TDA Progress Report 42-122 August 15, 1995

Analysis of Automatic Repeat Request Methods for Deep-Space Downlinks

F. Pollara and L. Ekroot
Communications Systems and Research Section

Automatic repeat request (ARQ) methods cannot increase the capacity of a memoryless channel. However, they can be used to decrease the complexity of the channel-coding system to achieve essentially error-free transmission and to reduce link margins when the channel characteristics are poorly predictable. This article considers ARQ methods on a power-limited channel (e.g., the deep-space channel), where it is important to minimize the total power needed to transmit the data, as opposed to a bandwidth-limited channel (e.g., terrestrial data links), where the spectral efficiency or the total required transmission time is the most relevant performance measure. In the analysis, we compare the performance of three reference concatenated coded systems used in actual deep-space missions to that obtainable by ARQ methods using the same codes, in terms of required power, time to transmit with a given number of retransmissions, and achievable probability of word error. The ultimate limits of ARQ with an arbitrary number of retransmissions are also derived.

I. Introduction

A major concern in data communications is the control of transmission errors caused by channel noise so that error-free data can be delivered to the user. Error control systems that rely only on a one-directional (forward) channel are called forward error correction (FEC) systems. Systems that make use of a reverse or feedback channel are called automatic repeat request (ARQ) systems and are based on protocols that request retransmission of data blocks when errors are detected. ARQ systems trade time for link margin and aim at returning *all* of the data reliably.

It is well known [2, p. 213] that the capacity of a memoryless channel cannot be increased by using a feedback channel.[1] However, the implementation complexity for a given performance goal can be considerably reduced by using the feedback channel for requesting retransmission of data frames. Situations where the probability of frame error needs to be virtually zero, such as for heavily compressed data, or where large link margins are imposed by poorly predictable channel impairments, are particularly suitable for retransmission schemes.

[1] In this article, the feedback channel is assumed to be noiseless. This is a realistic assumption since the uplink channel used for deep space typically operates at a high signal-to-noise ratio (SNR), includes powerful error detection capabilities, and already employs ARQ in the uplink direction.

We consider the use of ARQ methods on the downlink channel of deep-space missions. Some rudimentary forms of ARQ are already used in deep-space missions, where blocks of corrupted or lost data are isolated and requested again from the spacecraft. Retransmission methods have been considered, particularly as a playback strategy to overcome weather outages. We propose to use retransmission methods in a more systematic and automatic fashion, based on the error detection capability of existing codes. This implies a wider use of the uplink channel, not only for commands, but as a true "reverse" channel representing an integral part of the communication system. The effects of ARQ methods on the downlink performance must, therefore, be included in the link analysis. This article considers ARQ methods on a power-limited channel (e.g., the deep-space channel), where it is important to minimize the total power needed to transmit the data, as opposed to a bandwidth-limited channel (e.g., terrestrial data links), where the spectral efficiency or the total required transmission time is the most relevant performance measure.

We consider an interplanetary spacecraft transmitting frames of data to Earth. The frame consists of at least one Reed–Solomon (RS) codeword and may have an identification header. In this article, we limit the analysis to the case where a frame is one RS codeword. The frame is convolutionally encoded before transmission. Each RS codeword has redundancy, or parity symbols, that may be used for either error correction or error detection.

An (n, k) RS code with minimum distance $d_{min} = n - k + 1$ can correct all received words containing c errors and e erasures within the constraint $2c + e < d_{min}$, when hard-decision decoding is used. If the received word is within c errors and e erasures of a valid word, then the decoder will output the correct codeword. If the selected codeword is not the transmitted one, then a decoder error has occurred. If there is no codeword within c errors and e erasures, then a decoder failure is declared, and retransmission is requested. A similar situation occurs when a soft-bounded distance decoder is used. Given the decoding radius, we can calculate analytically the exact probability of decoder error and decoder failure for linear block codes [5].

An ARQ system must at least rely on error detection. The system we will discuss here uses a convolutional code concatenated with a (255,223) RS code for FEC. The convolutional codes considered here are constraint length $K = 7$ rate 1/2, $K = 15$ rate 1/4, and $K = 15$ rate 1/6. The RS code is used for both error correction and detection. Such a system is in the category of type I hybrid ARQ [10, pp. 393–423].

Section II describes the concatenated system at the heart of the ARQ system described in Section III. More thorough coverage of the concatenated system without ARQ can be found in [1], [3], and [9]. Section V describes features that are not examined in depth in this article, such as the effect of combining multiple copies of received words.

We have not yet fully considered problems that may be encountered with errors in the frame header or with frames that contain more than one codeword. Thus far, we have assumed a noiseless feedback channel, and we have ignored the effect of undetected errors, which occur with extremely low probability for the codes considered.

II. The Reference System

The nominal system is a concatenated system without ARQ. We consider a t-error-correcting (n, k) RS code on GF(2^8), with $n = 255$, $k = n - 2t = 223$, and $t = 16$. This means that the code sends k 8-bit symbols of information using n 8-bit symbols and that it can correct all patterns of t or fewer errors. This code is used as the outer code of a concatenated system, as shown in Fig. 1. We assume for this article that there is infinite interleaving between the RS encoder and the convolutional encoder, which implies that the errors at the input of the RS decoder can be modeled as independent. This allows a simpler analysis that can be used to bound the performance of a more practical implementation.

