Set Reconciliation via Counting Bloom Filters

Deke Guo, *Member*, *IEEE*, and Mo Li, *Member*, *IEEE*

Abstract—In this paper, we study the set reconciliation problem, in which each member of a node pair has a set of objects and seeks to deliver its unique objects to the other member. How could each node compute the set difference, however, is challenging in the set reconciliation problem. To address such an issue, we propose a lightweight but efficient method that only requires the pair of nodes to represent objects using a counting Bloom filter (CBF) of size $O(d)$ and exchange with each other, where d denotes the total size of the set differences. A receiving node then subtracts the received CBF from its local one via minus operation proposed in this paper. The resultant CBF can approximately represent the union of the set differences and thus the set difference to each node can be identified after querying the resultant CBF. In this paper, we propose a novel estimator through which each node can accurately estimate not only the value of d but also the size of the set difference to each node. Such an estimation result can be used to optimize the parameter setting of the CBF to achieve less false positives and false negatives. Comprehensive analysis and evaluation demonstrates that our method is more efficient than prior BF-based methods in terms of achieving the same accuracy with less communication cost. Moreover, our reconciliating method needs no prior context logs and it is very useful in networking and distributed applications.

Index Terms—Set reconciliation, set difference, Bloom filters

1 Introduction

Consider a pair of nodes A and B, each holding a set S_A and a set S_B. The goal of set reconciliation is for A and B to compute $S_A \cup S_B$ with the minimum communication cost. The set reconciliation is a fundamental task in a variety of systems where distributed information needs to be reconciliated. It was proposed in the literature [1] in attempt to improve gossip protocols and has an increasing number of applications outside of gossip protocols. For example, in peer-to-peer networks [2], [3], any two peers wish to receive only missing blocks from each other if they have already received a majority of blocks of a file from other peers. In wireless sensor networks [4], [5], each member of any sensor pair periodically updates the out-of-date objects at the other member using its new objects. The sink node thus can reliably deliver data to all the nodes, for example in remote reprogramming [6], [7]. In cloud applications [8], [9], a user needs to synchronize across its multiple data repositories, such as the phone, the desktop, and in the cloud. In mobile social networks [10], [11], two friends may wish to synchronize data when their portable devices are connected by opportunities.

A straightforward way of the set reconciliation between any two nodes involves wholesale exchanging their sets of objects. The amount of data transferred would be proportional to the total number of objects in the two nodes. Such a method is inefficient when there are few actual differences between the two nodes. This situation is common in many applications, for example, gossip protocols, particularly when information is introduced into the systems at a low rate [12]. An efficient and practical way, however, is to only exchange the objects within the set difference between the two nodes. Thus, the set reconciliation can be modeled as a problem of having the pair of nodes compute and exchange their set differences. The set difference to each node denotes all of its unique objects that are not within another one's set.

Several prior methods for computing the set differences between any two sets, S_A and S_B, have been proposed. A possible method is to use a logging system that records the node status at the time when the two nodes last communicated. If they communicate again, they only exchange the updates with each other. In such a setting, each node has to maintain one log for every other node that may wish to synchronize with itself. The number of such nodes, however, is nontrivial in many applications. Generally, the use of logs requires prior context, which we seek to avoid.

The intrinsic approach without prior context requires the two nodes to exchange the lists of identifiers for all of their objects. The set difference to each node can thus be found by scanning the received list. This requires $O(|S_A| + |S_B|)$ communication and $O(|S_A| \times |S_B|)$ search time. To reduce the overhead, each node can deliver a Bloom filter (BF) that compresses all identifiers of its objects to the other node. Each node then queries the received BF for identifying its set difference.

Although such an approach saves the communication cost by a constant factor, each BF still transfers data proportional to the size of the object sets. Moreover, such a method lacks a way to estimate the size of the set difference to each of S_A and S_B. Thus, it fails to optimize the configurations of each BF so as to bound the resultant false positives. Thus, some unique objects in one set may fail to be discovered and synchronized to the other set since they may be misidentified as common objects due to false positives. Approximate reconciliation of the set difference

- *D. Guo is with the Key laboratory of Information System and Engineering, College of Information System and Management, National University of Defense Technology, Changsha, P.R. China 410073. E-mail: guodeke@gmail.com.*
- *M. Li is with the School of Computer Engineering, Nanyang Technological University, Singapore. E-mail: limo@ntu.edu.sg.*

Manuscript received 19 Mar. 2012; revised 29 Aug. 2012; accepted 9 Oct. 2012; published online 23 Oct. 2012.
Recommended for acceptance by N. Mamoulis.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2012-03-0180.
Digital Object Identifier no. 10.1109/TKDE.2012.215.

may suffice for many applications, however, could not sustain nontrivial number of missed unique objects.

To efficiently solve the set reconciliation problem without the use of logs or other prior context, we propose a lightweight method. It computes the set difference with communication proportional not to the size of the sets but to the size of the set difference between any two sets. In this paper, we are particularly interested in optimizing the case when the set difference is small (e.g., the two nodes have a large amount of duplicate data blocks). Our approach will achieve particular performance gain in such circumstances at the cost of an approximate set reconciliation due to the impact of false positives and false negatives. The main contributions of this paper are summarized as follows:

1. Our first innovation is to utilize counting Bloom filter (CBF) and introduce a subtraction operator to compute the set difference with a round of communication of size $O(d)$, where d denotes the size of the total set difference. In such a setting, each node compresses all identifiers of its objects using a CBF of size $O(d)$ and exchanges it with the other node. Each node subtracts the received CBF from its local one via the minus operation. The set difference to each node can thus be discovered after querying the resultant CBF that provides an approximate representation of the total set difference.
2. The CBF-based reconciliating method requires each CBF to be sized appropriately so as to guarantee its accuracy. An important component of our method is a novel estimator through which each node can accurately estimate not only the size of the total set difference, d, but also the size of the set difference to each node. Being an important part of our reconciliating method, the difference estimator can also be generalized to optimize other BF-based reconciliating approaches.
3. We show that our set difference estimator is accurate for various sizes of the total set difference between two nodes. Comprehensive experiments show that our set reconciliation method is more efficient than prior BF-based methods in terms of achieving the same accuracy with less communication cost.

The remainder of this paper is organized as follows: We briefly describe the preliminaries of this work in Section 2. We present the design and analysis for our set reconciliating method based on counting Bloom filters in Section 3. We propose the set difference estimating method in Section 4. We comprehensively evaluate our approaches in Section 5 and conclude this work in Section 6.

2 Preliminaries

2.1 Problem Formulation

Given two sets S_A and S_B, the relative complement of S_A in S_B is $D_{A\backslash B} = S_A - S_B = \{s \in S_A \mid s \notin S_B\}$ and the relative complement of S_B in S_A is $D_{B\backslash A} = S_B - S_A = \{s \in S_B \mid s \notin S_A\}$. We assume that S_A and S_B are stored at two distinct nodes A and B. We then attempt to compute the set differences to S_A and S_B, denoted as $D_{B\backslash A}$ and $D_{A\backslash B}$, respectively, with the minimal communication overhead. The union of $D_{B\backslash A}$ and $D_{A\backslash B}$ forms the total set difference, denoted as D. Let $d_{A\backslash B}$ and $d_{B\backslash A}$ denote the cardinality of $D_{B\backslash A}$ and $D_{A\backslash B}$, respectively, while the cardinality of D is $d = d_{A\backslash B} + d_{B\backslash A}$. To reconciliate the two sets, the two nodes only need to exchange their unique objects in $D_{A\backslash B}$ and $D_{B\backslash A}$ with each other so as to let each node obtain $S_A \cup S_B$.

In such a setting, it is sufficient to complete the set reconciliation if the two nodes, A and B, have computed $D_{A\backslash B}$ and $D_{B\backslash A}$, respectively. In this paper, we focus on efficiently computing the set difference at each side of the node pair when the size of the total set difference, $D_{B\backslash A} \cup D_{A\backslash B}$, is small compared with S_A and S_B.

2.2 Bloom Filters

A standard Bloom filter is an effective data structure for representing a set to support approximate membership queries. A BF consists of an array of m cells, each of which is a bit with an initial value 0, and k *independent* random hash functions. We denote them by $A[i](1 \le i \le m)$ and $H = \{h_i(\cdot)\} \mid 1 \le i \le k, 1 \le h_i(\cdot) \le m$, respectively.

Given any set denoted by $S = \{s_1, s_2, \ldots, s_n\}$, every element s_j in S should be represented by the BF by setting all bits $A[h_i(s_j)]$ to one for $1 \le i \le k$ and $1 \le j \le n$. In this way, the membership information of s_j in S is encoded into the BF. After representing the total set S, we can answer the membership query "Is x a member of S" according to the resultant BF instead of the set. To answer the query, we check whether all bits at $A[h_i(x)]$ are set to one for $1 \le i \le k$. If not, we derive that x is not a member of S. Otherwise, x is considered as a member of S. A BF may yield a *false positive* due to hash collisions for which it wrongly reports that an item x belongs to the set S when it is actually not. The cause is that all bits at $A[h_i(x)]$ for $1 \le i \le k$ have been set to one by other items in the set S [13]. The probability that such a membership query gets a false positive response can be theoretically derived as follows [14], [15]:

$$f = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx (1 - e^{-k \times n/m})^k. \quad (1)$$

The deletion of any item $x \in S$ resets all bits at $A[h_i(x)]$ for $1 \le i \le k$ to 0. If other items in S also hash to one or more bits at $A[h_i(x)]$ for $1 \le i \le k$, the BF no longer reflects the data set correctly and it may respond one or more false negatives. To address such a problem, Fan et al. [16] propose the CBF in which each of the m cells is replaced by a count. The item inserting and deleting operations are extended to increasing and decreasing the value of each relevant count by one, respectively. To answer whether an item x is contained in a CBF, one needs to examine if all counts at $A[h_i(x)]$ for $1 \le i \le k$ are nonzero.

We define two operations of CBF, which will be used in the remainder of this paper. Such operations follow the assumption that two CBFs, $CBF(S_A)$ and $CBF(S_B)$, use the same configurations of hash functions and the number of cells.

Definition 1. *The minus of $CBF(S_A)$ and $CBF(S_B)$ is operated by setting the ith cell as $CBF(S_A)[i] - CBF(S_B)[i]$, for $1 \le i \le m$.*

Definition 2. *The union of $CBF(S_A)$ and $CBF(S_B)$ is to set the ith cell as $CBF(S_A)[i] + CBF(S_B)[i]|$, for $1 \leq i \leq m$. It is clear that $CBF(S_A) \cup CBF(S_B)$ is equivalent to $CBF(S_A \cup S_B)$.*

2.3 Set Reconciliation via BF

To reconciliate two sets between two arbitrary nodes, each node needs a method for computing the set difference to itself so as to exchange unique objects between the two nodes. The BF-based method is an efficient one among existing methods. In such a setting, each node delivers a BF that represents the identifiers of its all objects to the other node. Upon receiving a remote BF, each node can query with all identifiers of its objects to identify the set difference to itself.

