Coding for Skew-Tolerant Parallel Asynchronous Communications

Mario Blaum, *Senior Member, IEEE,* and Jehoshua Bruck, *Member, IEEE*

Abstract— **Consider a communication channel that consists of several subchannels transmitting simultaneously and asynchronously. As an example of this scheme, consider a board with several chips. The subchannels represent wires connecting between the chips where differences in the lengths of the wires might result in asynchronous reception. In current technology, the receiver acknowledges reception of the message before the transmitter sends the following message. Namely, pipelined utilization of the channel is not possible. The main contribution is a scheme that enables to transmit without an acknowledgement of the message, therefore enabling pipelined communication and providing a higher bandwidth. Moreover, the scheme allows for a certain number of transitions from a second message to arrive before reception of the current message has been completed, a condition that we call skew. Necessary and sufficient conditions for codes that can detect skew as well as for codes that are skew-tolerant, i.e., they can correct the skew and allow continuous operation, are derived. Codes have been constructed that satisfy the necessary and sufficient conditions, their optimality studied, and efficient decoding algorithms devised. To the best of the authors' knowledge, this is the first known scheme that permits efficient asynchronous communications without acknowledgement. Potential applications are in on-chip, on-board and board to board communications, enabling much higher communication bandwidth.**

Index Terms— **Parallel asynchronous communications, error-correcting codes, unordered codes, skew, pipelined channel.**

I. Introduction

A. Motivation and Background

CONSIDER a communication channel that consists of several subchannels transmitting simultaneously. As an example of this scheme consider a board with several chips where the sub-channels represent wires connecting between the chips and differences in the lengths of the wires might result in asynchronous reception. Namely, we would like to transmit a binary vector of length n using n parallel channels/wires. Every wire can carry only one bit of information. Each wire represents a coordinate of the vector to be transmitted. In this model, an electrical transition corresponds to a 1, while absence of a transition corresponds to a 0. The propagation delay in the wires varies. The problem is to find an efficient communication scheme that will be delay-insensitive.

Clearly, this problem is very common and arises in every system that incorporates transmission of information over parallel lines. Currently, there are two approaches for solving it in practice.

1) There is a clock that is shared by both the transmitter and the receiver and the state of the wire at the time of the clock represents the corresponding bit of information. This is a synchronous type of communication (which is not always feasible due to the difficulties in clock distribution and the fact that the transmitter might be part of an asynchronous system).
2) Asynchronous type of communications. Here, the idea is to send one vector at a time and have a handshake mechanism. Namely, the transmitter sends the following vector only after getting an acknowledgment that the current vector was completely received by the receiver.

A natural question with regard to the asynchronous type of communication is: how does the receiver know that the reception is complete? This problem was studied by Verhoeff [9]. He describes the forgoing physical model as a scheme in which the sender communicates with the receiver via parallel tracks by rolling marbles (that correspond to a logical 1) in the tracks. The assumption of rolling marbles is equivalent to the transmission of electrical signals. Although the marbles are sent in parallel, the channels are asynchronous. This means that marbles are received randomly and at different instants.

Before presenting Verhoeff's result, we introduce some notation. Let us represent the channels with the numbers $1, 2, \cdots, n$. After the mth transition has arrived, the receiver obtains a sequence $\hat{X}_m = x_1, x_2, \cdots, x_m$, where $1 \leq x_i \leq n$, and x_i represents the fact that the ith transition was received at the x_ith channel. The set $\{x_1, x_2, \cdots, x_m\}$ is the support (i.e., the set of nonzero coordinates) of a vector and determines uniquely a binary vector. From now on, $\hat{X}_m = x_1, x_2, \cdots, x_m$ denotes a sequence as defined above, and $X_m = \{x_1, x_2, \cdots, x_m\}$ the binary vector as defined by its support corresponding to sequence $\hat{X}_m$. For instance, assume that we have five channels and we receive the sequence $\hat{X}_4 = 2, 3, 2, 4$. This means, the first transition arrived in channel 2, the second one in channel 3, the third one in channel 2 and the fourth one in channel 4. The support of the corresponding binary vector is $X_4 = \{2, 3, 4\}$ (repeated arrivals count only once!) and the binary vector itself is $X_4 = 0\,1\,1\,1\,0$. In words, capital letters with a hat will denote sequences, while capital letters denote either vectors or their supports.

Manuscript received July 23, 1991. This work was presented in part at the IEEE International Symposium on Information Theory, Budapest, Hungary, June 24–28, 1991 and in part at the Third IMA Conference on Cryptography and Coding, Cirecester, England, December 1991. This work is dedicated to the memory of Joseph T. Downey.

The authors are with the IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120.

IEEE Log Number 9204356.

The following example shows the difficulty of choosing indiscriminate vectors for parallel asynchronous communications. Assume that a vector $X = 0110$ and a vector $Y = 0100$ are transmitted in some order. In the language of sets, we have $X = \{2, 3\}$ and $Y = \{2\}$. When the receiver gets a transition in channel number 2, it is not clear whether he just received Y or he should wait to get a transition in channel 3 (this will correspond to receiving X).

In general, the parallel asynchronous transmission model considered in [9], is the following: assuming that a vector X is transmitted, once reception has been completed, the receiver acknowledges receipt of the message. The next message is sent by the sender only after the receipt of the acknowledgement. The problem is finding a code C whose elements are messages such that the receiver can identify when transmission has been completed. It is easy to see, as shown in [9] and as suggested in the previous example, that the codes having the right property are the so called *unordered codes*, i.e., all its elements are unordered vectors (we say that two binary vectors are unordered when their supports are unordered as sets—one set is not a subset of the other).

One of the disadvantages of using the asynchronous type of communication is the fact that the channel is not fully utilized. Namely, there is at most one vector in the wires at any given time. This becomes very critical when the transmission rates are getting higher and lines are getting longer.

B. The New Paradigm

In this paper, we present a novel scheme that enables a pipelined utilization of the channel. In addition, our scheme has the important feature of not using a handshake (acknowledgement) mechanism. Hence, there is no need of communication between receiver and sender.

We note here that if one is ready to pay in performance, then a possible strategy, if acknowledgement of messages is not allowed, is that the sender will wait long enough between messages. So, if the sender sends a codeword X followed by a codeword Y, it will be very unlikely that a transition from Y will arrive before the reception of X has been completed. With this scheme, we can again use unordered codes as in [9].

The purpose of this paper is to study parallel asynchronous pipelined communication without acknowledgement. The main difficulty in this scheme is that a certain number of transitions from the second message might arrive before reception of the current message has been completed, a condition that we call *skew*.

We give next a precise mathematical definition of the concept of skew. Assume that a vector X is transmitted followed by other vectors. At reception, we obtain a sequence $\hat{Z} = x_1, x_2, \cdots, x_i, \cdots$. If there is no skew of X with respect to $\hat{Z}$, all the transitions from X arrive first and then the transitions from the next messages. However, this is not the case when there is skew.

