19th ICCRTS

Set Reconciliation in Two Rounds of Communication

Primary Topic: Data, Information, and Knowledge

Alternate Topic 1: Cyberspace, Communications, and Information Networks

Alternate Topic 2: Modeling and Simulation

Name of Author(s): Ryan Gabrys and Ayodeji Coker

POC Name: Ryan Gabrys

POC Organization: SSC Pacific

POC Address: 53560 Hull Street San Diego, CA 92152-5001

POC Telephone: 847 308 9893

POC E-mail: gabrys@spawar.navy.mil

Abstract

In this work, we propose an approach, known as the C2SS-BF method, to synchronizing similar sets of data that uses an Invertible Bloom Filter (IBF). The C2SS-BF method builds on previous work by Eppstein et al. in [6]. By allowing two rounds of communication, we show that in many cases the proposed approach requires substantially less throughput than the algorithm proposed in [6]. The C2SS-BF compares favorably to the work by Guo and Li in [9], and, in particular, it requires less computational complexity and throughput.

Set Reconciliation in Two Rounds of Communication

Ryan Gabrys and Ayodeji Coker
Spawar Systems Center, San Diego
gabrys@spawar.navy.mil, ayodeji.coker@navy.mil

***Abstract*—In this work, we propose an approach, known as the C2SS-BF method, to synchronizing similar sets of data that uses an Invertible Bloom Filter (IBF). The C2SS-BF method builds on previous work by Eppstein et al. in [6]. By allowing two rounds of communication, we show that in many cases the proposed approach requires substantially less throughput than the algorithm proposed in [6]. The C2SS-BF compares favorably to the work by Guo and Li in [9], and, in particular, it requires less computational complexity and throughput.**

I. Introduction

There has been an increasing need to maintain a Common Operational Picture (COP) between a collection of hosts within a disconnected, intermittent and low-bandwidth (DIL) maritime environment. Existing Command and Control (C2) systems that use event-based protocols may conserve bandwidth, but they do not guarantee a COP in a DIL environment. The purpose of the Command and Control Data Synchronization Service (CS22) is to develop synchronization tools that not only ensure synchronization occurs, but also guarantee synchronization is achievable within a DIL environment. In particular, the unreliable nature of the DIL environment is accounted for in our framework by ensuring that our method addresses the following core properties:

1) The method uses minimal communication rounds, and
2) The throughput between any two hosts on the network is limited.

The document is organized as follows. In Section II, we review the current approach to reconciling data sets used by C2SS and detail our contribution. In Section III we define the notation to be used in this paper. In Section IV we describe our approach to synchronizing similar sets of data. In Section V, we provide simulation results comparing our proposed algorithm to existing approaches. Section VI concludes the paper.

II. Current Approach to Reconciling Data in C2SS and Our Contribution

Suppose there are two hosts A and B where Host A has access to the set $\mathcal{S}_A \subseteq GF(2)^b$ and Host B has access to the set $\mathcal{S}_B \subseteq GF(2)^b$. The ***set reconciliation problem*** is to determine which information must be sent between Host A and Host B so that each host can compute $\mathcal{S}_A \cup \mathcal{S}_B$. We use the terms set reconciliation, data reconciliation, and synchronization to refer to the process by which Host A and Host B compute $\mathcal{S}_A \cup \mathcal{S}_B$.

In [6], [10], and [13] the authors considered the set reconciliation problem under the additional constraint that only a single round of communication was allowed. The goal in this work is to provide a solution to the set reconciliation problem that requires two rounds of communication. It is also desirable that any proposed algorithm posses low encoding/decoding complexity properties.

The current approach taken by the C2SS software (for set reconciliation) is to use a set of hashes along with a Merkle tree. The hashes are used to represent some unit of information and the Merkle tree organizes the hashes in a hierarchical manner to facilitate comparison. For shorthand, we refer to this approach as the ***C2SS-HM method***. The C2SS-HM method has been demonstrated to provide very reliable data synchronization; however, it needs to be optimized in the following areas:

1) For wide Merkle trees, many hashes need to be compared/exchanged at the same time, and
2) for tall Merkle trees, set reconciliation requires many rounds of communication.

In this work, we draw from the analysis provided in [12], which also uses a method similar to the C2SS-HM method. In the following analysis we assume the Merkle tree is balanced. Assuming w is the width of the tree and the size of the symmetric difference is $d = |\mathcal{S}_A \triangle \mathcal{S}_B| = |(\mathcal{S}_A \setminus \mathcal{S}_B) \cup (\mathcal{S}_B \setminus \mathcal{S}_A)|$, it was shown in [12] that the expected number of rounds of communication (or the height of the tree) is $O(2\log_w(\frac{d}{\bar{d}+1})) + O(1)$ for some positive integer $\bar{d}$. We note that given the unreliable nature of a DIL environment, a protocol with fewer rounds of communication is desirable. In addition, the C2SS-HM method also requires maintaining a tree structure (preferably balanced) in memory.

