Synchronization from Insertions and Deletions Under a Non-Binary, Non-Uniform Source

Nicolas Bitouzé and Lara Dolecek
Electrical Engineering Department
University of California, Los Angeles
Los Angeles, USA
Email: bitouze@ucla.edu, dolecek@ee.ucla.edu

***Abstract*—We study the problem of synchronizing two files X and Y at two distant nodes A and B that are connected through a two-way communication channel. We assume that file Y at node B is obtained from file X at node A by inserting and deleting a small fraction of symbols in X. More specifically, we consider the case where X is a non-binary non-uniform string, and deletions and insertions happen uniformly with rates β_d and β_i, respectively. We propose a synchronization protocol between node A and node B that needs to transmit $O(\frac{q}{H_2}(\beta_d+\beta_i)n\log\frac{1}{\beta_d+\beta_i})$ bits (where n is the length of X, q is the alphabet size and H_2 is the collision entropy of X) and reconstructs X at node B with error probability exponentially low in n. This protocol readily generalizes the recent result by Tabatabaei Yazdi and Dolecek that dealt with synchronization from binary uniform source and under only deletion errors.**

I. Introduction

Motivated by the pervasive use of file synchronization in modern data storage technologies, in this work we seek to develop a synchronization protocol that is more efficient than the existing algorithms. In particular, the popular RSYNC method can be in general very inefficient and the number of transmitted bits can be exponentially larger than the optimal number.

Our starting point is an information-theoretically oriented scheme recently developed in [1]. In [1], a synchronization protocol that synchronizes an altered copy of a binary file with the original version of that file was proposed. In this scheme, the owner of the altered file requests additional information from the owner of the original file to ensure proper synchronization. It was assumed that the altered copy was obtained from the original copy by i.i.d. deletions at the bit-level and that the original file was generated from an i.i.d. uniform binary source. It was then shown that the rate of the proposed scheme asymptotically matches the optimal rate for this channel, developed earlier in [2]. That is, in the scheme of [1], the number of bits needed to synchronize two files can be kept very small while achieving exponentially low probability of error.

There are many practical scenarios where the files cannot be modeled as binary and uniform. For example, a file is usually not structured as bits, but by bytes or by even longer atomic elements. If the source is a text file, not only are some characters more frequent than others, but there is a large autocorrelation within the file. Additionally, some symbols may be inserted as well as deleted. As a result, our objective is to suitably generalize the scheme in [1], while maintaining low cost of transmission and low error of mis-synchronization.

Specifically, our model encapsulates the following generalizations of the model in [1]:

1) We consider errors as being insertions or deletions instead of being restricted to deletions only,
2) We consider non-binary source symbols,
3) We allow the source symbols to have an arbitrary distribution; uniform distribution is then a special case.

However, this development is just a first step toward properly modeling real-case scenarios. In the example of text files, the symbols are not independent, and insertions and deletions are likely to happen in bursts. This more general setup is the subject of our future work.

The rest of the paper is organized as follows. In Section II we outline the overall synchronization protocol. Necessary notation and background results are presented in Section III. Two key components of our synchronization protocol, the matching module and the edit recovery module, are discussed in detail in Sections IV and V, respectively. Section VI concludes the paper.

II. The Synchronization Protocol

In [1], the following setup is considered: two distant nodes A and B are connected by a low-bandwidth high-latency network. A contains a file X which is a uniform i.i.d. binary string of length n, and B contains a file Y of length n' that is obtained by deleting bits of X independently with probability $\beta \ll 1$.

We consider a generalized setting in which the file $X = X_1, \ldots, X_n$ is i.i.d. on alphabet $\mathcal{X} = \{0, \ldots, Q-1\}$, where for all $1 \le t \le n$, X_t's are distributed according to $\mu(x)$. For simplicity, we consider Q to be a power of two, say $Q = 2^q$. Insertions and deletions occur respectively with probability β_i and β_d. Let us define an *edit pattern* $E = E_1, \ldots, E_n$ as a string in $\{-1, 0, 1\}^n$ such that Y is obtained from X in the following way: for t from 1 to n,

- if $E_t = 0$, transmit X_t,
- if $E_t = -1$, delete (do not transmit) X_t,
- if $E_t = 1$, transmit X_t, then insert (transmit) a new symbol of $\mathcal{X}$ drawn with distribution $\mu(x)$.