Consider a transmitted vector X followed by some other vectors, giving a received sequence $\hat{Z}$. There are two parameters that are related to the skew. The first one, denoted $m(X; \hat{Z})$, denotes the index of the last transition in X before the occurrence of skew, i.e., the last transition in X before the arrival of either a transition not in X or a repeated arrival. The second one, denoted $r(X; \hat{Z})$, denotes the index of the last arrival in X. If there is no skew, $m(X; \hat{Z}) = r(X; \hat{Z})$. For instance, if $X = \{1, 2, 4\}$ and $\hat{Z} = 2, 3, 1, 1, 4, 5, \cdots$, we can see that $m(X; \hat{Z}) = 1$ and $r(X; \hat{Z}) = 5$. More precisely, if $\hat{Z} = x_1, x_2, \cdots, x_i, \cdots$ is a sequence, and we define the truncated sequence $\hat{Z}_j = x_1, x_2, \cdots, x_j$, and Z_j denotes the vector corresponding to $\hat{Z}_j$, $j \geq 1$, and X is a vector, then

$$m(X; \hat{Z}) = \min \{j: Z_j \subseteq X \text{ and } (x_{j+1} \notin X \text{ or } x_{j+1} \in Z_j)\}. \quad (1)$$

and

$$r(X, \hat{Z}) = \min \{j: X \subseteq Z_j\}. \quad (2)$$

Notice that, if $x_1 \notin X, m(X; \hat{Z}) = 0$. We are ready now to define the concept of skew of a vector X with respect to a sequence $\hat{Z}$.

Definition 1: Let X be a subset of $\{1, 2, \cdots, n\}$ (equivalently, X is a binary vector of length n). Let $\hat{Z} = x_1, x_2, \cdots, x_j, \cdots$ be a sequence whose elements are in $\{1, 2, \cdots, n\}$, $\hat{Z}_i = x_1, x_2, \cdots, x_i$ and Z_i the set corresponding to $\hat{Z}_i$. Let $m = m(X; \hat{Z})$ and $r = r(X; \hat{Z})$ be as defined by (1) and (2), respectively.

We say that the skew of X with respect to $\hat{Z}$ is equal to (l_1, l_2) (notation, $\mathcal{S}(X; \hat{Z}) = (l_1, l_2)$), if and only if

$$l_1 = |(Z_r - Z_m) \cap X| \quad \text{and} \quad l_2 = r - m - l_1,$$

where $|S|$ denotes the cardinality of a set S.

Let $\mathcal{S}(X; \hat{Z}) = (l_1, l_2)$. We say that $\mathcal{S}(X; \hat{Z})$ does not exceed (s_1, s_2), denoted $\mathcal{S}(X; \hat{Z}) \leq (s_1, s_2)$, if $l_1 \leq s_1$ and $l_2 \leq s_2$. Otherwise, we say that $\mathcal{S}(X; \hat{Z})$ exceeds (s_1, s_2) (notation, $\mathcal{S}(X; \hat{Z}) > (s_1, s_2)$).

Example 1: Assume that $X = 11000$ is transmitted followed by other vectors. As a set, $X = \{1, 2\}$. At reception, assume that the sequence $\hat{Z} = 231425\cdots$ is obtained. Equations (1) and (2) give $m = m(X; \hat{Z}) = 1$ and $r = r(X; \hat{Z}) = 3$, respectively. Therefore, we obtain $Z_m = Z_1 = \{2\}$ and $Z_r = Z_3 = \{1, 2, 3\}$, giving $Z_r - Z_m = Z_3 - Z_1 = \{1, 3\}$.

According to Definition 1, $l_1 = |(Z_r - Z_m) \cap X| = |\{1\}| = 1$ and $l_2 = r - m - l_1 = 1$, so $\mathcal{S}(X; \hat{Z}) = (1, 1)$.

Similarly, if we receive $\hat{Z} = 224135$, we can see that $m = m(X; \hat{Z}) = 1$ and $r = r(X; \hat{Z}) = 4$.

Now, we obtain $Z_m = Z_1 = \{2\}$ and $Z_r = Z_4 = \{1, 2, 4\}$, giving $Z_r - Z_m = Z_4 - Z_1 = \{1, 4\}$. According to Definition 1, $l_1 = |(Z_r - Z_m) \cap X| = |\{1\}| = 1$ and $l_2 = r - m - l_1 = 2$, so $\mathcal{S}(X; \hat{Z}) = (1, 2)$.

The next step is defining codes that can either detect or correct skew. Our approach to dealing with skew is to use coding theory methodology and identify the properties of a family of vectors (a code) that can handle the skew. In some applications, we might merely want to detect that skew has occurred, and then invoke a protocol that will halt transmission and allow for retransmission. Codes detecting skew are called *skew-detecting* codes.

Definition 2: Let t_1 and t_2 be nonnegative integers and let C be a code. We say that C is (t_1, t_2)-skew-detecting (SD) if, whenever a codeword X in C is transmitted followed

by other codewords giving a received sequence $\hat{Z}$, then, by examining $\hat{Z}$, the code will correctly decode X provided that $\mathcal{S}(X; \hat{Z}) = (0, 0)$ (i.e., no skew), and will detect the occurrence of skew when there is a repeated arrival as long as $(0, 0) < \mathcal{S}(X; \hat{Z}) \leq (t_1, t_2)$ (i.e., the skew does not exceed (t_1, t_2)).

Definition 2 states that the skew will be detected when a repeated arrival occurs, provided that the skew does not exceed (t_1, t_2). In a worst case situation, the detection of skew will occur after n arrivals.

In other applications, we might want to go further and correct the skew, allowing for continuous operation. Codes capable of correcting skew are called *skew-tolerant* codes. Formally,

Definition 3: Let t_1 and t_2 be nonnegative integers and let $\mathcal{C}$ be a code. We say that $\mathcal{C}$ is (t_1, t_2)-skew-tolerant (ST) if, whenever a codeword X in $\mathcal{C}$ is transmitted followed by other codewords, and $\hat{Z}$ is the received sequence, then, by examining $\hat{Z}$, the code will correctly decode X as long as $\mathcal{S}(X; \hat{Z}) \leq (t_1, t_2)$ (i.e., the skew does not exceed (t_1, t_2)).

We will present a decoding algorithm that will correct the skew when the last transition corresponding to X, i.e., x_r, arrives, where $r = r(X; \hat{Z})$ is given by (2).

We illustrate Definitions 2 and 3 with an example.

Example 2: Consider the following code: $\mathcal{C} = \{X; Y\}$, where $X = 10000$ and $Y = 01111$.

a) Code $\mathcal{C}$ is $(3, 3)$-SD. The decoder checks for the arrivals, and when either X or Y arrive they are decoded and the process restarted. Assume that X is transmitted first and then Y followed by other transmissions of either X of Y, giving a received sequence $\hat{Z}$. If the first transition arrives in channel 1, then we conclude that X has been transmitted. If the first transition does not arrive in channel 1, since $\mathcal{S}(X, \hat{Z}) \leq (3, 3)$, up to 3 transitions from Y may arrive before the transition in channel 1 arrives. Therefore, at least one transition remains in Y when the transition in channel 1 arrives. For each arrival x_j, the receiver checks if the received sequence Z_j coincides with either X or Y. Z_j cannot be X, since X has only one element. Z_j cannot be equal to Y neither, since 1 does not belong in Y, and 1 is received before the last transition in Y has arrived. Hence, the last transition in Y to arrive will be the 5th transition in the received sequence, corresponding to the complete set $\{1, 2, 3, 4, 5\}$. The sixth arrival will be a repeated arrival, detecting the occurrence of skew. Something similar occurs when Y is transmitted followed by X. If the skew exceeds $(3, 3)$, it may be undetected. In effect, assume, as before, that X is transmitted first but the received sequence is $\hat{Z} = 2\,4\,3\,5\,1\cdots$. According to Definition 1, $\mathcal{S}(X; \hat{Z}) = (1, 4)$. However, when $x_4 = 5$ arrives, the receiver will conclude (incorrectly) that Y was the transmitted codeword.

b) Code $\mathcal{C}$ is $(1, 2)$-ST. Assume that X is transmitted first and $\hat{Z}$ is the received sequence. If $\mathcal{S}(X; \hat{Z}) \leq (1, 2)$, one of the first three transitions arrives in channel 1. On the other hand, if Y is transmitted first, $\hat{Z}$ is the received sequence, and $\mathcal{S}(Y; \hat{Z}) \leq (1, 2)$, when a transition arrives in channel 1, only one transition may remain in Y. This means, the first three transitions do not arrive in channel 1. So, the receiver has a clear decoding strategy: it observes the arrival of the first three transitions; if one of them arrives in channel 1, then it decides that X has been transmitted; if neither of them arrives in channel 1, it decides that Y has been transmitted. As long as the skew between the transmitted codeword and the received sequence does not exceed $(1, 2)$, this strategy will successfully decode the transmitted codeword.

