Transmission Parameter Estimation for an Autoconfigurable Receiver[1][2][3]

Raphael J. Lyman
(505) 646-3811
rlyman@nmsu.edu

Qingsong Wang
(505) 646-8862
qwang@nmsu.edu

Phillip De Leon
(505) 646-3771
pdeleon@nmsu.edu

Stephen Horan
(505) 646-4856
shoran@nmsu.edu

Klipsch School of Electrical and Computer Engineering
New Mexico State University
P.O. Box 30001, MSC 3-O
Las Cruces NM 88003-8001

Abstract—A satellite-to-ground communication link may sometimes be interrupted by an unexpected event, causing the satellite transponder to reset to an unknown state and begin transmitting in a different mode, resulting in a loss of data. We describe procedures to aid in reconfiguring the ground-station receiver by determining the new transmission parameters automatically from the received signal itself. We discuss procedures based on the signal statistics as well as on information about the data-frame structure. Simulations show that the latter approach can work for reasonably sized problems.

TABLE OF CONTENTS

1. INTRODUCTION

In order to demodulate a satellite-to-ground transmission correctly, a ground-station receiver must have knowledge of certain transmission parameters; e.g., data rate, data format, details of error-correction coding, etc. Normally, these are known ahead of time, but sometimes, when an unexpected event occurs, the satellite transponder may reset to an unknown state and begin transmitting data using a different set of parameters. This can cause data to be lost until the receiver operator determines that an event has occurred and makes appropriate adjustments.

The task of reconfiguring the receiver for the new parameters could be eased if the parameters could be determined automatically from the received signal itself. For example, data formats such as NRZ and BiΦ have distinctive statistical signatures [1]. Thus the data format, as well as the data rate, could be determined by estimating the power spectral density of the received signal. As we shall see, there are also time-domain approaches to these problems that offer simplicity and a method for calculating an error bound.

Some other parameters present more of a challenge. For example, there is no statistical distinction between random data formatted as NRZ-L and data formatted as NRZ-M. Such a distinction can be made, though, if we make use of *a priori* information regarding the frame structure of the transmitted data. Even if the link-layer protocol is not known at the ground-station receiver, if the number of possible protocols is small, we can distinguish between NRZ-L and NRZ-M by assuming that a particular one of these data formats was used in the transmission. We then demodulate accordingly and determine if the resulting data "makes sense."

In this paper, we apply this approach to a satellite-to-ground link through the TDRSS Multiple-Access service [2]. Satellites using the Space Network services will generally use either the Single Access (SA) service or the Multiple Access (MA) service for their forward and return link communications. The SA service may utilize Ka-Band, K-Band or S-Band frequencies while the MA service uses S-Band exclusively. The MA return service is a spread spectrum service and supports data rates up to 300 kbps through the original TDRS fleet and 3 Mbps on the newer satellites. The return data channels can be rate 1/2

[1] 0-7803-8155-6/04/$17.00© 2004 IEEE

[2] IEEEAC paper #1066, Version 2, Updated December 3, 2004

[3] This project was funded by NASA's Goddard Space Flight Center Grant #NAG5-13189.

convolutionally encoded on the original TDRS fleet or selectable between rate 1/2 and rate 1/3 on the newer satellites. The same chip rate is used on the return link, regardless of the convolutional rate or the underlying data rate. The RF transmission is formatted as a typical I/Q channel for QPSK signaling. At the ground station, the signal is downconverted and despread prior to processing the FEC-encoded data. In this paper, we assume that the downconversion and despread operations have been successfully completed since these are common to all MA users regardless of the data rate or FEC method.

We develop frequency-domain and time-domain approaches for determining the data rate and the basic data format, and we compute a bound on the probability of an incorrect decision. We also describe how other parameters may be determined using data-structure information, and show by simulation that this approach can work in the TDRSS MA environment.

2. USING THE SIGNAL STATISTICS

When examining a signal to determine the transmission parameters, keeping the algorithm simple is very important. For this reason, we wish to minimize the number of assumptions we make about the received signal. Fortunately, some parameters may be determined by examining the signal statistics alone. Examples of this include the data format and the data rate.