The purpose of this document will be to discuss a new approach to set reconciliation that overcomes several of the drawbacks to the C2SS-HM method. More specifically, we propose an approach to set reconciliation that requires at most two rounds of communication and does not require a tree structure. The principal tool used in our proposed method is a Bloom Filter and so we refer to our method as the ***C2SS-BF method***. The basic idea behind the C2SS-BF method is similar to that proposed in [6] and [9]. We compute a hash on Host A and a hash on Host B. Then, Host A and B exchange their hashes. On Host A we determine $\mathcal{S}_A \setminus \mathcal{S}_B$ and similarly on Host B we determine $\mathcal{S}_B \setminus \mathcal{S}_A$. Finally, the information $\mathcal{S}_A \setminus \mathcal{S}_B$ and $\mathcal{S}_B \setminus \mathcal{S}_A$ is exchanged between Host A and Host B.

The C2SS-BF method has the following important attributes:

1) Requires less throughput than current approaches to set reconciliation ([6], [13]).
2) Requires only two rounds of communication.

In Section V and Appendix B, we provide comparisons between existing approaches in the literature and the proposed algorithm in this paper.

III. Some notation

The following is a description of the notations used in the remainder of this paper. We assume that $f_1, \ldots, f_k$ are hash functions where for any $i \in \{1, 2, \ldots, k\}$, $f_i : GF(2)^b \rightarrow \{(i-1) \cdot \frac{d(k+1)}{k} + 1, (i-1) \cdot \frac{d(k+1)}{k} + 2, \ldots, i \cdot \frac{d(k+1)}{k}\}$ for positive integers d, k. For an element $\boldsymbol{x} \in GF(2)^b$, let $f^k(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$. We refer to the vector $f^k(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$ as the ***locations*** of the element $\boldsymbol{x}$.

IV. The C2SS-BF Method

In this section, we begin by describing at a high level the basic ideas behind the C2SS-BF method. Afterwards, we describe in detail the algorithm for computing $\mathcal{S}_A \cup \mathcal{S}_B$.

Recall, our problem is that given two hosts A and B where Host A has access to the set $\mathcal{S}_A \subseteq GF(2)^b$ and Host B has access to the set $\mathcal{S}_B \subseteq GF(2)^b$ to determine which information must be sent between Host A and Host B using two rounds of communication so that Host B and Host A can each compute $\mathcal{S}_A \cup \mathcal{S}_B$. Let $t = |\mathcal{S}_B| \geq |\mathcal{S}_A|$ and let $d = |\mathcal{S}_A \triangle \mathcal{S}_B|$. We assume that t and d are known to both Host A and Host B. We note that estimates of d can be obtained using sampling techniques such as those described in [6] and [7]. After two rounds of communication, the result is that Host A and Host B will recover $\mathcal{S}_A \triangle \mathcal{S}_B$ (and consequently can determine $\mathcal{S}_A \cup \mathcal{S}_B$) with probability $O(d^{-k+2})$.

In Section IV-A, we describe the encoding process followed by the decoding process in Section IV-B. Afterwards, we comment on the encoding/decoding complexity and the throughput required for the C2SS-BF method.

The algorithm proceeds as follows:

1) Host A inserts all the elements in $\mathcal{S}_A$ into a Bloom Filter (BF), denoted $h_{\mathcal{S}_A}$. Host B inserts the elements from $\mathcal{S}_B$ into a Bloom Filter, denoted $h_{\mathcal{S}_B}$.
2) Host A transmits $h_{\mathcal{S}_A}$ to Host B and Host B transmits $h_{\mathcal{S}_B}$ to Host A.
3) Host B receives $h_{\mathcal{S}_A}$ and uses it to compute $\mathcal{S}_B \setminus \mathcal{S}_A$, and Host A receives $h_{\mathcal{S}_B}$ and uses it to compute $\mathcal{S}_A \setminus \mathcal{S}_B$.
4) Host B transmits $S_B \setminus S_A$ to Host A, and Host A transmits $S_A \setminus S_B$ to Host B.

Since $\mathcal{S}_A \cup \mathcal{S}_B = \mathcal{S}_A \cup (\mathcal{S}_B \setminus \mathcal{S}_A) = \mathcal{S}_B \cup (\mathcal{S}_A \setminus \mathcal{S}_B)$, after the completion of step 4), both Host A and Host B can determine $\mathcal{S}_A \cup \mathcal{S}_B$.

The computation of h_{S_A} and h_{S_B} are identical and are described in Section IV-A.