Although the BF-based method can save the communication cost by a constant factor compared to directly exchanging the lists of identifiers, each BF still transfers data proportional to the size of the sets. That is, it still requires $O(|S_A| + |S_B|)$ communication. The time complexity for such a method is $O(|S_A| + |S_B|)$ since it involves $|S_A| + |S_B|$ BF queries and the time cost for each BF query is a constant value. The CBF-based reconciliating method proposed in this paper only requires $O(d)$ communication. The time complexity for our method is also $O(|S_A| + |S_B|)$.

A challenging issue that arises here is that a query based on BF may yield a false positive and thus every unique object to one node may be identified as a common one. Consequently, every unique object to one node may fail to be discovered and synchronized to the other node. This motivates the careful design of BF for each node such that the missed unique objects at both nodes are as few as possible. For example, we require that the total number of missed unique objects at both sides does not exceed a threshold, for example, at most one. Note that BFs at any two nodes usually adopt the same settings of hash functions and the number of cells so as to support the query and other operations of BF at remote nodes.

To do so, we first derive from (1) that every unique object to S_A may be misidentified as a common one with probability $(1 - e^{-k \times |S_B|/m})^k$ by the node A. Similarly, every unique object to S_B may be misidentified as a common one with probability $(1 - e^{-k \times |S_A|/m})^k$ by the node B. Recall that the number of unique objects to S_A and S_B are denoted by $d_{A \setminus B}$ and $d_{B \setminus A}$, respectively. Thus, we derive that the total number of misidentified unique objects at both sides of any node pair is given by

$$d_{A \setminus B} \times \left(1 - e^{-k \times |S_B|/m}\right)^k + d_{B \setminus A} \times \left(1 - e^{-k \times |S_A|/m}\right)^k. \quad (2)$$

Thus, given k, $|S_A|$, $|S_B|$, $|D_{A \setminus B}|$, and $|D_{B \setminus A}|$, both nodes A and B can estimate the minimum value of m if we impose a constraint that (2) is ≤ 1.

In practice, neither the node A nor the node B, is aware of $|D_{A \setminus B}|$ and $|D_{B \setminus A}|$ without prior communication between them. Thus, a method for estimating $|D_{A \setminus B}|$ and $|D_{B \setminus A}|$ is essential for implementing the set reconciliation between the two nodes.

2.4 Estimation of the Size of Set Difference

The size of set difference between two sets, S_A and S_B, can be estimated by comparing a random sample from each set [17]. The Min-wise sketches [18], [19], [20] was originally proposed for estimating the set similarity, defined as $r = \frac{|S_A \cap S_B|}{|S_A \cup S_B|}$, in a sampling way. It also can estimate the size of the total set difference as $d = \frac{1-r}{1+r}(|S_A| + |S_B|)$. The accuracy of the Min-wise method depends on the performance of random permutations and does not suffice for small set differences. Similarly, sketch-based methods for estimating large differences in traffic were proposed in [21] and [22].

Although such methods can estimate the size of the total set difference $D_{A \setminus B} \cup D_{B \setminus A}$, they cannot be directly used to estimate the size of its two parts, $D_{A \setminus B}$ and $D_{B \setminus A}$. As discussed later, such methods cannot support our set reconciliating method and prior BF-based methods. On the contrary, the novel estimating method proposed in this paper can accurately measure not only the size of $D_{A \setminus B} \cup D_{B \setminus A}$ but also that of $D_{A \setminus B}$ and $D_{B \setminus A}$.

Another related problem is the estimation of set cardinality. Flajolet and Martin (FM) [23] introduced hash sketches as a means of estimating the number of distinct items (not differences) using a bit vector. Each bit i in the vector is set to 1 if at least 1 element is sampled when sampling the set with probability $1/2^i$. Intuitively, if there are $2^5 = 32$ distinct values in the set, it is likely that bit 5 will be set when sampling with probability $1/32$. Thus the estimator returns 2^I as the set size, where I is the highest bit such that bit I is set to 1. Nikos et al. proposed distributed hash sketches for accurate cardinality estimation of distributed multisets [24], [25], [26]. Cormode et al. [27] estimated set sizes using a hierarchy of samples by providing a method for dynamic sampling from a data stream. Although such methods can estimate the cardinality for each of S_A and S_B, it is still not clear whether the size of $D_{A \setminus B} \cup D_{B \setminus A}$, $D_{A \setminus B}$, and $D_{B \setminus A}$ can be derived from such methods.

The most related research work revolves around the estimation method of set cardinality based on Bloom filters [28], [29]. For two nodes each hosting a set S_A and S_B, each node delivers a Bloom filter, which compresses all items in its set, to the other node. Moreover, the inner product of the two Bloom filters can be used to approximate the intersection between the two sets, S_A and S_B. Let Z_A and Z_B denote the number of zero bits in the Bloom filters for S_A and S_B, respectively. Let Z_{AB} be the number of zero bits in the inner product of the two Bloom filters. We obtain that

$$\frac{1}{m}\left(1 - \frac{1}{m}\right)^{-k|S_A \cap S_B|} \approx \frac{Z_A + Z_B - Z_{AB}}{Z_A Z_B}.$$

Accordingly, each node can calculate an estimation for $|S_A \cap S_B|$ and then derive $D_{A \setminus B}$ and $D_{B \setminus A}$ only given the two Bloom filters for the two sets. In the state-of-the-practice, the accuracy of such a method is still unknown, especially for a small set difference between S_A and S_B. Moreover, the settings of Bloom filters bring a significant impact on the estimating accuracy and thus need to be tackled well before utilizing such methods.

TABLE 1
Symbols and Notations

Term	Definition
BF	a standard Bloom filter
$BF(S_A)$	a BF that represents a set S_A
CBF	a Counting Bloom filter
$CBF(S_A)$	a CBF that represents a set S_A
$D_{A\setminus B}$	the set difference to S_A and is $\{s \in S_A \mid s \notin S_B\}$
$D_{B\setminus A}$	the set difference to S_B and is $\{s \in S_B \mid s \notin S_A\}$
$D_{A\setminus B} \cup D_{B\setminus A}$	the total set difference between two sets S_A and S_B
$d_{A\setminus B}$	the cardinality of $D_{B\setminus A}$ and is also denoted as d_1
$d_{B\setminus A}$	the cardinality of $D_{A\setminus B}$ and is also denoted as d_2
d	$d_{A\setminus B} + d_{B\setminus A}$
m	the number of cells in each BF or CBF
k	number of hash functions used by a BF or CBF
f	the false positive probability of a BF or CBF

The method for estimating the size of set difference proposed in this paper incurs a round of exchange of two CBFs between any two nodes. As we will show, the size of each CBF is proportional to the total size of set difference, $D_{A\setminus B} \cup D_{B\setminus A}$, between any two sets S_A and S_B. For the estimating methods based on Bloom filters [28], [29], the size of each Bloom filter is proportional to the total number of objects in S_A or S_B. In this paper, we desire to reconciliate any two large sets with a small set difference. In such a setting, our method for estimating the size of set difference incurs less communication overhead than the existing methods.

3 CBF-Based Set Reconciliating Method

We start with introducing the CBF and a subtraction operator to compute the set difference $D_{A\setminus B}$ or $D_{B\setminus A}$ using a round of communication of size $O(d)$. We then measure the accuracy of our method in terms of the number of resultant false negatives and false positives. Table 1 lists the symbols and notations used in the remainder of this paper.

3.1 Reconciliating Differences

In prior methods, nodes A and B generate two BFs, $BF(S_A)$ and $BF(S_B)$, for representing the identifiers of all objects in S_A and S_B by using $\alpha \times |S_A|$ and $\alpha \times |S_B|$ bits, respectively. Here, $\alpha = \log_f / \log_{0.6285}$ and f is the false-positive probability of $BF(S_B)$ or $BF(S_A)$. After exchanging the BFs, each node can query the remote BF using the identifier for each of its objects to answer whether a particular object is unique, with given false-positive probability.

Consider that each node allocates only a few cells, for example, only 100 cells, to its BF for representing the identifiers of a large set of objects, for example, 10,000 objects. If each object is hashed to only three cells ($k = 3$), an average of 300 identifiers hash into each cell. What can we do with such a small number of cells and such a large number of collisions? Actually, the resultant BF exhibits a false-positive probability of almost 100 percent and almost all unique objects to a node would be wrongly identified as the common objects. In this case, the previous method fails to compute the set difference to each node. Contrarily, we find that two CBFs each with $O(d)$ space can be utilized to calculate the set differences, $D_{A\setminus B}$ and $D_{B\setminus A}$, by the two nodes, respectively.

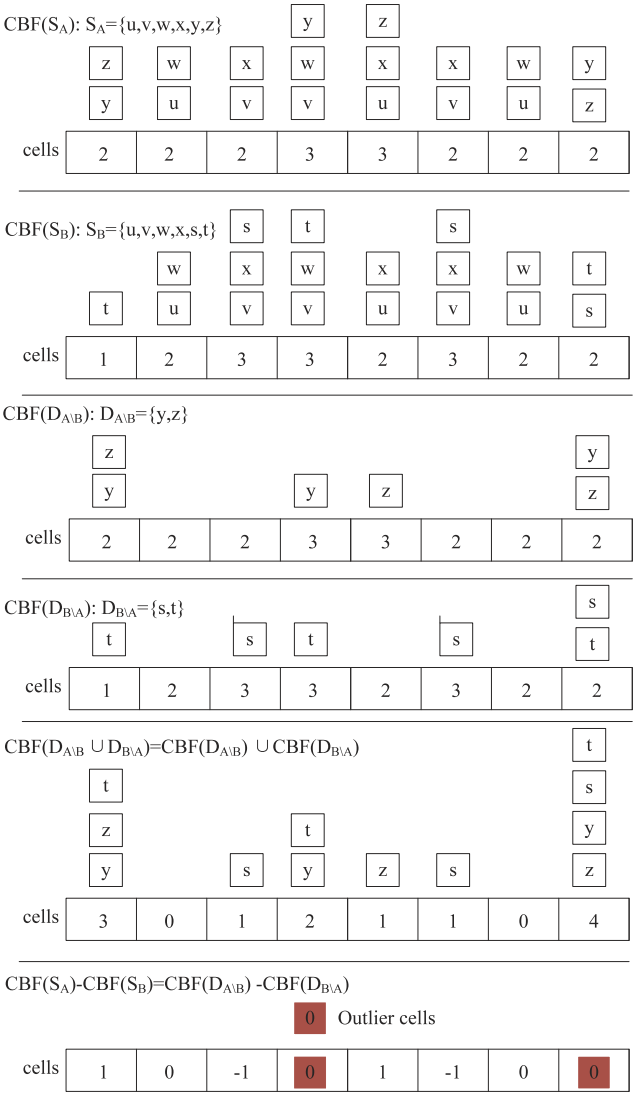


Fig. 1. An example of minus of two CBFs.