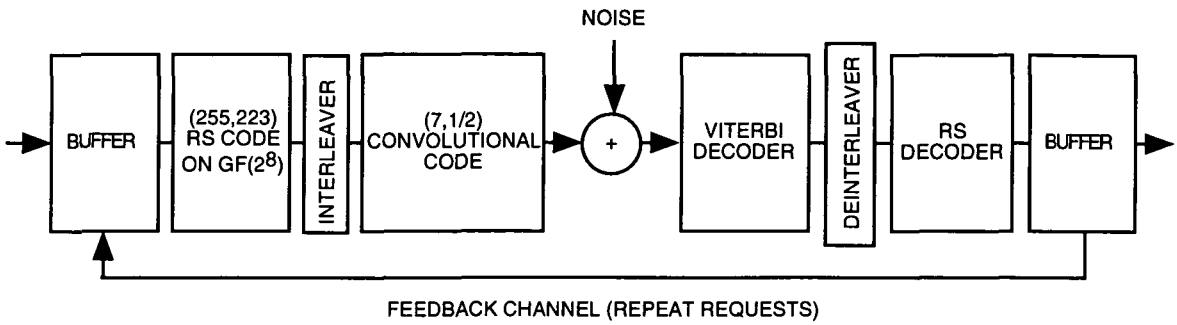

Fig. 1. The ARQ system.

For this system, there are several different SNRs that will be discussed. There is an SNR for bits that are convolutionally encoded and decoded. Let s_V be the SNR for the channel and inner (convolutional) code only. The concatenated system uses more power-per-information bit because k symbols are sent using n coded symbols, increasing the signal power by a factor of n/k. Let $s = (n/k)s_V$ denote the SNR of the concatenated system. Since this concatenated RS and convolutional system is the nominal system in this article, s will be called the nominal SNR.

We can measure the symbol error rate, $v_s(s)$, at the output of the Viterbi decoder by simulation. Since errors at the RS decoder input may be assumed to be independent because of infinite interleaving, the probability of decoder failure and decoder error, i.e., the probability that more than the correctable number of symbols are in error, is given by

$$p(s) = \sum_{i=t+1}^{n} \binom{n}{i} v_s(s)^i [1 - v_s(s)]^{n-i} \tag{1}$$

In this article, we use $p(s)$ to approximate the probability of decoding failure, and we ignore the probability of decoding error, since it is negligible for the (255,223) RS code. Specifically, if more than $t = 16$ errors occur, the probability of decoder error is less than $1/t! \approx 4.8 \times 10^{-14}$ [8].

The simulated symbol error rate at the output of the Viterbi decoder for the (7,1/2) code and the calculated word error rate of the (255,223) RS code assuming infinite interleaving are shown in Table 1.[2] Similar simulated data for the (15,1/4) and (15,1/6) codes are given in Tables 2 and 3.

Table 1. Simulated symbol error rate at the output of the Viterbi decoder (7,1/2) and the calculated word error rate of the (255,223) RS code assuming infinite interleaving.

s_V, dB	s, dB	$v_s(s)$	$p(s)$
1.05	1.63	8.53×10^{-2}	8.83×10^{-1}
1.55	2.13	3.41×10^{-2}	7.11×10^{-3}
1.85	2.43	1.91×10^{-2}	1.08×10^{-5}
1.95	2.53	1.51×10^{-2}	4.90×10^{-7}
2.05	2.63	1.14×10^{-2}	9.48×10^{-9}
2.55	3.13	3.20×10^{-3}	2.50×10^{-17}

[2] F. Pollara and S. Dolinar, "Concatenated Codes Performance at Low Bit Error Rates," JPL Interoffice Memorandum 331-88.2-043 (internal document), Jet Propulsion Laboratory, Pasadena, California, July 13, 1988.

Table 2. Simulated symbol error rate at the output of the Viterbi decoder (15,1/4) and the calculated word error rate of the (255,223) RS code assuming infinite interleaving.

s_V, dB	s, dB	$v_s(s)$	$p(s)$
0.10	0.68	4.22×10^{-2}	4.397×10^{-2}
0.30	0.88	2.24×10^{-2}	7.735×10^{-5}
0.50	1.08	1.08×10^{-2}	4.327×10^{-9}

Table 3. Simulated symbol error rate at the output of the Viterbi decoder (15,1/6) and the calculated word error rate of the (255,223) RS code assuming infinite interleaving.

s_V, dB	s, dB	$v_s(s)$	$p(s)$
0.10	0.68	2.28×10^{-2}	9.453×10^{-5}
0.30	0.88	1.19×10^{-2}	1.674×10^{-8}
0.50	1.08	5.79×10^{-3}	3.364×10^{-13}

Fitting a curve to the simulated Viterbi decoder symbol error rates give us the approximation $V_s(s) \approx v_s(s)$. The fit is of the form $V_s(s) = e^{a_0+a_1 s+a_2 s^2}$, where the coefficients in the exponent are given for each code in Table 4.[3] The comparisons of original data $v_s(s)$ and curve fits $V_s(s)$ for each code are shown in Fig. 2. Using the curve fit $V_s(s)$ for $v_s(s)$ in Eq. (1), we get an approximation for the probability of RS decoder failure $P(s)$:

$$P(s) = \sum_{i=t+1}^{n} \binom{n}{i} V_s(s)^i [1 - V_s(s)]^{n-i}$$

The comparison of the probability of an undecodable word at the output of the RS decoder from both the simulated v_s and from the fit V_s is shown in Fig. 3.