A. Encoding

On each host, we begin by creating a special type of Bloom Filter known as an Invertible Bloom Filter (IBF). Our IBF is comprised of a collection of $n = d(k+1)$ cells where $d = |\mathcal{S}_A \triangle \mathcal{S}_B|$ and k is some integer. We assume, for convenience, that n is a power of two. As before, we refer to the IBF on Host A as $h_{\mathcal{S}_A}$ and similarly we refer to the IBF on Host B as $h_{\mathcal{S}_B}$. We use the terms hash and IBF interchangeably for the remainder of the paper.

To encode $h_{\mathcal{S}_A}$ (and similarly for $h_{\mathcal{S}_B}$) we simply insert all the elements in $\mathcal{S}_A$ (or $\mathcal{S}_B$) into an IBF. In the following, we let $\mathcal{S} = \mathcal{S}_A$ if the procedure is being performed on Host A and $\mathcal{S} = \mathcal{S}_B$ if the procedure is being performed on Host B.

When an element is inserted into an IBF it is hashed to k different cells, and we assume the hash functions $f_1, \ldots, f_k$ are the same on both Host A and Host B. Let q be a positive integer to be defined later. Each cell contains two fields:

1) c: an integer which is simply the number of times the cell has been hashed to modulo q.
2) l: the sum of all the element locations that have hashed into the cell modulo n.

To fix ideas, we include the encoding algorithm for the C2SS-BF method below along with an example. We refer to k-th cell in the IBF below as $h_{\mathcal{S}}[k]$. We assume that $h_{\mathcal{S}}[k]$ is initialized so that the count field for every cell is zero and the locations field is simply all-zeros.

Algorithm 1: C2SS-BF Encode

```
input : The set $\mathcal{S} \subset GF(2)^b$
output: The IBF $h_{\mathcal{S}}$
1 for every $\boldsymbol{x} \in \mathcal{S}$ do
2     for $i = 1 : k$ do
3         $h_{\mathcal{S}}[f_i(\boldsymbol{x})].c = h_{\mathcal{S}}[f_i(\boldsymbol{x})].c + 1 \mod q$;
          $h_{\mathcal{S}}[f_i(\boldsymbol{x})].l = h_{\mathcal{S}}[f_i(\boldsymbol{x})].l + f^k(\boldsymbol{x}) \mod n$;
4     end
5 end
```

Example 1. *Assume $d = 4$, $k = 2$ and $\mathcal{S} = \{(0,0,0), (1,1,0)\}$ where $f_1((0,0,0)) = 5$, $f_2((0,0,0)) = 8$, $f_1((1,1,0)) = 5$, and $f_2((1,1,0)) = 9$. Assume that Algorithm 1 is performed on the elements from $\mathcal{S}$ resulting in the IBF $h_{\mathcal{S}}$. Then, $h_{\mathcal{S}}$ would appear as shown in Figure 1.*

We have the following corollary.

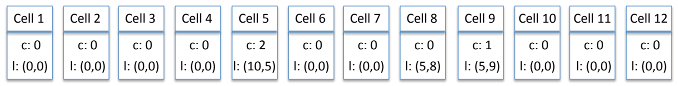

Fig. 1. IBF $h_{\mathcal{S}}$ when $\mathcal{S} = \{(0,0,0),(1,1,0)\}$

Corollary 1. *The hash $h_{\mathcal{S}}$ contains*

$$d(k+1)\left(\log_2(q) + k\log_2(d(k+1))\right)$$

bits of information.

Proof: According the encoding procedure, the hash $h_{\mathcal{S}}$ is comprised of two fields. The first field (c field) requires $\log_2(q)$ bits of information and the second field (l field) requires $k\log_2(n(k+1))$ bits of information. ■

In the next subsection, we show how compute $\mathcal{S}_A \triangle \mathcal{S}_B$ from $h_{\mathcal{S}_A}, h_{\mathcal{S}_B}$.

B. Decoding

In this subsection, we enumerate the decoding procedure for the C2SS-BF method performed on Host B or Host A given $h_{\mathcal{S}_A}, h_{\mathcal{S}_B}$. For the purposes of this section, we assume the local host is Host B; however the algorithm is identical for the case where the decoding takes place on Host A.

We now explain in words the decoding algorithm. Let $h_{\mathcal{S}_B}.c = (h_{\mathcal{S}_B}[1].c, h_{\mathcal{S}_B}[2].c, \ldots, h_{\mathcal{S}_B}[n].c)$ and $h_{\mathcal{S}_A}.c = (h_{\mathcal{S}_A}[1].c, h_{\mathcal{S}_A}[2].c, \ldots, h_{\mathcal{S}_A}[n].c)$. To determine $\mathcal{S}_B \setminus \mathcal{S}_A$, we first compute the vector $h_c = h_{\mathcal{S}_B}.c - h_{\mathcal{S}_A}.c$.