Although Example 2 is very simple, the reader is urged to comprehend it, since the general case involves a similar reasoning. The necessary and sufficient conditions for a code to be (t_1, t_2)-SD or ST, to be given in the next two sections, will allow to explain immediately why the code in Example 2 is $(3, 3)$-SD or $(1, 2)$-ST.

C. Contributions and Organization

Clearly, it is not enough to just define (t_1, t_2)-SD and (t_1, t_2)-ST codes. Our real goal is to identify the properties that characterize those codes and use them in order to construct the codes. Indeed, we were able to derive necessary and sufficient conditions for both (t_1, t_2)-SD and (t_1, t_2)-ST codes. These conditions are given using global distance properties between codewords. They fully characterize a set of vectors that can enable operation in the desired new paradigm.

We also provide efficient encoding and decoding algorithms both for the case of detection of skew and for the skew-tolerant case (continuous operation).

We have used the characterization theorems in order to construct efficient families, in terms of redundancy, of (t_1, t_2)-SD and ST codes. In particular, we have generalized the known construction by Berger [1] for unordering of vectors and constructed the so called error-correcting unordered codes (ECU's); these are codes that have both distance properties and are unordered. We also proved that the ECU's constructed out of Hamming codes and certain BCH codes are optimal in a certain sense.

In summary, we have used coding theory methodologies in order to create an efficient scheme for parallel pipelined asynchronous communication. As it turned out, new families of codes as well as new encoding and decoding algorithms are needed in order to address this problem.

The paper is organized as follows. In Section II, we prove the characterization theorem for (t_1, t_2)-SD codes and present an algorithm for detection of skew. In Section III, we prove the characterization theorem for (t_1, t_2)-ST codes and present a skew correction algorithm. In Section IV, we use the characterization theorems to construct efficient (t_1, t_2)-SD and ST codes. In Section V, we address the issue of the optimality of the codes obtained in Section IV.

II. Necessary and Sufficient Conditions and Decoding for (t_1, t_2)-SD Codes

In this section, we study (t_1, t_2)-SD codes (Definition 2). We give a characterization in terms of distance between code-

words, starting with necessary conditions and then proving that these conditions are also sufficient. The sufficient conditions are proven by providing a decoding algorithm, and showing that the decoding algorithm correctly decodes a codeword when there is no skew, and detects the presence of skew when this skew does not exceed (t_1, t_2).

Given two binary vectors X and Y of length n, we denote by $N(X, Y)$ the number of coordinates in which X is 1 and Y is 0 [2]. For example, if $X = 10110$ and $Y = 00101$, we have $N(X, Y) = 2$ and $N(Y, X) = 1$. Notice that $N(X, Y) + N(Y, X) = d_H(X, Y)$, where d_H denotes Hamming distance. In the language of sets, $N(X, Y) = |X - Y|$.

The following theorem gives necessary conditions for a code to be (t_1, t_2)-SD.

Theorem 1: Let $\mathcal{C}$ be a (t_1, t_2)-SD code and let X and Y be any two different codewords in $\mathcal{C}$. Let $t = \min\{t_1, t_2\}$ and let $T = \max\{t_1, t_2\}$. Then, at least one of the following two conditions occurs:

a) $\min\{N(X, Y), N(Y, X)\} \geq t + 1$
or
b) $\min\{N(X, Y), N(Y, X)\} \geq 1$ and
$\max\{N(X, Y), N(Y, X)\} \geq T + 1$.

Proof: Assume that for a given (t_1, t_2)-SD code $\mathcal{C}$ the conditions are not satisfied. Namely, there exist X and $Y \in \mathcal{C}$ such that

$$\min\{N(X,Y), N(Y,X)\} \leq t \tag{3}$$

and

$$\min\{N(X,Y), N(Y,X)\} = 0 \text{ or}$$
$$\max\{N(X,Y), N(Y,X)\} \leq T. \tag{4}$$

Given a set W, let $\hat{W}$ denote a sequence with the elements of W in some order (no repetitions).

Assume that $N(X, Y) = 0$. Therefore, $X \subseteq Y$. Let $V = Y - X$. Assume that the following sequence is received:

$$\hat{Z} = \hat{X}, \hat{V}, \hat{X}, \cdots.$$

Notice that both $\mathcal{S}(X; \hat{Z}) = (0, 0)$ and $\mathcal{S}(Y; \hat{Z}) = (0, 0)$. Therefore, there is no skew between either X of Y and the received sequence $\hat{Z}$, so the receiver cannot distinguish which codeword was transmitted first, whether X or Y. This contradicts the fact that $\mathcal{C}$ is (t_1, t_2)-SD, so, $\min\{N(X, Y), N(Y, X)\} \geq 1$, giving, from (3) and (4),

$$1 \leq \min\{N(X, Y), N(Y, X)\} \leq t \text{ and}$$
$$\max\{N(X, Y), N(Y, X)\} \leq T. \tag{5}$$

Without loss of generality, assume that $N(X, Y) \leq t_1$ and $N(Y, X) \leq t_2$.

Let $U = X \cap Y$, $V = Y - X$ and $W = X - Y$. Assume that the following is the received sequence:

$$\hat{Z} = \hat{U}, \hat{V}, \hat{W}, \hat{U}, \cdots. \tag{6}$$

Since $\mathcal{S}(Y; \hat{Z}) = (0, 0)$, the receiver will conclude, after arrival of the last transition in $\hat{V}$, that Y was transmitted.

However,

$$\mathcal{S}(X; \hat{Z}) = (|W|, |V|) = (N(X, Y), N(Y, X)) \leq (t_1, t_2),$$

therefore, it could have well happened that X was transmitted first and the skew with respect to the received sequence does not exceed (t_1, t_2), contradicting the fact that $\mathcal{C}$ is (t_1, t_2)-SD. □

Theorem 1 states necessary conditions for a code to be (t_1, t_2)-SD. It turns out that these conditions are also sufficient. The proof is based on showing that, given a code that satisfies the conditions, the following decoding algorithm correctly decodes the received sequence when there is no skew and detects an error provided that the skew does not exceed (t_1, t_2): given a received sequence $\hat{Z} = x_1, x_2, \cdots$, the receiver examines each arrival x_j checking for a codeword. If there is a repeated arrival, a skew error is detected and an error-detecting protocol is invoked. If a codeword is found, it is produced as output and the process is restarted. This can be formalized as follows.

