Software Package for Performing Experiments About the Convolutionally Encoded Voyager 1 Link

U. Cheng
Communications Systems Research Section

A software package enabling engineers to conduct experiments to determine the actual performance of long constraint-length convolutional codes over the Voyager 1 communication link directly from JPL has been developed. Using this software, engineers are able to enter test data from the Laboratory in Pasadena, California. The software encodes the data and then sends the encoded data to a personal computer (PC) at the Goldstone Deep Space Complex (GDSC) over telephone lines. The encoded data are sent to the transmitter by the PC at GDSC. The received data, after being echoed back by Voyager 1, are first sent to the PC at GDSC, and then are sent back to the PC at the Laboratory over telephone lines for decoding and further analysis. All of these operations are fully integrated and are completely automatic. Engineers can control the entire software system from the Laboratory. The software encoder and the hardware decoder interface were developed for other applications, and have been modified appropriately for integration into the system so that their existence is transparent to the users. This software provides (1) data entry facilities, (2) communication protocol for telephone links, (3) data displaying facilities, (4) integration with the software encoder and the hardware decoder, and (5) control functions.

I. Introduction

For several years, a goal of TDA Advanced Systems has been to find a convolutional code which, when concatenated with an appropriate outer Reed-Solomon code, would perform 2 dB better than the concatenated code currently used by the Voyager spacecraft [1]. When such a code was found, an "experiment" was planned: data would be encoded, transmitted to a spacecraft (e.g., Voyager 1) on a ranging channel, recovered at Goldstone, and decoded. Because the Galileo mission adopted a coding experiment with a code similar to this 2-dB code [2], [3], the original experiment is now unlikely to be performed. The software package developed for the experiment may have another use, and hence it is presented here.

The software and hardware configuration is illustrated in Fig. 1. The software package comprises two programs, both running on IBM personal computers (PCs) or compatibles. The first program is run by the engineers at the Laboratory; it is referred to as Program I in this article. The second program,

which runs on a PC at the Goldstone Deep Space Complex (GDSC), is the gateway to the transceiver; it is referred to here as Program II. The software package has eight features:

(1) menu-driven user interface

(2) data entry facilities

(3) interface to the software convolutional encoder

(4) interface to the hardware convolutional decoder

(5) stop-and-wait protocol for the telephone link with variable packet size and error detection

(6) split-screen display for the transmitted and returned data

(7) command coding with error-correction and error-detection capabilities for software operational control (see the subsequent discussion), and

(8) dedicated coding protection for the carriage-return byte

Program II has three operational modes: command, receiving, and transmitting. Program II always recognizes the "enter the command mode" command regardless of the current operational mode. For instance, by issuing this command, Program I can interrupt Program II to obtain control during the transmission of the returned data. Program I always puts Program II in the command mode before issuing any further commands. In the command mode, Program II is able to accept other commands from Program I, such as (1) enter the receiving mode, (2) enter the transmitting mode, or (3) clear all buffers. In this manner, the users of Program I can always control Program II remotely. In the receiving mode, Program II receives the encoded data from Program I. In the transmitting mode, Program II sends the returned data to Program I.

The software encoder and the hardware decoder interface were developed by Charles Lahmeyer of the Communications Systems Research Section for other applications. They have been modified appropriately for integration into the system. During transmission, the data entered by the users are first converted into a bit stream which is saved in a file. The software encoder is then invoked to encode the data. During reception, Program I converts the received data into a bit stream which is saved in a file. The decoder interface software is then invoked to decode the data.

The command coding for software operational control is explained in Section II. The stop-and-wait protocol is explained in Section III. The coding protection for the carriage-return byte is described in Section IV.

II. Command Coding

Coding for every operational command is necessary because of the noisy telephone link. Since the data are sent by bytes through the asynchronous telephone link, which has a low byte error rate, the command coding should be designed in bytes (i.e., a 256-symbol alphabet). Let a command code consisting of N bytes be denoted by $\underline{A} = \{a_1, a_2, \ldots, a_N\}$. The detection of this command code is done by straight correlation. Let $r_1, r_2, r_3, \ldots,$ denote the received bytes and let

$$D(j) = \sum_{i=1}^{N} \left(1 - W(a_i, r_{j+i})\right)$$

where $W(x, y)$ is the Hamming distance between two bytes, i.e., $W(x, y) = 0$ if $x = y$, and $W(x, y) = 1$ if $x \neq y$. Then if $D(j) > \delta$, one declares that $\underline{A}$ has been detected at the jth received symbol; otherwise, the symbol stream starting at the next received symbol will be tested. The threshold δ is predetermined, based on the error rate of the asynchronous telephone link.

In this software package, five command codes are provided to trigger the following actions, namely:

(1) enter the command mode

(2) enter the receiving mode

(3) enter the transmitting mode

(4) clear the operational buffer, and

(5) acknowledge command acceptance

The same acknowledgment code is also used in the stop-and-wait protocol for acknowledging acceptance of the most recently transmitted data packet. The first four command codes are referred to as the active command codes. They can only be issued by Program I. The acknowledgment code is a passive command code. It is issued by either program upon acceptance of a data packet or a command code. In order to allow Program I to maintain control of Program II even in noisy situations, an active command code must be repeated until an acknowledgment is received from Program II.

III. Stop-and-Wait Protocol

The stop-and-wait protocol is used for both the forward (JPL to GDSC) and the return (GDSC to JPL) telephone links to control the byte error rate. The probability of undetected errors over the telephone links must be made very low compared to the error rate of the Voyager 1 link. Low byte error rate can be accomplished through the addition of an adequate

number of parity-check bytes. In the current design, every data packet is protected by five parity-check bytes. These parity-check bytes are generated as follows:

(1) one byte derived by the overall bytewise exclusive-OR operation, and

(2) four bytes derived by the overall long-integer sum [4]

Each packet can be of any size up to 100 bytes. The first two bytes of each packet are the packet length. Two bytes are reserved for packet length out of consideration for software expandability. This allows the software to be usable should packets of length greater than 256 bytes be handled in the future. Each packet is transmitted repeatedly until an acknowledgment is received. On the receiving side, an acknowledgment is sent if a packet is received correctly.