More precisely, each node represents the identifiers of all its objects with a CBF of m cells and exchanges its CBF with the other node. Given its local $CBF(S_A)$ and a remote $CBF(S_B)$, the node A proceeds to "subtract" the remote $CBF(S_B)$ from its $CBF(S_A)$. It repeats cell by cell, by subtracting the counts in the corresponding cells of the two CBFs, as shown in Fig. 1. Intuitively, the identifier, id, of any common object between S_A and S_B is hashed into the same cells in both $CBF(S_A)$ and $CBF(S_B)$ if both nodes use the same hash functions and number of cells. When we minus such cells of the two CBFs, the membership information about id will disappear, because each of these cells is incremented by the same value when representing id in the two CBFs. Therefore, the resultant $CBF(S_A) - CBF(S_B)$ at the node A only represents the unique objects to nodes A and B.

Fig. 1 gives an illustrative example of reconciliating two sets, $S_A = \{u, v, w, x, y, z\}$ and $S_B = \{u, v, w, x, s, t\}$. Each element of S_A and S_B is hashed into 3 cells in $CBF(S_A)$ and $CBF(S_B)$, respectively. Moreover, every common object between the two sets is hashed into the same corresponding cells in the two CBFs. Upon exchanging

the two CBFs, the node A achieves $CBF(S_A) - CBF(S_B)$ that just represents all unique objects in $D_{A\backslash B} \cup D_{A\backslash B} = \{y, z, s, t\}$ not those common objects. As shown in Fig. 1, $CBF(S_A) - CBF(S_B)$ provides an approximate representation for the total set difference that is exactly represented by $CBF(D_{A\backslash B} \cup D_{A\backslash B})$.

Similarly, the node B can generate $CBF(S_B) - CBF(S_A)$ according to its $CBF(S_B)$ and a remote $CBF(S_A)$. One can imagine that the ith cells in $CBF(S_B) - CBF(S_A)$ as well as $CBF(S_A) - CBF(S_B)$ are zero or a pair of opposite numbers for $1 \leq i \leq m$. Although $CBF(S_B) - CBF(S_A)$ is the opposite one of $CBF(S_A) - CBF(S_B)$, they appear to be the same when answering any CBF query, which is only sensitive to whether each related cell is nonzero. Therefore, we will not distinguish them explicitly in the remainder of this paper.

In this way, although the nodes A and B are aware of neither $D_{A\backslash B}$ nor $D_{B\backslash A}$, $CBF(S_A) - CBF(S_B)$ and $CBF(S_B) - CBF(S_A)$ can provide them with an approximation of $CBF(D_{A\backslash B} \cup D_{B\backslash A})$, respectively. So far, nodes A and B can query the resultant $CBF(S_A) - CBF(S_B)$ and $CBF(S_B) - CBF(S_A)$ with the identifier of each object in S_A and S_B, respectively. Thus, the time complexity for our method is $O(|S_A| + |S_B|)$ since it involves $|S_A| + |S_B|$ CBF queries and the time cost for each CBF query is a constant value. In this way, nodes A and B can discover all objects in $D_{A\backslash B}$ and $D_{B\backslash A}$ and only exchange discovered unique objects with each other so as to efficiently achieve the set reconciliation.

The above procedure for identifying $D_{A\backslash B}$ and $D_{B\backslash A}$ may yield false positives each of which wrongly identifies a common object as a unique object. As a result, a few false-positive objects may be exchanged between the two nodes. This will not hurt the accuracy of the set reconciliation and is not a significant problem if the number of such false-positive objects is very limited. Another possible result of the above procedure is that it may result in false negatives each of which wrongly identifies a unique object as a common object. Each node thus fails to identify and deliver some unique objects to the other node. As discussed in [20], if the set difference is large, the failure to send some unique objects to the other node is not a significant problem. Fortunately, for many applications we do not need exact reconciliation of the set difference, for example, reconciliation of encoded content. Approximations will suffice and allow us to determine a large portion of $D_{A\backslash B}$ and $D_{B\backslash A}$ with very little communication overhead.

Although the false-positive and false-negative problems of $CBF(S_A) - CBF(S_B)$ are inherent features of approximate reconciliation of the set differences, they can be controlled at a very low level if we seek a near-exact reconciliation.

3.2 Accuracy Analysis of False-Negative Problem

In this section, we first reveal the root cause of false negative in $CBF(S_A) - CBF(S_B)$ and we then measure its potential impact to the set reconciliation.

We derive from Definition 2 and the generating procedure of $CBF(S_A) - CBF(S_B)$ that

$$\begin{aligned} &CBF(S_A) - CBF(S_B) \\ &\quad = CBF(D_{A\backslash B} \cup (S_A \cap S_B)) - CBF(D_{B\backslash A} \cup (S_A \cap S_B)) \\ &\quad = CBF(D_{A\backslash B}) \cup CBF(S_A \cap S_B) \\ &\qquad - CBF(D_{B\backslash A}) \cup CBF(S_A \cap S_B) \\ &\quad = CBF(D_{A\backslash B}) - CBF(D_{B\backslash A}). \end{aligned}$$

A natural question is whether $CBF(D_{A\backslash B}) - CBF(D_{B\backslash A})$ is the same as $CBF(D_{A\backslash B} \cup D_{B\backslash A})$. In addition, it is clear that $CBF(D_{A\backslash B} \cup D_{B\backslash A}) = CBF(D_{A\backslash B}) \cup CBF(D_{B\backslash A})$. They only differ in the minus and union operations of CBF presented in Definitions 1 and 2. To measure their similarity, we analyze the different impacts of the two operations on the two components. We do this cell by cell and partition the values of any given cell in the two components as three cases as follows:

In the first case, the responses of the two different operations are the same zero at a cell if the corresponding cells in the two components are zero. The cells 2 and 7 of the two CBFs in Fig. 1 belong to such a case. In the second case, the two operations result in different nonzero outputs if the two components have different values at that cell. Fortunately, membership queries about objects in $D_{A\backslash B} \cup D_{B\backslash A}$ based on the $CBF(D_{A\backslash B}) - CBF(D_{B\backslash A})$ and $CBF(D_{A\backslash B}) \cup CBF(D_{B\backslash A})$ would not be influenced by such different nonzero values. The cells 1, 3, 5, and 6 of the two CBFs in Fig. 1 belong to such a case. In the third case, the minus operation results in zero while the union operation does not if the two components have the same nonzero values at that cell such as the cells 4 and 8 in Fig. 1. In this paper, such a cell in $CBF(D_{A\backslash B}) - CBF(D_{B\backslash A})$ is called an *outlier cell* compared to $CBF(D_{A\backslash B}) \cup CBF(D_{B\backslash A})$ that is equivalent to $CBF(D_{B\backslash A} \cup D_{A\backslash B})$.

In summary, only the third case among the above three ones makes $CBF(D_{A\backslash B}) - CBF(D_{B\backslash A})$ and $CBF(D_{A\backslash B}) \cup CBF(D_{B\backslash A})$ respond different results when being queried any object in the set difference $D_{A\backslash B} \cup D_{B\backslash A}$. More precisely, the former CBF responds a false negative due to its outlier cells, while the latter one answers correctly. Therefore, we need to know how many outlier cells would appear in $CBF(D_{A\backslash B}) - CBF(D_{B\backslash A})$. We derive the answer through Theorem 1.

Theorem 1. *Assume that $CBF(D_{A\backslash B})$ and $CBF(D_{B\backslash A})$ represent two disjoint sets of size d_1 and d_2, respectively, by using m cells and k hash functions. The expected number of outlier cells in $CBF(D_{A\backslash B}) - CBF(D_{B\backslash A})$ is given by*

$$m \times \sum_{j=1}^{\min\{d_1,d_2\}\times k} \binom{k \times d_1}{j} \binom{k \times d_2}{j} \frac{(1 - 1/m)^{k\times(d_1+d_2)}}{(m-1)^{2j}}. \quad (3)$$

Proof. The value of any cell in $CBF(D_{A\backslash B})$ is a discrete random variable, denoted by Y. It's possible values are the integers ranging from 0 to $k \times d_1$. Its probability mass function is

$$Pr(Y = j) = \binom{k \times d_1}{j} \frac{(1 - 1/m)^{k\times d_1 - j}}{m^j}.$$

Additionally, the value of any cell in $CBF(D_{B\backslash A})$ is also a discrete random variable, denoted by Z, whose

possible values are the integers ranging from 0 to $k \times d_2$. Its probability mass function is

$$Pr(Z = j) = \binom{k \times d_2}{j} \frac{(1 - 1/m)^{k \times d_2 - j}}{m^j}.$$

Thus, the probability that the ith cells for any $1 \leq i \leq m$ in both $CBF(D_{A\setminus B})$ and $CBF(D_{B\setminus A})$ hold the same nonzero value is given by

$$\sum_{j=1}^{\min\{d_1,d_2\}} Pr(Y = j) \times Pr(Z = j).$$

Consequently, the expected number of outlier cells in $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$ can be calculated by (3). □

It is clear that the value of (3) is dominated by four parameters, m, k, d_1, and d_2. We can derive that the two parts, $\min\{d_1, d_2\}$ and $\binom{k\times d_1}{j}\binom{k\times d_2}{j}$, in (3) could be maximized only when $d_{A\setminus B} = d_{B\setminus A} = d/2$, where d is the size of the total set difference between S_A and S_B. Thus, the upper bound on the expected number of outlier cells is given by

$$m \times \sum_{j=1}^{d\times k/2} \binom{k \times d/2}{j}^2 \frac{(1 - 1/m)^{k\times d}}{(m-1)^{2j}}. \quad (4)$$

The expected number of resultant false negatives after querying $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$ is also maximized when $d_{A\setminus B} = d_{B\setminus A} = d/2$, as discussed later.

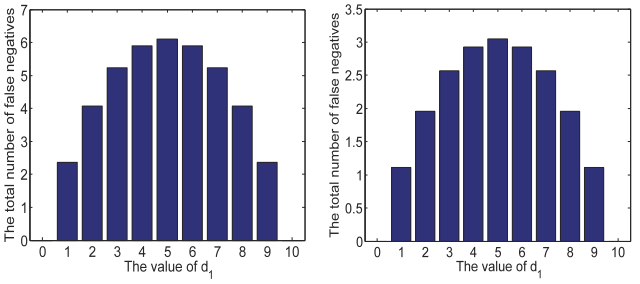
Moreover, one can find from (3) that there exist no outlier cells in $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$ when $d_{A\setminus B}$ or $d_{B\setminus A}$ is zero and the other one is d, i.e., one of S_A and S_B is a subset of the other one. A direct result of such a setting is that $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$ and $CBF(D_{A\setminus B}) \cup CBF(D_{B\setminus A})$ are the same. The number of false negatives in $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$ is thus minimized to 0. For a general case where $d_{A\setminus B}$ ranges from 0 to d, the number of outlier cells ranges from 0 to the upper bound given by (4) and the number of false negatives ranges from 0 to the upper bound, as we will show in (6).

After evaluating the outlier cells under two special and one general cases, we further measure the number of resultant false negatives in $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$, an essential metric for evaluating the accuracy of a set reconciliating method.

