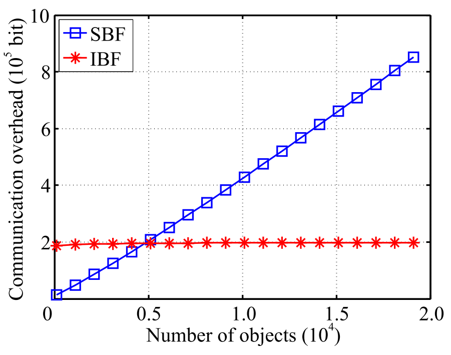


(c) $n$=10 000

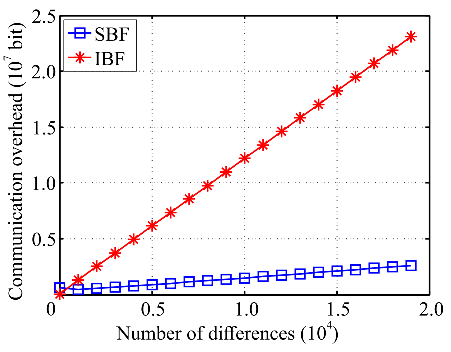


(d) $n$=20 000

**Fig. 2 Impact of the storage space size of each object in a set, when $n$=1000, 5000, 10 000, 20 000, and $d$= 0.03$n$, where $n$ is the cardinality of a set and $d$ denotes the number of different objects between two sets.**



(a) $d$=100



(b) $n$=10 000

**Fig. 3 (a) The impact of the set cardinality when $n$ ranges from 100 to 20 000. (b) The impact of the number of differences when $d$ ranges from 1 to 20 000.**

increasing trends, as $n$ grows from 100 to 20 000. The curve of the SBF-based method increases faster than that of the IBF-based method. That is because the number of the cells used by the SBF-based method is sensitive to the set cardinality, according to Formula (2), while the number of cells used by the IBF-based method is sensitive to the set differences, according to Ref. [11]. We again observe that neither of the methods has absolute advantages under different settings of $n$. In Fig. 3a, the two curves of SBF- and IBF-based methods intersect at $n = 4843$, i.e., the two methods achieve equal communication overhead when $n = 4843$, $d = 100$, and $M = 128$ bit. The SBF-based method outperforms the IBF-based method when $n$ is less than 4843. Only when $n$ is more than 4843, does the IBF-based method outperform the SBF-based method.

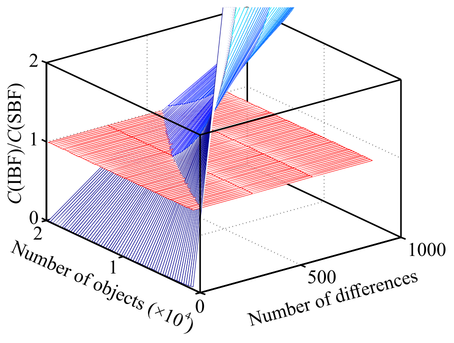
## 4.4 Impact of set differences

We examine the influence of the set differences ($d$) on the communication overheads of SBF- and IBF-based set reconciliation methods. Let $n = 10\,000$ and $M = 128$ bit. Because $d$ equals $2n$ at most, let $d$ vary from 1 to 20 000. In Fig. 3b, we report the experimental results. We see that the curves of the SBF- and the

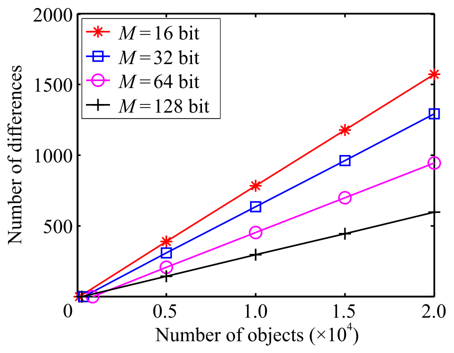
IBF-based set reconciliation methods, under different settings of $d$, follow similar trends of increase, as $d$ increases from 1 to 20 000. The curve of the IBF-based method increases faster than that of the SBF-based method. That is because the number of the cells used by the IBF-based method is sensitive to the set differences according to Ref. [11], while the number of the cells used by the SBF-based method is sensitive to the set cardinality. In addition, we observe that the SBF-based method outperforms the IBF-based method as $d$ increases from 1 to 20 000 in most cases. Only when the set differences are few (e.g., 10 unique objects between any pair of sets), the IBF-based method outperforms the SBF-based method.

