Proximity Detecting Codes *

(Preliminary Version)

Nitin H. Vaidya Srinivas Perisetty
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
{vaidya,speriset}@cs.tamu.edu

Technical Report 96-014
June 28, 1996

Abstract

A t-proximity detecting (t-PD) code can detect when a received word is at distance $\leq t$ from the transmitted codeword. For non-trivial values of t, a t-proximity detecting code must be unordered. However, not all unordered codes are t-proximity detecting. This report characterizes t-proximity detecting (t-PD) codes, and presents some properties of such codes. Designs of t-PD codes and lower bounds on number of checkbits are also presented. Proximity detecting codes may be useful to improve the performance of asynchronous communication channels.

*Research reported is supported in part by National Science Foundation grant MIP-9423735 and Texas Advanced Technology Program grant 009741-052-C.

Contents

Note

If you are familiar with past work on *proximity detection* (or similar), please inform the authors. We have been unable to locate any previous work on this subject.

1 Introduction

Unordered codes have been proposed for asynchronous channels, where delays on individual wires are independent and arbitrary [6, 5, 15]. In such a system, the sender encodes the data using an unordered code, and transmits the codeword. The receiver waits until a codeword is received, and then decodes it to recover the data. As different non-zero bits may take different amount of time to reach the receiver, an unordered code is necessary [6, 5, 15].

The notion of *proximity detection* is motivated by the need to improve the performance of an asynchronous channel. Presently, we are considering binary channels (that can take values 0 or 1). We consider the four-phase protocol for asynchronous communication using unordered codes [12]. In this protocol, the channel (or the bus) begins in the all-0 state (i.e., all wires are 0). The sender encodes the data as a codeword from an unordered code, and transmits the codeword. When the receiver receives the codeword, it sends an acknowledgement back to the sender. Upon receiving the acknowledgement, sender returns the bus to an all-0 state (called spacer) – this all-0 state is acknowledged by the receiver before the next codeword is transmitted by the sender [12].

While we are not concerned with actual specifics of the communication protocol, we do assume that the bus is all-0 before the sender begins transmitting a codeword. Thus, any word received by the receiver must be *covered* by the transmitted codeword.

The problem of designing proximity detecting codes is, at the very least, an interesting theoretical problem. Proximity detecting codes can be useful in asynchronous systems where certain assumptions can be made on *relative* delays on the wires of a bus (even if absolute delays are unpredictable) [1]. A t-proximity detecting code can be used to allow the receiver to send an acknowledgement to the sender when "all but t" non-zero transmitted bits have been received. If the "skew" in receiving the last t bits is guaranteed to be less than the *minimum* round trip delay, then the early acknowledgement can improve performance, without affecting correctness of the received data.

It is possible to conceive formulations of codes, other than proximity detecting, that can be used to improve performance of asynchronous channels under various assumptions regarding delay. We only consider the *proximity detecting* codes in this report. Two interesting properties of t-PD codes are listed below, and proved later in the report.

- Consider the following procedure: Take a t-PD code. To each codeword from this code, append one bit, whose value is arbitrarily chosen to be 0 or 1. (The appended value is not same for all codewords.) The resulting code may *not* be t-PD.
  This is unlike traditional error detecting/correcting codes, where adding arbitrary extra bits retains the original capability of the code.

- A t-error correcting code is not necessarily t-PD. Even an *unordered* t-error correcting code is not necessarily t-PD. (As discussed later, this observation raises an interesting issue regarding applicability of unordered SEC codes [3] for asynchronous communication.)

This report is organized as follows. Section 2 presents characterization and some properties of t-PD codes. A lower bound on the number of checkbits for 1-PD codes is obtained in Section 3. Sections 4 and 5 present design of some t-PD code. Section 6 summarizes the results. Appendix A presents some efficient 1-PD codes obtained by a manual search. Appendix B presents a table that can be used in proving optimality of some of our codes.

2 Characterization and Properties of t-PD Codes

For future reference, we present a few definitions.

Definition 1 *An n-bit word X is said to cover an n-bit word Y if X has a 1 in each bit position where Y has a 1.*

For instance, 011 covers 010, but does not cover 110.

Definition 2 *Two n-bit words X and Y are said to be unordered if neither word covers the other word.*

For instance, 011 and 101 are unordered.

Definition 3 *A codeword X is said to be* reachable *from a received word V if X covers V.*

Thus, if V is received by a receiver, the transmitted codeword must be one of the codewords *reachable from V* (assuming no errors).

Let $N(X,Y)$ denote the number of positions where X has a 1 and Y has a 0. For instance, $N(0111, 1001) = 2$ and $N(1001, 0111) = 1$. Let $d(X,Y)$ denote the Hamming distance between X and Y. Then, $d(X,Y) = N(X,Y) + N(Y,X)$ [14]. For future reference, note that if weight of X is smaller than weight of Y, then $N(X,Y) < N(Y,X)$; if the two weights are equal then $N(X,Y) = N(Y,X)$.

Definition 4 *A code C is t-proximity detecting (or t-PD) provided it can be used to determine if a received word is at distance $\leq t$ from the <u>transmitted</u> codeword.*

To clarify the definition, note two points:

- The received word, say W, is necessarily covered by the transmitted codeword, say X. Thus, $N(W,X) = 0$, and Hamming distance $d(X,W)$ is equal to $N(X,W)$.

- The keyword transmitted codeword in the above definition is important. Using a proximity detecting code, the receiver must always be able to correctly determine if a received word is at distance $\leq t$ from the *transmitted* codeword.

- If the received word is at distance $\leq t$ from any *reachable* codeword, it must also be at distance $\leq t$ from the transmitted codeword. When the received word is at distance $\leq t$ from the *transmitted* codeword, it may possibly be at distance $\leq t$ from another reachable codeword. (This observation leads to the characterization of t-PD codes presented below.)