Theorem 2. *Assume that d_1 and d_2 denote the cardinality of two set differences, $D_{A\setminus B}$ and $D_{B\setminus A}$. The expected number of resultant false negatives by querying $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$ with all objects in $D_{A\setminus B} \cup D_{B\setminus A}$ as inputs is given by*

$$d_1 \times \left(1 - (1 - f_1)^k\right) + d_2 \times \left(1 - (1 - f_2)^k\right). \quad (5)$$

Proof. We first consider the number of false-negative objects in the set difference $D_{A\setminus B}$ as follows: The probability that any given cell in $CBF(D_{A\setminus B})$ is nonzero is $1 - (1 - 1/m)^{k\times d_1}$; hence, the number of nonzero cells is $m(1 - (1 - 1/m)^{k\times d_1})$. Additionally, the number of outlier cells in $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$ is given by (3). Given any outlier cell, for example, ith cell, the ith cells of both $CBF(D_{A\setminus B})$ and $CBF(D_{B\setminus A})$ have to be nonzero. Thus, any nonzero cell in $CBF(D_{A\setminus B})$ becomes an outlier cell in $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$ with probability



(a) $m = 60$ (b) $m = 200$

Fig. 2. The number of resultant false negatives varies along with the increase of d_1, where $d_1 + d_2 = 10$ and $k = 4$.

$$f_1 = \frac{m \times \sum_{j=1}^{\min\{d_1,d_2\}\times k} \binom{k\times d_1}{j}\binom{k\times d_2}{j} \frac{(1-1/m)^{k\times(d_1+d_2)}}{(m-1)^{2j}}}{m \times (1 - (1 - 1/m)^{k\times d_1})}.$$

An object in $D_{A\setminus B}$ would be identified as a false negative if at least one of its k nonzero cells in $CBF(D_{A\setminus B})$ becomes an outlier cell in $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$. Therefore, the probability that any object in $D_{A\setminus B}$ becomes a false negative is $1 - (1 - f_1)^k$; hence, there exist $d_1 \times (1 - (1 - f_1)^k)$ false negatives among d_1 unique objects in $D_{A\setminus B}$.

Similarly, any nonzero cell in $CBF(D_{B\setminus A})$ can become an outlier cell in $CBF(D_{A\setminus B}) - CBF(D_{B\setminus A})$ with probability

$$f_2 = \frac{m \times \sum_{j=1}^{\min\{d_1,d_2\}\times k} \binom{k\times d_1}{j}\binom{k\times d_2}{j} \frac{(1-1/m)^{k\times(d_1+d_2)}}{(m-1)^{2j}}}{m \times (1 - (1 - 1/m)^{k\times d_2})}.$$

Therefore, the probability that any object in $D_{B\setminus A}$ becomes a false negative is given by $1 - (1 - f_2)^k$; hence, there are $d_2 \times (1 - (1 - f_2)^k)$ false negatives among d_2 unique objects in $D_{B\setminus A}$. In summary, the total number of false negatives in $D_{A\setminus B}$ and $D_{B\setminus A}$ is given by (5). Thus, Theorem 2 holds. □

The total number of false negatives should be as small as possible so as to obtain a near-exact reconciliation. Therefore, if k, d_1, and d_2 are known in prior, the least number of cells required by $CBF(S_A)$ and $CBF(S_B)$ would be calculated by (5), given a threshold on the number of false negatives. If sufficient numbers of cells are allocated to $CBF(S_A)$ and $CBF(S_B)$, the number of resultant false negatives can be very small and even near-zero. We find that (5) is a symmetric function of d_1 and d_2 and is maximized when $d_1 = d_2 = d/2$, as shown in Fig. 2. Thus, the upper bound on the resultant false negatives mentioned in Theorem 2 is given by

$$d \times \left(1 - \left(1 - \frac{m \times \sum_{j=1}^{d\times k/2} \binom{k\times d/2}{j}^2 \frac{(1-1/m)^{k\times d}}{(m-1)^{2j}}}{m \times (1 - (1 - 1/m)^{k\times d/2})}\right)^k\right). \quad (6)$$

Additionally, we can see that the value of (5) can be minimized to zero when one of d_1 and d_2 is zero while the other is d. For a more general case where d_1 ranges from 0 to d but is not $d/2$, one has to use the upper bound given by (6) as an estimation of the real number of resultant false negatives since the values of d_1 and d_2 are not given. This will allocate some more but unnecessary cells to $CBF(S_A)$

and $CBF(S_B)$ and hurt the benefit of our method, especially when one of d_1 and d_2 is zero since no false negative occurs in practice. In addition, even the calculation of missed unique objects in prior BF-based methods requires prior knowledge, about both d_1 and d_2, which is still missing in the state-of-the-practice. Bearing these in mind, we design a novel method to accurately estimate not only $d_1 + d_2$ but also d_1 and d_2 in Section 4.

3.3 Accuracy Analysis of False-Positive Problem

Given any node pair, another issue of our set reconciliating method is that each node may wrongly identify a common object as a unique one with given probability when querying $CBF(S_A) - CBF(S_B)$. Therefore, another design metric of our method is the number of resultant false positives at each side of the node pair. Although such false positives do not influence the accuracy of the set reconciliation, they cause a few additional traffic cost. For this reason, such a metric should be minimized.

Consider that $CBF(S_A) - CBF(S_B)$ is equivalent to the minus of $CBF(D_{A\setminus B})$ to $CBF(D_{B\setminus A})$ and approximately represents the set $D_{A\setminus B} \cup D_{B\setminus A}$ using m cells and k hash functions. According to the construction process of $CBF(S_A) - CBF(S_B)$, we can know that its ith cell is zero if the corresponding cell in a CBF that exactly represents the set $D_{A\setminus B} \cup D_{B\setminus A}$ is zero for $1 \le i \le m$. Note that the probability that any cell in such a CBF is zero is given by $(1 - \frac{1}{m})^{kd}$. Additionally, $CBF(S_A) - CBF(S_B)$ may also contain a few outlier cells that are also zero. The probability that any cell in $CBF(S_A) - CBF(S_B)$ is an outlier cell has been presented in the proof of Theorem 1. Therefore, any given cell in $CBF(S_A) - CBF(S_B)$ is zero with probability

$$p_0 = \left(1 - \frac{1}{m}\right)^{kd} + \sum_{j=1}^{\min\{d_1,d_2\}k} \frac{\binom{k\times d_1}{j}\binom{k\times d_2}{j}\left(1-\frac{1}{m}\right)^{kd}}{(m-1)^{2j}}.$$

$CBF(S_A) - CBF(S_B)$ would yield a *false-positive* item x only if all cells at $A[h_i(x)]$ for $1 \le i \le k$ are nonzero. The probability that a membership query gets such a false-positive response is $(1 - p_0)^k$ and is given by

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kd} - \sum_{j=1}^{\min\{d_1,d_2\}k} \frac{\binom{k\times d_1}{j}\binom{k\times d_2}{j}\left(1-\frac{1}{m}\right)^{kd}}{(m-1)^{2j}}\right)^k. \quad (7)$$

It is clear that such a formula is dominated by parameters d_1, d_2, m, and k, where $d = d_1 + d_2$. As the value of d_1 varies from 0 to d, the false-positive probability of $CBF(S_A) - CBF(S_B)$ reaches an upper bound $f = (1 - (1 - \frac{1}{m})^{kd})^k$, if no outlier cells exists when $d_1 = 0$ or $d_1 = d$; it also exhibits a lower bound

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kd} - \sum_{j=1}^{d\times k/2} \frac{\binom{k\times d/2}{j}^2 (1-1/m)^{kd}}{(m-1)^{2j}}\right)^k, \quad (8)$$

if the number of outlier cells is maximized when $d_1 = d_2 = d/2$.

As discussed in Section 3, $CBF(S_A) - CBF(S_B)$ at the node A and $CBF(S_B) - CBF(S_A)$ at the node B appear to be the same when answering any CBF query. Thus, they would yield a false positive with the same probability. Since the two nodes have number of $|S_A| - d_1 = |S_B| - d_2$ common objects, they would expose $(|S_A| - d_1) \times (1 - p_0)^k$ false positives. If we limit the number of false positives at every side of the node pair, for example, no more than one in this paper, one can derive the minimum number of cells for $CBF(S_A)$ and $CBF(S_B)$ if d_1, d_2, k, $|S_A|$, and $|S_B|$ are given.

3.4 Discussions

Our set reconciliating method exhibits different features under three scenarios for an arbitrary set pair S_A and S_B.

First, one of the two sets is a subset of the other set. In this case, the unique objects to one is empty while that to the other is of size d, i.e., $d_{A\setminus B} = 0, d_{B\setminus A} = d$ or $d_{A\setminus B} = d, d_{B\setminus A} = 0$. $CBF(S_B) - CBF(S_A)$ thus provides an exact representation of the total set difference, $D_{A\setminus B} \cup D_{B\setminus A}$, although the two nodes are unaware of both $D_{A\setminus B}$ and $D_{B\setminus A}$. As a result, the number of outlier cells and that of false negatives are minimized to zero at each of the two nodes. In such a setting, our method would exhibit far better performance than prior methods, as we will show in Section 5.3.1.

Second, the two sets are nearly the same size but not the same one. In this case, the unique objects to S_A and that to S_B are about half of the total set difference between the two sets, i.e., $d_{A\setminus B} \approx d_{B\setminus A} = d/2$. Given $CBF(S_B) - CBF(S_A)$ at each of the two nodes, A and B, the number of outlier cells and that of false negatives are maximized. That is, the mismatch between $CBF(D_{A\setminus B} \cup D_{B\setminus A})$ and its approximation $CBF(S_B) - CBF(S_A)$ would be maximized in this scenario. Our method still outperforms prior BF-based methods in terms of many essential metrics, as we will show in Section 5.3.2.

Third, the number of unique objects to each of the two nodes ranges from 0 to d. In such a general scenario, the number of outlier cells and that of false negatives are covered by the above lower bound and upper bound, respectively. Thus, our method achieves a better performance in such a general scenario than in the second scenario but with a little loss of performance than that in the first scenario, as we will show in Section 5.3.3.

Consider that the total number of resultant false negatives in $CBF(S_A) - CBF(S_B)$ at both side of the node pair should be controlled to a low level, for example, at most only one in this paper. To meet such a constraint, $CBF(S_A)$ and $CBF(S_B)$ should be allocated only necessary cells whose size is $O(d)$ and can be derived according to (5) and (6) in the general and second cases, respectively. In the first case, such a constraint is obviously satisfied for any number of cells since no false negatives will happen in $CBF(S_A) - CBF(S_B)$.

Additionally, we know that the number of resultant false positives in $CBF(S_A) - CBF(S_B)$ at each side of the node pair should be as small as possible, for example, at most one in this paper. To meet such a constraint, $CBF(S_A)$ and $CBF(S_B)$ should be allocated only necessary cells whose size can be derived according to formulas proposed in Section 3.3.

In summary, it is not necessary to allocate large number of cells for $CBF(S_A)$ and $CBF(S_B)$ based on the sizes of S_A and S_B. The number of cells for $CBF(S_A)$ and $CBF(S_B)$ should be the larger one between the two values derived from the constraints of both false positives and false negatives. In general, the CBF-based reconciliating method requires $O(d)$ communication.