In the TDRSS MA service [2], allowable data formats include *non return to zero* and *biphase* types (NRZ and BiΦ, see Figure 1) [1]. NRZ-L formatted data, for example, responds to the logic "level" of the data, assigning one signal level to a logic 1 and another signal level to a logic 0. By contrast, BiΦ-L assigns a signal-level *transition* to each logic level. The transition for logic 1 is in the opposite direction of the transition for logic 0.

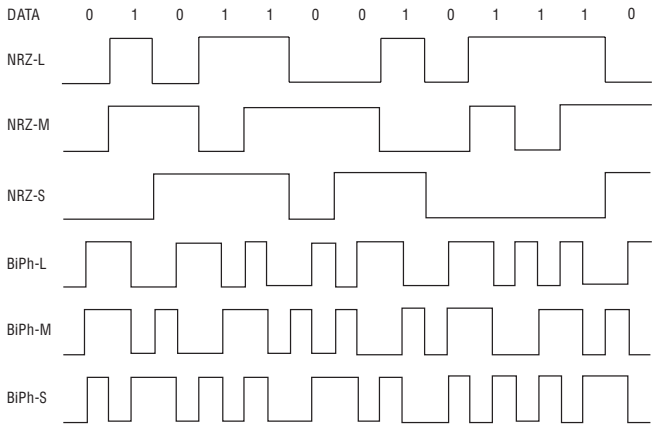

Figure 1 - Data Formats

NRZ-M and BiΦ-M respond to a logic 1 or "mark." For NRZ-M, a logic 1 is indicated by a change in signal level from the previous symbol. A logic 0 is indicated by no change in signal level. For BiΦ-M, a logic 1 is indicated by a change in the direction of the signal-level transition with respects to the previous symbol. NRZ-S and BiΦ-S are similar to NRZ-M and BiΦ-M, except that they respond to a logic 0 or "space," instead of a logic 1.

In the Multiple Access service, the formatted symbols are further modulated using direct-sequence spread spectrum, but the chip rate and details of modulation are the same for all data rates. Thus, we assume that the signal has been despread using well-known code acquisition and tracking procedures [3].

Note that, in contrast to NRZ, all BiΦ formats guarantee a signal-level transition during each symbol. For this reason, the power spectral densities of the two format types are distinct (see Figure 2). This suggests a frequency-domain approach for determining the data format of the received signal. We simply compute the periodogram of the signal and compare it to the spectra in Figure 2. For example, we note that the PSD of NRZ-formatted data has a negative slope at $f = 0$, whereas BiΦ-formatted data has a positive slope there. If $S[n]$ is the periodogram of the received signal, we may approximate the slope of the PSD using the forward difference $S[1] - S[0]$ [4]. If this difference is negative, we conclude that the data are NRZ formatted. A positive difference indicates BiΦ. The data rate could also then be estimated by determining the frequencies of the spectral nulls.

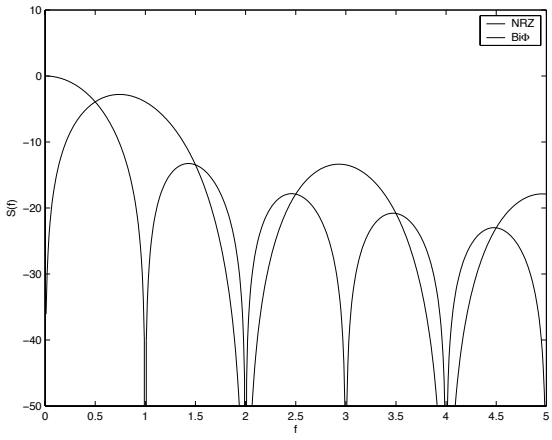

Figure 2 – Power Spectral Densities for Data Formats

Unfortunately, in order to obtain an accurate estimate of the PSD using the periodogram, it is usually necessary to buffer a long data record, and there is no convenient way to estimate the probability that the data-format determination procedure described above will yield an incorrect result.

To address these problems, we consider a time-domain approach. For example, we may determine the data rate by sampling the received signal at a rate much higher than the greatest expected data rate. We then search the sampled

signal for transitions in level. The shortest distance between adjacent transitions we call $W_{\min}$. By examining Figure 1, we see that if the data format is NRZ, then $W_{\min}$ is almost certainly the bit period. If the data format is BiΦ, then $W_{\min}$ will be one-half the bit period.

