A Novel ARQ Technique using the Turbo Coding Principle

Krishna R. Narayanan, *Student Member, IEEE,* and Gordon L. Stüber, *Senior Member, IEEE*

***Abstract*—A novel Automatic Repeat reQuest (ARQ) technique based on the Turbo coding principle is presented. The technique uses the log-likelihood ratios generated by the decoder during a previous transmission as *a priori* information when decoding retransmissions. Simulation results show a significant decrease in frame error rate, especially at low-to-moderate E_b/N_0 .**

I. Introduction

TURBO CODING refers to parallel concatenation of recursive systematic convolutional (RSC) codes, in which the data sequence is first encoded by an RSC encoder, then permuted using a pseudorandom interleaver and encoded by an identical RSC encoder. The structure of the Turbo encoder is shown in Fig. 1. The systematic output, along with the parity outputs of the two RSC encoders, RSC1 and RSC2, are then transmitted over the channel. The parity bits can be punctured to produce different rate Turbo codes.

Since their introduction by Berrou *et al.* [1], Turbo codes have been shown to provide tremendous coding gains for both additive white Gaussian noise (AWGN) and flat Rayleigh-fading channels. Recently, Benedetto and Montorsi have shown that the main ingredients of the Turbo coding scheme are the recursive encoder and the pseudorandom interleaver [2]. The interleaver permutes sequences that produce low output weights in RSC1, such that the outputs of RSC2 corresponding to the permuted sequences are of higher output weight. This results in a larger overall free distance and, hence, lower bit-error rates.

The Turbo decoder consists of many identical stages of a decoding module. Each module in turn consists of two soft-output decoders, capable of accepting and delivering soft outputs. We consider the maximum *a posteriori* (MAP) decoder as the soft-output decoder. In particular, the log-likelihood ratio (LLR) output of the MAP decoder for each bit can be written as [3]

$$L(d_k) = L_{\text{sys}} + L_{\text{parity}} + L_{\text{ext}} \tag{1}$$

where L_{sys} is the contribution of the systematic bit at time k L_{parity} is the contribution of all other systematic bits except the one at time k, and all parity bits of one encoder (say, RSC1) to the information bit at time k are based on the constraints imposed by the encoder (RSC1). L_{ext} is the extrinsic information made available by the other MAP decoder. Analogous to L_{parity}, L_{ext} at time k is the contribution of all systematic bits except the one at time k and all parity bits of the other encoder (RSC2) to the information bit at time k, based on the code constraint (RSC2). L_{parity} is then used as L_{ext} in the next decoder, which leads to an iterative structure.

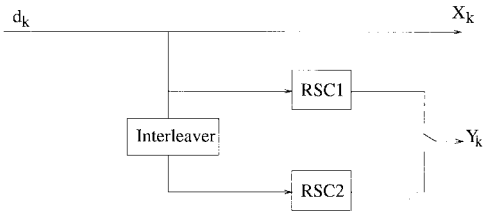

Fig. 1. Rate-1/2 Turbo encoder.

When data is transmitted in the form of packets, it is common to use Automatic Repeat reQuest (ARQ) or retransmission schemes. Whenever a packet is deemed to be erroneously decoded (assumed here to be from the failure of a cyclic redundancy code (CRC) check), the receiver requests retransmission of the packet until the packet is deemed to be correctly decoded.

When convolutional codes or block codes are used, various techniques have been suggested for combining the retransmissions of a packet to improve performance. Maximum-likelihood packet combining [4], in which a maximum-likelihood rule is used to combine the retransmissions is one such a technique. Another technique is code combining, or Type III ARQ scheme [5], in which a punctured code is used during the first transmission and the punctured bits are transmitted during the retransmission. The receiver then combines the two packets, which together correspond to the unpunctured lower rate code. In this letter, we propose a new ARQ and packet combining technique, that uses the principles of Turbo coding and iterative decoding. This technique is applicable to RSC codes, block codes, and Turbo codes that can be decoded using a soft-output decoder. The proposed ARQ scheme does not require significant additional computational complexity or storage requirements at the receiver when a soft-output decoder is used to decode the transmissions.

II. Turbo ARQ

This section explains the modifications in the Turbo encoder and decoder structure necessary to permit Turbo combining

Manuscript received November 26, 1996. This work was supported by the National Science Foundation under Grant NCR-9304185. The associate editor coordinating the review of this paper and approving it for publication was Dr. A. Elhakeem.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: stuber@eecom.gatech.edu).

Publisher Item Identifier S 1089-7798(97)02936-0.

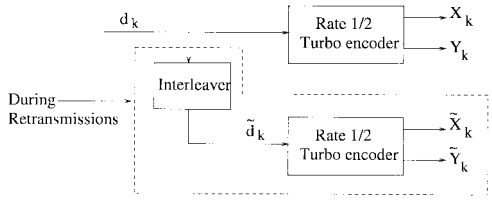

Fig. 2. Encoder structure.