For instance, consider X and Y defined over a quaternary alphabet, $X = 0\underset{D}{0}12213\underset{D}{3}10$ and $Y = 012\underset{I}{0}2\underset{I}{3}1\underset{I}{0}310$. Here Y is derived from X by 2 deletions and 3 insertions where deleted (inserted) symbols are denoted by D (I). The edit pattern is thus $E = (0, -1, 0, 1, 1, 1, 0, -1, 0, 0)$. Node B aims to synchronize its file Y with the (original) file X by requesting carefully chosen additional information from A over a reliable (and expensive) channel. Clearly, A can simply resend the entire file X but this would be extremely wasteful. We show that in fact with only a logarithmic additional cost, B can synchronize the file, provided that the additional transmission spans carefully chosen short substrings.

We now describe our synchronization protocol, which has the same overall structure as in the scenario with uniform binary source and deletions only, previously considered in [1]. First, we split X into a concatenation

$$X = D_1, P_1, D_2, P_2, \dots, D_{k-1}, P_{k-1}, D_k \tag{1}$$

of substrings P_i and D_i, respectively referred to as *pivot strings* (of small length $L_P \triangleq |P_i|$) and *data strings* (of length $L_D \triangleq |D_i|$). Both A and B know the values of L_P and L_D. A then sends all the pivot strings P_i to B. The decoding process at B is split into three tasks each solved using an individual module. The overall diagram is shown in Figure 1.

1) The *matching* module attempts to find the exact copies of each pivot string P_i in Y. Because edits may have modified a pivot, we may only be able to match a subset $\{P_{i_1}, \dots, P_{i_{k'-1}}\}$ of these. Based on their positions, the matching module divides Y into

$$Y = \bar{S}_1, P_{i_1}, \bar{S}_2, P_{i_2}, \dots, \bar{S}_{k'-1}, P_{i_{k'-1}}, \bar{S}_{k'}. \tag{2}$$

2) The *edit recovery module* then transmits the indices $\{i_1, \dots, i_{k'-1}\}$ of the matched pivots back to A. On A's side, X can now be divided into

$$X = S_1, P_{i_1}, S_2, P_{i_2}, \dots, S_{k'-1}, P_{i_{k'-1}}, S_{k'}. \tag{3}$$

If two successive pivots $P_{i_{j-1}}$ and P_{i_j} are both correctly matched in Y, then $\bar{S}_j$ can be derived from S_j by inserting and deleting a certain number of symbols. In our setup the expected number of such edits is small. Under this assumption, A and B use a modified version of the synchronization protocol of Venkataramanan *et al.* [3] to recover, for each j, an estimate $\tilde{S}_j$ of S_j at B. This modified version will be described and analyzed in Section V. Once this step is complete, B has a first estimate

$$\tilde{X} = \tilde{S}_1, P_{i_1}, \tilde{S}_2, P_{i_2}, \dots, \tilde{S}_{k'-1}, P_{i_{k'-1}}, \tilde{S}_{k'} \tag{4}$$

of X that it transmits to the next module.