To determine whether the signal is NRZ or BiΦ, we note from Figure 1 that for BiΦ, the longest time between signal-level transitions is $2W_{\min}$. Thus, if our sampled signal contains a segment of length $3W_{\min}$ without a transition, then it is certainly NRZ (see Figure 3).

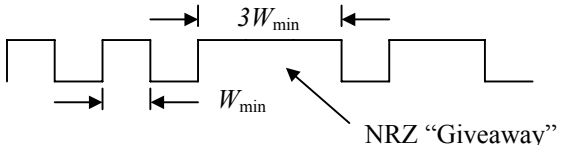

Figure 3 – Confirmed NRZ Waveform

If we examine our sampled signal and find that there is no segment of length greater than $2W_{\min}$ without a transition, then we conclude that it is formatted as BiΦ. This conclusion may be in error for certain transmitted data sequences. For example, if the transmitted sequence is "11001011" formatted as NRZ, $W_{\min}$ will be equal to one bit interval, and the time between signal-level transitions will always be less than $3W_{\min}$. This is because the data we are transmitting happens to contain no sequence of three consecutive like bits; i.e., "111" or "000".

3. PROBABILITY OF ERROR

To compute the probability that the above procedure results in an incorrect decision, we note that the error event may be written as

$$E = (\text{BiΦ} \cap \text{dNRZ}) \cup (\text{NRZ} \cap \text{dBiΦ}), \tag{1}$$

where, e.g., "BiΦ∩dNRZ" means the event that BiΦ data were transmitted, but we decided that the signal was formatted as NRZ. The two error types in (1) are mutually exclusive so from elementary probability we have [5]

$$P(E) = P(\text{BiΦ} \cap \text{dNRZ}) + P(\text{NRZ} \cap \text{dBiΦ}). \tag{2}$$

If BiΦ data are transmitted, then from Figure 1 we can see that, neglecting noise-induced errors, the signal will never contain a segment of length $3W_{\min}$ without a transition in level. Thus, we will never decide that NRZ was transmitted when the data are in fact formatted as BiΦ,

$$P(\text{BiΦ} \cap \text{dNRZ}) = 0. \tag{3}$$

Also,

$$P(\text{NRZ} \cap \text{dBiΦ}) = P(\text{dBiΦ} \mid \text{NRZ})P(\text{NRZ}). \tag{4}$$

We see from (4) and (2) that the error probability depends on the *a priori* probability that the transmitted sequence was formatted as NRZ. Since this probability may not be known, we can bound the error probability by noting that $P(\text{NRZ}) \le 1$. Making use of (4), (3) and (2) we have

$$P(E) \le P(\text{dBiΦ} \mid \text{NRZ}). \tag{5}$$

If the received data are formatted as NRZ, the probability that we will decide in favor of BiΦ is just the probability that the transmitted data contain no sequence of three consecutive like symbols; i.e., no "111" and no "000". The reason for this is that if such a sequence did occur, it would cause NRZ-L formatted data to hold a signal level for $3W_{\min}$ or longer.

The same probability holds for NRZ-M and NRZ-S as well. To see this, note from Figure 1 that an NRZ-M waveform may be obtained by first differentially encoding the data, and then formatting the result as NRZ-L. If a random sequence is differentially encoded, then the result is also random, and the probability of three consecutive like symbols is unchanged. In the case of NRZ-S, the original data must be inverted before differential encoding, but if a logic 1 is as likely to occur as a logic 0, the probabilities will again not be changed. We will make further use of these observations in the following section.

Suppose that the transmitted data consist of N binary symbols, $\{b_1, b_2, \ldots, b_N\}$. Let us use the symbol NA_n to mean the event that b_n, b_{n-1}, and b_{n-2} are not all alike. Then

$$P(\text{dBiΦ} \mid \text{NRZ}) = P\left(\bigcap_{n=3}^{N} \text{NA}_n\right). \tag{6}$$

The events $\{\text{NA}_n\}$ are not independent. For example, if $4 \le n \le N$, NA_n is not independent from NA_{n-1}, because the sequences $\{b_{n-2}, b_{n-1}, b_n\}$ and $\{b_{n-3}, b_{n-2}, b_{n-1}\}$ have the bits b_{n-2} and b_{n-1} in common. Thus, we evaluate the joint probability in (6) in terms of conditional probabilities [5]. For $N \ge 5$, this yields

$$P\left(\bigcap_{n=3}^{N} \text{NA}_n\right) = P(\text{NA}_4 \cap \text{NA}_3) \bullet \left(\prod_{n=5}^{N} P(\text{NA}_n \mid \text{NA}_{n-1} \cap \text{NA}_{n-2})\right). \tag{7}$$