III. The ARQ System

A. ARQ Protocols

The possible implementations of ARQ protocols fall into different categories, all of which include automatic requests for retransmission of data that are deemed unreliable by the receiver. The performance of ARQ protocols is often measured by the accepted-packet error rate and the throughput, which is the average number of information bits accepted by the receiver per packet sent. The variables of interest for deep-space applications are the required transmitter power for a given data rate, the required probability of block error, the time available for transmission, the number of retransmissions, the error detection capability of the code, and the round-trip delay. These variables affect the onboard complexity and memory requirements and the ground operational complexity.

[3] This fit is for s measured in dB, which yields a "quadratic" exponent instead of the more commonly used "linear" exponent for s not in dB reported, for example, in S. Dolinar, "Empirical Formula for the Performance of the Recommended (15,1/6) Convolutional Code," JPL Interoffice Memorandum 331-90.2-060 (internal document), Jet Propulsion Laboratory, Pasadena, California, October 12, 1990.

Table 4. Viterbi symbol error rate fits, where each $V_S(s)$ is of the form $e^{a_0+a_1 s+a_2 s^2}$ and s is in dB.

Code	a_0	a_1	a_2
(7,1/2)	−0.993617	−0.235839	−0.409607
(15,1/4)	−1.731157	−1.291763	−1.201945
(15,1/6)	−2.095216	−1.882523	−0.877300

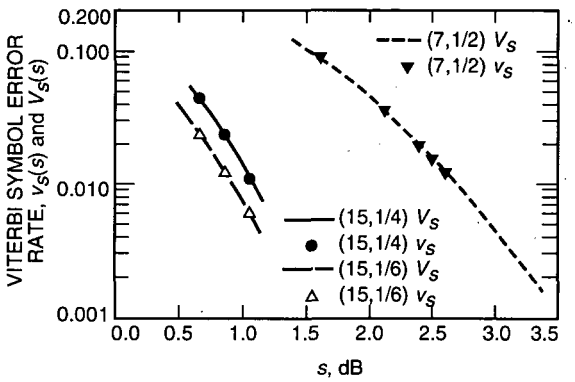

Fig. 2. Original data v_S (s) and curve fit $V_S(s)$.

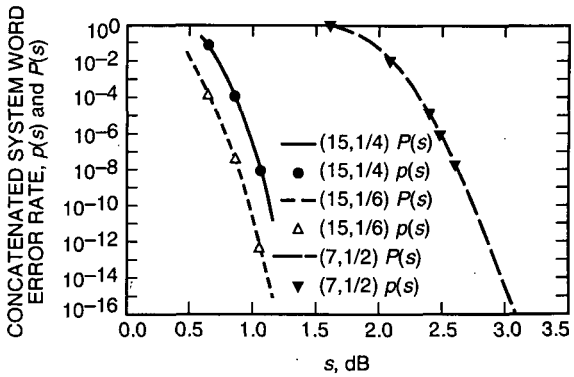

Fig. 3. For the concatenated systems, the word error rates from the simulation $p(s)$ and from the fitted curves $P(s)$ are shown as a function of $s(E_b/N_0)$.

The most appropriate protocols are based on selective repeat, where the transmitter retransmits only the packets requested by the receiver. Here the throughput does not depend on the round-trip transmission delay [10, p. 399]. Other protocols that have a throughput depending on round-trip delay are described here for contrast.

(1) The "stop and wait" ARQ protocol is one in which the transmitter sends one packet and waits for either an acknowledgment or a repeat request before sending either the next packet or the previous packet again. Although this is simple to implement at both the transmitter and receiver, the throughput is very dependent on the round-trip transmission time. For deep-space applications, this is not a practical protocol.

(2) The "go back N" protocol is one in which the transmitter sends packets sequentially. When the receiver is unable to decode a packet, it sends a repeat request and stops listening to the transmitter until the requested packet arrives. When the transmitter receives the repeat request, it goes back N and starts sending sequentially again. This version has a better throughput than the stop and wait protocol. The throughput and N depend on the round-trip transmission time, but the effect of the round-trip transmission time is lower for a relatively noiseless channel than for a noisy one.

In a selective repeat system, the transmitter must keep a buffer of what it has sent so that retransmission is possible. The receiver may store unreliable frames and combine them with subsequent retransmissions to improve reliability, as we will discuss in Section V. Let R be the number of retransmissions that are either allowed or that are necessary. If some maximum number of retransmissions is allowed, then we ask what fraction of the data is received error free. If it is essential to return all of the data, then we ask how many retransmissions are necessary to return the data. The answer to both questions depends on the channel SNR.