3) The *LDPC decoder module*[1] performs the last step, in which we attempt to recover from potential errors made by the first two modules. The input is $\tilde{X}$ and the output is referred to as $\hat{X}$. The first module may have erroneously matched P_{i_j} at the wrong place. The substrings $\bar{S}_j$ and $\bar{S}_{j+1}$ may therefore not be realizable by inserting and deleting a small number of symbols in S_j and S_{j+1} respectively, which may in turn make the estimates $\hat{S}_j$ and $\hat{S}_{j+1}$ different from S_j and S_{j+1}, respectively. The second module may also introduce errors due to its reliance on hash functions in the sub-protocol adopted from [3] to test for the equality of two strings. However, the length of $\tilde{X}$ is the same as that of X, and if we assume that the error probabilities of the first two modules are such that $\Pr\{\tilde{S}_j \neq S_j\} \leq \zeta$, then we also have $\Pr\{\tilde{X}_i \neq X_i\} \leq \zeta$. It therefore suffices to model $\tilde{X}$ as the output of a Q-ary Symmetric Channel with error probability at most ζ ($\zeta/(Q-1)$ per error transition), and recover from potential errors using an additive-error-correcting code. If A sends a sufficient number of parity-check bits of a binary systematic LDPC code (considering Q-ary symbols as the concatenation of $q = \log Q$ bits[2]), it is possible to guarantee that the error probability after decoding will be as low as $\Pr\{\hat{X}_i \neq X_i\} \leq 2^{-\Omega(n)}$.

[1]In principle, any suitably implemented high-performance systematic substitution-error-correcting code can be used in this module; an LDPC code is a clear choice.

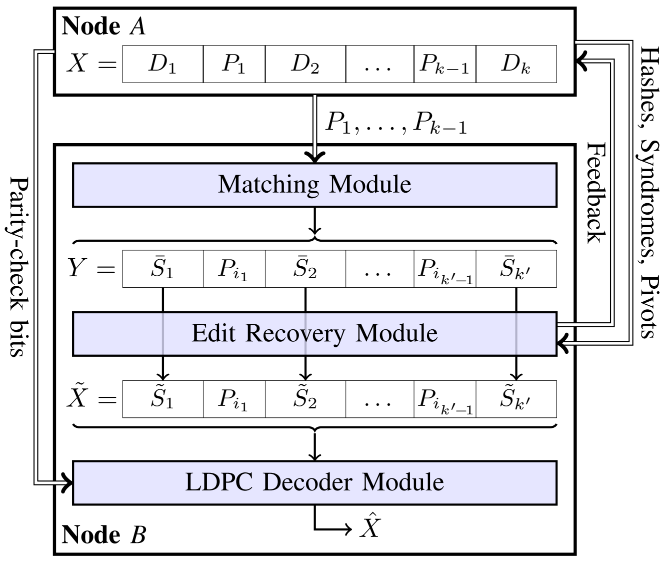
Node A
$X =$ D_1 P_1 D_2 $\dots$ P_{k-1} D_k
$P_1, \dots, P_{k-1}$
Matching Module
$Y =$ $\bar{S}_1$ P_{i_1} $\bar{S}_2$ $\dots$ $P_{i_{k'-1}}$ $\bar{S}_{k'}$
Edit Recovery Module
$\tilde{X} =$ $\tilde{S}_1$ P_{i_1} $\tilde{S}_2$ $\dots$ $P_{i_{k'-1}}$ $\tilde{S}_{k'}$
LDPC Decoder Module
$\hat{X}$
Node B
Parity-check bits
Feedback
Hashes, Syndromes, Pivots

Fig. 1. Overview of the Synchronization Protocol. The communications between A and each of the three modules at B are represented by doubled arrows. The edit recovery protocol involves multiple rounds, and after each round, B informs A of what information it needs for the next round (see Section V, and Fig. 2).

Working in a model that is more general than that of [1] mainly affects the synchronization protocol in the choice of the length L_P of the pivots, the algorithm used to match the pivots in Y, and the adapted protocol used by the edit recovery module. These changes are non-trivial and constitute our contribution in the remainder of the paper.

III. Notation and Preliminary Results

We introduce some notation and tools that we need to describe and prove our results. First, we denote by $X_t^{t'}$ the substring $X_t, \dots, X_{t'}$ of X from indices t to t'. Because we do not consider uniform sources anymore, we need an information-theoretic tool to measure the likelihood of two independent substrings being equal (our protocol relies on matching pivots in X and Y):

[2]All logarithms in this paper are base-2 logarithms.