It is easy to show that $P(\text{NA}_4 \cap \text{NA}_3) = 5/8$. The conditional probabilities are given by

$$P\left(\mathrm{NA}_n \mid \mathrm{NA}_{n-1} \cap \mathrm{NA}_{n-2}\right) = \frac{4}{5}, \quad 5 \le n \le N. \quad (8)$$

Substituting these values into (7), and making use of (6) and (5), we have

$$P(E) \le \left(\frac{5}{8}\right)\left(\frac{4}{5}\right)^{N-4}. \quad (9)$$

The bound in (9) may be used to show, for example, that an error probability $P(E) \le .0001$ may be achieved if the transmitted data are $N \ge 44$ bits long.

Recall that at the receiver, we sample a segment of the received signal in order to determine the data format and the data rate. Suppose that, after doing the $P(E)$ computations as above, we determine that we must sample enough of the signal to include at least N bits of transmitted data. How many seconds of the received signal does this represent? Since we do not yet know the data format or rate, we consider a minimum allowable data rate. For the Multiple Access service, the minimum data rate is 100 bps. If data transmitted at this rate are formatted as NRZ, then we expect W_{min} = 1/100, since the minimum pulse width for NRZ is just the bit interval. If we must have $N \ge 49$ for $P(E) \le .0001$, then we should sample a segment about .5 sec long.

In practice, the low data rate 100 bps is seldom used. If it is reasonable to assume a higher minimum data rate, the length of the sampled segment could be reduced.

4. USING THE FRAME STRUCTURE

We mentioned previously that an NRZ-M waveform may be obtained by first differentially encoding the data, and then formatting the result as NRZ-L. Data may be formatted as BiΦ-M in a completely analogous manner. For NRZ-S or BiΦ-S, the original data sequence is inverted before differential encoding. From this we see that once the format type, NRZ or BiΦ, is determined, the signal may be demodulated as if it were NRZ-L or BiΦ-L. Then the format suffix of the signal may be determined by whether the resulting sequence is differentially encoded.

As discussed previously, NRZ and BiΦ may be distinguished using a statistical approach. But there is no statistical difference between a binary sequence before and after differential encoding. A distinction can be made, though, if we know something beforehand about the data being encoded. For example, all of the data-link protocols most commonly used for space-to-ground transmission employ known binary sequences or *sync words* for proper frame synchronization [1]. If we differentially decode a received sequence, and if the decoded sequence has an occurrence of one of these sync words, then it is highly likely that the transmitted sequence was differentially encoded.

A similar approach may be used to determine the details of the convolutional encoding that was applied. For example, if Viterbi decoding of a rate-½ convolutional code is applied to a received signal, and if a recognized sync word is found in the resulting sequence, we conclude that the transmitted signal was coded in this way.

The line code and convolutional encoding may be determined jointly by trying all the possibilities. For example, suppose a received signal is known to be either L- or M-form modulation, and either uncoded or rate-½ encoded. We form the following four sequences: 1) the original sequence unchanged, 2) the original sequence after differential decoding, 3) the original sequence after Viterbi decoding of the convolutional code and 4) the original sequence after differential decoding and Viterbi decoding. Note that because convolutional encoding is applied before modulation at the transmitter, the operations are decoded in opposite order at the receiver to obtain sequence 4. The line code and convolutional encoding may be determined by searching each of the four sequences for sync word occurrences.

This approach provides a simple means of determining the type of modulation and convolutional encoding that have been applied to a received signal. A problem occurs, though, if one of the possible sync words is very short. For example, the HDLC *flag byte,* "01111110", is only eight bits long [6]. On average, we would expect this sequence to appear about once every 32 bytes in a random data stream. Thus, if a sequence is sufficiently long, it may contain a spurious occurrence of the flag byte even if it has not been properly decoded.