### 4.5 Synthetic impact of set cardinality and set differences

We synthetically discuss the impact of $n$ and $d$ on the communication overheads. In this case, we have $M = 128$ bit. Let $n$ change from 1 to 20 000. Since the number of the difference between two sets is at most $2n$, we have $d \leqslant 2n$ ($d \leqslant n$). In Fig. 4a, we see that the ratio of $C$(IBF) to $C$(SBF) grows as the increase



(a) $M$=128



(b) $M$=16, 32, 64, 128, respectively

**Fig. 4 The impact of the set cardinality and the number of different objects in a set when $n$ changes from 1 to 20 000 and $d$ changes from 1 to 2000 ($d \leqslant n$).**

of $n$ or $d$. The red plane presents $\dfrac{C(\text{IBF})}{C(\text{SBF})} = 1$. We again observe that neither of two methods has absolute advantages under different settings of the set cardinality and set differences. The communication overhead of SBF-based method is less than that of the IBF-based method above the red plane. Also, the communication cost of the IBF-based method is less than that of the SBF-based method below the red plane. Let $\rho$ denote the ratio of $d$ to $n$ on the red plane. As a result, only when $\dfrac{C(\text{IBF})}{C(\text{SBF})}$ exceeds the threshold $\rho$, does the SBF-based method outperform the IBF-based method in communication overhead. In order to derive $\rho$, we conduct an additional experiment as follows.

In our experiments, for any instance of $n$ that changes from 1 to 20 000 and $d$ changes from 1 to $2n$ ($d \leqslant n$), we focus on four representative settings of $M$, $M = 16$ bit, $M = 32$ bit, $M = 64$ bit, and $M = 128$ bit. Note that we report the average results from 200 rounds of experiments. We plot the lines that represent $\dfrac{C(\text{IBF})}{C(\text{SBF})} = 1$ with varying $n$ and $d$. From Fig. 4b, we can compute the value of $\rho$ in different settings $M$, as shown in Table 4. For example, when $M = 16$ bit and $\rho \leqslant 0.078$, does the IBF-based method outperform the SBF-based method.

### 4.6 Impact of the false positive probability

Given any pair of nodes (e.g., A and B), another issue of the SBF-based set reconciliation method is that each node may wrongly identify a common object as a unique one, with a given probability when querying SBF($S_A$) − SBF($S_B$) (i.e., false positive probability[24, 25]). Since each cell contains three fields (i.e., idSum, hashSum, and count) in the IBF-based set reconciliation method, the IBF-based method avoids the effects of the false positive probability. In this subsection, we only examine how the false positive probability affects the communication overhead for the SBF-based set reconciliation method.

According to Formula (4), the false positive probability is decided by the set cardinality and the number of differences. We vary $f_A$ from $\dfrac{0.01 \times d}{n}$ to

**Table 4 The ratio of set size and different objects between two sets with different $M$ values when $C$(SBF)=$C$(IBF).**

| $M$(bit) | $\rho$ | $M$(bit) | $\rho$ |
|---|---|---|---|
| 16 | 0.078 | 64 | 0.047 |
| 32 | 0.064 | 128 | 0.031 |

$\frac{10\times 0.01\times d}{n}$, and $f_{\rm B}$ from $\frac{0.01\times d}{n}$ to $\frac{10\times 0.01\times d}{n}-f_{\rm A}$. In this case, the set cardinality is 10 000, the number of different objects is 1000, and we fix $M=128$ bit. Figure 5 reports the experimental results. We can see that the resulting communication overhead of the SBF-based method decreases as the false positive probability increases. That is because the higher the false positive probability of the set, the fewer identified unique objects are exchanged between the two nodes. In practice, most approximate reconciliations could not sustain non-trivial numbers of missed unique objects. Therefore, the users should distribute a reasonable number of cells in SBFs to obtain a small false positive probability.