2.1 Characterization of t-PD codes

Theorem 1 *A code C with minimum codeword weight $\geq t$ is t-PD if and only if, for each $X, Y \in C$, $X \neq Y$, one of the following conditions is true:*

1. *for some $i \leq t$, $N(X,Y) = N(Y,X) = i$, or*
2. *$N(X,Y) > t$ and $N(Y,X) > t$.*

Proof: The minimum codeword weight is $\geq t$, as assumed in the theorem. (If minimum codeword weight is $< t$, then Theorem 2 is applicable.)

Necessity: Without loss of generality assume that weight(X)$\leq$weight(Y). Then, it follows that $N(X,Y) \leq N(Y,X)$.

We provide proof of necessity by contradiction. Assume that there exist $X, Y \in C$ such that, neither of the two conditions in the theorem is satisfied. This results in two possibilities. We show that each possibility leads to a contradiction that the code is not t-PD.

- $N(X,Y) \leq t$, $N(Y,X) \leq t$ and $N(X,Y) \neq N(Y,X)$: $N(X,Y) \neq N(Y,X)$ implies that weight(X)$\neq$weight(Y). This, together with our initial assumption that weight(X)$\leq$weight(Y), implies that weight(X)$<$weight(Y), and therefore, $N(X,Y) < N(Y,X)$. This situation is pictorially illustrated in Figure 1. In Figure 1, V is equal to $X.Y$ (bit-wise and of X and Y). Thus, V is the largest weight vector covered by X and Y both. Also, distance between X (Y) and V is equal to $N(X,Y)$ ($N(Y,X)$). Now, as weight of X is at least t, there must exist W such that W is covered by V, and distance of W from X is t. As distance of V from X is smaller than that from Y, it follows that distance of W from X is also smaller than that from Y.[1] Thus, distance(X,W) $= t$ and distance(Y,W) $> t$. As X and Y are both *reachable* from W, when W is the received word, the receiver cannot correctly determine if W is in t-proximity of the transmitted codeword.

[1] Note that X and Y both cover V and W. Therefore, $d(X,W) = N(X,W)$ and $d(Y,W) = N(Y,W)$.

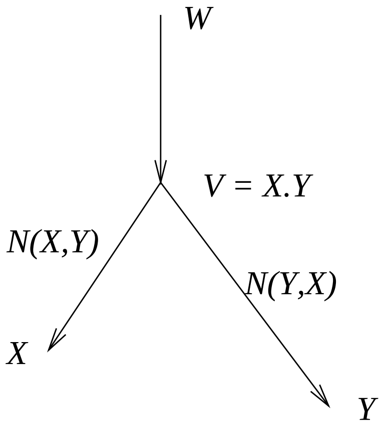

Figure 1: Proof of necessity: Theorem 1

- $N(X,Y) \leq t$ and $N(Y,X) > t$: This implies that weight(X)<weight(Y). The proof in this case similar to the previous case. Consider $V = X.Y$ (bit-wise and of X and Y). Clearly, V can potentially be received as an intermediate word when X or Y are transmitted. Additionally, because $N(X,Y) \leq t$ and $N(Y,X) > t$, V is at distance $\leq t$ from X and distance $> t$ from Y. When V is received, the receiver cannot correctly determine if it is at distance $\leq t$ from the transmitted codeword (as the receiver does not know which of X and Y may be transmitted). Thus, necessity of the condition in the theorem is proved.

Sufficiency: Assume that the condition stated in the theorem is satisfied.

Let X be the transmitted codeword, and let W be an intermediate word received by the receiver. Trivially, X must be reachable from W. Now we show that if there exists *any* codeword Y such that Y is reachable from W, and W is at distance $\leq t$ from Y, then W will also be at distance $\leq t$ from X. Consider the two alternative conditions in the theorem:

- $N(X,Y) = N(Y,X) \leq t$: As X and Y are both reachable from W, and $N(X,Y) = N(Y,X)$, it follows that $N(X,W) = N(Y,W)$. As $N(W,X) = N(W,Y) = 0$ (because X and Y cover W), W is at the same distance from both X and Y. Therefore, W must be at distance $\leq t$ from X.

- $N(X,Y) > t$ and $N(Y,X) > t$: This condition cannot be true if W is at distance $\leq t$ from Y, and X and Y are both reachable from W. To see this, note that the largest weight vector from which both X and Y are reachable is $X.Y$; distance of Y from $X.Y$ is $N(Y,X) > t$. Thus, as $N(Y,W) \leq t$, weight of W must be larger than $X.Y$, and therefore, X could not be reachable from W – this is a contradiction.

The above shows that if a received word is distance $\leq t$ from any reachable codeword, then it is distance $\leq t$ from the transmitted codeword. This implies that t-proximity can be correctly detected. $\square$

Observe that the condition in the above theorem can, equivalently, be stated as follows: For any codewords $X, Y \in C$, either (i) $N(X,Y) = N(Y,X)$, or (ii) $N(X,Y) > t$ and $N(Y,X) > t$.

Theorem 2 *A code with minimum codeword weight S and maximum codeword weight L, such that $S < L$, cannot be t-PD for $S \le t < L$.*

Proof: The initial all-0 received word is at distance S from a minimum weight codeword, and distance L from a maximum weight codeword. As $S \le t < L$, the all-0 received word is at distance $\le t$ from a reachable weight S codeword, and distance $> t$ from a reachable weight L codeword. Therefore, the code cannot be used to correctly determine t-proximity. □

Corollary 1 *A code is t-PD for all t if and only if it is a constant-weight code (i.e., all codewords have same weight).*

Proof:

Sufficiency: Observe that for two codewords X and Y of the same weight, $N(X,Y) = N(Y,X)$. Thus, a condition in Theorem 1 is satisfied for all t.

Necessity: If the minimum codeword weight S and maximum codeword weight L are not equal, then the code cannot be S-PD (according to Theorem 2). Therefore, S and L must be equal. In other words, all codewords must have the same weight. □

For a constant weight code, such as w-out-of-n, where all codewords are of weight w, a received word of weight u is at distance $\le t$ from the transmitted codeword if and only if $u + t \ge w$. Thus, if a received word has weight $\ge w - t$, then it is in t-proximity of the transmitted codeword.