In this paper, we are particularly interested in optimizing the case when d is small. Our approach will achieve a particular performance gain in such circumstances. When d is relatively large, our CBF-based set reconciliation method will deteriorate to the existing BF-based method. The difference estimator approach proposed in this paper can provide an accurate estimation about not only d but also its two components, $d_{A\backslash B}$ and $d_{B\backslash A}$. In this way, applications can choose to use our method or the existing method according to the measured results. Note that both our method and existing method need to estimate the values of d and its two components. It, thus, does not bring additional overhead to our method.

4 Estimating Methods for the Size of the Set Difference

Recall that each member of any node pair, A and B, requires $O(d)$ cells to represent the identifiers of its objects so as to accurately compute the set differences, $D_{A\backslash B}$ and $D_{B\backslash A}$. More precisely, the calculations about the number of outlier cells, false positives, and false negatives in our method require that not only $d_{A\backslash B} + d_{B\backslash A}$ but also $d_{A\backslash B}$ and $d_{B\backslash A}$ are given in advance. Additionally, prior BF-based set reconciliating methods in [20] also has the same requirement when evaluating its false positives. In summary, to use both our set reconciliating method and prior ones effectively, we have to determine the approximate size of not only $D_{A\backslash B} \cup D_{B\backslash A}$ but also $D_{A\backslash B}$ and $D_{B\backslash A}$.

As aforementioned, the values of $d_{A\backslash B}$ and $d_{B\backslash A}$ fall into one of three cases. In this section, we will present two dedicated and one general methods for estimating $d_{A\backslash B}$, $d_{B\backslash A}$, and d. The common idea of such three methods is that the nodes A and B represent the identifiers of their objects with $CBF(S_A)$ and $CBF(S_B)$ and then exchange the two CBFs with each other at small communication overhead. A receiving node can subtract the remote CBF from its local CBF by invoking the minus operation in Definition 1 and thus $CBF(S_A) - CBF(S_B)$ could be established by the two nodes. Therefore, each node can account the number of zero cells, denoted as α, in $CBF(S_A) - CBF(S_B)$.

4.1 One of Two Arbitrary Sets Has No Unique Objects

In such a scenario, we have $d_{A\backslash B} = 0$ or $d_{B\backslash A} = 0$. Consequently, $CBF(S_A) - CBF(S_B)$ is an exact representation of the total set difference, $D_{A\backslash B} \cup D_{B\backslash A}$, although the two nodes are unaware of both $D_{A\backslash B}$ and $D_{B\backslash A}$. It is easy to derive that any cell in $CBF(S_A) - CBF(S_B)$ is zero with probability $(1 - 1/m)^{k\times d}$ and there exist number of $m \times (1 - 1/m)^{k\times d}$ zero cells in theory. If we use the theoretical number of zero cells to match the practical number of zero cells in $CBF(S_A) - CBF(S_B)$, the size of the total set difference could be simply estimated as $d = -(m/k) \times \ln(\alpha/m)$. This kind of estimation is called the first estimator. After achieving an estimation of d, one has to further answer which one of $d_{A\backslash B}$ and $d_{B\backslash A}$ is 0 and the other is d. This can be done by simply comparing the sizes of S_A and S_B since only the larger set has d unique objects in such a setting. If $|S_A|$ and $|S_B|$ are not given, the two nodes A and B can compare the number of nonzero cells in $CBF(S_A)$ and $CBF(S_B)$. Actually, a CBF that contains more nonzero cells should represent a larger set with high probability.

The first estimator can accurately estimate the size of the set difference when the number of cells in $CBF(S_A)$ and $CBF(S_B)$ is sufficient such that $CBF(S_A) - CBF(S_B)$ contains at least a small percent of zero cells. As we will show in Section 5.2.1, such a method works well even if the number of cells allocated to the two CBFs is only $2d$.

4.2 Unique Objects to a Set Pair Are the Same Size

As aforementioned, we have $d_{A\backslash B} = d_{B\backslash A} = d/2$ in this case. According to the construction process, we know that the zero cells inside $CBF(S_A) - CBF(S_B)$ consist of two parts.

The first part comes from the fact that there exist some cell pairs each of which are at the same locations in $CBF(D_{A\backslash B})$ and $CBF(D_{B\backslash A})$ and are zero. Thus, the minus of $CBF(S_A)$ to $CBF(S_B)$ results in zero at the related cells in $CBF(S_A) - CBF(S_B)$. We can derive from Theorem 1 that the total number of the first part zero cells is $m \times Pr(X = 0) \times Pr(Y = 0)$, i.e., $m \times (1 - 1/m)^{k\times d}$. If the theoretical number of such zero cells is used to match the practical number of zero cells in $CBF(S_A) - CBF(S_B)$, we actually adopt the aforementioned first estimator. Such a method may not accurately estimate the size of the total set difference due to the existence of the second part of zero cells.

The second part of zero cells involves all outlier cells, where the corresponding cells in $CBF(D_{A\backslash B})$ and $CBF(D_{B\backslash A})$ of each outlier cell have the same nonzero value. Thus, the minus of the two CBFs results in zero at the relevant cells in $CBF(S_A) - CBF(S_B)$. The number of all outlier cells can be calculated by (3) and is upper bounded by (4) that is just achieved in the case of $d_{A\backslash B} = d_{B\backslash A} = d/2$.

To enhance the accuracy of the first estimator, we further propose the second estimator that utilizes the total number of zero cells that belong to the two parts. In summary, the expected number of zero cells in $CBF(S_A) - CBF(S_B)$ should be

$$m \times \left(1 - \frac{1}{m}\right)^{k\times d} + m \times \sum_{j=1}^{d\times k/2} \frac{\binom{k\times d/2}{j}^2 (1 - 1/m)^{kd}}{(m-1)^{2j}} = \alpha. \quad (9)$$

After solving (9), we achieve an accurate estimation about the size of the total set difference even if each CBF only uses a small number of cells, for example, only 10 cells for estimating $d = 10$ different objects, as shown in Fig. 4.

4.3 General Case

If the distribution of $d_{A\backslash B}$ and $d_{B\backslash A}$ are known to be one of the above two cases in advance, we can choose a suitable method to estimate them accurately. If such a distribution is not one of the two cases or is not given, the first and second estimators would fail to estimate with high accuracy. In practice, a more general estimating method is essential to support an arbitrary distribution of $d_{A\backslash B}$ and $d_{B\backslash A}$. For this reason, we further focus on a general case where $d_{A\backslash B}$ and $d_{B\backslash A}$ may range from 0 to d and $d_{A\backslash B} + d_{B\backslash A} = d$.

In such a general case, the theoretical number of the first part of zero cells in $CBF(S_A) - CBF(S_B)$ is still given by $m(1 - 1/m)^{k\times d}$. On the other hand, the number of outlier cells can be calculated by (3). We adopt the second

estimator that utilizes the total number of all zero cells of the two parts to match the practical number of zero cells. That is, the expected number of zero cells in $CBF(S_A) - CBF(S_B)$ should be

$$m\left(1-\frac{1}{m}\right)^{kd} + m \sum_{j=1}^{\min\{d_1,d_2\}k} \frac{\binom{kd_1}{j}\binom{kd_2}{j}\left(1-\frac{1}{m}\right)^{kd}}{(m-1)^{2j}} = \alpha, \quad (10)$$

where d_1 and d_2 denote $d_{A\setminus B}$ and $d_{B\setminus A}$, respectively. We can see from the above equation that it generalizes the two equations in the aforementioned special cases.

After solving the above equation given m, k, and α, one can achieve accurate estimations about d, $d_{A\setminus B}$, and $d_{B\setminus A}$. The challenging issue that arises here is that the above equation is unsolvable since it has two variables d_1 and d_2. An intrinsic solution is to replace d_1 and d_2 with $d/2$ so as to estimate the value of d. This can generate an upper bound on d but cannot give any hint about $d_{A\setminus B}$ and $d_{B\setminus A}$. It is worthy noticing that those estimating methods using random sampling [30] or Min-hash also cannot distinguish $d_{A\setminus B}$ and $d_{B\setminus A}$ from d although they can estimate the value of d.

We present a general estimator that utilizes an inherent feature of the minus operation of CBF to address such a challenge. That is, the nonzero cells in $CBF(S_A) - CBF(S_B)$ may be positive or negative integers that can be used as hints for distinguishing $d_{A\setminus B}$ and $d_{B\setminus A}$ from d.

Given $CBF(S_A) - CBF(S_B)$, the node A can definitely infer that $d_{A\setminus B} = 0$ if no positive cells exist and $d_{B\setminus A} = 0$ if no negative cells exist. For other cases, we define the ratio of the number of positive cells to that of negative cells as r that can be accounted from $CBF(S_A) - CBF(S_B)$. Thus, it is easy to derive that $d_{B\setminus A} = d_2 = d/(1+r)$ and $d_{A\setminus B} = d_1 = d - d/(1+r)$. After substituting d_1 and d_2 into (10), the formula only contains a single variable d and becomes solvable. The node A can thus obtain $d_{A\setminus B}$ and $d_{B\setminus A}$.

Additionally, given $CBF(S_B) - CBF(S_A)$, the node B can definitely infer that $d_{B\setminus A} = 0$ if the number of positive cells is zero and $d_{A\setminus B} = 0$ if the number of negative cells is zero. For other cases, we have $d_{B\setminus A} = d - d/(1+r)$ and $d_{A\setminus B} = d/(1+r)$ in the similar way. The node B can further calculate the values of d, $d_{A\setminus B}$, and $d_{B\setminus A}$ after solving (10).

4.4 Discussion

In our CBF-based and prior BF-based reconciliating methods, each of the two nodes, A and B, should be aware of not only an approximate size of $D_{A\setminus B} \cup D_{B\setminus A}$ but also that of both $D_{A\setminus B}$ and $D_{B\setminus A}$. Only in this way each CBF or BF can be appropriately sized so as to guarantee the accuracy of the set reconciliation. Although some efforts have been made to estimate $D_{A\setminus B} \cup D_{B\setminus A}$, existing methods paid less attention to $D_{A\setminus B}$ and $D_{B\setminus A}$. The general estimator proposed in this paper can address such a fundamental issue with high accuracy in a single round at the cost of exchanging two small CBFs.

Such an estimator contributes to optimize the performance of not only our approach but also prior BF-based set reconciliating methods. At the same time, the reconciliating methods involve additional communication cost due to the estimating process for the difference size. This part of communication cost is small since $6d$ cells are sufficient for each exchanged CBF, as we will show in Section 5.2.3. Additionally, this paper desires to reconciliate two large sets with a small set difference, for example, no more than 300 different elements. Thus, a CBF of 1,800 cells at each node is sufficient to support our general estimator. If the largest value of d is underestimated such that $d > 300$, such a setting of each CBF is still sufficient if there exist a fraction of zero cells in the resultant $CBF(S_A) - CBF(S_B)$.

5 Evaluation

This section will provide the performance details of our set reconciliating and difference estimating methods. We address two questions. First, what is the accuracy of our estimating method for the size of the set difference between any two sets? Second, how do our method compare with prior BF-based set reconciliating methods.

