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.
14. 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 $S_A \subseteq GF(2)^b$ and Host $B$ has access to the set $S_B \subseteq GF(2)^b$. The set reconciliation problem is to determine which information must be sent between Host $A$ and Host $B$ so that each host can compute $S_A \cup S_B$.

We use the terms set reconciliation, data reconciliation, and synchronization to refer to the process by which Host $A$ and Host $B$ compute $S_A \cup S_B$.

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 possess 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 $d = |S_A \Delta S_B| = |(S_A \setminus S_B) \cup (S_B \setminus S_A)|$, it was shown in [12] that the expected number of rounds of communication (or the height of the tree) is $O(2\log_w(\frac{d}{\delta})) + O(1)$ for some positive integer $\delta$. 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 $S_A \setminus S_B$ and similarly on Host B we determine $S_B \setminus S_A$. Finally, the information $S_A \setminus S_B$ and $S_B \setminus S_A$ 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 attributes:

\begin{itemize}
\item \begin{itemize}
\item Requires only two rounds of communication.
\item \end{itemize}
\end{itemize}

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 $d = |S_A \Delta S_B|$ 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 $h_{S_A}$ and similarly we refer to the IBF on Host B as $h_{S_B}$. We use the terms hash and IBF interchangeably for the remainder of the paper.

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

\begin{itemize}
\item 1) $c$: an integer which is simply the number of times the cell has been hashed to modulo $q$.
\item 2) $l$: the sum of all the element locations that have hashed into the cell modulo $n$.
\end{itemize}

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 $h_S[k]$. We assume that $h_S[k]$ is initialized so that the count field for every cell is zero and the locations field is simply all-zeros.

\begin{algorithm}
1 \textbf{for every} $x \in S$ \textbf{do}
2 \hspace{1cm} \textbf{for} $i = 1 : k$ \textbf{do}
3 \hspace{2cm} $h_S[f_i(x)].c = h_S[f_i(x)].c + 1 \text{ mod } q$;
4 \hspace{2cm} $h_S[f_i(x)].l = h_S[f_i(x)].l + f_i(x) \text{ mod } n$;
5 \textbf{end}
6 \textbf{end}
\end{algorithm}

Example 1. Assume $d = 4$, $k = 2$ and $S = \{(0,0,0),(1,1,0)\}$ 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 $S$ resulting in the IBF $h_S$. Then, $h_S$ would appear as shown in Figure 1.

We have the following corollary.
Corollary 1. The hash $h_S$ 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 $h_S$ is comprised of two fields. The first field (e 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 $S_A \triangle S_B$ from $h_{S_A}, h_{S_B}$.

B. Decoding

In this subsection, we enumerate the decoding procedure for the C2SS-BF method performed on Host B or Host A given $h_{S_A}, h_{S_B}$. 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 $h_{S_B}.c = (h_{S_B[1]}.c, h_{S_B[2]}.c, \ldots, h_{S_B[n]}.c)$ and $h_{S_A}.c = (h_{S_A[1]}.c, h_{S_A[2]}.c, \ldots, h_{S_A[n]}.c)$. To determine $S_B \setminus S_A$, we first compute the vector $h_c = h_{S_B}.c - h_{S_A}.c$.

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

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

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

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

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

Algorithm 2: C2SS-BF Decode

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

We note that for every element $y \in S_A \triangle S_B$, Algorithm 2 requires that, in order to produce the estimate $y'$, we have to search through the elements in $S_B$. Assuming the elements $S_B$ are sorted, each search operation would require $O(\log(|S_B|))$ operations, and so the total complexity of Algorithm 2 is $O(d \log(|S_B|))$. 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.
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 $S_A \subseteq GF(2)^{131072}$ and Host $B$ has access to the set $S_B \subseteq GF(2)^{131072}$ where $|S_A| = 200$, $|S_B| = 200$. 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 $d = |S_A \triangle S_B|$. 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 $S_A, S_B$ given that $|S_A \triangle S_B| = d$. In other words, for every trial we attempted to compute $S_A \cup S_B$ 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 $S_A \setminus S_B$ on Host $A$ and a CBF was used to determine the set difference $S_B \setminus S_A$ on Host $B$. Then $S_A \setminus S_B$ was sent from Host $A$ to $B$ and similarly $S_B \setminus S_A$ 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 $2d$. The polynomial interpolation approach from [13] (like the approaches in [6], [10], [12]) require at least $2d$ 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.

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 $O(d \max(\log |S_A|, \log |S_B|))$ whereas the CBF approach in [9] had decoding complexity $O(|S_A| + |S_B|)$. Recall the method from [13] has decoding complexity $O(d^3)$, which in many cases, renders it impractical.