B. Buffer Size On Board the Spacecraft

Using ARQ, every frame transmitted must be stored long enough for it to be received, processed, and (if necessary) for a repeat request to be received and processed by the spacecraft. So, the buffer size on board the spacecraft needs to be large enough to store all the data transmitted in that amount of time. The buffer size B is a function of the data rate D, the round-trip transmission time τ, and the on-ground processing time η. Specifically,

$$B = D(\tau + \eta)$$

If we consider a Mars–Earth–Mars round-trip transmission time of approximately 25 min, a ground processing time of 1 min, and a data rate of 512 kb/s, the buffer size must be approximately 100 Mbytes. The ground processing time is also an operations question since it depends on how often uplink transmissions are desired.

C. ARQ Analysis

We consider the use of ARQ to transmit a large data set. After the first transmission of the whole data set, subsequent retransmissions will repeat only the frames that were flagged as unreliable. The average amount of data that must be sent after the Rth retransmission is the amount of data sent during the Rth retransmission times the probability of decoder failure. If successive retransmissions of the same frame are decoded without reference to previous transmissions, then the fraction of data that is still in error after R retransmissions is $P(s)$ where s is the nominal SNR. Thus, the fraction of the original data that is still in error after R retransmissions is

$$P_R(s) = P(s)^{R+1}$$

If previous copies of a retransmitted word are combined with the current copy, as explained in Section V, then $P_R(s)$ may be smaller.

In this article, time refers to the amount of time the transmitter is actively sending coded data and, correspondingly, the amount of time the receiver is actively receiving data. In this way, time relates well to the amount of transmitter energy used for sending data.[4] For our purposes, references to time are normalized by the amount of time it takes to send the whole data set once. Thus, a system with the same coding, but without ARQ, sends the data back in 1 time unit. The average amount of time it takes to send the data set when R retransmissions are allowed is given by

$$t_R(s) = 1 + \sum_{i=0}^{R-1} P_i(s) = \sum_{i=0}^{R} P^i(s) = \frac{1 - P(s)^{R+1}}{1 - P(s)}$$

Note that a system that allows no more than R retransmissions of the data is done transmitting before $R+1$ units of time have past, i.e., $t_R \leq R+1$. For high SNR, few transmissions are necessary, so $t_R \approx 1$. For low SNR, the maximum number of transmissions will almost always be used, so $t_R \approx R+1$. If the number of retransmissions is unlimited, the average amount of time is given by

[4] The amount of time between when the first and last bytes of data are received is different and depends on operations. If the ground station cannot send and receive simultaneously, then there is a delay while a clump of data is being received and before a batch of repeat requests can be sent. There may also be some delay on the ground for processing. As a result, the amount of clock time it takes to send the data is different from the amount of time the transmitter is sending data.

$$t_\infty(s) = \frac{1}{1 - P(s)}$$

Since, on average, each frame is transmitted $t_R > 1$ times, for a system that allows no more than R retransmissions, the effective SNR, $s_R(s)$, is larger than the nominal SNR, s:

$$s_R(s) = st_R(s) = s\sum_{i=0}^{R} P^i(s) = s\frac{1 - P(s)^{R+1}}{1 - P(s)}$$

Note that $s_0(s) = s$. If the number of retransmissions is unlimited, the effective SNR is given by

$$s_\infty(s) = \frac{s}{1 - P(s)}$$

Performance for low SNRs is described in the Appendix, as it is primarily of academic interest. The effective rate at which data are received is $r_R(s) = 1/t_R(s)$. Note that $t_\infty(s) = 1/r_\infty(s) = 1/[1 - P(s)]$.

1. ARQ Performance Limits: Limiting Performance for Zero Error Probability. It is interesting to establish the performance limits of ARQ for a given coding system when the maximum number of retransmissions is arbitrary. The minimum effective SNR at which the system can produce vanishingly small probability of word error for arbitrary R is given by

$$s^*_\infty = \min_s[s_\infty(s)] \tag{2}$$

Accordingly, $P_R(s) \to 0$ for large R, if $s_R(s) > s^*_\infty$. Conversely, if R and s are sufficiently small so that the effective SNR $s_R(s)$ is less than s^*_∞, then vanishingly small word error probability is impossible. The minimum effective SNR s^*_∞ for a given ARQ coded system can be compared with the ultimate Shannon limit $[(2^{2r} - 1)/2r] < s^*_\infty$, where r is the rate of the concatenated system.