Definition 1. (Rényi [4]) The *collision entropy* of a discrete random variable $Z \sim \mu(z)$ is defined by

$$H_2(Z) \triangleq -\log \mathbb{E}(\mu(z)) = -\log \sum \mu^2(z). \tag{5}$$

The collision entropy is related to the probability that there is a collision between two i.i.d. samples $Z, Z' \sim \mu(z)$ by $\Pr\{Z = Z'\} = 2^{-H_2(Z)}$. In our context, because X is i.i.d., $H_2(X_t)$ does not depend on t, and we therefore write it as H_2. Because X is i.i.d., the collision probability between two distinct substrings of X of equal length l is $\Pr\{X_t^{t+l-1} = X_{t'}^{t'+l-1}\} = 2^{-lH_2}$.

We also need the following concentration theorem:

Theorem 1. *(Hoeffding [5]) Let $Z_1, \dots, Z_l$ be i.i.d. random variables with expected value M that take values in an interval of length I. Then for every $\epsilon > 0$,*

$$\Pr\left\{\left|\sum_{t=1}^{l} Z_t - lM\right| \geq l\epsilon\right\} \leq 2\exp\left(-\frac{2l\epsilon^2}{I^2}\right). \tag{6}$$

IV. The Matching Module

We now follow Section III of [1] in our generalized framework. Our first modification to the matching module is our choice of the pivot length L_P: we set $L_P = O(\frac{1}{H_2}\log\frac{1}{\beta})$, to conserve the same probability of collision between a pivot and another substring as in [1]. Here, $\beta \triangleq \beta_d + \beta_i$ so that β is the probability $\Pr\{E_t \neq 0\}$ that there is an edit at index t. We keep $L_D = \frac{1}{\beta}$ as in [1], so that the number k of data strings remains $\approx n\beta$. The proofs of Lemmas 1 to 5 follow the same sketches as their counterparts in [1] and are therefore omitted for the sake of brevity.

A. Good and Bad Matches

For a given pivot P_i, the matching module finds the list of all the substrings in Y that are equal to P_i. Let us first define the *good match of the symbol* X_t in X, under the assumption that $E_t \geq 0$ (X_t was not deleted), as the symbol $Y_{t'} = X_t$ that resulted from the transmission of X_t. (In the case of insertion, $Y_{t'}$ is followed by an inserted symbol). Let $\check{p}_i$ and $\hat{p}_i$ respectively denote the indices of the first and of last symbol of P_i. We now define a *good match of the pivot* P_i as a substring $Y_t^{t+L_P-1}$ of Y that such that

- $P_i = Y_t^{t+L_P-1}$,
- The edit pattern $E_{\check{p}_i}^{\hat{p}_i}$ corresponding to P_i has at most one non-zero element (P_i has at most one edit), and
- Either Y_t is the good match of $X_{\check{p}_i}$ or Y_{t+l-1} is the good match of $X_{\hat{p}_i}$.

Notice that it is possible that P_i has two good matches if there is an edit in P_i.

Lemma 1. *With probability $1 - \beta L_P + o(\beta)$, P_i has no edit and there is a good match for P_i within Y.*

The following result is shown by treating the single insertion and the single deletion cases, and using $L_P = O(\frac{1}{H_2}\log\frac{1}{\beta})$.

Lemma 2. *With probability $2\beta + o(\beta)$ there is a single edit within P_i and there is a good match for P_i within Y.*

Lemma 3. *With probability $o(\beta)$, P_i has at least two edits.*

We now define $R \triangleq 1 - L_P\beta + 2\beta$ as in [1] and use Lemmas 1 to 3 to get the following result.