We now consider the following scenario. Suppose $\boldsymbol{x} \in \mathcal{S}_A \cap \mathcal{S}_B$. If $\boldsymbol{x}$ hashes to cell i in $h_{\mathcal{S}_B}$ (that is, if there exists some $j \in [k]$, where $f_j(\boldsymbol{x}) = i$), then since Host A and B have the same hash functions, $\boldsymbol{x}$ would also be hashed to cell i in $h_{\mathcal{S}_A}$ since $\boldsymbol{x} \in \mathcal{S}_A$. Thus, any increments to the vector $h_{\mathcal{S}_B}.c$ caused by $\boldsymbol{x} \in \mathcal{S}_B$ are canceled out in h_c since the same increments are made to the vector $h_{\mathcal{S}_A}.c$ since $\boldsymbol{x} \in \mathcal{S}_A$.

From the previous paragraph, if cell i in h_c has a value $+1$ it follows that one of the elements from $\mathcal{S}_B$ that hashed to cell i in $h_{\mathcal{S}_B}$ is from the set $\mathcal{S}_B \setminus \mathcal{S}_A$. Let $\boldsymbol{y} \in \mathcal{S}_B \setminus \mathcal{S}_A$ be an element that hashed to cell i. In this case, we produce an estimate for $\boldsymbol{y} \in \mathcal{S}_B \setminus \mathcal{S}_A$ by finding an element $\hat{\boldsymbol{y}} \in \mathcal{S}_B$ where $f^k(\hat{\boldsymbol{y}}) = h_{\mathcal{S}_B}[i].l$. We will show in Appendix A that with high probability, $\boldsymbol{y} = \hat{\boldsymbol{y}}$. If an element $\hat{\boldsymbol{y}}$ can be found, then we proceed by removing the contribution of $\hat{\boldsymbol{y}}$ from $h_{\mathcal{S}_B}$. In other words, we decrement all the c fields for cells where $\hat{\boldsymbol{y}}$ hashed to and we subtract $f^k(\hat{\boldsymbol{y}})$ from all the l fields for cells where $\hat{\boldsymbol{y}}$ hashed to.

Now suppose cell i in h_c has a value -1. From the discussion, it follows that one of the elements, say $\boldsymbol{y}' \in \mathcal{S}_A$, that hashed to cell i in $h_{\mathcal{S}_A}$ is such that $\boldsymbol{y}' \in \mathcal{S}_A \setminus \mathcal{S}_B$. Since we assumed the decoding is being performed on $\mathcal{S}_B$, we do not produce an estimate of $\boldsymbol{y}'$. However, with high probability, we have that $f^k(\boldsymbol{y}') = h_{\mathcal{S}_A}[i].l$. Since $f^k(\boldsymbol{y}')$ contains the locations $\boldsymbol{y}'$ hashed to, we proceed in this case by removing the contribution of $\boldsymbol{y}'$ from $h_{\mathcal{S}_A}$.

The algorithm thus proceeds by successively searching for positions where h_c is equal to ± 1 and removing the elements (as described in the previous two paragraphs) from either $h_{\mathcal{S}_A}$ or $h_{\mathcal{S}_B}$ until both $h_{\mathcal{S}_A}$ and $h_{\mathcal{S}_B}$ are empty. The details are provided in Algorithm 2.

Algorithm 2: C2SS-BF Decode

```
input : $\mathcal{S}_B$, $h_{\mathcal{S}_A}$, $h_{\mathcal{S}_B}$
output: An estimate $\mathcal{F}$ of $\mathcal{S}_B \setminus \mathcal{S}_A$
1  $\mathcal{F} = \emptyset$;
2  $\ell = 1$;
3  while $\ell \leq n$ do
4      $h_c = h_{\mathcal{S}_B}.c - h_{\mathcal{S}_A}.c$;
5      if $h_c[\ell] = 1$ then
6          if $\exists! \hat{\boldsymbol{y}} \in \mathcal{S}_B : f^k(\hat{\boldsymbol{y}}) = h_{\mathcal{S}_B}[i].\ell$ then
7              Add $\hat{\boldsymbol{y}}$ to $\mathcal{F}$;
8              for $i = 1 : k$ do
9                  $h_{\mathcal{S}_B}[f_i(\hat{\boldsymbol{y}})].c = h_{\mathcal{S}_B}[f_i(\hat{\boldsymbol{y}})].c - 1 \mod q$;
10                 $h_{\mathcal{S}_B}[f_i(\hat{\boldsymbol{y}})].l = h_{\mathcal{S}_B}[f_i(\hat{\boldsymbol{y}})].l - f^k(\hat{\boldsymbol{y}})$
                   $\mod n$;
11             end
12         end
13         else
14             STOP. A decoding error occurred.
15         end
16         $\ell = 0$;
17     end
18     else if $h_c[\ell] = -1$ then
19         $(j_1, j_2, \ldots, j_k) = h_{\mathcal{S}_A}[\ell].l$;
20         for $i = 1 : k$ do
21             $h_{\mathcal{S}_A}[j_i].c = h_{\mathcal{S}_A}[j_i].c - 1 \mod q$;
22             $h_{\mathcal{S}_A}[j_i].l = h_{\mathcal{S}_A}[j_i].l - (j_1, j_2, \ldots, j_k)$
               $\mod n$;
23         end
24         $\ell = 0$;
25     end
26     $\ell = \ell + 1$;
27 end
28 If $h_c$ does not contains all-zeros, then a decoding error
   has occurred.
```