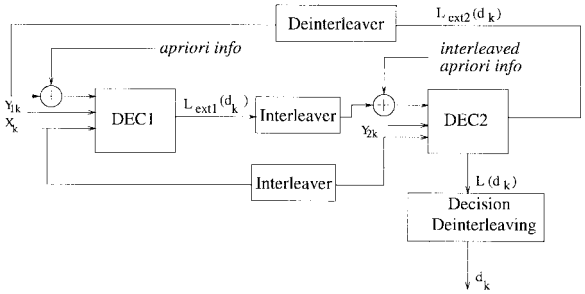

Fig. 3. Decoder structure.

of retransmissions. We assume a rate-1/2 Turbo code as the error correcting code during every transmission. The use of a Turbo code as the error-correcting code is not mandatory. Any recursive systematic code can be used as the error-correcting code. In fact, any code capable of being decoded using a soft-output decoder can be used, which also includes block codes. A simple stop-and-wait ARQ protocol is assumed; that is, the transmitter waits until an acknowledgment (ACK) is received from the receiver before transmitting the next packet.

A. Encoder Structure

The encoder structure, shown in Fig. 2, is a modified version of the Turbo encoder. During the first transmission of a packet, a rate-1/2 Turbo code is used. If a negative ACK is received, indicating an erroneously decoded packet at the receiver, the data sequence is interleaved using a pseudo-random interleaver and then encoded by an identical Turbo encoder and retransmitted. The pseudorandom interleaver is different from that used within the Turbo encoder. This is essential and helps in improving the performance by permuting input sequences that produce low output weight sequences during the first transmission, so that the output weight during retransmissions is higher. Now, the prior transmission and the retransmission can be effectively combined at the receiver. During every additional retransmission, should it be necessary, a different pseudo-random interleaving pattern is used.

B. Decoder Structure

The decoder structure is similar to a Turbo decoder, with a few modifications. Whenever the packet is deemed to be in error, a negative ACK is sent and the LLR's of each bit are stored. When a retransmission is received these LLR's are used as *a priori* information during the decoding of the packet. Recall that the LLR output of the component MAP decoders (DEC1 and DEC2 in Fig. 3) corresponding to each decoded bit can be expressed as in (1). Normally, at the beginning of decoding, there is no *a priori* or extrinsic information about each bit, i.e., $\Pr(d_k = 0) = \Pr(d_k = 1) = 1/2$. Therefore, $L_{\text{ext}}(d_k) = 0$, at the beginning of the first decoding stage. However, in the case of retransmissions, the LLR's corresponding to each bit from the previous transmission can be used as *a priori* information. Therefore, instead of setting $L_{\text{ext}}(d_k) = 0$, it can be set to $L'(d_k)$, the LLR from the previous transmission. The *a priori* information, $L'(d_k)$, can be combined with the parity information produced by DEC1 and used in the second component decoder, DEC2. Specifically, the new extrinsic information input to DEC2 can be set to be $L_{\text{parity}} + L'(d_k)$, where L_{parity} has same meaning as in (1). This is similar to the technique used in [6] for combining the extrinsic information when decoding Turbo codes with more than two component encoders. This can be done, only because the *a priori* information was generated from a previous transmission, and is uncorrelated with the extrinsic information generated by the component decoders themselves. $L'(d_k)$ can be used in a similar fashion, in the future stages of the decoding also. The output of the MAP decoders can then be written as

$$L(d_k) = L_{\text{sys}} + L_{\text{parity}} + L_{\text{ext}} + L'(d_k). \tag{2}$$

Turbo coding then continues in an iterative fashion, until the packet is deemed to be correctly decoded. Thus after every retransmission, the code corresponds to a lower rate code. For example, after the first retransmission, the two packets together constitute a rate-1/4 Turbo code.

III. Simulation Results

The above technique was simulated using a rate-1/2, constraint length 5 (or memory 4, or 16-state), Turbo code with generator polynomials $(37, 21)_8$ as the error-correction code. The block length was 1024 b, and the channel was assumed to be an AWGN channel. A 16-b CRC code with generator polynomial $x^{16} + x^{15} + x^2 + 1$ was used to detect errors. The pseudorandom interleaving pattern was generated using the recursion

$$X_{n+1} = (aX_n + c) \bmod N \tag{3}$$

where N is the block length, and a and c are parameters, such that $a - 1$ is a multiple of all prime factors of N and c is relatively prime to N and $a - 1$ is divisible by 4 if N is divisible by 4 [7]. Every time a new interleaving pattern is required, different values of a and c are used. This obviates the need to store the interleaving pattern in the receiver, but it is enough to simply store the different combinations of a and c.