The data transmission session is started by Program I using one of two commands: "enter the receiving mode" or "enter the transmitting mode." The session is terminated by Program I using the "enter the command mode" command. During data transmission, the programs display the number of packets transmitted, the number of packets received, and the length of the current packet. Program I attempts to collect the returned data from Program II whenever possible. For instance, when users are looking at the menu or examining the returned data, Program I will put Program II in the transmitting mode automatically in order to obtain as much returned data as possible. This strategy maximizes the link utilization and minimizes the waiting time for the users.

IV. Coding Protection for the Carriage-Return Byte

An important feature of this software package is to let users send plain-English text. Users can easily compare the transmitted text with the returned text. Errors which are not corrected by the convolutional decoder will be obvious to them.

In displaying the plain-English text, there is a control character of particular importance, the carriage-return character. This character tells when a new line begins. If the carriage-return character is received in error, the received text will have a disturbed format that makes the visual comparison between the transmitted and the received text more difficult. Therefore, it is worth protecting this character by coding. The codeword must be designed so that when Program I searches for it in the data bit stream, the probability of detecting it is high but that of false alarm is low. The data bit stream has a bit error rate ranging from 10^{-5} to 0.2 (typical of the convolutionally encoded Voyager 1 link). The command coding concept described in Section II can be applied here. In this case, however, the codeword should be over binary alphabets and bitwise correlation should be performed:

$$D(j) = \sum_{i=1}^{M} \left(1 - W(c_i, b_{j+i})\right)$$

where $b_1, b_2, b_3, \ldots$, are the received bits, $\underline{C} = (c_1, c_2, \ldots, c_M)$ is the carriage-return codeword, and $W(x, y)$ is the Hamming distance between two bits, i.e., $W(x, y) = 0$ if $x = y$, and $W(x, y) = 1$ if $x \neq y$. Then if $D(j) > \delta_1$, one declares that $\underline{C}$ has been detected at the jth received bit; otherwise, the bit stream starting at the next received bit will be tested. The threshold is predetermined based on the bit error rate.

When Program I is transmitting, it examines each byte prior to transmission. If a carriage-return byte is encountered, the carriage-return codeword is sent instead. When Program I is receiving, it examines the received bit stream constantly to detect the carriage-return codeword. If this codeword is found, it is removed from the data and a carriage-return byte is inserted at that position.

V. Conclusion

The software described in this article enables engineers to conduct experiments to determine the actual performance of long constraint-length convolutional codes over the Voyager 1 communication link from the Laboratory. The concept behind this software should also be useful for other applications. Its control features allows engineers to conduct off-lab experiments from the Laboratory. The implemented protocol guarantees nearly error-free transmission of data from the remote sites to the Laboratory over standard telephone lines. High-speed modems can be used if a great amount of data must be transferred.

References

[1] J. H. Yuen and Q. D. Vo, "In Search of a 2-dB Coding Gain," *TDA Progress Report 42-83*, vol. July–September 1985, Jet Propulsion Laboratory, Pasadena, California, pp. 26-33, November 15, 1985.

[2] S. Dolinar, "A New Code for Galileo," *TDA Progress Report 42-93*, vol. January–March 1988, Jet Propulsion Laboratory, Pasadena, California, pp. 83–96, May 15, 1988.

[3] S. Arnold and F. Pollara, "A Software Simulation Study of the Long Constraint Length VLSI Viterbi Decoder," *TDA Progress Report 42-94*, vol. April–June 1988, Jet Propulsion Laboratory, Pasadena, California, pp. 210–221, August 15, 1988.

[4] *Microsoft C Language Reference Manual*, Microsoft Corporation, Bothell, Washington, 1987.

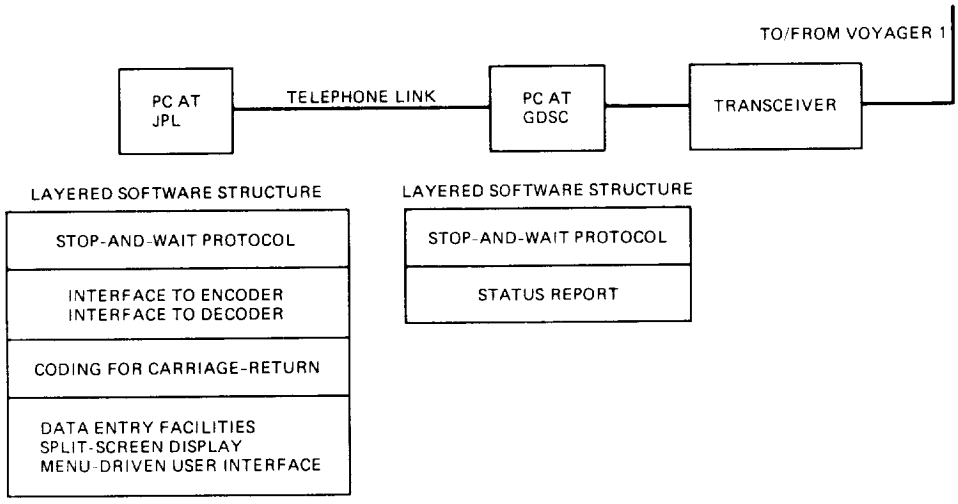

Fig. 1. Software and hardware configuration.