Optimal Frame Synchronization

W. Kizner
DSN Engineering and Operations Office

Optimal frame synchronization algorithms are developed which will reject bad data as well as provide high probabilities for obtaining correct frame synchronization with data which has an error rate consistent with project requirements. The exact analysis to obtain these probabilities is outlined. The amount of computation to obtain these quantities may be very large, hence easily computed approximations are also given.

I. Introduction

The aim of this study is to develop optimal frame synchronization algorithms under a very general set of assumptions. The definition of optimality can be specified to a large extent by the user. The problem of frame synchronization may be posed in the following way. Initially, data is received starting from some unknown part of the frame. The beginning of the frame is determined by recognizing a sync word, or a binary sequence of given length which is not likely to be confused with a string of information bits. When further identifications of sync words occur at intervals equal to integer multiples of a length of a frame, then a final determination can be made that frame synchronization has been attained.

The figure of merit of an algorithm depends on how reliably good data is frame-synched, and how few errors are made in labeling as synched data which is not properly synched or data which has a large number of errors. Another is the computation time involved. Here, the user can specify his own figure of merit consistent with his goals.

One optimal property of the present technique is that if the quality of the data should be seriously degraded after some point, then this algorithm insures that the good frames up to this point are kept and the bad frames after and including this point are rejected. This is in contrast to other algorithms which have a time lag. It also contrasts to the use of the computed signal-to-noise ratio, which also lags.

Definitions of terms used are given in Table 1.

II. Stages of the Algorithm

The method consists of three states:

(1) *Search mode.* Initially, the information train is examined one bit at a time in a serial shift register, N bits in length, where N is the number of bits in the sync word. When the contents of the shift register match the sync word, allowing up to E_p error bits, the provisional lock mode is entered. A pattern with up to E_p errors is called good.

Table 1. Definition of terms

E_g	maximum number of bad patterns allowed in group
E_p	number of allowable bit errors in a pattern
N	number of bits in sync word
N_F	number of bits in frame
N_g	number of patterns in group, or number of representations of sync word that are used to determine mode of algorithm
p_b	probability of a bit error
p_c	probability that a sequence of N bits of the information train is confused as a pattern when starting in the wrong position
p_L	probability of a bad pattern (one which has more than E_p errors)
p_s	probability that a group of N_g sequences of N bits is mistaken for the condition that the system is in lock when starting in the wrong position
q_L	probability of a good pattern
Type I error	good data which should be in sync but is declared out of sync
Type II error	locking in at the wrong position in N_F or fewer tries of consecutive positions

(2) *Provisional lock mode.* The contents of the shift register are examined only once every N_F bits, where N_F is the length of the frame. Next, a sequence of patterns, or group, is tested as follows: N_g patterns at a time are examined. Thus, $N_g - 1$ new patterns are considered. If $N_g - E_g$ or more good patterns are observed, lock mode is entered. If not, search mode is entered again.

(3) *Lock mode.* Here, groups of N_g patterns, each spaced N_F bits apart, are examined. If $N_g - E_g$ or more good patterns are observed, the lock mode is continued. Here, a new pattern would be added to the group and the oldest one eliminated. If more than E_g bad patterns are observed, the lock mode is discontinued and the search mode is entered again at the approximate location of the next sync word. The frames up to the last good sync word are accepted and the others are rejected.

In the model used here, it is assumed that no bits are created or destroyed in error. Hence, the normal assumption is that loss of lock is due to bad data.

III. Approximate Calculation of the Possibilities of Some of the Errors

One type of error, called type I, is to declare good data as out of sync or bad. Another kind of error is to lock on data in the wrong position, which we call a type II error. Still another kind of error is to include as good data that data which has a high error rate. It will turn out that it is possible to calculate the probability of a type I error accurately, but the cost of computation may be high. Hence, a simple approximate formula is given here. It is not possible to calculate the probability of a type II error with any accuracy without a detailed knowledge of the kind of data transmitted, since this error consists of the possibility of confusing this data or part of this data with a sync word. However, some approximate optimistic estimates will be given.

Now to the calculation of the different types of errors: the criterion for labeling a pattern as good or bad may be generalized from what has been said before to take into account dependence of errors within a pattern. However, it is assumed that the probability that a pattern is bad p_L can be calculated, and that the probability that any pattern is good or bad is independent of the state of other patterns.

A heuristic method of computing the probability of a type I error can now be given. It consists of finding the probability of observing a group of N_g patterns with more than E_g errors, or

$$\text{prob (type I)} = 1 - \sum_{i=0}^{E_g} B(N_g; i, p_L) \tag{1}$$

where $B(n; k, p)$ is the probability of having k "successes" in n Bernoulli trials with probability p for "success,"

$$B(n; k, p) = \binom{n}{k} p^k (1-p)^{n-k} \tag{2}$$

and

$$\binom{n}{k}$$

is the binomial coefficient.

A closer examination of the reasoning reveals many flaws. For instance, part of a bad group may be usable.

In addition, the algorithm may reject some good groups in the process of losing lock and reacquiring it. Later, an exact treatment will be given.