We note that for every element $\boldsymbol{y} \in \mathcal{S}_A \triangle \mathcal{S}_B$, Algorithm 2 requires that, in order to produce the estimate $\hat{\boldsymbol{y}}$, we have to search through the elements in $\mathcal{S}_B$. Assuming the elements $\mathcal{S}_B$ are sorted, each search operation would require $O(\log(|\mathcal{S}_B|)$ operations, and so the total complexity of Algorithm 2 is $O(d\log(|\mathcal{S}_B|)$. Furthermore, as described in the theorem below, the probability of incorrect synchronization is $O(d^{-k+2})$. The proof of the theorem is included in Appendix A.

Theorem 1. *If $|\mathcal{S}_B| \leq k\frac{\log_2(\frac{1}{d})}{\log_2(1-\frac{k^k}{d^k(k+1)^k})}$ and $q \leq k\log_e(d) + e - 1$, then with probability $O(d^{-k+2})$, the output of Algorithm 2 is such that $\mathcal{F} \neq \mathcal{S}_B \setminus \mathcal{S}_A$.*

In the next section, we present simulation results illustrating some properties of the C2SS-BF method.

V. Simulation Results

In this section, we evaluate the C2SS-BF method against the set reconciliation algorithms from [9], [13]. We assumed that there were two hosts A and B where Host A has access to the set $\mathcal{S}_A \subseteq GF(2)^{131072}$ and Host B has access the the set $\mathcal{S}_B \subseteq GF(2)^{131072}$ where $|\mathcal{S}_A| = 200, |\mathcal{S}_B| = 200$. We chose to test the synchronization of bit-strings of size 131072 (16 kilobytes). The choice of 16 kilobytes was motivated by the setup where two databases are synchronizing their pages (which are usually between 4KB and 32KB [3]). We then tested the performance of the algorithms for varying sizes of $d = |\mathcal{S}_A \triangle \mathcal{S}_B|$. For every value of d from the set $\{10, 20, 30, 40, 50, 60, 70, 80\}$, we ran 10,000 trials where we attempted to synchronize the sets $\mathcal{S}_A, \mathcal{S}_B$ given that $|\mathcal{S}_A \triangle \mathcal{S}_B| = d$. In other words, for every trial we attempted to compute $\mathcal{S}_A \cup \mathcal{S}_B$ on both Host A and Host B.

We used the CBF from [9] in a manner analogous to the usage of the IBF in the C2SS-BF method. In particular, a CBF from [9] was used to determine the set difference $\mathcal{S}_A \setminus \mathcal{S}_B$ on Host A and a CBF was used to determine the set difference $\mathcal{S}_B \setminus \mathcal{S}_A$ on Host B. Then $\mathcal{S}_A \setminus \mathcal{S}_B$ was sent from Host A to B and similarly $\mathcal{S}_B \setminus \mathcal{S}_A$ was sent from Host B to Host A. For the results shown in Figures 2 and 3, we constructed CBFs of the following lengths: 1) 65100 2) 303600 3) 492800 4) 2077200 5) 2698800 6) 3338400 7) 3993300 8) 4661100. The CBF of length 65100 was used for data reconciliation when $d = 10$; the CBF of length 303600 was used to reconcile data when $d = 20$, and so on. The CBFs constructed consisted of simply an array of cells containing binary numbers.

In Figure 2, we plotted the error rates for the C2SS-BF method and the approach from [9] for varying values of d. We assumed, for the purposes of the C2SS-BF method, that d was known and that $k = 4$. Since the approach from [13] is exact, the probability of correct synchronization is 1 and so no data is present for the polynomial interpolation approach described in [13] in Figure 2. It can be seen from Figure 2 that as d increases, the probability of incorrect synchronization for the C2SS-BF method decreases, which is consistent with the analysis from the previous section (since the probability of incorrect synchronization is $O(d^{-k+2})$). Such a trend did not seem to hold for the approach from [9] even though the size of the CBF was increased for larger values of d.

In Figure 3, we plotted the total number of bits that were sent between Hosts A and B using the C2SS-BF method, the approach from [9], and the approach from [13]. As a frame of reference, we also plotted a lower bound of db. The polynomial interpolation approach from [13] (like the approaches in [6], [10], [12]) require at least $2db$ bits of information exchange since these approaches only require a single round of communication. The C2SS-BF method as well as the one from [9] use two rounds of communication and, as a result, these approaches reduced the total throughput given our test scenario.