Corollary 2 *A t-PD code is u-PD if $t > u$.*

Proof: Follows from Theorem 1. □

Theorem 3 *Consider n-bit code C such that maximum weight of any codeword in C is L. Then, C is t-PD for $t \ge L$.*

Proof: Trivial, as *any* received word is distance $\le t$ from the transmitted codeword. □

Values of t identified in Theorem 3 (i.e., $t \ge L$) are said to be the "trivial" values of t. We will only be interested in "non-trivial" values of t.

From Theorem 1, it follows that, for a non-trivial value of t (including $t = 0$), all t-PD codes are unordered codes. Note that the $\lfloor n/2 \rfloor$-out-of-n is an optimal unordered code [9], as well as a t-PD code (for all t). As a t-PD code must be unordered, the $\lfloor n/2 \rfloor$-out-of-n code is also an *optimal* t-PD code (for all non-trivial t). Thus, the problem of designing efficient *non-systematic* t-PD codes is already solved.

The t-PD codes designed in this report are all *systematic* codes. Although an optimal *non-systematic* unordered code is also an optimal non-systematic t-PD code, an optimal *systematic* unordered code (e.g., Berger code [2]) is **not** t-PD for any *non-trivial* non-zero t. (All unordered codes are 0-PD, however.)

Example 1 *Berger codes with number of databits $k = 2^r - 1$ are not 1-PD.* For instance, consider the (5,3) Berger code shown below.

Data	checkbits
000	11
001	10
010	10
011	01
100	10
101	01
110	01
111	00

Assume that 011 01 is the transmitted codeword and 000 01 is the received word. Clearly, this results in ambiguity as 000 01 is at distance 1 from the codeword 000 11 but at distance 2 from the transmitted codeword 011 01. Thus, the above code is not 1-PD. This example can be generalized to arbitrary r.[2]

2.2 Anomalous behavior of t-PD codes

In error detecting and correcting codes, adding an extra checkbit to the words from a given code does not *decrease* its capability. However, PD codes are anomalous in the sense that adding checkbits can reduce the proximity detection capability. For instance, the code $C_1 = \{011, 110\}$ is 1-PD. Appending a 1 checkbit to 011 and a 0 checkbit to 110 results in code $C_2 = \{0111, 1100\}$. Code C_2 is not 1-PD. Note that when received word is 0100, it is at distance 1 from 1100 but distance 2 from 0111 – this does not permit unambiguous 1-proximity detection.

2.3 Relation between t-PD and other unordered codes

As noted previously, all unordered codes are 0-PD. Now we consider unordered codes with additional capabilities.

[2]The proof, that the Berger code is not 1-PD, is based on the observation that, for some u, v, w, checkbits of weight u and $u+1$ are assigned to data of weight v and w, where $|v - w| > 1$. As a t-PD code, with $t > 1$, must also be 1-PD (by Corollary 2), the Berger code is not t-PD for any non-trivial non-zero t.

Corollary 3 *Any t-EC-AUED (t-error correcting-all unidirectional error detecting) code is t-PD.*

Proof: A code C is t-EC-AUED iff for every $X, Y \in C$ the following condition is true: $N(X,Y) \geq t+1$ and $N(Y,X) \geq t+1$ [4, 6, 11, 13]. If this condition is satisfied, then condition 2 in Theorem 1 is also satisfied. Therefore, a t-EC-AUED code is also t-PD. □

Example 2 *A t-PD code is not necessarily t-EC-AUED :* This is expected, as the necessary and sufficient conditions for t-PD code are weaker than those for t-EC-AUED code. 2-out-of-4 code is an example of a code that is 1-PD code but not 1-EC-AUED (or SEC-AUED). The codewords from the 2-out-of-4 code are: 0011, 0101, 1001, 0110, 1010, 1100. Note that $N(0011, 0101) = N(0101, 0011) = 1$. Therefore, the code is not SEC-AUED. The code is 1-PD, however, as all codewords are of weight 2.

Example 3 *An unordered single error correcting (unordered SEC) code is not necessarily 1-PD :* A code is unordered SEC [3] iff for every $X, Y \in C$ the following condition is true: $N(X,Y) \geq 1$ and $N(Y,X) \geq 2$ or vice versa. As satisfying this condition is not sufficient to satisfy Theorem 1, as expected, an unordered SEC code is not necessarily 1-PD. $C = \{001, 110\}$ is an example code that is unordered SEC but not 1-PD. (Note that $N(001, 110) = 1$ and $N(110, 001) = 2$.)

The above observation raises an issue regarding applicability of unordered SEC codes for asynchronous communication. Consider the code $C = \{001, 110\}$. When 000 is received by the receiver, there are two possibilities: (a) 001 may be the transmitted codeword, but the least significant bit possibly contains an error, or (b) 110 is the transmitted codeword, and the 1-bits have not been received yet. In this case, the receiver cannot correctly determine which possibility to assume, and hence, it may not be possible to correct single errors (assuming channel delays are arbitrary). This situation is similar to impossibility of achieving distributed consensus in the presence of single failure when channels are asynchronous [8].

Example 4 *A 1-PD code is not necessarily an unordered SEC code :* A code is unordered SEC [3] iff for every $X, Y \in C$, $N(X,Y) \geq 1$ and $N(Y,X) \geq 2$ or vice versa. As satisfying Theorem 1 is not sufficient to satisfy requirements for an unordered SEC code, as expected, a 1-PD code is not necessarily unordered SEC. The 2-out-of-4 code presented in Example 2 is 1-PD but not unordered SEC, because $N(0011, 0101) = N(0101, 0011) = 1$.

Example 5 *Dual-rail code [15] is a systematic t-PD code for all t :* Dual-rail code with k databits is a $(2k, k)$ constant-weight code; each codeword has weight k. Therefore, by Corollary 1, the dual-rail code is t-PD for all t. It is trivial to show that, dual-rail code is the most efficient *systematic* constant-weight code; therefore, it is also the most efficient systematic code that is t-PD for <u>all</u> t.

3 A Bound for 1-PD Codes

Theorem 4 *The number of databits k, and the number of checkbits r, in a systematic 1-PD code must satisfy the following relation:*

$$k + 1 \leq \sum_{i=0}^{r} F_i$$

where F_i is the maximum cardinality of an r-bit constant weight code with minimum distance 4 such that each codeword has weight i.