The problem may be addressed by handling HDLC separately. After forming the various decoded sequences from our received signal we test each to determine if it contains a valid HDLC frame. If one of them does, then we conclude that this is the correctly decoded sequence. Otherwise, we proceed to search for the occurrence of a sync word *other than* the flag byte.

It is possible to determine whether a data sequence contains a valid HDLC frame by using more detailed knowledge of the HDLC frame format. Specifically, error detection in HDLC is accomplished by appending a 16-bit frame-check sequence to the body of the frame [6]. Then a bit-stuffing operation is performed before the flag byte is attached to the beginning and end of the frame.

Bit stuffing prevents the spurious occurrence of the flag byte within the body of the HDLC frame. The flag byte contains a sequence of six consecutive ones. Thus, each time a sequence of five consecutive ones occurs in the frame body, a zero is stuffed. For example, "0111111"

becomes "01111101" and "0111110" becomes "01111100". This operation is easily reversed at the receiver. Note that bit stuffing affects the frame-check sequence as well as the user data.

To determine if a data sequence contains a valid HDLC frame, it is searched for two consecutive flag bytes. The data between these bytes is then destuffed and checked for the presence of a properly computed frame-check sequence. If the FCS is correct, then the sequence is declared to be valid HDLC data. Also, note that during idle periods, when there are no user data to transmit, a continuous sequence of flag bytes is transmitted instead. Thus, a sequence is also declared as valid HDLC data if it contains several consecutive flag bytes with no user data.

5. ESTIMATION ALGORITHM

To test the approach discussed in the previous section, we have developed an algorithm that uses frame-structure information to recover transmission parameters for a data sequence that might be transmitted on a space-to-ground link over the Multiple Access service. The transmission parameters of interest are the data-format suffix (-L, -M or -S), the convolutional code rate, the node synchronization, and the inversion pattern. These are some of the parameters required by the ground-station receiver in order to correctly demodulate and decode the received signal. The data rate and the format type (NRZ or BiΦ), which were discussed in Section 2, are assumed to have been estimated already.

As discussed in the previous section, the data-format suffix may be determined by demodulating the received signal as if it were NRZ-L or BiΦ-L, and then determining if the resulting sequence is differentially encoded. For the Multiple Access service, data may be transmitted without error-correction coding, or with convolutional encoding at rate ½ or rate 1/3 [2]. In practice, rate-1/3 encoding is rare, so we check only for the possibility of uncoded or rate-½ encoded data.

To decode a data sequence properly, it is also necessary to know the generator polynomials that were used by the convolutional encoder, but since only one set of polynomials is specified for each coding rate used in the Multiple Access service, it was not necessary to make a separate determination of these.

If a sequence has been convolutionally encoded, though, the Viterbi decoder must know which bits were produced by each polynomial. Figure 4 shows the operation of the convolutional encoder. Unencoded bits are clocked into a seven-bit shift register from the left. These seven bits are then combined using modulo-2 adders. The adders shown above the shift register represent the operation of polynomial "G1", and the adders below represent the polynomial "G2". Bits produced by G1 and G2 are clocked out alternately onto the output stream. For a rate-½ encoder, two bits are clocked out every time one bit is clocked in. Note that often the output from G2 is inverted, as shown in the figure.

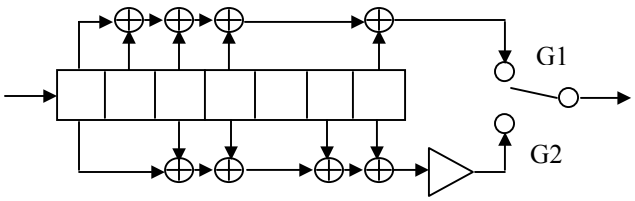

Figure 4 – Convolutional Encoder

It is clear from the above that the data presented to the Viterbi decoder consist of a sequence which alternates between G1 bits and G2 bits. When we begin to demodulate the data, we will not know if the first bit we receive is from G1 or G2. The process of determining this is called *node synchronization.*

As was mentioned previously, Figure 4 shows that the G2 bits are inverted before they are clocked out. This is done to avoid long strings of 1's or 0's in the output stream. It is typical in convolutional encoding to invert the bits from one or more polynomials. The specification for which polynomials are to be inverted we call the *inversion pattern.* It is of course necessary to know this in order to decode the signal properly.