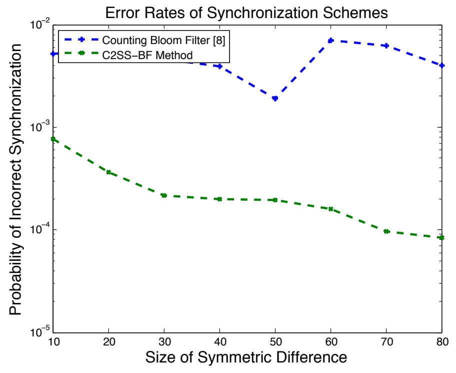

Fig. 2. Error Rates for Synchronization Algorithms

From Figures 2 and 3 it can be seen that the C2SS-BF method has a lower probability of incorrect synchronization and it requires less throughput than the approach in [9]. This lower probability of incorrect synchronization is a result of using an IBF instead of a CBF. We note that in addition to requiring the transmission of fewer bits, the C2SS-BF decoder has complexity $O(d \max(\log |\mathcal{S}_A|, \log |\mathcal{S}_B|)$ whereas the CBF approach in [9] had decoding complexity $O(|\mathcal{S}_A| + |\mathcal{S}_B|)$. Recall the method from [13] has decoding complexity $O(d^3)$, which in many cases, renders it impractical.

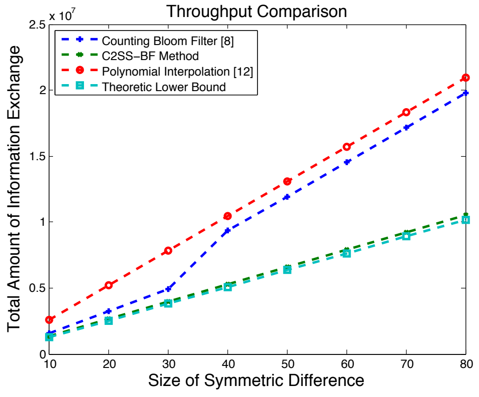

Fig. 3. Total Throughput for Synchronization Algorithms

VI. Conclusion

In this work, we considered an algorithm for synchronizing similar sets of data. In particular, we considered an approach to the set reconciliation problem, known as the C2SS-BF method, which requires only two rounds of communication. It was demonstrated that the C2SS-BF method has the potential to reduce the throughput as well as computational complexity of many alternative schemes in the literature.

We note that one potential limitation of the C2SS-BF method is that the C2SS-BF method requires an upper bound for $d = |\mathcal{S}_A \triangle \mathcal{S}_B|$. Thus, additional communication may be necessary to produce accurate estimates for d. However, if accurate upper bounds for d can be determined, the C2SS-BF method can significantly reduce the rounds of communication required to determine $\mathcal{S}_A \cup \mathcal{S}_B$ on either Host A or Host B. Future work involves incorporating our algorithm into future releases of C2SS and providing mechanisms that estimate the symmetric difference.

References

[1] "Test Plan for Open Track Manager (OTM) Testing for Trident Warrior 2013 (TW13)," Document Version 0.5, 2013.

[2] "Integrated Shipboard Network System (ISNS) Application Integration (AI) Test Report," Open Track Manager (OTM) v1.0, 2013.

[3] "Understanding Pages and Extents," available at http://technet.microsoft.com/en-us/library/ms190969(v=sql.105).aspx.

[4] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," Internet Mathematics, 2004.

[5] J. Canny, "Lecture 10, Chernoff Bounds," Notes from CS174 offered at UC Berkeley, 2012, available at http://www.cs.berkeley.edu/∼jfc/cs174/lecs/lec10/lec10.pdf.

[6] D. Eppstein, M. Goodrich, F. Uyeda, G. Varghese, "What's the difference? Efficient set reconciliation without prior context," *SIGCOMM* 2011.

[7] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Computer and System Sciences*, 1985.

[8] M. T. Goodrich and M. Mitzenmacher, "Invertible Bloom Lookup Tables," ArXiv e-prints, 2011.

[9] D. Guo and M. Li, "Set Reconciliation via Counting Bloom Filters," *IEEE Trans. Knowledge and Data Eng.*, 2013.

[10] M. Karpovsky, L. Levitin, and A. Trachtenberg, "Data verification and reconciliation with generalized error-control codes," *IEEE Trans. Info. Theory*, July 2003.

[11] R. J. Lipton, "Efficient checking of computations," *STACS*, 1990.

[12] Y. Minsky and A. Trachtenberg, "Practical set reconciliation," *Tech. Rep.*, Department of Electrical and Computer Engineering, Boston University, 2002.

[13] Y. Minsky, A. Trachtenberg, R. Zippel, "Set reconciliation with nearly optimal communication complexity," *IEEE Trans. Inform. Theory*, 2003.