Consider next the probability of a type II error, or the probability of achieving lock at the wrong position. We assume that the probability that any particular pattern of N bits is 2^{-N}. If E_p errors are allowed, then the probability of confusion of a pattern on a single try when out of sync p_c is

$$p_c = \sum_{i=0}^{E_p} \binom{N}{i} 2^{-N} \tag{3}$$

The probability of a type II error on a single try when out of sync is the probability that the first pattern in the group is confused as good (to gain provisional lock) and that at least $N_g - 1 - E_g$ in $N_g - 1$ are also taken as good, or

$$P_s = p_c \sum_{i=N_g-1-E_g}^{N_g-1} B(N_g - 1; i, p_c) \tag{4}$$

The probability of a type II error in N_F tries is the probability of a type II error in at least one of the $N_F - 1$ bad positions with the correct position rejected. Hence,

$$\begin{aligned} \text{prob (type II)} &= [p_L + (1 - p_L) \sum_{i=E_g+1}^{N_g-1} B(N_g - 1; i, p_L)] \\ &\times \sum_{j=1}^{N_F-1} B(N_F - 1; j, p_s) \\ &= [p_L + (1 - p_L) \sum_{i=E_g+1}^{N_g-1} B(N_g - 1; i, p_L)] \\ &\times [1 - (1 - p_s)^{N_F-1}] \end{aligned} \tag{5}$$

IV. Exact Solution for Type I Error

The method for obtaining the exact estimate of the probability of a type I error will now be outlined. First, we recognize that loss of lock is a recurrent event. This means roughly that the initial situation is restored again after each loss of lock, or that the experiment starts all over again. Next, the experiment is divided into two phases: gaining and losing lock. The expectation of the number of frames that are lost before gaining lock and the number of frames that are accepted when in lock are calculated. Lastly, the expectation of the number of frames that are lost when going out of lock before the process starts again is calculated.

To find the expectation of the number of frames lost before lock is achieved, one uses some results from the theory of Markov chains. A state is defined so as to specify the sequence of good and bad patterns in the group. The initial probability of each state is easily found, given the probability of a bad pattern p_L. The distribution of succeeding states is found by setting up the transition matrix, or the matrix which specifies the probability of the transition from state i to state j, given state i. States which are in lock are defined as absorbing, and the expectation of the time for absorption, or the expected number of frames lost, is calculated. The process is then reversed. Starting with a known distribution of states in lock, the expected number of frames that are accepted (before going out of lock) is calculated. Lastly, a correction is made for some additional frames lost.

V. Calculation of the Probability of Good or Bad Patterns

For uncoded data, the probability of a good pattern q_L is easily computed if it is assumed that the bit errors are independent.

$$q_L = 1 - p_L = \sum_{i=0}^{E_p} B(N; i, p_b) \tag{6}$$

where p_b is the probability of a bit error.

For coded data, the situation is different. For block coded data, for instance, one would like to try to determine the number of code words in error, rather than the number of bits in error in a sync word, because the number of wrong code words is a much better indicator of the status of the communications system. Also, although errors in code words are independent errors, the resulting bits are not. The problem of defining bad patterns for coded data is being pursued at present.

VI. Some Numerical Results

A few numerical results (Tables 2 and 3) will now be given to show the validity of the approximation for type I error and the possibility of obtaining algorithms which meet the objectives stated. First, a few results are given for establishing how accurate the approximation Eq. (1) is for the probability of type I error. These results and the others listed suggest that the approximation Eq. (1)

Table 2. Numerical results for $N_g = 5$, $E_g = 1$, $N = 15$, $N_F = 140$, and $E_p = 1$

Parameter	$p_b = 0.005$	$p_b = 0.05$
Probability of a bad pattern	0.2513×10^{-2}	0.1709
Expected number of consecutive frames lost	2.25	3.09
Expected number of consecutive frames in lock	40,081	18.09
Probability of type I error	0.5622×10^{-4}	0.1456
Approximation for probability of type I error, using Eq. (1)	0.6287×10^{-4}	0.2045
Probability of type II error	0.8061×10^{-13}	0.9014×10^{-11}

Table 3. Numerical results for $N_g = 7$, $E_g = 1$, $N = 15$, $N_F = 140$, and $E_p = 1$

Parameter	$p_b = 0.005$	$p_b = 0.05$
Probability of a bad pattern	0.2514×10^{-2}	0.1709
Expected number of consecutive frames lost	2.17	4.09
Expected number of consecutive frames in lock	26,925	17.25
Probability of type I error	0.8061×10^{-4}	0.1916
Approximation for probability of type I error, using Eq. (1)	0.1316×10^{-3}	0.3422
Probability of type II error	0.2946×10^{-9}	0.4494×10^{-17}

is good to within a factor of 2 and is increasingly accurate as the probability gets smaller.

Another conclusion that we can draw is that it is possible to meet the requirements of having small type I and II errors for bit error rates within project requirements and having a large type I error when the bit error rate exceeds this amount by, say a factor of 8. If $N_g = 15$, $E_g = 4$, $E_p = 0$, $p_b = 0.005$, and the other quantities are unchanged from those used in Tables 2 and 3, then Eq. (1) yields 0.0032 for the approximate type I error. If $p_b = 0.04$, then Eq. (1) becomes 0.89. In other words, there is a good chance that the poor data will be rejected. For this case, the probability of a type II error is essentially zero.