Proof: Consider the $k+1$ data vectors $D_0,\ D_1, \cdots D_k$, each containing k bits, such that D_j covers D_i if $i < j$. Thus, D_i is of weight i.

Now assume that, for some i and j, $i < j$, D_i and D_j are assigned r checkbits C_i and C_j, such that C_i and C_j are of the same weight and $d(C_i, C_j) = 2$. Now consider $V = D_iC_i.D_jC_j$ (bit-wise and of D_iC_i and D_jC_j). Thus, $V = D_iC_*$, where $C_* = C_i.C_j$. Observe that $d(D_iC_i, V) = 1$ and $d(D_jC_j, V) > 1$, while D_iC_i and D_jC_j are both reachable from V. Thus, the code cannot be 1-PD.

Thus, a necessary condition for a 1-PD code is that no two r-bit checkbits (such as C_i and C_j) of the same weight should be at distance 2. Or, in other words, two r-bits checkbits of same weight must be at distance at least 4 (same weight vectors can only have even distance). If F_i denotes maximum number of checkbits combinations of weight i, such that no two of them are at distance 2, then the theorem follows from the above observation. □

Appendix B enumerates F_i for small values of r, and shows how the above bound can be used to prove optimality of some of our codes.

The bound in the above theorem, in general, is not tight. For some values of k, the bound specified in the theorem yields the minimum possible value of r. For some other small k, the bound can be further refined to obtain a tight bound. The refinement procedure we use is somewhat *ad hoc*, therefore, omitted here. We expect to provide a better bound in a future revision of this report.

4 Design of 1-PD Codes

4.1 1-PD codes: Construction 1

This construction is simple, though not the most efficient. In this construction (and all other constructions described here), checkbits assigned to given databits are completely determined by weight of the databits. That is, two data with the same weight are assigned the same checkbits. The construction, described below, is later illustrated with an example.

1. First, a $(k + r_1, k)$ unordered code is constructed by appending $r_1 = \lceil \log_2 (k+1) \rceil$ checkbits to given k-bit data. The r_1 checkbits, similar to Berger codes [2], denote the number of zeros in the data.

2. Another $r_2 = \lceil \log_2 (k+1) \rceil$ bits are appended by representing the number of zeros in the $k + r_1$ bits obtained in the first step. (We will later prove that $\lceil \log_2 (k+1) \rceil$ bits are sufficient for this step.)

The total number of checkbits for k data bits is $r_1 + r_2 = 2\lceil \log_2 (k+1) \rceil$. Now we present an example code, followed by a proof that the above construction yields a 1-PD code. For the example code, $k = 7$. Thus, data weight can be between 0 and 7. The table below lists the checkbits assigned to data of each possible weight.

```
data weight     r_1  r_2

     0          111  111       Observation:
     1          110  111           This code remains 1-PD if 111,110,100,011
     2          101  110           in r_2 column are replaced by 11,10,01,00,
     3          100  110           respectively. We will return to this
     4          011  100           observation in the next section.
     5          010  100
     6          001  011
     7          000  011
```

The first r_1 bits represent number of zeros in the data. The next r_2 checkbits represent the number of zeros in the $k + r_1$ bits. For instance, data of weight 3 contains $7 - 3 = 4$ zeros, and the r_1 checkbits 100 assigned to this data contain 2 zeros, for a total of 6 zeros – therefore, the corresponding r_2 checkbits are 110.

Now we prove that the above construction yields a 1-PD code.

Proof: We claim that, in step 2, r_2 need only be $\lceil \log_2 (k+1) \rceil$ – that is, the number of zeros in the first $k + r_1$ bits of the codeword never exceeds $2^{\lceil \log_2 (k+1) \rceil} - 1$. The proof of this claim is deferred till Section 5.1. Assuming this claim is correct, we now prove that the above construction yields a 1-PD code.

One key observation, that is used in all construction presented here, is that if two codewords X and Y are of the same weight, then they satisfy the condition stated in Theorem 1 (because, for codewords of same weight, $N(X,Y) = N(Y,X)$). By appending the same checkbits to the data of same weight, we ensure that the codewords corresponding to data of same weight also have the same weight.

Now consider two data d_1 and d_2 of different weights. During step 1, these two data are appended r_1 checkbits c_1 and c_2 respectively. Now there are two possibilities:

- d_1c_1 and d_2c_2 are of the same weight: In this case, the second step will assign them identical r_2 checkbits, and resultant weight of the two codewords will be identical. Therefore, the two codewords will satisfy the condition in Theorem 1.

- d_1c_1 and d_2c_2 are of different weight: Without loss of generality, assume that weight of d_1c_1 is greater than d_2c_2. Thus, $N(d_2c_2, d_1c_1) < N(d_1c_1, d_2c_2)$. Note that the first step ensures that d_1c_1 and d_2c_2 are unordered. Therefore, $1 \leq N(d_2c_2, d_1c_1) < N(d_1c_1, d_2c_2)$. If $N(d_2c_2, d_1c_1) = 2$, then $N(d_1c_1, d_2c_2) > 2$, and the condition in Theorem 1 will be satisfied, irrespective of the checkbits appended in the second step.
  Therefore, now consider the case when $N(d_2c_2, d_1c_1) = 1$ and $N(d_1c_1, d_2c_2) \geq 2$.

  In this case, the second step (similar to Berger code) appends r_2 checkbits that represent the number of zeros. Let the r_2 checkbits appended to d_1c_1 (d_2c_2) be f_1 (f_2). As the number of zeros in d_1c_1 is smaller, f_1 will contain a 0 at a place where f_2 contains a 1. This implies that $N(d_2c_2f_2, d_1c_1f_1) \geq N(d_2c_2, d_1c_1) + 1 = 2$. As $N(d_1c_1, d_2c_2) \geq 2$, $N(d_1c_1f_1, d_2c_2f_2) \geq 2$. Thus, codewords $d_1c_1f_1$ and $d_2c_2f_2$ satisfy the condition in Theorem 1.