2. ARQ Performance Limits: Limiting Performance for Nonzero Error Probability. If the required word error probability P_ϵ is nonzero, then one has to simultaneously optimize the required effective SNR, s_ϵ, and the number of retransmissions, R. The limiting performance is given by the envelope of the curves for different R's (as shown later for the (7,1/2) code in Fig. 7) or, equivalently, by the locus of the minimum effective SNRs for each R. Since a given data volume can be transmitted reliably with power proportional to s^*_∞, a fraction $1 - P_\epsilon$ of the data volume can be transmitted with power proportional to $s_\epsilon = s^*_\infty(1 - P_\epsilon)$. Then the limiting ARQ performance is given by

$$P_\epsilon = 1 - \frac{s_\epsilon}{s^*_\infty}, \quad s_\epsilon \le s^*_\infty \tag{3}$$

It has been suggested[5] that Eq. (3) can also be explained by considering the ARQ system as an erasure channel, whose capacity can be achieved by just retransmitting codewords until they get through. Since RS codewords get through with probability $1 - P_\epsilon$, the capacity of the erasure channel is $C = 1 - P_\epsilon$, and we must have $r \le C$, i.e., the rate is limited by capacity. It can be verified[6] that $r = s_\epsilon/s^*_\infty$.

[5] Personal communication with M. Costa, Jet Propulsion Laboratory, Pasadena, California, August 1994.

[6] Ibid.

D. The (7,1/2) Convolutional Code

Three plots showing the pairwise relationships between nominal SNR, effective SNR, and probability of word error for the (7,1/2) convolutional code are shown in Fig. 4. In designing the coding system, we start from a given available nominal SNR. The effective SNR is then a measure of the average power actually used for transmission, including the penalty due to retransmissions. For example, if we start out at a nominal SNR of 2.5 dB, we would have a resulting effective SNR also of 2.5 dB, as shown by point A in Figs. 4(b) and 4(c), for $R = 1$. Point B in Figs. 4(b) and 4(c) represents an anomalous operation of the system, where the given nominal SNR is insufficient for good performance, even though the effective SNR has the same value of 2.5 dB due to the retransmission overhead. Note that, for $R = 1$, Fig. 4(c) shows that, for high nominal SNR, the effective SNR is equal to the nominal SNR and, for low nominal SNR, the effective SNR is 3-dB more than (i.e., double) the nominal SNR. The latter follows from the fact that for finite R the data can be transmitted at most $R+1$ times (and at low SNR, the maximum number of retransmissions will almost always be used), so the effective SNR due to retransmissions is approximately $10\log_{10}(R+1) + s$ for low SNR. This phenomenon at low SNR happens for all finite R, but not when the number of retransmissions is unrestricted. This effect appears on all the curves, but is at such low nominal SNRs that discussion is deferred to the Appendix.

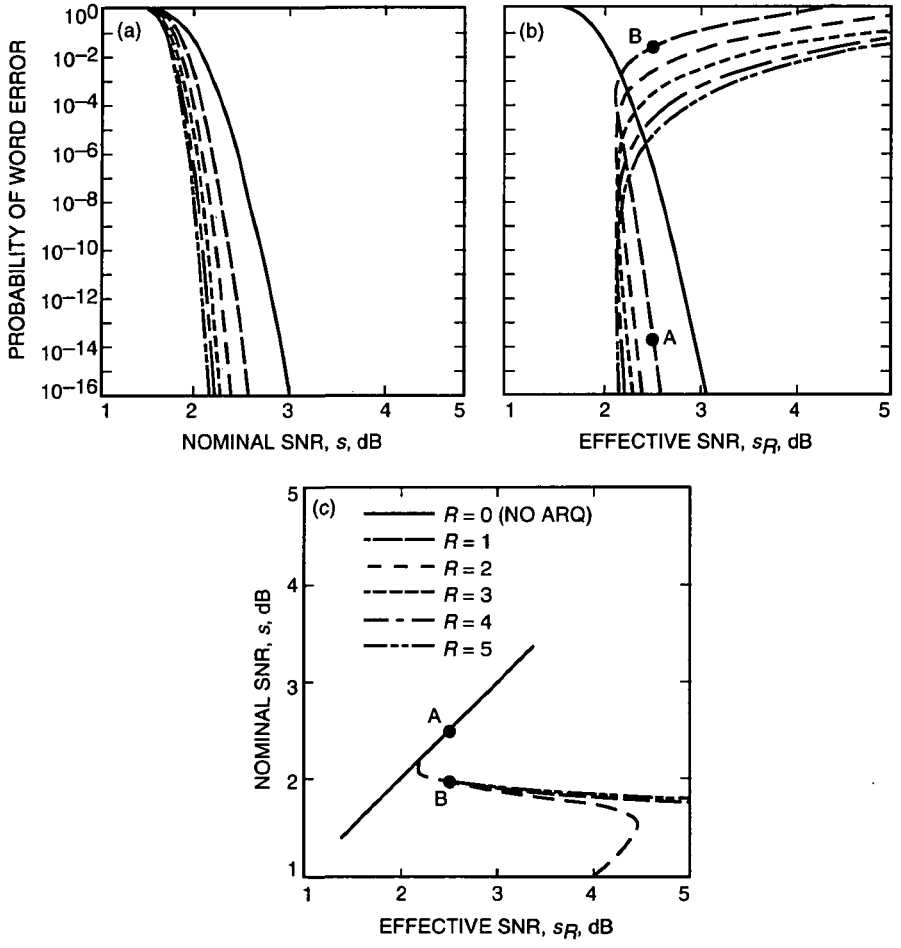

Fig. 4. Pairwise relationships between (a) nominal SNR s, (b) effective SNR $s_R(s)$, and (c) probability of word error $P_R(s)$ for the (7,1/2) + (255,223) convolutional code.