[14] R. Perkins, F. Dejesus, J. Durham, R. Hastings, "C2 Data Synchronization in Disconnected, Intermittent, and Low-Bandwidth (DIL) Environments," *ICCRTS*, 2013.

Appendix A
Proof of Theorem 1

In this section, we consider the probability the decoding algorithm described in Section IV fails. There are 3 possible scenarios under which the decoding algorithm would fail. First, Algorithm 2 can fail at step 14 if there is any element in $\mathcal{S}_B$ that hashes to all the same cells as an element in $\mathcal{S}_A \triangle \mathcal{S}_B$. Second, the decoding can fail if a cell is hashed to q or more times by an element in $\mathcal{S}_A \triangle \mathcal{S}_B$. Finally, the decoding can fail at step 28 if the following scenario holds: Suppose $\mathcal{S}'$ is a subset of $\mathcal{S}_A \triangle \mathcal{S}_B$ and suppose $\mathcal{L}$ is the set of all cells hashed to by the elements of $\mathcal{S}'$. Then, Algorithm 2 can fail if for any $\ell \in \mathcal{L}$, there are at least two elements in $\mathcal{S}'$ hash to ℓ. The three conditions are stated mathematically below:

1) $\exists \boldsymbol{x} \in \mathcal{S}_B, \exists \boldsymbol{y} \in \mathcal{S}_A \triangle \mathcal{S}_B$ where $f^k(\boldsymbol{x}) = f^k(\boldsymbol{y})$.
2) $\exists i, 1 \leq i \leq n$ where $|\{(\boldsymbol{x}, j) : \boldsymbol{x} \in \mathcal{S}_A \triangle \mathcal{S}_B, j \in \{1, \ldots, k\}, f_j(\boldsymbol{x}) = i\}| \geq q$.
3) Suppose $\mathcal{S}' \subseteq (\mathcal{S}_A \triangle \mathcal{S}_B)$ and $\mathcal{L} = \{f_i(\boldsymbol{x}) : i \in \{1, 2, \ldots, k\}, \boldsymbol{x} \in \mathcal{S}'\}$ and $\forall \ell \in \mathcal{L}, \exists \boldsymbol{x}, \exists \boldsymbol{y} \in \mathcal{S}', \exists i, \exists j \in \{1, \ldots, k\}$ where $f_i(\boldsymbol{x}) = f_j(\boldsymbol{y})$.

In the following, we refer to the first event listed as item 1) above as ξ_1, the second event as ξ_2, and the third event as ξ_3. For any event, ς, we let $P(\varsigma)$ denote the probability the event ς occurs. Let ξ denote the event that $\mathcal{F} \neq \mathcal{S}_A \triangle \mathcal{S}_B$ where $\mathcal{F}$ is computed according to Algorithm 2. Then, by the union bound we have

$$P(\xi) \leq P(\xi_1) + P(\xi_2) + P(\xi_3). \tag{1}$$

We begin with the following lemma.

Lemma 1. $P(\xi_1) \leq d(1 - \frac{k^k}{d^k(k+1)^k})^{|\mathcal{S}_B|}$.

Proof: For any $\boldsymbol{y} \in \mathcal{S}_A \triangle \mathcal{S}_B, \boldsymbol{x} \in \mathcal{S}_B$, $P(f^k(\boldsymbol{x}) = f^k(\boldsymbol{y})) = \left(\frac{k}{d(k+1)}\right)^k = \frac{k^k}{d^k(k+1)^k}$. Therefore, for $\boldsymbol{y} \in \mathcal{S}_A \triangle \mathcal{S}_B$,

$$P(\nexists \boldsymbol{x} \in \mathcal{S}_B : f^k(\boldsymbol{x}) = f^k(\boldsymbol{y})) = (1 - \frac{k^k}{d^k(k+1)^k})^{|\mathcal{S}_B|}.$$

Then, since $|\mathcal{S}_d| = d$ we have $P(\xi_1) \leq d(1 - \frac{k^k}{d^k(k+1)^k})^{|\mathcal{S}_B|}$ as desired. ■

The following corollary follows from Lemma 1.

Corollary 2. *If* $|\mathcal{S}_B| \leq k\frac{\log_2(\frac{1}{d})}{\log_2(1-\frac{k^k}{d^k(k+1)^k})}$, *then* $P(\xi_1) \leq d^{-k+1}$.

We next bound the probability of the $P(\xi_2)$.

Lemma 2. *If* $q \leq k\log_e(d) + e - 1$, *then* $P(\xi_2) \leq O(d^{-k+2})$.

Proof: For some integer i where $1 \leq i \leq d(k+1)$ and any $\boldsymbol{x} \in \mathcal{S}_A \triangle \mathcal{S}_B$, let $\mathcal{X}$ be a random variable that is equal to 1 when $f_j(\boldsymbol{x}) = i$ and 0 otherwise for $j = \lceil \frac{i}{\frac{d(k+1)}{k}} \rceil$. Then $\mathcal{X}$ is a Bernoulli random variable with parameter $p = \frac{k}{d(k+1)}$.