$\square$

The above construction can be generalized as follows:

- Determine r_1 checkbits using any systematic unordered code construction that assigns same r_1 checkbits to all data of the same weight.

- Additional r_2 checkbits are appended to represent the number of zeros in the $k + r_1$ bits – r_2 is chosen such that it can represent the largest possible number of zeros in the $k + r_1$ bits.

4.2 1-PD codes: Construction 2

In the example of construction 1 observe that, codewords corresponding to data of weight i and $i - 1$, where i is odd, have identical r_2 checkbits. Using this observation, construction 2 reduces the number of checkbits by 1. Construction 2 requires $2\lceil \log_2(k+1) \rceil - 1$ checkbits for k databits. The steps of the construction are described below:

1. First, a $(k + r_1, k)$ unordered code is constructed by appending $r_1 = \lceil \log_2(k+1) \rceil$ checkbits to given k-bit data. The r_1 checkbits, similar to Berger codes [2], denote the number of zeros in the data.

2. Another $r_2 = \lceil \log_2(k+1) \rceil - 1$ bits are appended as follows: Let z be the number of zeros in the data. Then, the r_2 checkbits represent $\lfloor z/2 \rfloor$ in binary.

The total number of checkbits for k databits is $r_1 + r_2 = 2\lceil \log_2(k+1) \rceil - 1$. Now we present an example code, followed by a proof that the above construction yields a 1-PD code. For the example code, $k = 7$. Thus, data weight can be between 0 and 7. The table below lists the checkbits assigned to data of each possible weight.

data weight	r_1	r_2
0	111	11
1	110	11
2	101	10
3	100	10
4	011	01
5	010	01
6	001	00
7	000	00

Proof: Let $Z(i)$ denote the number of zeros in the r_1-bit binary representation of i.

The above construction appends same checkbits to different data of same weight. This ensures that the codewords corresponding to data of same weight also have the same weight, and therefore, these codewords satisfy the condition in Theorem 1.

Now consider data d_i and d_j, containing i and j zeros ($i \neq j$), respectively. Without loss of generality, let $i > j$. Thus, weight(d_i) <weight(d_j). Let c_i (c_j) and f_i (f_j) denote the checkbits appended to d_i (d_j) in steps 1 and 2, respectively. There are three possibilities:

- $j = i - 1$, and i is odd :

  Claim: After appending the first r_1 checkbits, the $(k + r_1)$-bit "partial" codewords $d_i c_i$ and $d_j c_j$ have the same weight. **Proof of the claim:** If i is odd, then the binary representation for i ends with a 1, and the binary representation for $j = i-1$ differs from i only in the last bit. Thus, the binary representation for $i-1$ contains one more 0 than i. Thus, $Z(i-1) = Z(i)+1$. Therefore, $i + Z(i) = (i-1) + Z(i-1)$. This proves correctness of the claim.

  Now, observe that, $\lfloor i/2 \rfloor = \lfloor j/2 \rfloor$, as $j = i-1$ and i is odd. Thus, the r_2 checkbits assigned to data of weight i and j will be identical, i.e., $f_i = f_j$. As the weight of the corresponding partial $(k+r_1)$-bit codewords $d_i c_i$ and $d_j c_j$ is identical, weight of the $(k+r_1+r_2)$-bit codewords $d_i c_i f_i$ and $d_j c_j f_j$ is also identical. Therefore, the codewords will satisfy the condition in Theorem 1.

- $j = i - 1$, and i is even : In this case, it is easy to see that $Z(j) \leq Z(i)$. Therefore, $j + Z(j) < i + Z(i)$. This implies that the weight of the $(k + r_1)$-bit partial codeword $d_i c_i$ has a smaller weight than $d_j c_j$. Also, step 1 ensures that $d_i c_i$ and $d_j c_j$ are unordered. These two conditions, together, imply that $N(d_i c_i, d_j c_j) \geq 1$ and $N(d_j c_j, d_i c_i) \geq 2$. As, i is even, and $j = i - 1$, it follows that checkbits f_i and f_j, assigned in step 2, are not identical. Also, as $i > j$, it follows that f_i has a 1 at a place where f_j has a 0. Thus, $N(d_i c_i f_i, d_j c_j f_j) = N(d_i c_i, d_j c_j) + 1 \geq 2$. Also, as $N(d_j c_j, d_i c_i) \geq 2$, $N(d_j c_j f_j, d_i c_i f_i) \geq 2$. Thus, the two codewords satisfy the condition in Theorem 1.

- $j \leq i - 2$: In this case, clearly, $N(d_j c_j f_j, d_i c_i f_i) \geq 2$. As $j \leq i - 2$, $f_i = \lfloor i/2 \rfloor \neq f_j$. As the number of zeros in d_i is larger than d_j, checkbits c_i and f_i both will contain a 1 at one place where c_j and f_j, respectively, contain a 0. Therefore, $N(d_i c_i f_i, d_j c_j f_j) \geq 2$.

In each of the above cases, the final codewords satisfy Theorem 1.

□

4.3 1-PD codes: Construction 3

The third construction is more efficient than construction 2 for large values of k. This construction uses SEC-AUED codes to design 1-PD codes, as described below.

- Obtain r_1 checkbits such that checkbits for data of different weight are different, and checkbits for data of identical weight are identical. Thus, r_1 must be at least $\lceil \log_2 (k+1) \rceil$. (For instance, the checkbits may denote number of zeros in the data.)
- Encode the r_1 checkbits using a $(r_1 + r_2, r_1)$ SEC-AUED code. Any SEC-AUED code may be used, for instance [4, 11, 7, 10].

The resulting $(k + r_1 + r_2, k)$ code is 1-PD, as proved next.

Proof: Consider two data d_1 and d_2. If weight of data d_1 and d_2 are same, they are appended same $r_1 + r_2$ checkbits, resulting in codewords of identical weight, which clearly satisfy Theorem 1. Now assume that weight of d_1 is different from d_2. Let the checkbits assigned to d_1 (d_2) in the first two steps be c_1 (c_2) and f_1 (f_2), respectively. Thus, the two resultant codewords are $d_1c_1f_1$ and $d_2c_2f_2$.