A parametric plot of $s_R(s)$ versus $t_R(s)$ in Fig. 5 shows a local minimum where the effective SNR is low and the average amount of time spent transmitting is close to 1. For the (7,1/2) code and $R > 1$, a nominal SNR $s = 2.109$ dB yields an effective SNR $s_R = 2.166$ dB, and the amount of time spent transmitting ($t_R = 1.013$) is only 1.3-percent more time than the time to transmit once. For nominal SNR higher than 2.25 dB, the channel is relatively noiseless and few retransmissions are requested. Thus, $s_R(s) \approx s$, and the ARQ system performance is comparable to the system without ARQ, as evidenced by t_R being nearly 1. As the nominal SNR gets lower than 2.1 dB, the amount of data transmitted increases and so the effective SNR s_R and time t_R also increase.

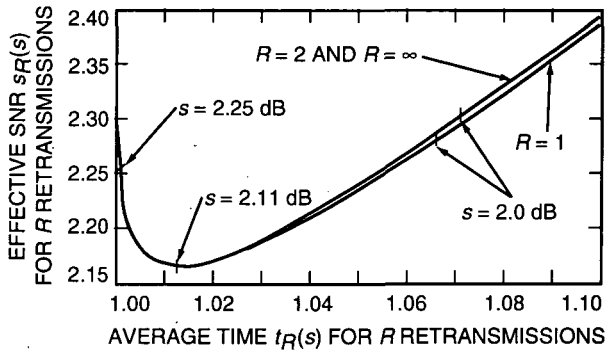

Fig. 5. Effective SNR (s_R) plotted with the average time to transmit (t_R) using the (7,1/2) + (255,223) code.

So, for an additional 0.057 dB = 2.166 dB − 2.109 dB of SNR (due to 1.3-percent more time required to transmit), the data will be delivered virtually error free, if an unlimited number of retransmissions is allowed, instead of with a probability of error of 0.013 for the same system without ARQ. This is especially useful for compressed data where errors can propagate through the data. Another advantage is that the link margin, for say a 10^{-6} word error rate, can be reduced by more than 0.3 dB, if retransmissions are used to combat unforeseen SNR reductions and 1.3-percent more time is allowed for transmission.

The fraction of the total data volume that is accepted (successfully decoded) after R retransmissions, $1 - P^{R+1}$, versus the amount of time for R retransmissions is shown in Fig. 6 for several values of nominal SNR.

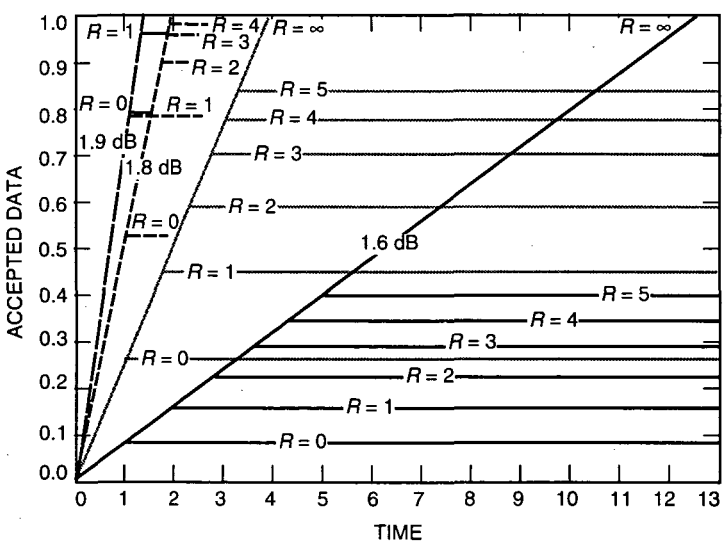

Fig. 6. Decoded data volume versus time for the (7,1/2) + (255,223) code.

For the concatenated system considered in this section, the minimum effective SNR [see Eq. (2)] is $s^*_\infty = 2.166$ dB, where this minimum is achieved at nominal SNR $s^* = 2.109$ dB. This limiting SNR value of s^*_∞ is still significantly higher than the Shannon limit $(2^{2r} - 1)/2r = -0.209$ dB, where $r = (1/2) \cdot (223/255)$. We can compute the limiting time and rate at nominal SNR s^* to be

$$t^*_\infty = t_\infty(s^*) = 1.013$$

$$r^*_\infty = r_\infty(s^*) = 0.987$$

The limiting performance curve [see Eq. (3)] is shown in Fig. 7, and it coincides with the envelope of the parametric curves of $s_R(s)$ and $P_R(s)$ for finite R.