5.1 Experimental Methodologies

We borrow the CBF implementation from one of our prior work [13] and extend it from two aspects to support our set reconciliating and difference estimating methods. First, we implement the CBF and let it support the minus operation introduced in this paper. Second, the CBF accounts the number of positive integers and that of negative integers. One critical issue for CBF is to generate a group of k independent random hash functions. We address such an issue by adopting the method used in [13] and [30]

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

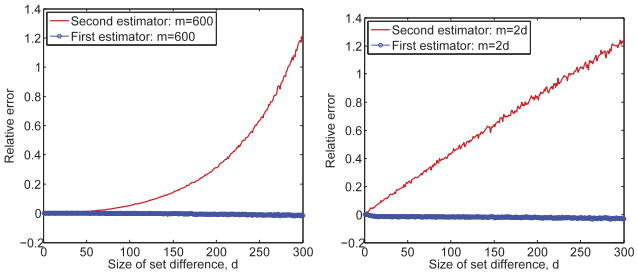
where $g_1(x)$ and $g_2(x)$ are two random and independent integers in the universe with a range $\{1, 2, \ldots, m\}$. The value of i ranges from 0 to $k-1$. We use the MersenneTwister, a random number generator, to generate the two random integers $g_1(x)$ and $g_2(x)$ for any item x. The seed of MersenneTwister comes from the output of the BUZ hash function.

The quality of the used hash function and random number generator has significant impacts on the performance of CBF. We select the BUZ hash function since it can quickly produce a near-perfect result even with an extremely skewed input data. As for the Mersenne twister, it results in a fast generation of very high quality pseudorandom numbers.

We have two degrees of freedom in creating a pair of sets, S_A and S_B, each at one node: the number of objects in S_A and S_B and the size of the total set difference between the two sets. For each setting of the set difference size, we report the average results among 200 rounds of experiments. In our experiments, each CBF utilizes only three hash functions since such a setting is sufficient to make our method work well.

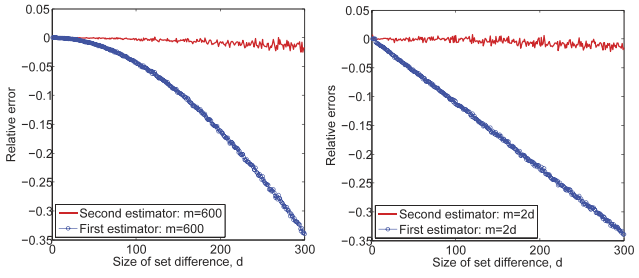
5.2 Accuracy of Estimating the Size of the Set Difference

Given two sets S_A and S_B, we first evaluate the accuracy of our two dedicated estimators that provide an estimation for d under two special cases. We then show that our general estimator can accurately estimate not only d but also $d_{A\setminus B}$ and $d_{B\setminus A}$ under more general settings. The absolute error is defined as that the estimated value of d minus its practical



(a) m=600. (b) m=2×d.

Fig. 3. The relative error about $d = d_1 + d_2$, where $d_1 = 0$, $k = 3$ and the set is of 6,000 size at each node.



(a) $m = 600$. (b) $m = 2\times d$.

Fig. 5. The relative errors of the two estimators, where $d_1 = d_2 = d/2$, $k = 3$, and each node has a set of 6,000.

value. The relative error denotes the ratio of the absolute error to the practical value of d. We use the relative error as a metric to evaluate the accuracy of our estimating methods.

5.2.1 One of Two Arbitrary Sets Has No Unique Objects

In this case, $CBF(S_A) - CBF(S_B)$ provides an exact representation of the total set difference, $D_{A\setminus B} \cup D_{B\setminus A}$, since one of $d_{A\setminus B}$ and $d_{B\setminus A}$ is zero in such a scenario. Fig. 3 shows the relative errors for various sizes of the total set difference under our two estimating methods.

We can see that the first estimator can generate an accurate estimate for d with very low relative errors that range from -3% to 0 when d increases from 1 to 300. As for the second estimator, its estimate errors are considerable large compared to the first estimator as expected, especially for large values of d. Thus, the second estimator is not suitable to such a scenario since it considers the impacts of both the first and second parts of zero cells, however, only the first part of zero cells exist in such a scenario. As a result, the second estimator only generates an upper bound on d while the first estimator can accurately estimate d.

We further adopt two representative settings of the number of cells m, allocated to $CBF(S_A)$ and $CBF(S_B)$, to evaluate the impacts on the accuracy of our estimators. We first fix m as two times of the largest one among all possible values of d and display our results in Fig. 3a. We then modify m such that it is always two times of d and display our results in Fig. 3b. We can see that the first estimator always exhibits a low relative error under the two settings, even very few cells are used by each of the two CBFs, for example, $m = 2d$.

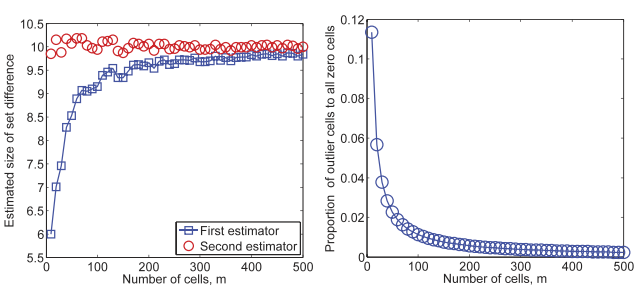
5.2.2 Unique Objects to a Set Pair Are the Same Size

We have $d_{A\setminus B} = d_{B\setminus A} = d/2$ in this case. In Fig. 4, we report the estimated values of d when the two estimators utilize various sizes of CBFs each of which represents a set of 6,000 elements, where $d = 10$ and $k = 3$. We see that the second estimator always generates a better estimate even if a CBF uses a small number of cells, for example, $m = d = 10$ cells. The first estimator, however, causes underestimations before the number of cells reaches a threshold. The root cause is that it uses the practical number of all zero cells to match the theoretical number of the first part of zero cells in $CBF(S_A) - CBF(S_B)$. This works well only when the proportion of outlier cells, the second part of zero cells, to all zero cells is near to zero, as shown in Fig. 4b. On the contrary, the second estimator always generates a good estimate since it uses the number of observed zero cells to match the number of both parts of zero cells in theory.

We next evaluate the relative errors of our two dedicated estimators under two representative settings of m, when d varies from 1 to 300. In the first case, the number of cells allocated to each CBF is fixed to 600, which is two times of the largest value of d. In the second case, the number of cells allocated to each CBF is set to two times of d. As shown in Fig. 5, the second estimator predicts d with high accuracy and outperforms the first estimator in both cases.

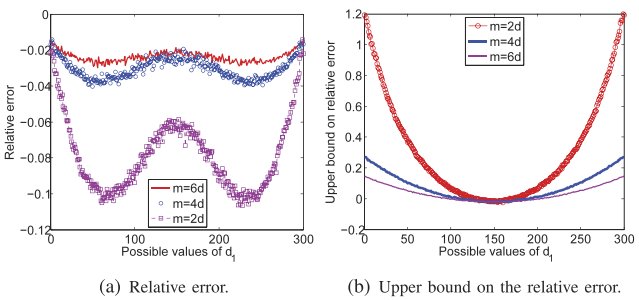
5.2.3 General Settings

The above evaluation results demonstrate that the two dedicated estimators exhibit high accuracy if the distribution of $d_{A\setminus B}$ and $d_{B\setminus A}$ falls into the desired working circumstances. At the same time, Figs. 3 and 5 demonstrate that the two estimators are no longer practical in undesired circumstances. To support the set reconciliation in more



(a) The estimated size of set difference. (b) Ratio of the number of outlier cells.

Fig. 4. The estimated size of the set difference between two sets each with 6,000 elements, where $k = 3$, $d = 10$, and $d_1 = d_2 = 5$.



(a) Relative error. (b) Upper bound on the relative error.

Fig. 6. The relative error and its upper bound, where $k = 3$, $d = d_1 + d_2 = 300$, and S_A and S_B have 6,000 common objects but only d_1 or d_2 unique ones, respectively.

general settings, we evaluate the accuracy of our general estimator. In such a setting, $d_{A\backslash B}$ and $d_{B\backslash A}$ can be arbitrary values ranging from 0 to d so long as $d_{A\backslash B} + d_{B\backslash A} = d$.

As shown in Fig. 6a, the three curves of our general estimator, under different settings of m, follow the similar trend as $d_1 = d_{A\backslash B}$ increases from 0 to 300. The general estimator achieves the highest accuracy when $d_1 = 0$ or $d_1 = d$ and suffers the decrease of its accuracy at a certain extent for other cases. Such a result indicates the difficulty of estimating the size of the set difference in more general cases. Fortunately, its accuracy can be enhanced to 12 percent for $m = 2d$, 4 percent for $m = 4d$, and 3 percent for $m = 6d$ underestimates. The resultant accuracy for $m = 6d$ is usually enough for many applications since the estimating result is only required to provide an approximate guidance to the upcoming set reconciliating method. Its accuracy can be further improved if more than $6d$ cells are allocated, for example, its accuracy is very close to 100 percent if $m = 8d$.

Fig. 6b further shows that the second estimator is not a good choice for more general settings of $d_{A\backslash B}$ and $d_{B\backslash a}$ and only generates an upper bound on the estimation of d derived from the general estimator.

5.3 Our Reconciliating Method versus Prior Work

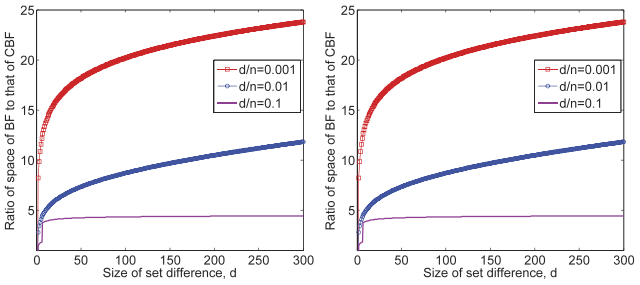
Given two arbitrary sets, S_A and S_B, the basic idea of our set reconciliating method is to approximate $CBF(D_{A\backslash B} \cup D_{B\backslash A})$ with $CBF(S_A) - CBF(S_B)$ at nodes A and B. To evaluate such an approximation, we compare our CBF-based and prior BF-based set reconciliating methods in terms of the communication cost.

5.3.1 One of Two Arbitrary Sets Has No Unique Objects

In many applications, one of any two nodes, A and B, may have no unique objects such that all objects in $D_{B\backslash A} \cup D_{A\backslash B}$ appear at one node, i.e., one of S_A and S_B is a subset of the other. In such a setting, $CBF(S_A) - CBF(S_B)$ does not contain any outlier cell and thus is an exact representation of $D_{A\backslash B} \cup D_{B\backslash A}$. Without loss of generality, we assume that S_B is a subset of S_A. Thus, $d_{B\backslash A} = 0$, $d_{A\backslash B} = d$, and the number of common objects between the sets S_A and S_B is $|S_B| = n$.