Note that checkbits c_1 and c_2 are encoded using a SEC-AUED code to obtain codewords c_1f_1 and c_2f_2 from the SEC-AUED code. From the characterization of SEC-AUED codes [4, 6, 11, 13] it follows that $N(c_1f_1, c_2f_2) \geq 2$ and $N(c_2f_2, c_1f_1) \geq 2$. This, in turn, implies that $N(d_1c_1f_1, d_2c_2f_2) \geq 2$ and $N(d_2c_2f_2, d_1c_1f_1) \geq 2$. Thus, the resulting codewords satisfy the condition in Theorem 1. □

4.4 Parameters for some 1-PD codes

The table below summarizes the number of checkbits required for a 1-PD code designed using our constructions. Construction 1 always requires more checkbits than construction 2. Construction 3 requires less checkbits than construction 2 for large k. In the table, second column lists the smallest number of checkbits obtained using constructions 2 and 3, and third column lists the construction number (2 or 3) that yields smaller number of checkbits. For comparison purposes, last column of the table lists number of checkbits for a SEC-AUED code [10] for given values of k.

For $k \leq 4$, the dual-rail code is optimal, i.e., optimal number of checkbits is k. Therefore, $k \leq 4$ are not listed in the table. The simple bound presented earlier implies that the 1-PD codes listed below are optimal for $k = 7, 15$ (refer Appendix B). Not all 1-PD codes listed below are optimal. Appendix A presents some 1-PD codes that are better than those listed in the table. The codes presented in Appendix A were obtained by a manual search, and (as yet) we do not have a systematic way of generating codes with same amount of redundancy.

number of data bits k	number of checkbits r	construction number	checkbits for a SEC-AUED code [7, 10]
5	5	2	6
6	5	2	7
7	5	2	7
8	7	2	7
9	7	2	7
10	7	2	8
12	7	2	9
15	7	2	10
20	9	2	10
25	9	2	11
31	9	2	12
32	11	2	13
50	11	2	13
55	11	2	14
63	11	2	15
64	13	2	15
127	13	2	17
128	15	2/3	17
139	15	2/3	17
255	15	2/3	19
256	16	3	19
282	16	3	19
512	18	3	22
857	18	3	22

5 Design of t-PD codes

This section presents two constructions for t-PD codes. These constructions have not yet been verified for optimality (in fact, they will not, in general, be optimal).

5.1 A recursive construction for t-PD codes: Construction 4

This construction is a generalization of construction 1 for 1-PD codes. Note that any unordered code is trivially a 0-PD code. A t-PD code, $t > 0$, can be obtained using a $(t-1)$-PD code as follows:

1. Encode given k data bits using a $(t-1)$-PD code. Let number of checkbits in the $(t-1)$-PD code be r_1. Let the resultant $(k + r_1)$-bit vector be named X.

2. Append to X additional r_2 checkbits that represent the number of zeros in X. The resultant $(k + r_1 + r_2)$-bit vector is the final codeword for the given data. r_2 is the smallest number of bits that can represent the largest possible number of zeros in X.

The resulting code is t-PD.

Proof: Consider two codewords X and Y, from the $(t-1)$-PD code, obtained in step 1. Let step 2 append checkbits c_1 and c_2 to X and Y, respectively. Based on Theorem 1, for codewords X and Y from a $(t-1)$-PD code, there are 4 possibilities:

- $N(X,Y) = N(Y,X)$: In this case, weights of X and Y are identical, so they will be assigned the same checkbits in step 2, i.e., $c_1 = c_2$. Therefore, the final codewords, say Xc_1 and Yc_2 will satisfy the condition $N(Xc_1, Yc_2) = N(Yc_2, Xc_1)$.
- $N(X,Y) > t$ and $N(Y,X) > t$: In this case, for the final codewords Xc_1 and Yc_2 the following condition will hold: $N(Xc_1, Yc_2) > t$ and $N(Yc_2, Xc_1) > t$.
- $N(X,Y) = t$ and $N(Y,X) > t$: In this case, as weight of X is smaller than that of Y, number of zeros in X is larger than that in Y. Therefore, c_1 will contain a 1 in at least one place, where c_2 contains a 0. Therefore, $N(Xc_1, Yc_2) \geq N(X,Y)+1 = t+1$. Also, as $N(Y,X) > t$, $N(Yc_1, Xc_1) > t$. Thus, we have $N(Xc_1, Yc_2) > t$ as well as $N(Yc_1, Xc_1) > t$.
- $N(X,Y) > t$ and $N(Y,X) = t$: This case is similar to the case above.

Note that in each of the above four cases, the final codewords Xc_1 and Yc_2 satisfy the condition in Theorem 1. Thus, the code obtained is t-PD. $\square$

The table below presents an example of the above construction.

data weight	r_1 = 6	r_2 = 3
0	111 111	111
1	110 111	111
2	101 110	111
3	100 110	111
4	011 100	110
5	010 100	110
6	001 011	100
7	000 011	100

The 2-PD code above is designed using the example 1-PD code in Section 4.1. Thus, the first $r_1 = 6$ checkbits are identical to those obtained for the 1-PD code. The next r_2 checkbits count the number of zeros in the $k + r_1 = 7 + 6 = 13$ bits. For instance, for data of weight 2 (i.e.

containing 5 zeros) the r_1 checkbits are 101 110. Thus, the total number of zeros in the $k + r_1$ checkbits is 7. Therefore, the r_2 checkbits for data weight 2 are 111.

Observe that the above code remains 2-PD if the r_2 bits 111, 110, 100 are replaced by 11, 10, 01, respectively. This observation can, in general, be used to reduce the number of checkbits required by the above construction. This point is elaborated later.

5.2 A Special Case and Some Observations

We consider the case when the recursive algorithm is applied to design a t-PD code, starting from the 0-PD Berger code. To get a t-PD code, the recursion is unfolded t times. For this application of the recursive algorithm, consider the case when number of data bits k is such that $2^{r-1} \leq k \leq 2^r - 1$. Then, $\lceil \log_2(k+1) \rceil = r$. Thus, the 0-PD Berger code is a $(k+r, k)$ code.