The algorithm that we have developed estimates all of these parameters using frame-structure information, as discussed in the previous section. We assume that the satellite transponder transmits data using either HDLC [6] or the CCSDS TM data-link protocol [7]. We decode the received data sequence under all the possible sets of assumptions regarding the values of the parameters we are trying to estimate. The various output sequences are stored in a table, and then each sequence is checked to see if it conforms to one of the possible frame formats. Such a sequence is assumed to have been decoded correctly. Thus, we conclude that the parameter values used in decoding it are correct, and these become our estimates for reconfiguring the receiver.

Our algorithm recognizes a decoded sequence as conforming to the TM frame format if the sequence contains an appropriate sync word. One of these has the hex designation 1ACFFC1D. TM uses other longer sync words for data that have been turbo coded. Because the HDLC flag byte is so short, we use more detailed information about its frame structure, as discussed previously, in determining whether a decoded sequence was generated under this protocol.

6. SIMULATION AND RESULTS

We have developed a Matlab simulation that exercises the algorithm described in the previous section. Under each of the possible frame structures, the simulator generates 1000 random data sequences encoded under each of the possible assumptions about the transmission parameters. Each time a data sequence is generated, it is fed to the parameter-estimation algorithm. The estimates returned by the algorithm are compared with the assumptions used in encoding the sequence. If one or more parameters were estimated incorrectly, an error is recorded, and the data sequence giving rise to the error is logged.

The random data sequences were generated using simple models of the assumed protocols. When the CCSDS TM protocol was assumed, the sequences consisted of five frames, each with 200 bits of random user data prefixed by an appropriate sync word. The first frame of each sequence is truncated at a random point, so that the estimation algorithm does not know the starting position of the data. The HDLC frames were generated similarly, except that a frame-check sequence is appended to the user data, and then a bit-stuffing operation is applied, as described in Section 4, before the flag byte is attached.

When an error occurs in our simulation, we use the logged sequence to analyze the cause. One problem we have discovered relates to the Viterbi decoder. Because convolutional encoding is intended to correct errors, if two input sequences are very similar, they may in fact be decoded to the same sequence. In our initial simulation runs, we found that sequences encoded under very different parameter assumptions sometimes in fact produced very similar output, close enough that the output of the Viterbi decoder under the two cases was the same. This meant that we could not use these output sequences to distinguish which set of parameter assumptions was correct. Because the two sets of parameters produce very similar output, if the receiver is configured using the wrong set of parameters, it may still decode the incoming signal correctly most of the time, but will be less robust to the types of bit errors against which convolutional encoding was meant to provide protection.

We were able to alleviate this problem by using the average value of the final state metric from the Viterbi decoder as a tiebreaker. The sequence with the highest average metric was considered to have been decoded correctly. With this modification in place, repeated runs of the simulator have resulted in no set of 1000 repetitions with more than one error. Recall that each set of repetitions is made under a specific assumption regarding the frame structure and the values of the transmission parameters. Under most sets of assumptions, 1000 repetitions will result in no errors. This is especially true for frame structures with long sync words, including all of the TM sync words. Thus, the total error rate is less than .001. We continue to investigate the reasons for the few residual errors.

7. CONCLUSIONS

Our simulation results show that using frame-structure information, as describe in Section 4, is a viable approach for estimating transmission parameters that do not have a clear statistical signature. We note that we had more success in the estimation task when the CCSDS TM protocol was employed, than when HDLC was used. This is because TM frames include long sync words that are unlikely to occur spuriously in invalid frames. This makes it easier to recognize a correctly decoded sequence.

We were aided in our task by the fact that the number of possible transmission-parameter values was small; e.g., only one rate for the convolutional encoding. Larger numbers of parameters, and larger numbers of possible values for each parameter, would have made it necessary to produce a much greater number of decoded sequences to examine. This would have made the computational task more burdensome, of course, but because we are considering the receiver at the ground station, we expect that sufficient computing power will be available. We note that the decoding task described here lends itself well to parallel computing.

The more serious problem is that large numbers of decoded sequences may make more difficult the problem of deciding which one of them was decoded correctly. An example of this is the Viterbi decoding problem discussed in the previous section.