A. Frame Error Rate (FER) Performance

Fig. 4 compares the FER against E_b/N_0 for the proposed Turbo ARQ scheme and a simple ARQ scheme with no packet combining, based on the same throughput. At low E_b/N_0, Turbo combining is shown to significantly reduce the FER. This is because of the effective use of the information in all of the transmissions of a packet, which corresponds to using a lower rate code. As the E_b/N_0 increases, the improvement in performance decreases, because the retransmissions are infrequent. However, when the E_b/N_0 decreases due to varying

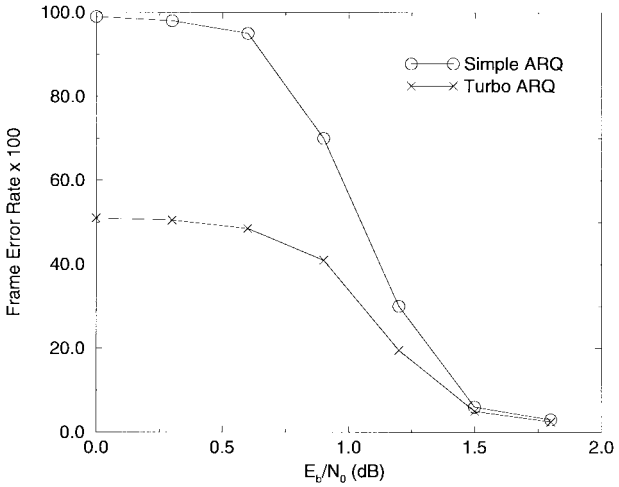

Fig. 4. FER comparison over AWGN channel.

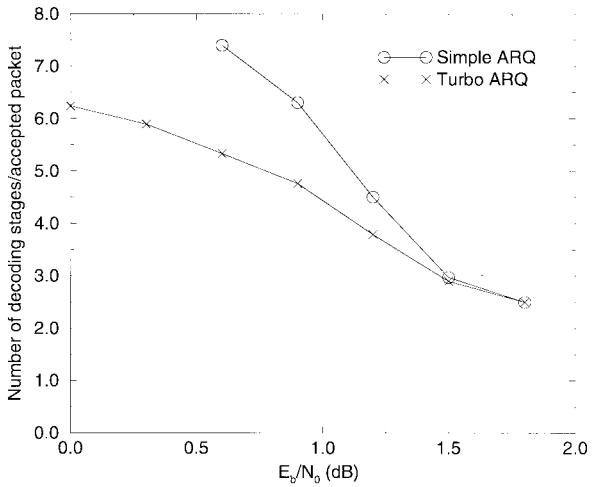

Fig. 5. Decoder complexity comparison over AWGN channel.

channel conditions, the Turbo ARQ scheme still maintains a low-to-moderate FER, while the FER for the simple ARQ scheme increases drastically.

B. Decoder Complexity

We propose to use the CRC check to stop the iterations in a Turbo decoder, i.e., after every decoding stage or iteration, the decoded data is checked for errors using the CRC check. If the checksum is zero, then no more iterative decoding stages are required and the packet is accepted. This guarantees that the minimum number of decoding stages are used to decode a packet. If the checksum is not zero after four decoding stages, then a retransmission is requested. This reduces the complexity and latency of the decoder. Furthermore, if the packet is not decoded correctly within three transmissions, then a ''Failure'' is declared and the next packet is transmitted. The use of *a priori* information from previous transmissions reduces the required number of decoding stages during every retransmission, thereby decreasing the decoder complexity. Fig. 5 compares the average number of decoding stages required for decoding each accepted packet. Each decoding stage consists of two MAP decoders, one for each component encoder. The average is calculated only for accepted packets, i.e., failures are not included in calculating the average. Observe that the average number of decoding stages required is significantly smaller with the Turbo ARQ scheme than with the simple ARQ scheme. As with the FER performance, the improvement in performance at high E_b/N_0 is only marginal, because the rate-1/2 Turbo code by itself has very good error-correction capability.

IV. Concluding Remarks

We have proposed a simple and effective ARQ technique based on the Turbo coding principle for packet data transmission. The technique is general and can be used with recursive systematic convolutional codes, Turbo code, or block codes. The technique is clearly not the best technique to combine the retransmissions. However, the advantages of the proposed technique are that it is simple, it does not increase the decoder complexity beyond that required for the normal decoding of the original rate Turbo code, and it does not impose storage requirements at the receiver (only the LLR's corresponding to the previous transmissions need to be stored). Furthermore, a technique has been proposed to stop the iterations in a Turbo decoder based on the CRC check, which reduces the complexity of decoding, without sacrificing the performance.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes,"in *Proc. ICC '93*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[2] S. Benedetto and G. Montorsi, "Unveiling Turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.

[3] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar 1996.

[4] D. Chase, "Code combining—A maximum-likelihood decoding approach for combining an arbitrary number of noisy packets," *IEEE Trans. Commun.,* vol. 33, pp. 385–393, May 1985.

[5] S. Kallel, "Complementary punctured convolutional (CPC) codes and their applications,"*IEEE Trans. Commun.*, vol. 43, pp. 2005–2009, June 1995.

[6] D. Divsalar, and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications," Jet Propulsion Lab., Pasadena, CA, Telecommunications and Data Acquisition Progress Rep. 42-121, pp. 66–77, May 1995.

[7] G. C. Clark, Jr. and J. B. Cain, *Error-Correction Coding for Digital Communications.* New York: Plenum, 1981.