Observation 1: In each iteration of the above recursion, r additional checkbits are sufficient ($r_2 = r$). That is, independent of how many times the recursion is applied, the number of bits needed to represent the number of zeros, in any iteration, is equal to r.

Proof: Consider the case when $k = 2^r - 1$. The proof for other values of k follows from the proof for $k = 2^r - 1$.

Let i be an integer $\leq k$ (thus, i can be represented as a r-bit binary number). Let $Z(i)$ and $N(i)$ denote the number of zeros and ones, respectively, in the r-bit binary representation of i. Thus, $Z(i) + N(i) = r$. We now prove that $i + Z(i) \leq k$ (recall that we have assumed $k = 2^r - 1$).

As $Z(i) \leq r$, clearly, if $i \leq k - r$, then $i + Z(i) \leq k$. Now consider i such that $i > k - r$. There are two possibilities:

- $i = k$: In this case, as $i = 2^r - 1$, $Z(i) = 0$. Thus, $i + Z(i) = k$.
- $k - 1 \geq i > k - r$: Now observe that, as $k = 2^r - 1$, $N(k-i) = Z(i)$ (i.e., the r-bit binary representation of $k - i$ is one's complement of i). Also, observe that, for any j, $N(j)$ can at most be $\log_2(j+1)$. The two observations together imply that $Z(i) = N(k-i) \leq \log_2(k-i+1)$. Thus, $i + Z(i) \leq i + \log_2(k-i+1)$. Now, let us represent i as $k - a$ where $1 \leq a < r$. Then, $i + Z(i) = k - a + \log_2(a+1)$. As $a \geq \log_2(a+1)$ for $a \geq 1$, $i + Z(i) \leq k$.

Thus, we have proved that,

$$i + Z(i) \leq k, \quad 0 \leq i \leq k$$

Now, consider the Berger code encoding. The Berger code, for a data containing i zeros, appends checkbits that represent i in r-bit binary. Thus, the number of zeros in a codeword from a Berger code, corresponding to data containing i zeros, is $i + Z(i) \leq k$. Thus, it follows that each codeword from the 0-PD Berger code contains at most k zeros. Now assume that each codeword from the $(t-1)$-PD code contains at most k zeros ($t \geq 1$).

Induction: Each codeword of the $(t-1)$-PD code contains at most k zeros. Step 2 of the recursive procedure appends to a codeword containing i zeros, checkbits that represent i in r-bit binary (as $i \leq k$, r bits are sufficient, i.e., $r_2 = r$). As $i + Z(i) \leq k$, the resultant $(k + r_1 + r_2)$-bit codeword contains at most k zeros.

The above result is easily extended to any k – in general, in each iteration, $\lceil \log_2(k+1) \rceil$ additional checkbits are needed. □

Observation 2: Observation 1 implies that number of zeros in a codeword cannot be larger than k, independent of how many times the recursion is applied. If a codeword from a $(t-1)$-PD code already contains k zeros, then the $r_2 = r$ checkbits appended in step 2 will be all-1, and the number of zeros in the resultant codeword will remain k. If, on the other hand, a codeword from the $(t-1)$-PD code contains $i < k$ zeros, the codeword obtained after appending r_2 checkbits will contain $> i$ but $\leq k$ checkbits. Thus, after some iterations, each codeword will contain exactly k zeros – in other words, the code will be a constant-weight code. Thus, after a finite number of iterations of the recursive procedure, a constant-weight code is obtained, which is t-PD for *all* values of t.

Observation 3: The number of checkbits needed can be minimized by using a technique similar to that used in 1-PD construction 2. The basic idea is that, if the number of *distinct* weights of the codewords from the $(t-1)$-PD code is ν, then r_2 need only be $\lceil \log_2 \nu \rceil$. The recursive algorithm can be modified as follows:

1. Encode given k data bits using a $(t-1)$-PD code. Let number of checkbits in the $(t-1)$-PD code be r_1. Let the resultant $(k + r_1)$-bit vector be named X.

2. Let ν be the distinct number of weights of the codewords from the $(t-1)$-PD code. Let these weights be $w_0, w_1, \cdots, w_{\nu-1}$ in an increasing order. If weight of X is w_i, then append to X additional $r_2 = \lceil \log_2 \nu \rceil$ checkbits that represent $(\nu - 1 - i)$ in binary.

The resulting code is t-PD. The proof is similar to that for the original recursive construction. The table below presents example of a 2-PD code obtained by the revised recursive procedure.

data weight	c_1	c_2	c_3
0	111	11	1
1	110	11	1
2	101	10	1
3	100	10	1
4	011	01	0
5	010	01	0
6	001	00	0
7	000	00	0

The code containing data and the c_1 checkbits (in above table) is the 0-PD Berger code. The code that contains the data and the c_1 and c_2 checkbits is 1-PD (as obtained using the revised recursive construction). Observe that c_2 contains only 2 checkbits (not 3). As the codewords from the 0-PD code, corresponding to data of weight 2 and 3 both have weight 4, they are assigned identical checkbits (specifically, 10). The code obtained by appending c_3 checkbits, in addition to c_1 and c_2, is 2-PD. Note that c_3 contains only 1 bit, as the codewords from the 1-PD code have only two distinct weights.

5.3 t-PD codes: Construction 5

This construction is a generalization of construction 3 for 1-PD codes. This construction uses t-EC-AUED codes to design t-PD codes, as described below.

- Obtain r_1 checkbits such that checkbits for data of different weight are different, and checkbits for data of identical weight are identical. Thus, r_1 must be at least $\lceil \log_2 (k+1) \rceil$. (For instance, the checkbits may denote number of zeros in the data.)
- Encode the r_1 checkbits using a $(r_1 + r_2, r_1)$ t-EC-AUED code. Any t-EC-AUED code may be used, for instance [4, 11, 7, 10].

The resulting $(k + r_1 + r_2, k)$ code is t-PD. The proof is similar to the proof of construction 3. It uses the fact that a code is t-EC-AUED iff for any two codewords X and Y from that code $N(X,Y) \geq t+1$ and $N(Y,X) \geq t+1$ [4, 11].