Algorithm 1 (Decoding Algorithm for (t_1, t_2)-SD codes):
Let $\hat{Z} = x_1, x_2, \cdots, x_j, \cdots$ be a received sequence. Then,
SET the initial conditions as $X \leftarrow \emptyset$ and $j \leftarrow 0$.
START: SET $j \leftarrow j + 1$.
IF $x_j \in X$, THEN detect an error and stop.
ELSE, SET $X \leftarrow X \cup \{x_j\}$.
IF $X \in \mathcal{C}$, THEN output X, set $X \leftarrow \emptyset$ and
GO TO START.
ELSE, GO TO START.

Theorem 2: Let t_1 and t_2 be positive integers, and let $t = \min\{t_1, t_2\}$ and $T = \max\{t_1, t_2\}$. Let $\mathcal{C}$ be a code such that, for any pair of distinct codewords X, $Y \in \mathcal{C}$, at least one of the following two conditions occurs:

a) $\min\{N(X, Y), N(Y, X)\} \geq t + 1$
or
b) $\min\{N(X, Y), N(Y, X)\} \geq 1$ and
$\max\{N(X, Y), N(Y, X)\} \geq T + 1$.

Then, $\mathcal{C}$ is (t_1, t_2)-SD.

Proof: We prove the theorem by showing that Decoding Algorithm 1 will correctly decode any codeword X when no skew with the received sequence has occurred, and will detect the occurrence of skew not exceeding (t_1, t_2).

Assume that $X \in \mathcal{C}$ has been transmitted and the received sequence is $\hat{Z} = x_1, x_2, \cdots, x_j, \cdots$ (without loss of generality, we may assume that $\hat{Z}$ starts at x_1, and the codewords transmitted before X have been correctly decoded).

If $\mathcal{S}(X; \hat{Z}) = (0, 0)$ (i.e., no skew), then the algorithm will correctly decode X. Let $Z_j = \{x_1, x_2, \cdots, x_j\}$, $j \geq 1$. We show that, if $\mathcal{S}(X; \hat{Z}) \leq (t_1, t_2)$ and $Z_j \neq X$, then $Z_j \notin \mathcal{C}$. If $Z_j \subset X$ or $X \subset Z_j$ and $Z_j \neq X$, then $Z_j \notin \mathcal{C}$ since the code is unordered.

So, assume that X and Z_j are unordered and $Z_j \in \mathcal{C}$ for some $j \geq 1$. Since $\mathcal{S}(X, \hat{Z}) \leq (t_1, t_2)$, $N(X, Z_j) \leq t_1$ and $N(Z_j, X) \leq t_2$, contradicting conditions a) and b) in the theorem.

Since the algorithm correctly decodes X when $\mathcal{S}(X; \hat{Z}) = (0, 0)$ and never outputs a codeword when $0 < \mathcal{S}(X; \hat{Z}) \leq (t_1, t_2)$, then a repeated arrival indicates the occurrence of skew. In a worst case situation, after n transitions in different channels arrive, the $(n+1)$th transition will necessarily be a repeated arrival. Hence, $\mathcal{C}$ is (t_1, t_2)-SD. □

The next two corollaries follow immediately from the necessary and sufficient conditions.

Corollary 1: A code $\mathcal{C}$ is (t_1, t_2)-SD, if and only if it is also (t_2, t_1)-SD. Moreover, the decoding algorithm is the same for the two pairs of conditions.

Corollary 2: A code $\mathcal{C}$ is (t, t)-SD, if and only if, for any pair of codewords $X, Y \in \mathcal{C}$,

$$\min \{N(X, Y), N(Y, X)\} \geq 1$$

and

$$\max \{N(X, Y), N(Y, X)\} \geq t + 1.$$

The next example illustrates the necessary and sufficient conditions and the decoding algorithm for (t_1, t_2)-SD codes.

Example 3: Let $\mathcal{C} = \{U, V, W\}$, where

$$U = 00110 \leftrightarrow \{3, 4\},$$
$$V = 01101 \leftrightarrow \{2, 3, 5\},$$
$$W = 10001 \leftrightarrow \{1, 5\}.$$

Notice that $N(U, V) = 1$ and $N(V, U) = 2$, $N(U, W) = 2$ and $N(W, U) = 2$, and $N(V, W) = 2$ and $N(W, V) = 1$. According to Corollary 2, code $\mathcal{C}$ is $(1, 1)$-SD.

Assume now that the sender transmits U followed by V, then by W and then by other codewords, and the receiver obtains the following sequence:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	$\cdots$
$\hat{Z}$	=	3	4	5	2	1	3	5	$\cdots$.

Table I illustrates the algorithm, with the relevant parameters at each step. We observe that skew has occurred between V and W: the fifth arrival, in channel 1, corresponds to W, and the sixth arrival, in track 3, corresponds to V. Since this skew does not exceed $(1, 1)$, the decoding algorithm detects it. The detection occurs when there is a repeated arrival. In that case, a skew detection protocol may be invoked and transmission is temporarily halted.

Let us complete the example by showing that if the $(1, 1)$ constraints are exceeded, skew may be undetected. This example also illustrates the proofs of Theorems 1 and 2.

Assume, as before, that $U, V, W, \cdots$ are sent in this order and the received sequence is

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	$\cdots$
$\hat{Z}$	=	3	4	5	1	2	3	5	$\cdots$.

is received. We observe that the fourth arrival occurs in channel 1, which corresponds to W. When this occurs, there are two transitions left in Y, 2, and 3. Hence, once X has been received, the skew between Y and the received sequence is $(2, 1)$, which exceeds $(1, 1)$. Notice that the received sequence corresponds to a skew-free reception of $U, W, V, \cdots$, therefore the skew is undetected.

TABLE I

j	x_j	Repeated Arrival?	X	Output
0			$\varnothing$	
1	3	No	{3}	
2	4	No	{3, 4}	00110
			$\varnothing$	
3	5	No	{5}	
4	2	No	{2, 5}	
5	1	No	{1, 2, 5}	
6	3	No	{1, 2, 3, 5}	
7	5	Yes		Halt!
⋮	⋮	⋮	⋮	⋮

In the next section, we give the conditions for (t_1, t_2)-ST codes.

III. Necessary and Sufficient Conditions and Decoding for (t_1, t_2)-Skew-Tolerant Codes

The structure of this section is similar to the previous one: we characterize (t_1, t_2)-ST codes in terms of distance properties between codewords. We give necessary and sufficient conditions, and we prove sufficiency by providing a decoding algorithm and showing that this decoding algorithm can successfully decode transmitted codewords when the skew does not exceed (t_1, t_2). Codes that can correct, and not merely detect skew, have the advantage of allowing continuous operation: the system does not need to be halted in the presence of skew. Therefore, the skew is not noticed by the user. However, it will come as no surprise to the reader that ST codes require more redundancy than SD codes. Their decoding algorithm, also, is more complex.

The following theorem gives necessary conditions for a code to be (t_1, t_2)-ST.

Theorem 3: Let $\mathcal{C}$ be a (t_1, t_2)-ST code, and let X and Y be any pair of distinct codewords in C. Let $t = \min \{(t_1, t_2)\}$. Then, at least one of the following two conditions occurs:

a) $\min \{N(X, Y), N(Y, X)\} \geq t + 1$
   or
b) $\min \{N(X, Y), N(Y, X)\} \geq 1$ and
   $\max \{N(X, Y), N(Y, X)\} \geq t_1 + t_2 + 1$.