Lemma 4. *The number of pivots with a good match in Y is on average $(R + o(\beta))k$.*

Using Theorem 1 and Lemma 4, we conclude that:

Lemma 5. *With probability $1 - 2^{-\Omega(n)}$, there are $(R+o(\beta))k$ pivots with a good match in Y.*

B. The Matching Graph

As in [1], we use a graph theoretic approach in the matching module. We define a graph $G(V, E)$, where the set V of vertices is built as in [1]: it is split into $k+1$ layers $\Lambda_0, \dots, \Lambda_k$ where for $1 \leq i \leq k-1$, each vertex v in layer Λ_i represents a match of pivot P_i in Y, and $\Lambda_0 = \{s\}$ and $\Lambda_k = \{f\}$ respectively represent the fact that the start of X matches the start of Y, and that the end of X matches the end of Y. We let $\check{v}$ and $\hat{v}$ denote respectively the first and last indices of the match of P_i in Y corresponding to vertex v, with $\hat{s} = 0$ and $\check{f} = |Y| + 1$. We also define *good* and *bad* vertices as vertices corresponding respectively to good and bad matches.

In [1], only deletions are considered. Therefore, if $i < j$ and $u \in \Lambda_i$ and $v \in \Lambda_j$ are good vertices, the distance $\mathrm{Dis}(u, v) \triangleq \check{v} - \hat{u} - 1$ is upper-bounded by the distance between the end of P_i and the beginning of P_j in X: $\mathrm{Dis}(u, v) \leq \mathrm{L}(j - i) \triangleq (j - i - 1)L_P + (j - i)L_D$, and this inequality defines the valid edges of G. However, in our context, if there are more insertions than deletions between P_i and P_j, the inequality is not satisfied.

Rather, let $\delta(u, v) = \mathrm{Dis}(u, v) - \mathrm{L}(j - i)$ denote the number of *net insertions* (the number of insertions minus the number of deletions) that must have occurred between P_i and P_j if u and v are good vertices. We consider the probability distribution for two good vertices u and v of $\delta(u, v)$ (neglecting the probability that good matches may have one edit):

$$\Pr\{\delta(u, v) = \delta\} = \Pr\left\{\sum_{t=\hat{p}_i+1}^{\check{p}_j-1} E_t = \delta\right\}. \tag{7}$$

As this probability depends only on $L = \mathrm{L}(j - i)$ and δ, we denote it by $\pi(L, \delta)$:

$$\pi(L, \delta) = \sum_{2d+\delta \leq L} \Pr\{d \text{ deletions and } d + \delta \text{ insertions occurred}\}. \tag{8}$$

We connect a vertex $u \in \Lambda_i$ to a vertex $v \in \Lambda_j$ if and only if $i < j$, and we assign to this connection a weight $\omega(u, v) = (\beta(L_P - 2))^{j-i} \cdot \pi(\mathrm{L}(j - i), \delta(u, v)) \cdot R$, where:

- $(\beta(L_P - 2))^{j-i}$ is the probability that there is no good vertex between layers i and j,
- $\pi(\mathrm{L}(j - i), \delta(u, v))$ is the probability that $\delta(u, v)$ net insertions occur over $\mathrm{L}(j - i)$ symbols,
- R is the probability that there is a good match for P_i in Y (we omit this term when $i = 0$ because it only represents the beginning of the file, and no actual pivot is sent).

Under this definition, the weight of a path from s to f is the probability that the edit pattern has a structure so that all vertices on the path are good vertices and no layer that contains good vertices is skipped.

We now state the claim central to this section.

Claim 1. *For a random file X and a random edit pattern E, taking $L_P = \frac{C}{H_2} \log \frac{1}{\beta}$ for a suitable constant $C > 0$, the shortest path from s to f has no less than $(R - \beta + o(\beta))k$ good vertices with probability at least $1 - 2^{-\Omega(n)}$.*

Here, we define the shortest path as the path that maximizes the product of the weights of its edges, so that the shortest path corresponds to the most likely edit pattern structure. To find this path in practice, it may be better to instead consider weights $\omega'(u, v) = -\log \omega(u, v)$, and use the classic version of Dijkstra's algorithm, to avoid precision problems.

The following list briefly describes our approach to prove Claim 1 (see [6]).