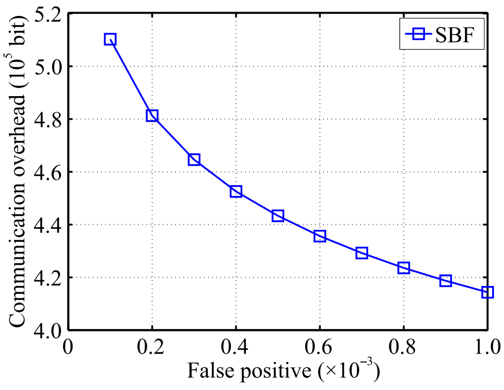


**Fig. 5 The communication overhead decreases when $f$ changes from 0.0001 to 0.001.**

### 4.7 Discussion

For the SBF-based set reconciliation method, due to false positives, some different objects unique to one set or the other cannot be identified. We can decrease the number of omitted unique objects by decreasing the false positive probability, while causing more communication overhead. For the IBF-based method, due to the probability of decoding failure, this degrades to a worst-case situation in which no unique object can be decoded. We demonstrate that the IBF-based set reconciliation method outperforms the SBF-based method when the ratio of the number of different objects to the set cardinality is smaller than a threshold, such as 10%, under different $\rho$ values. Moreover, we find that the communication overhead under the two methods increases linearly as the storage size of each object increases.

In summary, our study demonstrate that the IBF-based set reconciliation method works better in scenarios in which there exist fewer different objects between the two sets, while the SBF-based set reconciliation method works better when there are more different objects between the two sets. In practice, users should comprehensively consider the settings of parameters and choose the distinguished set reconciliation method with the least communication overhead.

## 5 Conclusion

Bloom filters and their variants have been widely used to solve set synchronization problem in many network applications. In this paper, we rethink the representative set reconciliation methods, the SBF-based and IBF-based set reconciliation methods, by evaluating their communication overheads. We propose communication overhead models for SBF-based and IBF-based set reconciliation methods. Comprehensive experiments demonstrate that neither of those two methods has absolute advantages in general scenarios. The SBF-based method outperforms the IBF-based method in most cases. But when the number of different objects between two sets is below a certain threshold, the IBF-based method outperforms the SBF-based method.

### Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Nos. 61422214 and 61402513), the National Key Basic Research and Development (973) Program of China (No. 2014CB347800), the Program for New Century Excellent Talents in University and Distinguished Young Scholars of National University of Defense Technology (No. JQ14-05-02), and the Preliminary Research Funding of National University of Defense Technology (No. ZDYYJCYJ20140601).

### References

[1] M. Satyanarayanan, A highly available file system for a distributed workstation environment, in *Proc. of the Second IEEE Workshop on Workstation Operating Systems*, Pacic Grove, CA, USA, 1989.

[2] A. J. Demers, D. H. Greene, C. Hause, W. Irish, and J. Larson, Epidemic algorithms for replicated database maintenance, in *Proc. of 6th Annual ACM Symposium on Principles of Distributed Computing*, Vancouver, Canada, 1987, pp. 1–12.

[3] D. Starobinski, A. Trachtenberg, and S. Agarwal, Efficient pda synchronization, *IEEE Trans. on Mobile Computing,* vol. 1, no. 2, 2003.

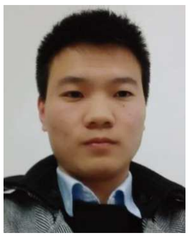
[4] K. Wu, H. Tan, Y. Liu, J. Zhang, Q. Zhang, and L. M. Ni, Side channel: Bits over interference, *IEEE Transactions on Mobile Computing,* vol. 11, no. 8, pp. 1317–1330, 2012.