Proof: Assume that there exist X and $Y \in \mathcal{C}$ such that a) and b) are not satisfied. Without loss of generality, let $N(X, Y) \leq N(Y, X)$, so, negation of a) and b) gives

$$N(X, Y) \leq t \tag{7}$$

and

$$N(X, Y) = 0 \quad \text{or} \quad N(Y, X) \leq t_1 + t_2. \tag{8}$$

If $N(X, Y) = 0$, we get a contradiction as in the proof of Theorem 1. So, $N(X, Y) > 0$ and conditions (7) and (8) become

$$1 \leq N(X, Y) \leq t \quad \text{and} \quad N(Y, X) \leq t_1 + t_2. \tag{9}$$

Since $N(Y, X) = |Y - X| \leq t_1 + t_2$, we may assume that $Y - X = A_1 \cup A_2$, where $A_1 \cap A_2 = \emptyset$, $|A_1| \leq t_1$ and $|A_2| \leq t_2$.

As in the proof of Theorem 1, given a set W, we denote by $\hat{W}$ a sequence with the elements of W in some order. Let $U = X \cap Y$ and $W = X - Y$.

Assume that we receive a sequence $\hat{Z}$ as

$$\hat{Z} = \hat{U}, \hat{A}_2, \hat{W}, \hat{A}_1, \hat{U}, \cdots. \tag{10}$$

According to Definition 1,

$$\mathcal{S}(X; \hat{Z}) = (|W|, |A_2|) \leq (t, t_2) \leq (t_1, t_2),$$

therefore, since $\mathcal{C}$ is (t_1, t_2)-ST, X will be decoded as the transmitted codeword.

On the other hand,

$$\mathcal{S}(Y; \hat{Z}) = (|A_1|, |W|) \leq (t_1, t) \leq (t_1, t_2),$$

therefore, since $\mathcal{C}$ is (t_1, t_2)-ST, Y will be decoded as the transmitted codeword. This is a contradiction. □

It turns out that the necessary conditions in Theorem 3 are also sufficient for a code to be (t_1, t_2)-ST. As in the previous section, the proof is based on a decoding algorithm, to be given next. In Algorithm 1, we simply checked, for each arrival, if the received vector is a codeword. Now we need a more complicated check operation. We need to look into a window in the immediate past and eliminate possible sets of arrivals that could have been caused by skew. The skew may appear in two possible ways: through arrivals of transitions that do not belong in the current word, and through repeated arrivals. The decoder needs to keep track of both sets. We denote by A the set of transitions that do not belong in the current codeword, and by B the set of repeated arrivals. The possible sets A are eliminated from the current word until we obtain a codeword. If no codeword is obtained after all possible sets A have been considered, the decoder considers the next transition x_j and repeats the process.

The repeated transitions are ignored in the decoding of the current codeword, but they must be stored because they belong in the next transmitted codeword. Also, if the number of repeated transitions (i.e., the cardinality of set B) exceeds t_2, then, since the skew has exceeded the skew-tolerance of the code, the decoder halts transmission and may invoke a skew-detection protocol as in the previous section.

In order to consider the possible sets A, we need to look into a window with the last $t_1 + t_2$ elements, which are stored with their order of arrival. We denote these last $t_1 + t_2$ arrivals by $W_1, W_2, \cdots, W_{t_1+t_2}$, where each W_i is either the empty set or a set with one element. When the current transition, say, x_j arrives, once it is established that x_j is not a repeated arrival, it is stored as $W_{t_1+t_2}$. The original value of W_1 is eliminated and replaced by W_2, W_2 by W_3, etc. This way, we have stored the last $t_1 + t_2$ transitions in the order they have arrived.

Next we give formally the Decoding Algorithm.

Algorithm 2 (Decoding Algorithm for (t_1, t_2)-ST codes): Let the received sequence be $\hat{Z} = x_1, x_2, \cdots, x_j, \cdots$. Then:

SET the initial conditions $j \leftarrow 0$, $X \leftarrow \emptyset$, $B \leftarrow \emptyset$ and $W_l \leftarrow \emptyset$ for $1 \leq l \leq t_1 + t_2$.

START: SET $j \leftarrow j + 1$.

IF $x_j \in X$, THEN:

IF $|B| = t_2$ or $x_j \in B$, THEN declare an uncorrectable error and stop.

ELSE, SET $B \leftarrow B \cup \{x_j\}$ AND GO TO START.

ELSE, SET $X \leftarrow X \cup \{x_j\}$, $W_l \leftarrow W_{l+1}$ for $1 \leq l \leq t_1 + t_2 - 1$ and $W_{t_1+t_2} \leftarrow \{x_j\}$.

For each $A \subseteq W_{t_2-i+1} \cup W_{t_2-i+2} \cup \cdots \cup W_{t_1+t_2-1}$ of size $|A| = i$, and for each $0 \leq i \leq t_2$, DO:

IF $X - A \in \mathcal{C}$ for some A, THEN output $X - A$.

IF $A \cup B \in \mathcal{C}$, THEN output $A \cup B$, SET $X \leftarrow \emptyset$, $W_l \leftarrow \emptyset$ for $1 \leq l \leq t_1 + t_2$, $B \leftarrow \emptyset$ AND GO TO START.

ELSE, SET $X \leftarrow A \cup B$, $W_l \leftarrow \emptyset$ for $1 \leq l \leq t_1 + t_2$, $B \leftarrow \emptyset$ AND GO TO START.

ELSE, IF $X - A \notin \mathcal{C}$ for any A, GO TO START.

Notice that Algorithm 2 performs at most $\sum_{i=0}^{t_2} \binom{t_1+i-1}{i}$ checks for each received transition. This is a fixed number depending only on t_1 and t_2, so the algorithm has low complexity.

Before proving that the conditions in Theorem 3 are also sufficient, we give an example to familiarize the reader with Algorithm 2.

Example 4: Let $\mathcal{C} = \{U, V, W\}$, where

$$U = 0001100 \leftrightarrow \{4, 5\}$$
$$V = 0110000 \leftrightarrow \{2, 3\}$$
$$W = 1100111 \leftrightarrow \{1, 2, 5, 6, 7\}.$$

Notice that $N(U, V) = 2$ and $N(V, U) = 2$, $N(U, W) = 1$ and $N(W, U) = 4$, and $N(V, W) = 1$ and $N(W, V) = 4$. As will be shown in Theorem 4, code $\mathcal{C}$ is $(1, 2)$-skew-tolerant.

Assume that the receiver obtains the sequence

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	$\cdots$
$\hat{Z}$ =	5	2	7	1	4	5	6	4	2	3	5	$\cdots$.

Table II implements Algorithm 2, with the relevant parameters at each step.

Theorem 4: Let t_1 and t_2 be positive integers and $t = \min\{t_1, t_2\}$. Let $\mathcal{C}$ be a code such that, for any $X, Y \in \mathcal{C}$, at least one of the following two conditions occurs:

a) $\min\{N(X, Y), N(Y, X)\} \geq t + 1$
or
b) $\min\{N(X, Y), N(Y, X)\} \geq 1$ and $\max\{N(X, Y), N(Y, X)\} \geq t_1 + t_2 + 1$.

Then, $\mathcal{C}$ is (t_1, t_2)-ST.

Proof: We prove the theorem by showing that Decoding Algorithm 2 will correctly decode any transmitted codeword Y when the skew between Y and a received sequence $\hat{Z}$ does not exceed (t_1, t_2).