1) Consider the symmetric case where $\beta_i = \beta_d = \frac{\beta}{2}$.
2) Estimate the ratio $\frac{\pi(L,|\delta|+1)}{\pi(L,|\delta|)}$.
3) For two successive good vertices on the shortest path from s to f in G, use the above ratio to upper-bound the probability that there is an alternative path between these two good vertices that visits at least one intermediate good vertex.
4) Use the local result from item 3 along with Theorem 1 to show that globally, with high probability, the number of good vertices missed by the shortest path is small.
5) Use Lemma 5 to reach the statement.

V. The Edit Recovery Module

The deletion recovery module from [1] needs to be adapted to our setting. We first build an error-free, one-round communication protocol to fix a single edit between two strings S and $\bar{S}$ on alphabet $\mathcal{X}$, assuming that both the encoder and the decoder know the lengths of S and $\bar{S}$. (Note that because this protocol is error-free, it does not actually depend on the distributions of the string S and of the edit pattern, as long as every possible string and edit pattern have non-zero probability). Using this result, we then define a protocol to recover from multiple edits using hashes, following the ideas from [3]. This protocol is not error-free because of the non-zero collision probability of the hash function, and its parameters do depend on the distribution of the strings and edits.

A. Synchronizing from a single edit

We adapt the protocol of Section III from [3] to the non-binary Varshamov-Tenengolts codes (*VT codes*) [7].

Definition 2. We call the (non-binary) *VT-syndrome* of a string S of length L the pair $\mathrm{VTS}(S) = (a, b) \in \mathcal{X} \times \{0, \dots, L-1\}$ defined by

$$a \equiv \sum_{t=1}^{L} S_t \mod Q, \quad b \equiv \sum_{t=1}^{L} (t-1)\alpha_t \mod L, \qquad (9)$$

where $\alpha_1 = 1$ and for $1 < t \leq L$, $\alpha_t = \begin{cases} 1 & S_t \geq S_{t-1}, \\ 0 & S_t < S_{t-1}. \end{cases}$

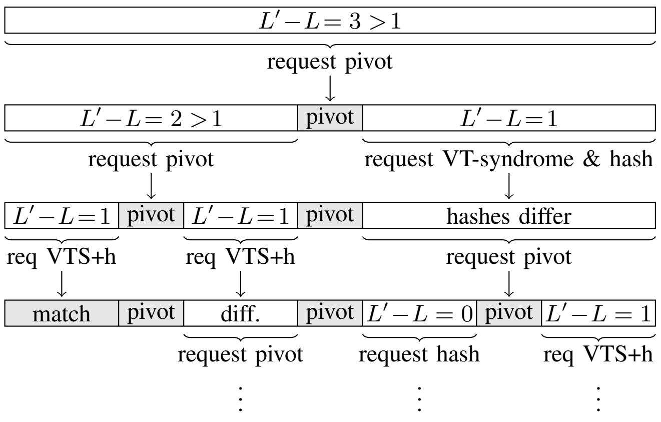

Fig. 2. Example of a run on the edit recovery module. At the first iteration (top line), because $L' - L > 1$, B requests a central pivot from A. This pivot is matched on B, and the algorithm goes on recursively on both sides of the pivot. Parts of the string that are considered as synchronized are grayed out.

The following theorem describes the main result of [7]:

Theorem 2. *For all $(a, b) \in \mathcal{X} \times \{0, \dots, L-1\}$, the set $\{S \in \mathcal{X}^L \mid \mathrm{VTS}(S) = (a, b)\}$ of all strings of length L over $\mathcal{X}$ of VT-syndrome (a, b), is a code that corrects any single edit. We refer to it as the VT-code $\mathrm{VT}_a^b(L)$.*

We use the argument in [8] to derive a one-way synchronization protocol that corrects a single edit, using the fact that we have a family of single-edit-correcting codes that covers any string S (because for any string S, if $(a, b) = \mathrm{VTS}(S)$, then $S \in \mathrm{VT}_a^b(|S|)$). The idea behind this argument is that if A knows S and B knows $\bar{S}$, where $\bar{S}$ differs from S by a single edit, then A can send the VT-syndrome $(a, b) = \mathrm{VTS}(S)$ to B. B then knows that $S \in \mathrm{VT}_a^b(|S|)$ and inputs $\bar{S}$ to the decoder of this VT-code in order to recover S. Because $(a, b) \in \mathcal{X} \times \{0, \dots, |S| - 1\}$, the number of bits that this protocol needs to send is $q + \lceil \log |S| \rceil$ (where $q = \log Q$).