[5] R. van Renesse, Y. Minsky, and M. Hayden, A gossip-style failure detection service, presented at the IFIP International Conference on Distributed Systems Platforms and Open Distributed, London, UK, 1998.
[6] R. van Renesse, Scalable and secure resource location, in *Proc. of 33rd Annual Hawaii International Conference on System Sciences*, 2000.
[7] K. Wu, H. Li, L. Wang, Y. Yi, Y. Liu, D. Chen, X. Luo, Q. Zhang, and Li. M. Ni, hJam: Attachment transmission in WLANs, *IEEE Transactions on Mobile Computing,* vol. 12, no. 12, pp. 2334–2345, 2013.
[8] H. Li, K. Wu, Q. Zhang, and L. M. Ni, CUTS: Improving channel utilization in both time and spatial domains in WLAN, *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 6, pp. 1413–1423, 2014.
[9] K. Wu, J. Xiao, Y. Yi, D. Chen, X. Luo, and L. M. Ni, CSI-based indoor localization, *IEEE Transactions on Parallel and Distributed Systems,* vol. 24, no. 7, pp. 1300–1309, 2013.
[10] B. H. Bloom, Space/time trade-offs in hash coding with allowable errors, *ACM Commun.,* vol. 13, no. 7, pp. 422–426, 1970
[11] D. Eppstein, M. T. Goodrich, F. Uyeda, and G. Varghese, What's the difference? Efficient set reconciliation without prior context, in *Proc. of ACM SIGCOMM*, Ontario, Canada, 2011, pp. 218–229.
[12] D. Guo, Y. He, and P. Yang, Receiver-oriented design of bloom filters for data-centric routing, *Elsevier Computer Networks Journal,* vol. 54, no. 1, pp. 165–174, 2010.
[13] D. Guo, J. Wu, H. Chen, Y. Yuan, and X. Luo, The dynamic bloom filters, *IEEE Trans. on Knowledge and Data Engineering,* vol. 22, no. 1, pp. 120–133, 2010.
[14] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, Theory and practice of bloom filters for distributed systems, *IEEE Communications Surveys and Tutorials,* vol. 14, no. 1, pp. 131–155, 2012.
[15] M. T. Goodrich and M. Mitzenmacher, Invertible bloom lookup tables: 1101.2245, 2011.
[16] L. Fan, P. Cao, J. Almeida, and A. Broder, Summary cache: A scalable wide-area web cache sharing protocol, *IEEE/ACM Transactions on Networking,* vol. 8, no. 3, pp. 281–293, 2000.
[17] D. Guo and M. Li, Set reconciliation via counting bloom filters, *IEEE Trans. on Knowledge and Data Engineering,* vol. 25, no. 10, pp. 2367–2380, 2013.
[18] D. Guo, Y. Liu, X. Li, and P. Yang, False negative problem of counting bloom filter, *IEEE Trans. on Knowledge and Data Engineering,* vol. 22, no. 5, pp. 651–664, 2010.
[19] A. Broder and M. Mitzenmacher, Network applications of bloom filters: A survey, *Internet Mathematics,* vol. 1, no. 4, pp. 485–509, 2004.
[20] D. Eppstein and M. Goodrich, Straggler identification in round-trip data streams via newton's identities and invertible bloom filter, *IEEE Trans. on Knowledge and Data Engineering,* vol. 23, no. 2, pp. 297–306, 2011.
[21] B. Bollobas, *Random Graphs*. New York, NY, USA: Academy Press, 1985.
[22] M. Dietzfelbinger, A. Goerdt, M. Mitzenmacher, A. Montanari, R. Pagh, and M. Rink, Tight thresholds for cuckoo hashing via XORSAT, in *Proc. of ICALP*, 2010, pp. 213–225.
[23] M. Molloy, The pure literal rule threshold and cores in random hypergraphs, in *Proc. of the 43rd Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004, pp. 672–681.
[24] K. Christensen, A. Roginsky, and M. Jimeno, A new analysis of the false positive rate of a bloom filter, *Inf. Processing Letters,* vol. 110, no. 21, pp. 944–949, 2010.
[25] 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, School of Comp. Sci., Carleton Univ., New York, USA, 2007.

**Zhiyao Hu** received the BEng degree from Beijing Institute of Technology, China, in 2015. He is currently a master student at National University of Defense Technology. His research interests include software-defined networking and data center networking.

**Deke Guo** received the BS 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, China, in 2008. He is an associate professor with the College of Information System and Management, National University of Defense Technology, China. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks.

**Zhong Liu** received the BS degree in physics from Central China Normal University, China, in 1990, and the PhD degree in management science and engineering from National University of Defense Technology, China, in 2000. He is a currently professor with the College of Information System and Management, National University of Defense Technology, China. His research interests include information systems, cloud computing, and big data.



**Xiaoqiang Teng** received the BS degree from Shenyang University of Technology, China, in 2013. Currently, he is a PhD student in the College of Information System and Management, National University of Defense Technology, China. His research interests include mobile computing, pervasive computing, and computer vision.

**Bangbang Ren** received the BS degree from National University of Defense Technology, China, in 2015. Currently, he is a master student in NUDT. His research interests include software-defined networking and data center networking.

**Pin Lv** received the BSc degree from Northeastern University, China, in 2006 and the PhD degree in computer science from National University of Defense Technology, China, in 2012. He is currently with the School of Computer, Electronics and Information, Guangxi University, Nanning, China. His research interests include wireless networks, network virtualization, cloud computing, etc.