Moreover, if $m = m(Y; \hat{Z})$ and $r = r(Y; \hat{Z})$ are defined by (1) and (2), respectively, and $\mathcal{S}(Y; \hat{Z}) \leq (t_1, t_2)$, we

TABLE II

j	x_j	B	X	W_1	W_2	W_3	A	$X-A$	Output
0		∅	∅	∅	∅	∅			
1	5	∅	{5}	∅	∅	{5}			
							∅	{5}	
2	2	∅	{2, 5}	∅	{5}	{2}			
							∅	{2, 5}	
							{5}	{2}	
3	7	∅	{2, 5, 7}	{5}	{2}	{7}			
							∅	{2, 5, 7}	
							{2}	{5, 7}	
							{2, 5}	{7}	
4	1	∅	{1, 2, 5, 7}	{2}	{7}	{1}			
							∅	{1, 2, 5 7}	
							{7}	{1,2,5}	
							{2, 7}	{1, 5}	
5	4	∅	{1, 2, 4, 5, 7}	{7}	{1}	{4}			
							∅	{1, 2, 4, 5, 7}	
							{1}	{2, 4, 5, 7}	
							{1, 7}	{2, 4, 5}	
6	5	{5}							
7	6	{5}	{1, 2, 4, 5, 6, 7}	{1}	{4}	{6}			
							∅	{1, 2, 4, 5, 6, 7}	
							{4}	{1, 2, 5, 6, 7}	1100111
		∅	∅	∅	∅	∅			0001100
8	4	∅	{4}	∅	∅	{4}			
							∅	{4}	
9	2	∅	{2, 4}	∅	{4}	{2}			
							∅	{2, 4}	
							{4}	{2}	
10	3	∅	{2, 3, 4}	{4}	{2}	{3}			
							∅	{2, 3, 4}	
							{2}	{3, 4}	
							{2, 4}	{3}	
11	5	∅	{2, 3, 4, 5}	{2}	{3}	{5}			
							∅	{2, 3, 4, 5}	
							{3}	{2, 4, 5}	
							{2, 3}	{4, 5}	0001100
		∅	∅	∅	∅	∅			0110000
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

show that Y will be decoded when x_r arrives, i.e., the last transition in Y. To complete the proof, we need to show that the decoder does not produce any codeword different from Y for any arrival x_j, $j \leq r$.

Since Algorithm 2 ignores repeated arrivals in the decoding of the present codeword Y, without loss of generality, we assume that $\hat{Z}_r$ does not contain repeated arrivals. Assume that $\mathcal{S}(Y; \hat{Z}) = (l_1, l_2) \leq (t_1, t_2)$. Let $C = Z_r - Y$, i.e., $Y = Z_r - C$. Hence, since there are no repeated arrivals, $|C| = l_2$. We have to show that $Z_r - C$ is a possible outcome of the decoding algorithm, i.e., $C \subseteq W_{t_2-l_2+1} \cup W_{t_2-l_2+2} \cup \cdots \cup W_{t_1+t_2-1}$. Notice that, by the definition of the W_i's in the decoding algorithm,

$$W_{t_2-l_2+1} \cup W_{t_2-l_2+2} \cup \cdots \cup W_{t_1+t_2-1} = \{x_{r-t_1-l_2+1}, x_{r-t_1-l_2+2}, \cdots, x_{r-1}\}.$$

Since $C \subseteq Z_{r-1} - Z_m = \{x_{m+1}, x_{m+2}, \cdots, x_{r-1}\}$, it is enough to show that $Z_{r-1} - Z_m \subseteq \{x_{r-t_1-l_2+1}, x_{r-t_1-l_2+2}, \cdots, x_{r-1}\}$, i.e.,

$$m + 1 \geq r - t_1 - l_2 + 1. \tag{11}$$

This last inequality is equivalent to $r - m \leq t_1 + l_2$. But this is true, since $r - m = l_1 + l_2$ and $l_1 \leq t_1$, therefore (11) holds.

So, $Y = Z_r - C$ is a possible outcome of the decoding algorithm. In order to complete the proof, we need to show that, for any $j \leq r$ and $Z_j - A \neq Y$, where A is as defined in the decoding algorithm, then $Z_j - A \notin \mathcal{C}$. In other words, $Y = Z_r - C$ is the only possible outcome of the decoding algorithm.

If $j \leq m$, $Z_j - A \subseteq Z_j \subseteq Z_m \subseteq Y$. Since code $\mathcal{C}$ is unordered, $Z_j - A \notin \mathcal{C}$. So, assume that $Z_j - A \in \mathcal{C}$, where

$$m + 1 \leq j \leq r,$$
$$|A| = i \leq t_2,$$
$$A \subseteq \{x_{j-t_1-i+1}, x_{j-t_1-i+2}, \cdots, x_{j-1}\} \quad \text{and}$$
$$Z_j - A \neq Y.$$

We will compute $N(Z_j - A, Y)$ and $N(Y, Z_j - A)$ and show that the values we obtain contradict the fact that both Y and $Z_j - A$ are in code $\mathcal{C}$ (by contradicting the conditions in the theorem).

There are two possibilities for A: either $A \cap Z_m = \emptyset$ or $A \cap Z_m \neq \emptyset$. Assume first that $A \cap Z_m = \emptyset$. Therefore, $Y - (Z_j - A) \subseteq Y \cap (Z_r - Z_m)$, giving $N(Y, Z_j - A) = |Y - (Z_j - A)| \leq |Y \cap (Z_r - Z_m)| = l_1$, i.e.,

$$N(Y, Z_j - A) \leq t_1. \tag{12}$$

On the other hand, $N(Z_j - A, Y) \leq |Z_r - Y|$. Hence,

$$N(Z_j - A, Y) \leq t_2. \tag{13}$$

Inequalities (12) and (13) contradict the hypothesis, so $A \cap Z_m = \emptyset$ cannot hold.

Now assume that $A \cap Z_m \neq \emptyset$. Then, $A = A_1 \cup A_2$, where $A_1 = A \cap Z_m$ and $A_2 = A - A_1$. Let $|A_1| = i_1$ and $|A_2| = i_2$, so $i = i_1 + i_2$.

First, we notice that, since $\mathcal{S}(Y; \hat{Z}) = (l_1, l_2) \leq (t_1, t_2)$, by Definition 1 of skew,

$$\begin{aligned} N(Z_j - A, Y) &\leq N(Z_j, Y) \leq N(Z_r, Y) \\ &= |Z_r - Y| = |C| = l_2 \leq t_2. \end{aligned} \tag{14}$$

Since, by the decoding algorithm, $A \subseteq \{x_{j-t_1-i+1}, x_{j-t_1-i+2}, \cdots, x_{j-1}\}$, then $A_1 \subseteq \{x_{j-t_1-i+1}, x_{j-t_1-i+2}, \cdots, x_m\}$. Therefore,

$$\begin{aligned} i_1 = |A_1| &\leq |\{x_{j-t_1-i+1}, x_{j-t_1-i+2}, \cdots, x_m\}| \\ &= m + t_1 + i - j. \end{aligned}$$

This gives the inequality

$$j - m \leq t_1 + i_2. \tag{15}$$

Now, observe that $N(Z_j - A, Y) = |(Z_j - A) - Y| = |(Z_j - Y) - A| \leq |(Z_j - Z_m) - A_2| = j - m - i_2$. Since, by inequality (15), $-i_2 \leq t_1 + m - j$, replacing the value of $-i_2$, we obtain

$$N(Z_j - A, Y) \leq t_1. \tag{16}$$

From inequalities (14) and (16), we obtain

$$N(Z_j - A, Y) \leq t, \tag{17}$$

where $t = \min\{t_1, t_2\}$.