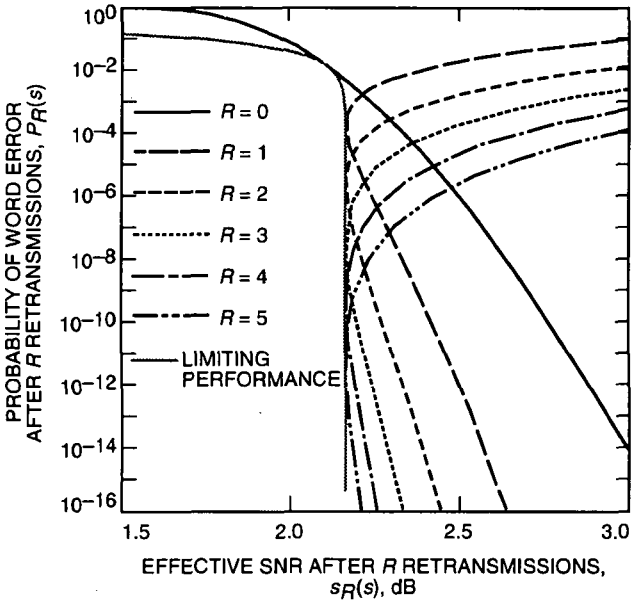

Fig. 7. $P_R(s)$ versus $s_R(s)$, (7,1/2) + (255,223) code.

E. The (15,1/4) Convolutional Code

Three plots showing the pairwise relationships between nominal SNR, effective SNR, and probability of word error for the (15,1/4) convolutional code are shown in Fig. 8. In Fig. 9, a parametric plot of effective SNR versus time shows a local minimum where the effective SNR is low and the average amount of time spent transmitting is close to 1. For the (15,1/4) code, a nominal SNR $s = 0.747$ dB yields an effective SNR $s_R = 0.782$ dB, and the amount of time spent transmitting ($t_R = 1.008$) is only 0.8-percent more time than transmitting once. For nominal SNR higher than 0.8 dB, the channel is essentially noiseless and few retransmissions are requested. As the nominal SNR gets lower than 0.740 dB, the amount of data transmitted increases and so the effective SNR s_R and time t_R also increase.

So, for an additional 0.035 dB = 0.782 dB − 0.747 dB of SNR (due to 0.795-percent more time to transmit), the data will be delivered without detected errors, if an unlimited number of retransmissions are allowed, instead of with a probability of error of 0.0079 for the same system without ARQ. The fraction of the total data volume that is accepted after R retransmissions, $1 - P^{R+1}$, versus the amount of time for R retransmissions is shown in Fig. 10.

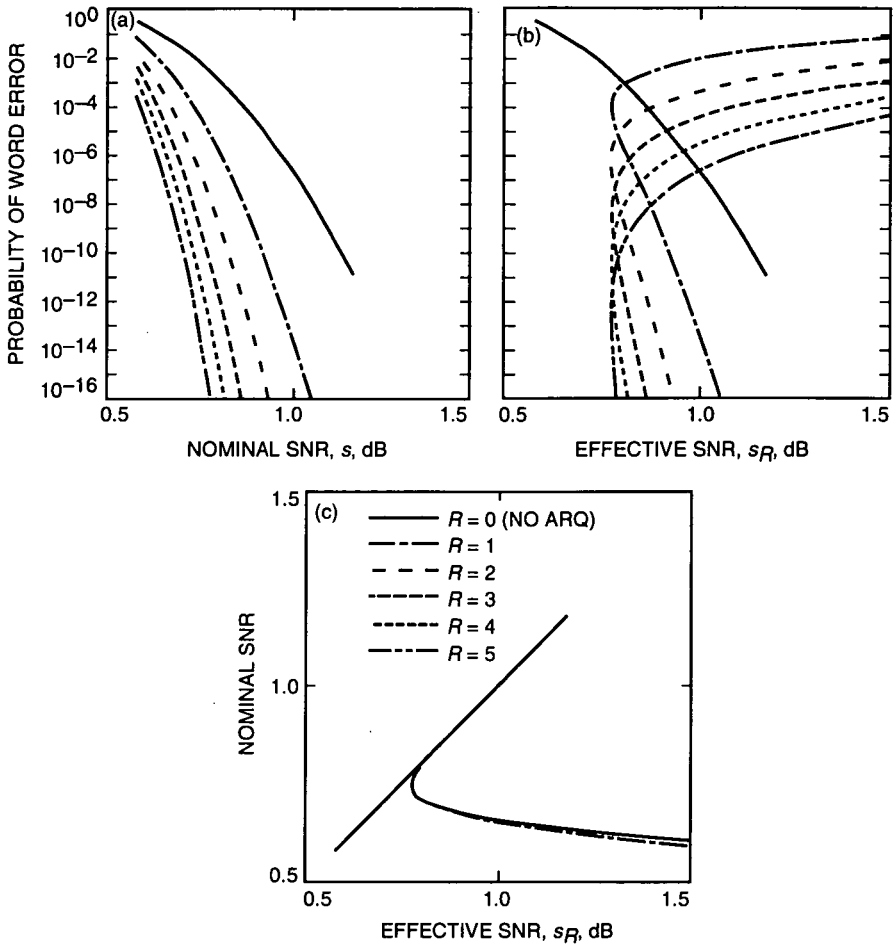

Fig. 8. Pairwise relationships between (a) nominal SNR *s*, (b) effective SNR $s_R(s)$, and (c) probability of word error $P_R(s)$ for the (15,1/4) + (255,223) convolutional code.