B. Synchronizing from several edits

The protocol for synchronizing from a single edit when the source is non-binary and non-uniform can be adapted to synchronize from multiple edits following the same ideas as in the uniform binary case presented in Section IV in [3]. To synchronize S (on A) with $\bar{S}$ (on B), A sends l_p central symbols of S, that B then tries to match in $\bar{S}$. These symbols are used as a pivot to recursively split the problem into smaller problems (on each side of the pivot) until we reach a situation in which the substrings respectively on A and on B either are equal, or only differ by a single edit. When the substrings differ by a single edit, the protocol from the previous subsection is used to correct it.

Determining whether two substrings S' and $\bar{S}'$ respectively of S and of $\bar{S}$ are equal, or that they differ by a single edit, is not trivial. A necessary condition for them to be equal is that $|S'| = |\bar{S}'|$, and a necessary condition for them to differ by a single edit is that $|S'| = |\bar{S}'| \pm 1$. When the first condition is met, A sends a hash $h(S')$ of S' to B, and B declares that $S' = \bar{S}'$ if $h(S') = h(\bar{S}')$: the expected behavior of these hashes is such that the probability that two distinct strings

S' and $\bar{S}'$ have the same hash is very small. We denote the length of the hashes by l_h (in bits). When $|S'| = |\bar{S}'| \pm 1$, A sends both the VT-syndrome of S' and its hash to B. B then compares hash the output $\hat{S}'$ of the VT-decoder to $h(S')$, and concludes that $\hat{S}' = S'$ if their hashes are equal. In both cases, if the hashes differ, we go back to splitting S' and $\bar{S}'$ around a central pivot and work on smaller strings. This protocol is represented in Fig. 2.

We fix a parameter $c > 1$, which in turn fixes the length of our central pivots to $l_p = \frac{c \log |S|}{H_2}$ (in symbols) and the length of the hashes to $l_h = cq \log(|S|)$ (in bits). Under these conditions, we assume the existence of a hashing function with collision probability $\frac{1}{|S|^c}$ (see, for instance, [9]). Our main theorem for this section follows Theorem 2 in [3].

Theorem 3. *Let S be an i.i.d. string of length L over $\mathcal{X}$, and let H_2 be the collision entropy of S_t for all t. Let $\bar{S}$ be a string obtained from S through $\delta = o\left(\frac{L}{\log L}\right)$ edits. For any parameter $c > 1$, there exists an interactive synchronization protocol that produces an estimate $\hat{S}$ of S from $\bar{S}$, such that:*

(a) The probability that the protocol does not synchronize correctly is $\Pr\{\hat{S} \neq S\} \leq \frac{\delta \log L}{L^c}$,

(b) If $N_{A\to B}$ and $N_{B\to A}$ respectively denote the number of bits sent from A to B and from B to A, then their expected values are such that

$$\begin{aligned} &\mathbb{E}[N_{A\to B}] < \delta\left(4q + \left(2qc + 2\frac{qc}{H_2} + 4\right)\log L\right), \\ &\mathbb{E}[N_{B\to A}] < 10\delta. \end{aligned} \quad (10)$$

(c) The probability that the algorithm terminates after r rounds is at least $(1-(\delta+1)2^{-r})^\delta$. The expected number of rounds taken by the protocol to terminate is therefore approximately $4 + 2\log\delta$.

As the proof of Theorem 3 mostly follows the appendix of [3], we only give a sketch and emphasize on the differences introduced by our non-binary non-uniform context.