Now, notice that $N(Y, Z_j - A) \leq |(Z_r - Z_m) \cap Y| + |A|$. By Definition 1, $|(Z_r - Z_m) \cap Y| = l_1 \leq t_1$. Also, we have $|A| = i \leq t_2$, so we obtain the inequality

$$N(Y, Z_j - A) \leq t_1 + t_2. \tag{18}$$

But inequalities (17) and (18) also contradict the hypothesis. This shows that $Y = Z_r - C$ is the only possible outcome of the decoding algorithm, completing the proof. □

The following corollary is clear from the necessary and sufficient conditions.

Corollary 3: A code is (t_1, t_2)-ST, if and only if it is also (t_2, t_1)-ST.

Notice that the decoding algorithm for a (t_1, t_2)-ST code is not the same as the decoding algorithm for a (t_2, t_1)-ST code when $t_1 \neq t_2$.

In the next sections, we discuss actual constructions of codes that are (t_1, t_2)-SD and ST as well as optimality issues.

IV. Constructions of (t_1, t_2)-SD and ST Codes

In this section, we give two constructions of (t_1, t_2)-SD and ST codes.

The first construction involves a family of codes well known in literature: the so called t-error-correcting/all unidirectional error-detecting (EC/AUED) codes [2]–[4], [8]. A t-EC/AUED code satisfies condition a) [8], which is sufficient for both (t_1, t_2)-skew detection and correction when the skew between the transmitted codeword and the received sequence does not exceed (t_1, t_2), and $t = \min\{t_1, t_2\}$. We state this fact in the next proposition.

Proposition 1: Let t_1 and t_2 be positive integers and $t = \min\{t_1, t_2\}$. Let $\mathcal{C}$ be a t-EC/AUED code. Then $\mathcal{C}$ is (t_1, t_2)-SD and ST.

An efficient way of constructing at t-EC/AUED is as follows: first encode the information bits using a t-error-correcting code. Then append a tail such that the code satisfies property a). Efficient tail matrices may be found in [2]–[4].

The second family of codes that we consider are the so called error-correcting unordered (ECU) codes. A t-ECU code is a code that can correct t errors and any two codewords are unordered.

Definition 4: We say that a code $\mathcal{C}$ is error-correcting unordered (ECU) with minimum distance d if, for any $X, Y \in \mathcal{C}$,

a) $d_H(X, Y) = N(X, Y) + N(Y, X) \geq d$.
b) $\min\{N(X, Y), N(Y, X)\} \geq 1$.

The connection between ECU codes and (t_1, t_2)-SD and ST codes is given by the following lemma.

Lemma 1: Let t_1 and t_2 be positive integers and $t = \min\{t_1, t_2\}$.

a) Let $\mathcal{C}$ be an ECU with minimum distance $\geq t_1 + t_2 + 1$. Then, $\mathcal{C}$ is (t_1, t_2)-SD.
b) Let $\mathcal{C}$ be an ECU with minimum distance $\geq t_1 + t_2 + t + 1$. Then, $\mathcal{C}$ is (t_1, t_2)-ST.

Proof:

1) Let $t = \min\{t_1, t_2\}$ and $T = \max\{t_1, t_2\}$. Let $X, Y \in \mathcal{C}$, and assume that condition a) is violated, say, $N(X, Y) \leq t$. The codewords are unordered, and also,

$$\begin{aligned} N(Y, X) &= d_H(X, Y) - N(X, Y) \\ &\geq t_1 + t_2 + 1 - t = T + 1. \end{aligned}$$

Hence, X and Y satisfy condition b) in Theorem 2, proving that the code is (t_1, t_2)-SD.

2) Let $X, Y \in \mathcal{C}$, and assume that condition a) is violated, say, $N(X, Y) \leq t$. The codewords are unordered, and also $N(Y, X) = d_H(X, Y) - N(X, Y) \geq t_1 + t_2 + 1$. Hence, X and Y satisfy condition b) in Theorem 4, proving that the code is $(t_1, t_2$-ST. □

In the particular case in which $t_1 = t_2 = t$, an ECU code with minimum distance $2t + 1$ gives a (t, t)-SD code, while an ECU with minimum distance $3t+1$ gives a (t, t)-ST code.

Next, we describe a method to construct systematic ECU codes. The construction is in fact a generalization of the well-known Berger construction [1].

Construction 1: Assume that we want to construct an ECU code $\mathcal{C}$ with minimum distance d and dimension k. Choose an $[n', k, d]$ error-correcting (EC) code $\mathcal{C}'$. Let $\underline{u}$ be an information vector of length k. Then proceed as follows.

a) Encode $\underline{u}$ into a vector $\underline{v} \in \mathcal{C}'$.
b) Let j be the Hamming weight of $\underline{v}$. Then append to $\underline{v}$ the complement of the binary representation of $\lfloor j/d \rfloor$.

The code obtained with this encoding procedure is ECU with minimum distance d.

Before proving that the code is ECU, we observe the following.

1) The code $\mathcal{C}$ has length $n' + \lceil \log_2 \lceil (n'+1)/d \rceil \rceil$.
2) The Berger construction corresponds to the special case in which $\mathcal{C}'$ is the $[k, k, 1]$ code.
3) The code $\mathcal{C}$ is systematic if the code $\mathcal{C}'$ is systematic.
4) We may sometimes make the construction more efficient when the all-1 vector is in $\mathcal{C}'$ by taking a coset of this code. The construction is analogous but we have less than $n' + 1$ different weights in the coset [4].
5) For a table with the best error-correcting codes, see [10].

Lemma 2: The code $\mathcal{C}$ obtained in Construction 1 is ECU with minimum distance at least d.

Proof: It is clear that the minimum distance is at least d. Assume that we have two codewords $\underline{u}$ and $\underline{v}$ in $\mathcal{C}'$ with weights i and j, respectively, where $i \leq j$. Notice that $N(\underline{v}, \underline{u}) > 0$. Let $t_{\underline{u}}$ and $t_{\underline{v}}$ be the tails when we encode using Construction 1. We will prove that $N((\underline{u}, t_{\underline{u}}), (\underline{v}, t_{\underline{v}})) > 0$.

We have two possibilities: either $\lfloor i/d \rfloor = \lfloor j/d \rfloor$ or $\lfloor i/d \rfloor \neq \lfloor j/d \rfloor$.

If $\lfloor i/d \rfloor = \lfloor j/d \rfloor$, then $j - i \leq d - 1$. If $\underline{u} \subseteq \underline{v}$, then $d_H(\underline{u}, \underline{v}) = j - i \leq d - 1$, a contradiction. So, in particular, $(\underline{u}, t_{\underline{u}})$ and $(\underline{v}, t_{\underline{v}})$ are unordered.

If $\lfloor i/d \rfloor \neq \lfloor j/d \rfloor$, then, in particular, $\lfloor i/d \rfloor < \lfloor j/d \rfloor$. According to Construction 1, $t_{\underline{u}}$ as a binary number is larger than $t_{\underline{v}}$ as a binary number. This means, $N(t_{\underline{u}}, t_{\underline{v}}) > 0$. Since we had that $N(\underline{v}, \underline{u}) > 0$, it follows that $(\underline{u}, t_{\underline{u}})$ and $(\underline{v}, t_{\underline{v}})$ are unordered. □

Example 5: Assume that we want to construct a $(1, 1)$-ST code of dimension k. The first approach is to encode the k information bits into an $[n', k, 3]$ Hamming code. We then append a tail in such a way that the code becomes 1-EC/AUED.

Take for instance $k = 57$. We first add six redundant bits in order to encode the information into a [63, 57, 3] Hamming code. Using the table in [4], we see that we have to add nine bits to make the code 1-EC/AUED (and therefore, $(1, 1)$-ST). This gives a total of 15 redundant bits.