Let $\mathcal{X}_i$ be a a random variable that has value $|\{\boldsymbol{x} \in \mathcal{S}_A \triangle \mathcal{S}_B, j = \{1, 2, \ldots, k\} : f_j(\boldsymbol{x}) = i\}|$. Notice that since $\mathcal{X}$ is a Bernoulli random variable, $\mathcal{X}_i$ is a Poisson random variable with mean $\lambda = \frac{dk}{d(k+1)}$. Applying the Chernoff bound (c.f. [5]), which states that $P(\mathcal{X}_i \geq a) \leq e^{-ta}M_{\mathcal{X}_i}(t)$ where $M_{\mathcal{X}_i}(t)$ denotes the moment generating function for $\mathcal{X}_i$, gives $P(\mathcal{X}_i \geq q) \leq e^{-tq}e^{\lambda(e^t-1)}$. Letting $t = 1$ and substituting $q = k\log_e(d) + e - 1$, gives that

$$\begin{aligned} P(\mathcal{X}_i \geq q) &\leq d^{1-k}\frac{1}{e^{e-1}}e^{\lambda(e-1)} \\ &\leq ed^{-k}. \end{aligned}$$

Since there are $d(k+1)$ possibilities for the index i, the probability that there exists an i where $|\{\boldsymbol{x} \in \mathcal{S}_A \triangle \mathcal{S}_B, j = \lceil \frac{i}{\frac{d(k+1)}{k}} \rceil : f_j(\boldsymbol{x}) = i\}| \geq q$ is at most $(k+1)d \cdot ed^{-k} = O(d^{-k+2})$ as desired. ■

Finally, we bound the probability of the third event in (1).

Lemma 3. $P(\xi_3) \leq O(d^{-k+2})$.

Proof: The probability of such an event occurring is equivalent to the probability a 2-core exists in a hypergraph (see [8]). This probability was shown to be at most $O(d^{-k+2})$. ■

Combining Equation 1 along with Lemmas 1, 2, and 3, we have the result.

Theorem 1. *If* $|\mathcal{S}_B| \leq k\frac{\log_2(\frac{1}{d})}{\log_2(1-\frac{k^k}{d^k(k+1)^k})}$ *and* $q \leq k\log_e(d) + e - 1$, *then* $P(\xi) \leq O(d^{-k+2})$.

Appendix B
Analytic Comparison of Proposed Approach with Existing Approaches

In this section, we provide an analytic comparison of the approach proposed in this paper with existing approaches to set reconciliation. We begin by considering the properties of our set reconciliation algorithm (as described in Section IV). Recall $\mathcal{S}_A, \mathcal{S}_B \subseteq GF(2)^b$ and the size of the symmetric difference is $d = |\mathcal{S}_A \triangle \mathcal{S}_B| = |(\mathcal{S}_A \setminus \mathcal{S}_B) \cup (\mathcal{S}_B \setminus \mathcal{S}_A)|$. Let k be a positive integer and suppose $t = |\mathcal{S}_B| \geq |\mathcal{S}_A|$ and $t \leq k\frac{\log_2(\frac{1}{d})}{\log_2(1-\frac{k^k}{d^k(k+1)^k})}$, where e is the base of the natural log. Our proposed approach requires approximately $2d(k+1)(\log_2(k\log_e(d)+e-1)+k\log_2(d(k+1)))+db$ total bits of information exchange and the probability of incorrect synchronization (or the probability the algorithm fails) is $O(d^{-k+2})$. For the case where $t, d << 2^b$, our approach requires approximately $db(1+\delta)$ total bits of information exchange where $0 < \delta < 1$ so that our approach is close to the optimum total number of bits that must be exchanged which is db. Furthermore, our approach has decoding complexity $O(d\log(t))$ and encoding complexity $O(t)$.

Recall that our algorithm requires an additional round of communication. As a result, in many cases our algorithm reduces the throughput required in both [6] and [13]. Recall that, if the methods from [6] and [13] are used, then the throughput is at least $2Kdb$ for some positive integer K.

If the approach from [9] is used with a Counting Bloom Filter (CBF) of size $d(k+1)$ (which is the size of the Bloom Filter used in our approach), then from equation (8) in [9] we have that the probability of incorrect synchronization is approximately $O(1-(1-\frac{1}{d(k+1)})^{kd} - \frac{(kd/2)^2(1-\frac{1}{d(k+1)})^{kd}}{(m-1)^2})^k)$ which is much larger than $O(d^{-k+2})$ for small k and large d. In addition, the CBF method in [9] has decoding complexity $O(|\mathcal{S}_A|+|\mathcal{S}_B|)$ which, for the case where $|\mathcal{S}_A|+|\mathcal{S}_B|$ is large, may be prohibitively expensive.