Also, it is clear that in general, deciding parameter values from statistics, as discussed in Section 2, is preferable to using frame-structure information. This is because the effects of various transmission parameters on the final output sequence are highly coupled, and each parameter adds a great deal of complexity to the problem. Note from Section 2 that the task of estimating the data rate and data format type, which uses only statistics, is completely independent from the estimation of the other parameters. In Section 2, the estimation procedure for each parameter is simple, and only slightly coupled with the other.

For this reason, it is worthwhile to try to discover statistical approaches for other parameters as well. A likely possibility is convolutional encoding, since this may be viewed, in a sense, as a filtering operation, and because it is intended to introduce redundancy.

When using frame-structure information, there is a tradeoff between the amount of such information you use and the complexity of the resulting algorithm. In order to make our algorithm more generally applicable, we used as few details of the data-link protocols as possible in performing our estimation task. We hoped only to use sync-word information, and though this approach did not prove to be

sufficient, we broke from it only when necessary. Our handling of HDLC is an example of this.

As the number of parameters to be estimated using frame–structure information increases, it is likely that the size and complexity of the estimation algorithms will grow quickly if we wish to keep the error rate low. In such a case, it may be worthwhile to reduce the number of possible parameter values that are tested. These values would be the ones that are used most commonly, and would result in a correct determination most of the time.

REFERENCES

[1] S. Horan, *Introduction to PCM Telemetering Systems,* 2nd ed., Boca Raton, FL: CRC Press, 2002.

[2] *Space Network Users' Guide,* Rev. 8, Mission Services Program Office, NASA Goddard Space Flight Center, Greenbelt, Maryland, 2002.

[3] R. L. Peterson, R. E. Ziemer, and D. E. Borth, *Introduction to Spread-Spectrum Communications,* Englewood Cliffs, NJ: Prentice-Hall, 1995

[4] P. DeLeon, Q. Wang, S. Horan, and R. Lyman, "A Design for Satellite Ground Station Receiver Autoconfiguration," *International Telemetering Conference*, Las Vegas, October 2003.

[5] A. Papoulis, *Probability, Random Variables, and Stochastic Processes,* 3rd ed., New York: McGraw-Hill, 1991.

[6] U. D. Black, *Data Link Protocols,* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[7] *TM Synchronization and Channel Coding,* CCSDS 130.0-R-1, Consultative Committee for Space Data Systems, 2002.

BIOGRAPHIES

Raphael J. Lyman *holds a Ph.D. in electrical engineering from the University of Florida, Gainesville. He is currently an assistant professor in the Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces. His teaching and research interests include satellite communications, mobile radio, signal processing and estimation theory.*

Qingsong Wang *holds a Master's degree in Physics from Bejing Normal University. He is currently pursuing his Ph.D. degree in electrical engineering from New Mexico State University.*

Phillip De Leon *received the B.S. Electrical Engineering and the B.A. in Mathematics from the University of Texas at Austin, in 1989 and 1990 respectively and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Colorado at Boulder, in 1992 and 1995 respectively. Previously he worked at AT&T (and later Lucent Technologies) Bell Laboratories in Murray Hill, N.J. Currently, he serves as an Associate Professor in the Klipsch School, Director of the Advanced Speech and Audio Processing Laboratory, and Associate Director of the Center for Space Telemetering and Telecommunications at NMSU. His research interests are in adaptive-, multirate-, real-time-, and speech-signal processing as well as wireless communications. Dr. De Leon is a senior member of IEEE.*

Stephen Horan *(S'79 - M'83 - SM'96) received an A.B. degree in Physics from Franklin and Marshall College in 1976, an M.S. degree in astronomy in 1979, the M.S.E.E degree in 1981, and the Ph.D. degree in electrical engineering in 1984 all from New Mexico State University. From 1984 through 1986, he was a software engineer and systems engineer with Space Communications Company at the NASA White Sands Ground Terminal where he was involved with the software maintenance and system specification for satellite command and telemetry systems and operator interfaces. In 1986 he joined the faculty at New Mexico State University where he is presently a Professor and holder of the Frank Carden Chair in Telemetering and Telecommunications in the Klipsch School of Electrical and Computer Engineering. His research and teaching interests are in space communications and telemetry systems. Dr. Horan is the author of* Introduction to PCM Telemetering Systems *published by CRC Press. He is also a Senior Member of both the IEEE and AIAA, and a member of Eta Kappa Nu.*