6 Summary

This report defines a new class of codes that are capable of t-proximity detection. The report characterizes the codes and studies their properties. It turns out that the optimal $n/2$-out-of-n unordered (non-systematic) code is also an optimal non-systematic t-PD code. An optimal systematic unordered code, however, is not t-PD for non-trivial non-zero t. This report presents designs of some systematic t-PD codes. A simple lower bound on number of checkbits in a 1-PD code is also presented. Work is in progress to improve the lower bound, as well as to design better t-PD codes.

Acknowledgements

A discussion, on asynchronous systems, with Venkatesh Akella of University of California–Davis prompted this research on proximity detecting codes. This work is funded in part by National Science Foundation grant MIP-9423735 and Texas Advanced Technology Program grant 009741-052-C.

A 1-PD Codes Obtained by Manual Search

This section presents some 1-PD codes generated by a manual search. These codes are better than those obtained by our constructions presented previously.

For each code below, we present a table that lists the checkbits for data of each possible weight. k is the number of databits.

The proof that the codes presented below are 1-PD is obtained by verifying that each pair of codewords X and Y satisfies the condition in Theorem 1.

A.1 1-PD code with $k = 5$ and $r = 4$

data weight	checkbits
0	1111
1	1110
2	1100
3	0011
4	0001
5	0000

A.2 1-PD code with $k = 12$ and $r = 6$

data weight	checkbits
0	111111
1	111110
2	111100
3	110011
4	001111
5	000111
6	011001
7	101010
8	001010
9	100001
10	010100
11	001000
12	000000

A.3 1-PD code with $k = 16$ and $r = 7$

data weight	checkbits

0	1111111
1	1111110
2	1111100
3	1110011
4	1001111
5	0001111
6	1010101
7	1101010
8	1101000
9	1000110
10	0110001
11	0011100
12	0011000
13	1000001
14	0100010
15	0100000
16	0000000

A.4 1-PD code with $k = 30$ and $r = 8$

data weight	checkbits
0	1111 1111
1	1111 1110
2	1111 1100
3	1111 0011
4	1100 1111
5	0011 1111
6	0001 1111
7	0110 0111
8	0111 1001
9	1010 1011
10	1101 0101
11	1101 0100
12	0011 1100
13	0011 0011
14	1100 0011
15	1010 0101
16	0101 1010
17	1001 1001
18	1001 0110
19	1110 1000

20	1110 0000
21	0000 0111
22	0010 1010
23	0100 1001
24	1000 1100
25	1000 1000
26	0100 0010
27	0010 0100
28	0001 0001
29	0000 0001
30	0000 0000

B F_i Table

For some small values of r, the table below presents F_i; recall that F_i is defined as the maximum cardinality of an r-bit constant weight code with minimum distance 4 such that each codeword has weight i. For $r \leq 3$, $F_i = 1$, for $i \leq r$. Note that, in general, $F_{r-i} = F_i$.

Caveat: The F_i's listed in the table were obtained by a manual "trial-and-error" procedure, and are believed to be accurate.

	$r = 4$	$r = 5$	$r = 6$
F_0	1	1	1
F_1	1	1	1
F_2	2	2	3
F_3	1	2	4
F_4	1	1	3
F_5	-	1	1
F_6	-	-	1
$\sum_{i=0}^{r} F_i$	6	8	14

From the above table, and the bound presented in Section 3, it follows that, $r = 6$ is inadequate for a 1-PD code with $k = 15$. Thus, at least 7 checkbits are necessary. Construction 2 yields an optimal (22,15) 1-PD code. Similarly, our construction yields optimal 1-PD code for $k = 7$ (5 checkbits)

References

[1] V. Akella. Personal communication, October 1995.

[2] J. M. Berger, "A Note on Error Detection Codes for Asymmetric Channels," *Information and Control*, vol. 4, pp. 68–73, 1961.

[3] M. Blaum and J. Bruck, "Unordered error-correcting codes and their applications," in *Digest of papers: The 22^{nd} Int. Symp. Fault-Tolerant Comp.*, pp. 486–493, July 1992.

[4] M. Blaum and H. van Tilborg, "On t-error correcting/all unidirectional error detecting codes," *IEEE Trans. Computers*, vol. 38, pp. 1493–1501, November 1989.

[5] B. Bose, "On Unordered Codes," *IEEE Transactions on Computers*, vol. 40, pp. 125–131, February 1991.

[6] B. Bose and T. R. N. Rao, "Theory of unidirectional error correcting/detecting codes," *IEEE Trans. Computers*, vol. 31, pp. 521–530, June 1982.

[7] J. Bruck and M. Blaum, "New techniques for constructing EC/AUED codes," *IEEE Trans. Computers*, vol. 41, October 1992.

[8] M. J. Fischer, N. A. Lynch, and M. Merritt, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, pp. 374–382, April 1985.

[9] C. V. Frieman, "Optimal error detection codes for completely asymmetric binary channels," *Information and Control*, vol. 5, pp. 64–71, March 1962.

[10] R. S. Katti and M. Blaum, "An improvement on constructions of t-EC/AUED codes," *IEEE Trans. Computers*, vol. 45, pp. 607–608, May 1996.

[11] D. J. Lin and B. Bose, "Theory and design of t-error correcting and $d(d > t)$-unidirectional error detecting (t-EC d-UED) codes," *IEEE Trans. Computers*, pp. 433–439, April 1988.

[12] C. A. Mead and L. Conway, *An Introduction to VLSI Systems.* Addison Wesley, 1980. *Chapter 7 by C. Seitz, entitled "System Timing".*

[13] D. K. Pradhan, "A new class of error-correcting/detecting codes for fault-tolerant computer applications," *IEEE Trans. Computers*, vol. 29, pp. 471–481, June 1980.

[14] T. R. N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems.* Prentice-Hall, 1989.

[15] T. Verhoeff, "Delay-insensitive codes – An overview," *Distributed Computing*, vol. 3, pp. 1–8, 1988.