Since no false negatives exist in $CBF(S_A) - CBF(S_B)$ and $CBF(S_B) - CBF(S_A)$, the only constraint that guides the designs of $CBF(S_A)$ and $CBF(S_B)$ is the false positives. Given $CBF(S_B) - CBF(S_A)$ and $CBF(S_A) - CBF(S_B)$, nodes B and A may misidentify a common object between S_A and S_B as one of the unique objects, respectively. The probability of such an event is given by $(1 - (1 - \frac{1}{m})^{kd})^k$. The number of resultant false positives at each node is thus given by $n \times (1 - (1 - \frac{1}{m})^{kd})^k$ due to $|S_B| = n$ common objects. Since this paper requires each node to exhibit at most one false positive, the minimum number of cells for $CBF(S_A)$ and $CBF(S_B)$ is given by $m = \frac{-k \times d}{\log 1 - n^{-1/k}}$. To derive the minimum size of the two CBFs, we still need to know the minimal size of each cell such that it will not overflow.

Consider that the process of throwing n balls into m bins uniformly at random. At the end of the process, the maximum load denotes the maximum number of balls in a bin. It is well known that such a process results in a maximum load of $\Theta(\log m / \log \log m)$ when $m = n$ [31], [32]. It was also shown that for $n \geq m \log m$, such a process results in a maximum load of $n/m + \Theta(\sqrt{n \log m / m})$ with high probability [34].



(a) d ranges from 1 to 300. (b) d/n ranges from 0.0001 to 0.01.

Fig. 7. The ratio of space of BF to that of CBF, where $k = 3$, $d_{B\backslash A} = 0$, and $d_{A\backslash B} = d$, and n is the number of common objects between S_A and S_B.

The construction of $CBF(S_B)$ is equivalent to the process of throwing $k|S_B|$ balls into m bins uniformly at random. Consider that $k|S_B| \geq m \log m$ is true since our method seeks to reconciliate two large sets, which have a small set difference, using two CBFs each with a small number of cells. This results in a maximum load of $k|S_B|/m + \Theta(\sqrt{k|S_B| \log m/m})$. Fig. 8c plots the experimental and theoretical maximum loads, where the theoretical maximum load is given by $k|S_B|/m + 1.5\sqrt{k|S_B| \log m/m}$. We find that the two curves match well along with the increase of d. Thus, each cell should be at least

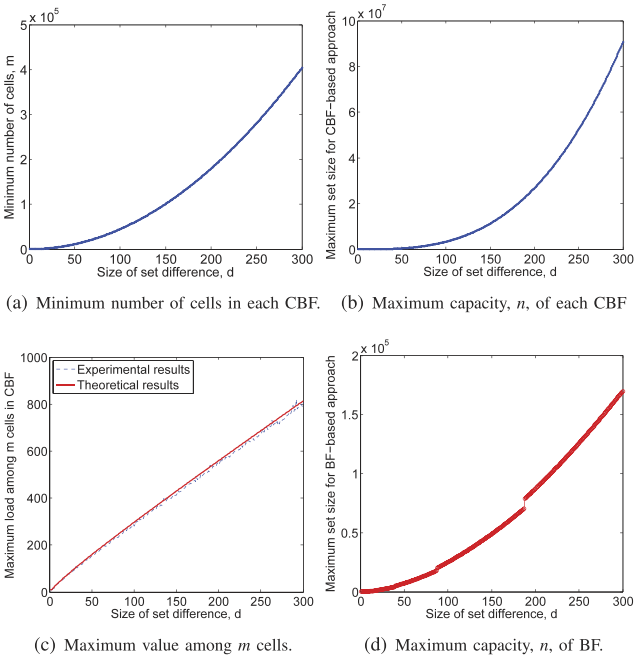
$$\lceil \log_2(k|S_B|/m + 1.5\sqrt{k|S_B| \log m/m} \rceil, \tag{11}$$

bits and thus the total number of bits used by $CBF(S_B)$ or $CBF(S_A)$ should be

$$\lceil \log_2(k|S_B|/m + 1.5\sqrt{k|S_B| \log m/m} \rceil \times \frac{-kd}{\log(1 - n^{-1/k})}.$$

As discussed in Section 2.3, prior BF-based reconciliating methods may misidentify a unique object in the total set difference as a common object. According to (2), the total number of misidentified unique objects at both sides of the node pair is $d \times (1 - e^{-k \times n/m})^k$ since $d_{A\backslash B} = d$, $d_{B\backslash A} = 0$, and $|S_B| = n$ and should not exceed one. Therefore, the minimum number of bits allocated to a BF at each node should be $-kn/\log(1 - d^{-1/k})$.

So far, we can compare our CBF-based method with prior BF-based methods in terms of the communication overhead, which is determined by the total space of two CBFs or two BFs the two methods adopted, respectively. Fig. 7 reports the ratio of the space of two BFs to that of two CBFs for various sizes of d and d/n. We can see that our method significantly outperforms prior methods since two BFs consumes more space than two CBFs, irrespective of the values of d and d/n. Additionally, our method can obtain more gains along with the increase of d or the decrease of d/n. In summary, given any two sets S_A and S_B one of which has no unique objects, the benefit of our reconciliating method is very prominent when d/n is relative low or d is relative large.



(a) Minimum number of cells in each CBF. (b) Maximum capacity, n, of each CBF

(c) Maximum value among m cells. (d) Maximum capacity, n, of BF.

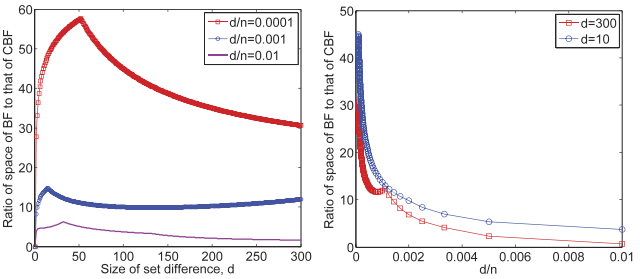
Fig. 8. Settings of m and n for our CBF-based and prior BF-based set reconciliating methods, where $k = 3$.

5.3.2 Unique Objects to a Node Pair Are the Same Size

In such a scenario, any two sets S_A and S_B are the same size of $n + d/2$, where $d_{A\backslash B} = d_{B\backslash A} = d/2$ and n denotes the number of common objects between the two sets. In our experiments, for any instance of d that ranges from 1 to 300, we focus on three representative settings of n, including $d/n = 0.01$, 0.001, and 0.0001. Recall that our CBF-based and prior BF-based reconciliating methods may misidentify a unique object in $D_{A\backslash B} \cup D_{B\backslash A}$ as a common object between the two sets.

We start with measuring the minimum number of cells required by each CBF such that our reconciliating method incurs at most one such misidentification. In this case, the upper bound on the total number of false negatives at both sides of the node pair, defined by (6), is one. To evaluate such an issue, we vary the value of d ranging from 1 to 300 and calculate the corresponding minimum number of cells in each CBF. Fig. 8a indicates that a CBF at each node requires more cells along with the increase of d. This verifies that the number of cells in $CBF(S_A)$ or $CBF(S_B)$ should scale with the size of the total set difference and not the set size of S_A or S_B. Thus, given the number of required cells under a fixed d, one may intuitively think that S_A and S_B can be arbitrary large. This is not true since the two CBFs would be constrained by the false positives, each of which means that a common object is misidentified as a unique object.

For this reason, the number of false positives at each side of the node pair is another design metric. Although such false positives do not influence the accuracy of the set reconciliation, they would cause a few unnecessary traffic cost and hence should be minimized, for example, only one in this paper. Recall that the false-positive probability of $CBF(S_A) - CBF(S_B)$ is given by (8) and there are n common objects between the two sets. Thus, given m, k, and d, we can derive the maximum cardinality of S_A and S_B



(a) d ranges from 1 to 300. (b) d/n ranges from 0.0001 to 0.01.

Fig. 9. The ratio of space of BF to that of CBF, where $k = 3$, $d_{B\backslash A} = 0$, and $d_{A\backslash B} = d$.

where $|S_A| = |S_B| = n + d/2$. Fig. 8b shows that the maximum size of the set represented by each CBF increases along with the increase of d.

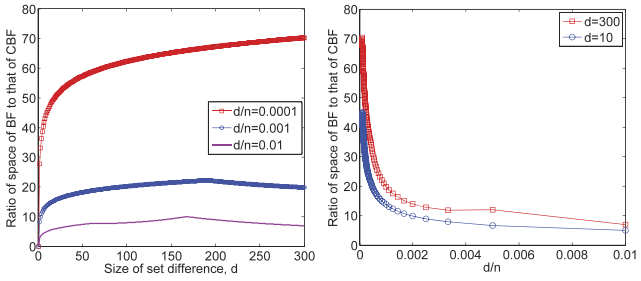
Although the number of cells in $CBF(S_A)$ and $CBF(S_B)$ scale with d, the total size of each CBF is also influenced by the size of each cell. The size of each cell is determined by the maximum load among all cells after an object set of size $n + d/2$ is represented by a CBF with m cells. So far, we are aware of the values of m and n under any given value of d. Fig. 8c plots the maximum load in $CBF(S_A)$ and $CBF(S_B)$ with respect to the change of d, where theoretical maximum load is

$$\lceil \log_2(k(n + d/2)/m + 1.5\sqrt{k(n + d/2)\log m/m} \rceil. \quad (12)$$

We can see that 10 bits per cell is always sufficient when d ranges from 1 to 300. Thus, the total size of each CBF under any given value of d could be calculated.

For prior BF-based methods, we also need to study the maximum sizes of S_A and S_B such that the total number of misidentified unique objects does not exceed one if each BF and CBF occupies the same size of space. In this case, a unique object may be misidentified as a common one with probability $(1 - e^{-k\times(n+d/2)/(10m)})^k$, where m is the number of cells in each CBF and each cell consists of 10 bits. Thus, the total number of false positives at both sides of the node pair is $d(1 - e^{-k\times(n+d/2)/(10m)})^k$ and should not exceed one. After solving such an inequation, we achieve the maximum sizes of S_A and S_B under various values of d if using prior BF-based methods, as shown in Fig. 8d. Our CBF-based reconciliating method obtains an increase of two orders of magnitude in terms of the maximum set size compared to prior BF-based methods when d ranges from 1 to 300. As shown in Fig. 9, each BF utilizes more space compared to each CBF if prior BF-based methods want to support the same set size as our CBF-based method, irrespective of the value of d.

Fig. 9 reports the ratio of the space of a BF to that of a CBF for various sizes of d and d/n. The benefit of our CBF-based method compared to prior BF-based methods increases along with the decrease of d/n from 0.01 to 0.0001. When $d/n = 0.001$, our method considerably outperforms prior BF-based methods since a BF consumes at least 10 times space than a CBF, irrespective of the value of d. When $d/n = 0.0001$, a BF consumes at least 30 times space than a CBF, irrespective of the value of d. If d/n is further reduced, our method would achieve more benefit.



(a) d ranges from 1 to 300. (b) d/n ranges from 0.0001 to 0.01.

Fig. 10. The ratio of space of BF to that of CBF, where $k = 3$. Given any d in the range of 1 to 300, we vary $d_1 = d_{A\backslash B}$ from 0 to d and $d_2 = d_{B\backslash A} = d - d_1$.

Additionally, our method can achieve more benefit as the decrease of d when d/n is fixed, as shown in Fig. 9b. In summary, our reconciliating method achieves far better performance compared to prior BF-based methods when d/n is relative small.