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 \( d = |S_A \triangle S_B| \). 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 round of communication required to determine \( S_A \cup S_B \) 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


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 \( S_B \) that hashes to all the same cells as an element in \( S_A \triangle S_B \). Second, the decoding can fail if a cell is hashed to \( q \) or more times by an element in \( S_A \triangle S_B \). Finally, the decoding can fail at step 28 if the following scenario holds: Suppose \( S' \) is a subset of \( S_A \triangle S_B \) and \( \mathcal{L} \) is the set of all cells hashed to by the elements of \( S' \). Then, Algorithm 2 can fail if for any \( \ell \in \mathcal{L} \), there are at least two elements in \( S' \) hash to \( \ell \). The three conditions are stated mathematically below:

1. \( \exists x \in S_A, \exists y \in S_A \triangle S_B \text{ where } j^k(x) = j^k(y) \).
2. \( \exists i, 1 \leq i \leq n \text{ where } |\{(x,j) \mid x \in S_A \triangle S_B, j \in \{1, \ldots, k\}, j^k(x) = i\}| \geq q \).
3. Suppose \( S' \subseteq (S_A \triangle S_B) \) and \( \mathcal{L} = \{f_j(x) \mid i \in \{1,2, \ldots, k\}, x \in S' \} \) and \( \forall \ell \in \mathcal{L}, \exists x, \exists y \in S' \text{ where } f_j(x) = f_j(y) \).

In the following, we refer to the first event listed as item 1) above as \( \xi_1 \), the second event as \( \xi_2 \), and the third event as \( \xi_3 \). For any event, \( \xi \), we let \( P(\xi) \) denote the probability the event \( \xi \) occurs. Let \( \xi \) denote the event that \( \mathcal{F} \neq S_A \triangle S_B \) 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. \( P(\xi_1) \leq d(1 - \frac{k^d}{d^k(k+1)^{k-1}})^{|S_B|} \).

Proof: For any \( y \in S_A \triangle S_B, x \in S_B \), \( P(f^k(x) = f^k(y)) = \left( \frac{k}{d(k+1)} \right)^k = \frac{k^d}{d^k(k+1)^{k-1}} \). Therefore, for \( y \in S_A \triangle S_B \),

\[
P(\exists x \in S_B : f^k(x) = f^k(y)) = (1 - \frac{k^d}{d^k(k+1)^{k-1}})^{|S_B|}. \]

Then, since \( |S_B| = d \) we have \( P(\xi_1) \leq d(1 - \frac{k^d}{d^k(k+1)^{k-1}})^{|S_B|} \) as desired.

The following corollary follows from Lemma 1.

Corollary 2. If \( |S_B| \leq k \frac{\log_2(\frac{1}{d})}{\log_2(1 - \frac{k^d}{d^k(k+1)^{k-1}})} \), then \( P(\xi_1) \leq d^{-k+1} \).

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

Lemma 2. If \( q \leq k \log_2(d) + e - 1 \), then \( P(\xi_2) \leq O(d^{-k+2}) \).

Proof: For some integer \( i \) where \( 1 \leq i \leq d(k+1) \) and any \( x \in S_A \triangle S_B \), let \( \mathcal{X} \) be a random variable that is equal to 1 when \( f_j(x) = i \) and 0 otherwise for \( j = \lfloor \frac{i + k}{k} \rfloor \). Then \( \mathcal{X} \) is a Bernoulli random variable with parameter \( p = \frac{k}{d(k+1)} \). Let \( \mathcal{X}_i \) be the random variable that has value \( \{x \in S_A \triangle S_B, j = \{1,2, \ldots, k\} : f_j(x) = i\} \). Notice that since \( \mathcal{X} \) is a Bernoulli random variable, \( \mathcal{X}_i \) is a Poisson random variable with mean \( \lambda = \frac{d^k}{d^k(k+1)^{k-1}} \). Applying the Chernoff bound (c.f. [5]), which states that \( P(\mathcal{X}_i \geq a) \leq e^{-a} M_{\mathcal{X}_i}(t) \) where \( M_{\mathcal{X}_i}(t) \) denotes the moment generating function for \( \mathcal{X}_i \), gives

\[
P(\mathcal{X}_i \geq q) \leq e^{-q} \leq e^{-q} e^{(e-1)} \text{ for } q = k \log_2(d) + e - 1, \text{ given that } d \leq \frac{1}{e} \text{ and } k \log_2(d) + e - 1 \leq d^{k-1}. \]

Letting \( t = 1 \) and substituting \( q = k \log_2(d) + e - 1 \), gives that

\[
P(\mathcal{X}_i \geq q) \leq \frac{1}{d^{k-1}} \text{ for } q = k \log_2(d) + e - 1 \leq d^{k-1}. \]

Since there are \( d(k+1) \) possibilities for the index \( i \), the probability that there exists an \( i \) where \( |\{x \in S_A \triangle S_B, j = \{1,2, \ldots, k\} : f_j(x) = i\}| \geq q \) is at most \((k+1)d \cdot d^{-k+1} = O(d^{-k+2}) \) 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 \( |S_B| \leq k \frac{\log_2(\frac{d}{k})}{\log_2(1-\frac{k}{2^k}d^k)} \) 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 \( S_A, S_B \subseteq GF(2)^b \) and the size of the symmetric difference is \( d = |S_A \triangle S_B| = |(S_A \setminus S_B) \cup (S_B \setminus S_A)| \). Let \( k \) be a positive integer and suppose \( t = |S_B| \geq |S_A| \) and \( t \leq k \frac{\log_2(\frac{1}{d})}{\log_2(1-\frac{1}{2^{k+1}})} \), 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 \( O(1 - (1 - \frac{1}{d(k+1)})^kd - (kd/2)^k) \) 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 \( O(|S_A| + |S_B|) \) which, for the case where \( |S_A| + |S_B| \) is large, may be prohibitively expensive.