Proof of (a): This follows from the corresponding proof in [3] by adapting the length l_p of the center-pivot from $l_p = c\log L$ to $l_p = \frac{c \log L}{H_2}$, and accordingly upper-bounding the probability of error by the sum of the probability $p_{e,p}$ that an error occurred in pivot matching and the probability $p_{e,h}$ that an error occurred because of a hash collision. Here,

$$p_{e,p} \leq \frac{\text{Number of central pivots comparisons}}{(2^{H_2})^{l_p}}, \quad (11)$$

where the numerator is at most $\frac{\delta}{2}\log L$, and the denominator is $2^{H_2 \frac{c \log L}{H_2}} = L^c$, so that $p_{e,p} \leq \frac{\delta \log L}{2L^c}$. Similarly, we have

$$p_{e,h} \leq \frac{\text{Number of hash comparisons}}{L^c} = \frac{\delta \log L}{2L^c}. \quad (12)$$

Hence, $\Pr\{\hat{S} \neq S\} \leq p_{e,p} + p_{e,h} \leq \frac{\delta \log L}{L^c}$. □

Proof of (b): Let N_c be the number of times that a central pivot is sent. We decompose $N_{A\to B}$ as a sum of three contributions:

1) The number of central pivot bits sent: $N_c q l_p$, as each central pivot of length l_p can be represented with ql_p bits,
2) The number of hash bits sent: $N_c l_h$,
3) The number of non-binary VT-syndrome bits sent: at most $2N_c\delta(q + \log L)$.

As in [3], we can show that $\mathbb{E}(N_c) < 2\delta$. Substituting this inequality and the expressions for l_p and l_h in $N_{A\to B}$ completes the proof of the first half of (10).

To prove the second half of (10), we show that each time a central pivot is sent from A to B and B treats the substring on the left and on the right individually, B then has to instruct A about what to do next. Five scenarios are possible (on each side of the central pivot): 1) Continue splitting, 2) Send non-binary VT-syndrome, 3) Hashes are synchronized, 4) Request verification hash, 5) Request different central pivot (no match found). Twenty-five scenarios are therefore possible, so B needs to send $\lceil \log 25 \rceil = 5$ bits to A. This is true each time a central pivot is sent, therefore in total, the expected number of bits sent from B to A is $5\mathbb{E}(N_c) < 10\delta$. □

The proof of (c) is identical to its counterpart in [3].

VI. Summary and Conclusion

Combining the results of Sections IV and V with discussion from [1], it follows that the proposed scheme has exponentially low error probability, and needs to transmit $O(\frac{q}{H_2}\beta n \log\frac{1}{\beta})$ bits to synchronize, which asymptotically matches the optimal transmission rate up to a multiplicative constant.

Motivated by the problem of synchronizing files between remote computers, we presented an interactive scheme to synchronize from insertions and deletions when the source is non-binary and not assumed to be uniform, thus extending the results from [1]. Future work will focus on the cases where the errors occur in bursts and the source symbols are not i.i.d.

Acknowledgement

Research supported in part by NSF grant CCF-1150212 and a gift from Intel Corporation.

References

[1] S. Tabatabatei Yazdi and L. Dolecek, "Synchronization from deletions through interactive communication," in *IEEE ISTC*, 2012.
[2] N. Ma, K. Ramchandran, and D. Tse, "Efficient file synchronization: A distributed source coding approach," in *IEEE ISIT*, 2011.
[3] R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions," in *Allerton Conf.*, 2010.
[4] A. Rényi, "On Measures of Entropy and Information," in *Berk. Symp. Math., Stat. and Prob.*, 1960.
[5] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Amer. Stat. Assoc.*, vol. 58, pp. 13–30, Mar. 1963.
[6] N. Bitouzé and L. Dolecek, "On Synchronization from Insertions and Deletions." Technical Report, Loris Group, EE Department, UCLA, 2013.
[7] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion," *IEEE Trans. Info Theory*, vol. 30, pp. 766–769, Sep. 1984.
[8] A. Orlitsky, "Interactive communication: balanced distributions, correlated files, and average-case complexity," in *FOCS*, 1991.
[9] L. J. Carter and M. N. Wegman, "Universal classes of hash functions," 1977.