5.3.3 General Settings

Given any two sets S_A and S_B, the number of unique objects to each of them may be any integer in the range of 0 to d in general scenarios. Therefore, we should compare our method with prior methods in terms of required space of a BF and a CBF, so as to generalize the application fields of our method.

The sizes of minimum space required by the BF and CBF can be calculated as follows: As aforementioned in Section 3.4, the number of cells required by the CBF should be the larger one between two values derived from the following constraints. The first is that at most one unique object to S_A or S_B may be misidentified as a common one between the two sets. The second is that at most one common object may be reported as a unique one at each side of the node pair.

To evaluate such an issue, given any value of d in a range of 1 to 300, we vary $d_1 = d_{A\backslash B}$ from 0 to d and $d_2 = d_{B\backslash A} = d - d_1$. For any given d, we focus on three representative settings of n, including $d/n = 0.01$, 0.001, and 0.0001. Thus, given d, d/n, and k, we have d different settings of d_1 and d_2. For each of such settings, the minimum number of cells allocated to the CBF under the above two constraints can be derived from (5) and (7), respectively. In this way one can achieve the final number of cells allocated to the CBF in such a setting. For any pair of d and d/n, the number of cells for the CBF is the average value of d outputs each of which comes from one setting of d_1. Furthermore, the size of each cell can be calculated using the same way mentioned in Section 5.3.1 and thus the minimum space allocated to the CBF can be derived.

For the BF-based methods, the number of cells allocated to the BF has to guarantee that at most one unique object to S_A or S_B may be misidentified as a common one between the two sets. Similarly, given any pair of d and d/n, we have d different settings of d_1 and d_2. For each of such settings, the number of cells for the BF can be derived from (2). Consequently, for any pair of d and d/n, the minimum number of cells for the BF is the average value among d results. Note that the minimum space for the BF is just the number of cells since each cell in the BF is only one bit.

Fig. 10 plots the ratio of the minimum space for the BF to that for the CBF under various settings of d and d/n. We see that our reconciliating method always outperforms prior BF-based method since the latter consumes more space than the former as d increases from 1 to 300 and d/n decreases from 0.01 to 0.0001. The performance of our method would be more prominent if d/n is further reduced. Additionally, the performance of our method in such a general scenario is bounded by the upper bound in the first scenario and the lower bound in the second scenario. Such experimental results confirm our theoretical analysis, as discussed in Section 3.4.

6 Conclusion

Although many solutions using logs have been proposed recently to the set reconciliation problem, they require prior context and suffer nontrivial update overhead and scale poorly when many nodes need to reconciliate with each other. In this paper, we propose an efficient method that discovers the objects belonging to the set difference in a single round and makes any two nodes exchange the unique objects solely from the set difference. Such a method only requires each node to exchange its CBF of size $O(d)$, which represents the identifiers of all its objects. An essential component in our method is a novel estimator that can accurately estimate not only the value of d but also the size of the set difference to each node. Such an estimator not only contributes to optimize the performance of our CBF-based approach, but also contributes to prior BF-based reconciliating methods. Comprehensive experiments demonstrate that our method is more efficient than prior BF-based methods in terms of achieving the same accuracy with less communication cost.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments. This work was partially supported by the National Basic Researcg Program (973 program) under grant no. 2014CB347800, the NSFC under grants nos. 61170284, 60903206, and 61272483, and the Nanyang Technological University NAP grant M4080738.020.

References

[1] J.W. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed Content Delivery across Adaptive Overlay Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 5, pp. 767-780, Oct. 2004.

[2] Y. Liu, "A Two-Hop Solution to Solving Topology Mismatch," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1591-1600, Nov. 2008.

[3] X. Cheng and J. Liu, "Nettube: Exploring Social Networks for Peer-to-Peer Short Video Sharing," *Proc. IEEE INFOCOM*, pp. 1152-1160, 2009.

[4] Y. Zhu and L.M. Ni, "Probabilistic Approach to Provisioning Guaranteed QoS for Distributed Event Detection," *Proc. IEEE INFOCOM*, pp. 592-600, Apr. 2008.

[5] Y. Liu, L.M. Ni, and C. Hu, "Generalized Probabilistic Topology Control in Wireless Sensor Networks," *ACM/IEEE J. Selected Area in Comm.*, vol. 30, no. 9, pp. 1780-1788, 2012.

[6] K. Lin and P. Levis, "Data Discovery and Dissemination with Dip," *Proc. Seventh Int'l Conf. Information Processing in Sensor Networks*, pp. 433-444, 2008.
[7] W. Dong, C. Chen, X. Liu, J. Bu, and Y. Gao, "A Lightweight and Density-Aware Reprogramming Protocol for Wireless Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 10, no. 10, pp. 1403-1415, Oct. 2011.
[8] A.J. Feldman, W.P. Zeller, M.J. Freedman, and E.W. Felten, "Sporc: Group Collaboration Using Untrusted Cloud Resources," *Proc. Ninth USENIX Conf. Operating Systems Design and Implementation (OSDI)*, pp. 337-350, Oct. 2010.
[9] K.P.N. Puttaswamy, C.C. Marshall, V. Ramasubramanian, P. Stuedi, D.B. Terry, and T. Wobber, "Docx2go: Collaborative Editing of Fidelity Reduced Documents on Mobile Devices," *Proc. Eighth Int'l Conf. Mobile Systems, Applications, and Services (MobiSys)*, pp. 345-356, June 2010.
[10] Q. Zondervan and A. Lee, "Data Synchronization of Portable Mobile Devices in a Distributed Database System," Technical Report: 98-02, IBM Watson Research Center, Mar. 2011.
[11] E. Lagerspetz, S. Tarkoma, and T. Lindholm, "Dessy: Demonstrating Mobile Search and Synchronization," *Proc. 11th Int'l Conf. Mobile Data Management Mobile Data Management (MDM)*, pp. 284-286, 2010.
[12] Y. Minsky and A. Trachtenberg, "Scalable Set Reconciliation," *Proc. 40th Ann. Allerton Conf. Comm., Control, and Computing*, Oct. 2002.
[13] D. Guo, Y. Liu, X.-Y. Li, and P. Yang, "False Negative Problem of Counting Bloom Filter," *IEEE Trans. Knowledge and Data Eng.*, vol. 22, no. 5, pp. 651-664, May 2010.
[14] D. Guo, J. Wu, H. Chen, Y. Yuan, and X. Luo, "The Dynamic Bloom Filters," *IEEE Trans. Knowledge and Data Eng.*, vol. 22, no. 1, pp. 120-133, Jan. 2010.
[15] S. Tarkoma, C.E. Rothenberg, and E. Lagerspetz, "Theory and Practice of Bloom Filters for Distributed Systems," *IEEE Comm. Surveys and Tutorials*, vol. 14, no. 1, pp. 131-155, Jan.-Mar. 2012.
[16] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *IEEE/ACM Trans. Networking*, vol. 8, no. 3, pp. 281-293, June 2000.
[17] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," *Proc. 13th Ann. ACM Symp. Theory of Computing (STOC)*, 1998.
[18] A.Z. Broder and U. Feige, "Min-Wise versus Linear Independence (Extended Abstract)," *Proc. 11th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA)*, pp. 147-154, Jan. 2000.
[19] M. Charikar, "Similarity Estimation Techniques from Rounding Algorithms," *Proc. 34th Ann. ACM Symp. Theory of Computing (STOC)*, pp. 380-388, 2002.
[20] J.W. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed Content Delivery across Adaptive Overlay Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 5, pp. 767-780, Oct. 2004.
[21] G. Cormode and S. Muthukrishnan, "What's New: Finding Asignificant Differences in Network Data Streams," *IEEE/ACM Trans. Networking*, vol. 13, no. 6, pp. 1219-1232, Dec. 2005.
[22] R. Schweller, Z. Li, Y. Chen, Y. Gao, A. Gupta, Y. Zhang, P.A. Dinda, M.-Y. Kao, and G. Memik, "Reversible Sketches: Enabling Monitoring and Analysis Over High-Speed Data Streams," *IEEE/ACM Trans. Networking*, vol. 15, pp. 1059-1072, Oct. 2007.
[23] P. Flajolet and G.N. Martin, "Probabilistic Counting Algorithms for Data Base Applications," *J. Computer and System Sciences*, vol. 31, no. 2, pp. 182-209, 1985.
[24] N. Ntarmos, P. Triantafillou, and G. Weikum, "Distributed Hash Sketches: Scalable, Efficient, and Accurate Cardinality Estimation for Distributed Multisets," *ACM Trans. Computer Systems*, vol. 27, no. 1, pp. 439-442, 2009.
[25] N. Ntarmos, P. Triantafillou, and G. Weikum, "Statistical Structures for Internet-Scale Data Management," *VLDB J*, vol. 18, no. 6, pp. 1279-1312, 2009.
[26] N. Ntarmos, P. Triantafillou, and G. Weikum, "Counting at Large: Efficient Cardinality Estimation in Internet-Scale Data Networks," *Proc. IEEE 22nd Int'l Conf. Data Eng. (ICDE)*, 2006.
[27] G. Cormode, S. Muthukrishnan, and I. Rozenbaum, "Summarizing and Mining Inverse Distributions on Data Streams via Dynamic Inverse Sampling," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB)*, 2005.
[28] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Math.*, vol. 1, no. 4, pp. 485-509, 2005.
[29] O. Papapetrou, W. Siberski, and W. Nejdl, "Cardinality Estimation and Dynamic Length Adaptation for Bloom Filters," *Distributed and Parallel Databases*, vol. 28, nos. 2/3, pp. 119-156, 2010.
[30] A. Kirsch and M. Mitzenmacher, "Less Hashing, Same Performance: Building a Better Bloom Filter," *Proc. 14th Ann. European Symp. Algorithms*, pp. 456-467, 2006.
[31] B. Godfrey, "Balls and Bins with Structure: Balanced Allocations on Hypergraphs," *Proc. 19th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA)*, pp. 511-517, 2008.
[32] C. Lenzen and R. Wattenhofer, "Tight Bounds for Parallel Randomized Load Balancing: Extended Abstract," *Proc. 43rd Ann. ACM Symp. Theory of Computing (STOC)*, pp. 11-20, 2011.
[33] S. Dutta, S. Bhattacherjee, and A. Narang, "Perfectly Balanced Allocation with Estimated Average Using Approximately Constant Retries," *Proc. Conf. CoRR*, 2011.

Deke Guo received the BS degree in industry engineering from the Beijing University of Aeronautic and Astronautic, China, in 2001, and the PhD degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is an associate professor with the College of Information System and Management, National University of Defense Technology, Changsha, China. His research interests include distributed systems, wireless and mobile systems, P2P networks, and interconnection networks. He is a member of the IEEE and the ACM.

Mo Li received the BS degree in computer science and technology from Tsinghua University, Beijing, China, in 2004, and the PhD degree in computer science and engineering from Hong Kong University of Science and Technology, in 2009. He is a Nanyang assistant professor with the Computer Science Division, School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include distributed systems, wireless sensor networks, pervasive computing and RFID, and wireless and mobile systems. He is a member of the IEEE.