The second approach is to use Construction 1 to obtain an ECU code with minimum distance 4. We first have to encode into a [64, 57, 4] extended Hamming code. Then, to make the code unordered, we have to add a tail of length $\lceil \log_2 \lceil 65/4 \rceil \rceil = 5$ bits. This gives a total of 12 redundant bits, so, for $k = 57$, the second method is more efficient than the first. If we take a coset of this code, the weight distribution goes from 1 to 63, so we have 63 different weights. Now, $\lceil \log_2 \lceil 63/4 \rceil \rceil = 4$ bits, so we save one bit in the total redundancy.

In the next section, we deal with the issues of optimality of ECU codes.

V. Optimal Error-Correcting Unordered Codes

In the previous section, we have presented two general constructions of codes that meet the necessary and sufficient conditions. The second construction is based on error-correcting codes to which a tail is added in such a way that the code is unordered.

We consider the optimality of Construction 1 in the following sense: the tail added to the error-correcting code has minimal length, i.e., it is impossible to find a shorter tail making the code unordered. In this sense, we prove that Construction 1 is optimal for the extended Hamming codes and for certain BCH codes (this does not mean that the code is globally optimal, in the sense that the tail is the shortest one that can be added to the information bits).

We begin by defining the concept of a *chain* of vectors;

Definition 5: A set of binary vectors $\{V_1, V_2, \cdots, V_m\}$ is a chain of length m if any two vectors in the set are ordered.

The idea in proving the optimality of our constructions is to exhibit a long enough chain of codewords in the error-correcting code. The following lemma gives the key.

Lemma 3: Let $\{C_1, C_2, \cdots, C_m\}$ be a chain of vectors, each being a codeword in a given code $\mathcal{C}$. Then the length of the tail that we have to add to $\mathcal{C}$ to make it unordered is at least $\lceil \log_2 m \rceil$ bits.

Proof: Since all the codewords in the chain are ordered, we need to have a different tail for every one of them to make them unordered. Hence, we need at least m different tails. □

We prove the optimality of some of our constructions by exhibiting chains of length $\lceil n/d \rceil + 1$ in an $[n, k, d]$ code. First we prove the optimality of our construction for the extended Hamming code by exhibiting a chain of $2^{m-2} + 1$ codewords in a code of length 2^m.

Proposition 2: The $[2^m, 2^m - m - 1, 4]$ extended Hamming code contains a chain of $2^{m-2} + 1$ codewords.

Proof: The columns of the parity check matrix of a $(2^m, 2^m - m - 1, 4)$ extended Hamming code are

$$\{(v_1, v_2, \cdots, v_m, 1)^T : (v_1, v_2, \cdots, v_m) \in \{0, 1\}^m\}.$$

Note that we can arrange the columns in the parity check matrix in pairs such that the first m bits are complementary. Namely, column $(v_1, v_2, \cdots, v_m, 1)^T$ is paired with $(\overline{v}_1, \overline{v}_2, \cdots, \overline{v}_m, 1)^T$. Hence, the sum of a pair of columns in this arrangement gives the vector $(1, 1, \cdots, 1, 0)^T$ and the sum of two pairs (four columns) is the all-0 vector. We call

this matrix H_m. For example,

$$H_3 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Consider the extended Hamming code that corresponds to the matrix H_m. It follows from the construction of H_m that the following set of $2^{m-2}+1$ codewords is a chain:

$$\{(1\,1\,1\,1)^i 0^{2^m-4i} : 0 \le i \le 2^{m-2}\},$$

where S^i, S a binary vector, is a vector obtained by concatenating S i times. □

The second result is related to BCH codes. We prove that in many cases we can exhibit chains of codewords that show the optimality of our construction. The key to exhibiting long chains is the following lemma [5].

Lemma 4: Consider a binary t-error-correcting BCH code defined in a standard way, i.e., as a cyclic code of length $2^m - 1$. Let a and b be two integers such that

$$a \cdot b = 2^m - 1$$

and

$$a \ge 2t + 1.$$

Then, the following b polynomials correspond to codewords:

$$z_1(X) = 1 + X^b + X^{2b} + \cdots + X^{(a-1)b}$$

and for $2 \le i \le b$,

$$z_i(X) = X^{i-1} z_1(X).$$

Using this lemma we can prove the following.

Proposition 3: Given a t-error-correcting BCH code of length $2^m - 1 = a \cdot b$ where a and b are integers, and $a \ge 2t+1$, we can exhibit a chain of length $b+1$.

Proof: The proof follows from Lemma 4. The chain consists of the all-0 vector and the set of b vectors that correspond to partial sums of the polynomials from Lemma 4 as

$$\left\{ \sum_{i=1}^{j} z_i : 1 \le j \le b \right\}.$$ □

Example 6: Consider the case $t = 2$, namely $2t + 1 = 5$. We can exhibit a chain of $((2^m - 1)/5) + 1$ codewords in all the cases in which $2^m - 1 \equiv 0 \pmod 5$. For example, for $m = 4$ we can exhibit a chain of length 4. In general, we can exhibit a long chain whenever $m \equiv 0 \pmod 4$ (by Fermat's Theorem). Similarly, for $2t + 1 = 7$, we can exhibit a long chain for all the cases in which $m \equiv 0 \pmod 6$.

To summarize, we proved in this section that our construction of a t-ECU code is optimal when we consider the extended Hamming codes and certain BCH codes.

VI. Conclusion

We have studied a problem in parallel asynchronous communications allowing a certain amount of skew between consecutive messages. We have shown that there are codes that can either detect or correct a certain amount of skew. We gave a precise mathematical definition of the concept of skew. We found necessary and sufficient conditions for codes that can either detect or tolerate a predetermined amount of skew. We constructed codes satisfying the necessary and sufficient conditions and we studied their optimality. Finally, we provided efficient encoding and decoding algorithms.

We note here that better (t_1, t_2)-ST codes were obtained in a recent paper [6].

Acknowledgment

The authors thank the referees for their comments that led to an improved presentation of the results in the paper.

References

[1] J. M. Berger, "A note on error detecting codes for asymmetric channels," *Inform. Control*, vol. 4, pp. 68–73, Mar. 1961.
[2] M. Blaum and H. van Tilborg, "On t-error-correcting/all unidirectional error-detecting codes," *IEEE Trans. Comput.*, vol. 38, pp. 1493–1501, Nov. 1989.
[3] F. J. H. Boinck and H. van Tilborg, "Constructions and bounds for systematic tEC/AUED codes," *IEEE Trans. Inform. Theory*, vol. 36, pp. 1381–1390, Nov. 1990.
[4] J. Bruck and M. Blaum, "New techniques for constructing EC/AUED codes," *IEEE Trans. Comput.*, vol. 41, pp. 1318–1324, Oct. 1992.
[5] R. H. Deng and M. A. Herro, "DC-free coset codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 786–792, July 1988.
[6] L. H. Khachatrian, "Construction of (t_1, t_2)-tolerant codes," to appear in *Proc. of Dilijan Conf.*, Sept. 1991.
[7] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.
[8] D. K. Pradhan, "A new class of error-correcting detecting codes for fault-tolerant computer application," *IEEE Trans. Comput.*, vol. 29, pp. 471–481, June 1980.
[9] T. Verhoeff, "Delay-insensitive codes—An overview," *Distributed Computing*, pp. 3:1–8, 1988.
[10] ——, "An updated table of minimum distance bounds for binary linear codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 665–680, Sept. 1987.