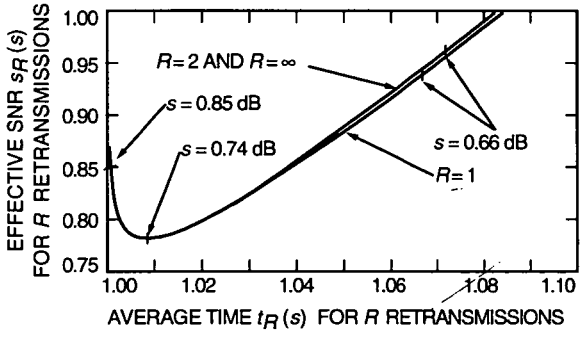

Fig. 9. Effective SNR, s_R, plotted with the average time to transmit, t_R, using the (15,1/4) + (255,223) code.

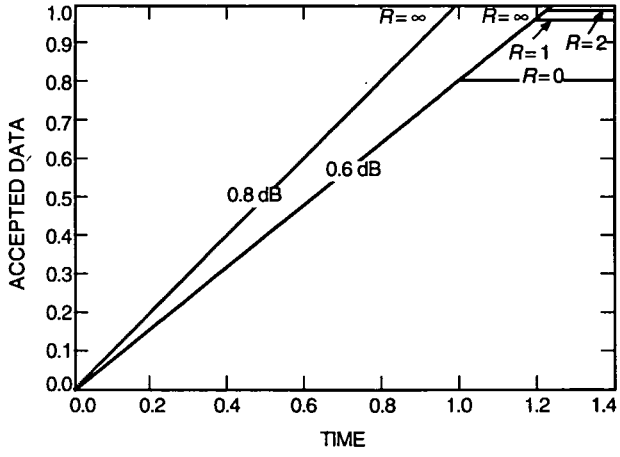

Fig. 10. Decoded data volume versus time for the (15,1/4) + (255,223) code.

For the concatenated system considered in this section, we have $s^*_\infty = 0.782$ dB, where the minimum is achieved at $s^* = 0.747$ dB. This limiting SNR value of s^*_∞ is still significantly higher than the Shannon limit $> (2^{2r} - 1)/(2r) = -0.917$ dB, where $r = (1/4) \cdot (223/255)$. We can compute the limiting time and rate at the channel SNR s^* to be

$$t^*_\infty = t_\infty(s^*) = 1.008$$

$$r^*_\infty = r_\infty(s^*) = 0.992$$

The limiting performance curve is shown in Fig. 11 as the envelope of the parametric curves of $s_R(s)$ and $P_R(s)$ for finite R.

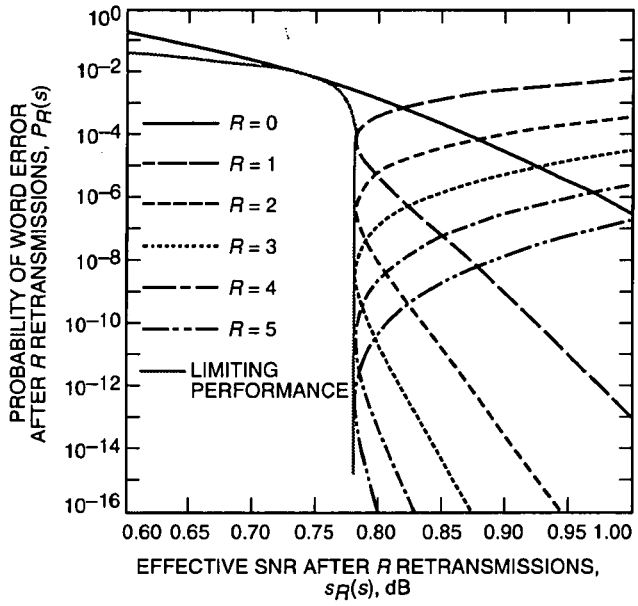

Fig. 11. $P_R(s)$ versus $s_R(s)$, (15,1/4) + (255,223) code.

F. The (15,1/6) Convolutional Code

Three plots showing the pairwise relationships between nominal SNR, effective SNR, and probability of word error for the (15,1/6) convolutional code are shown in Fig. 12.

In Fig. 13, a parametric plot of effective SNR versus time shows a local minimum where the effective SNR is low and the average amount of time spent transmitting is close to 1. For the (15,1/6) code, a nominal SNR $s = 0.536$ dB yields an effective SNR $s_R = 0.574$ dB, and the amount of time spent transmitting ($t_R = 1.009$) is only 0.9-percent more time than transmitting once. For nominal SNR higher than 0.6 dB, the channel is essentially noiseless and few retransmissions are requested. As the nominal SNR gets lower than 0.53 dB, the amount of data transmitted increases and so the effective SNR s_R and time t_R also increase. The amount that they can increase is limited because the number of retransmissions is limited to R.

So, for an additional 0.038 dB = 0.574 dB − 0.536 dB of SNR (due to 0.88-percent more time to transmit), the data will be delivered without detected errors, if an unlimited number of retransmissions is allowed, instead of with a probability of error of 0.0087 for the same system without ARQ. The fraction

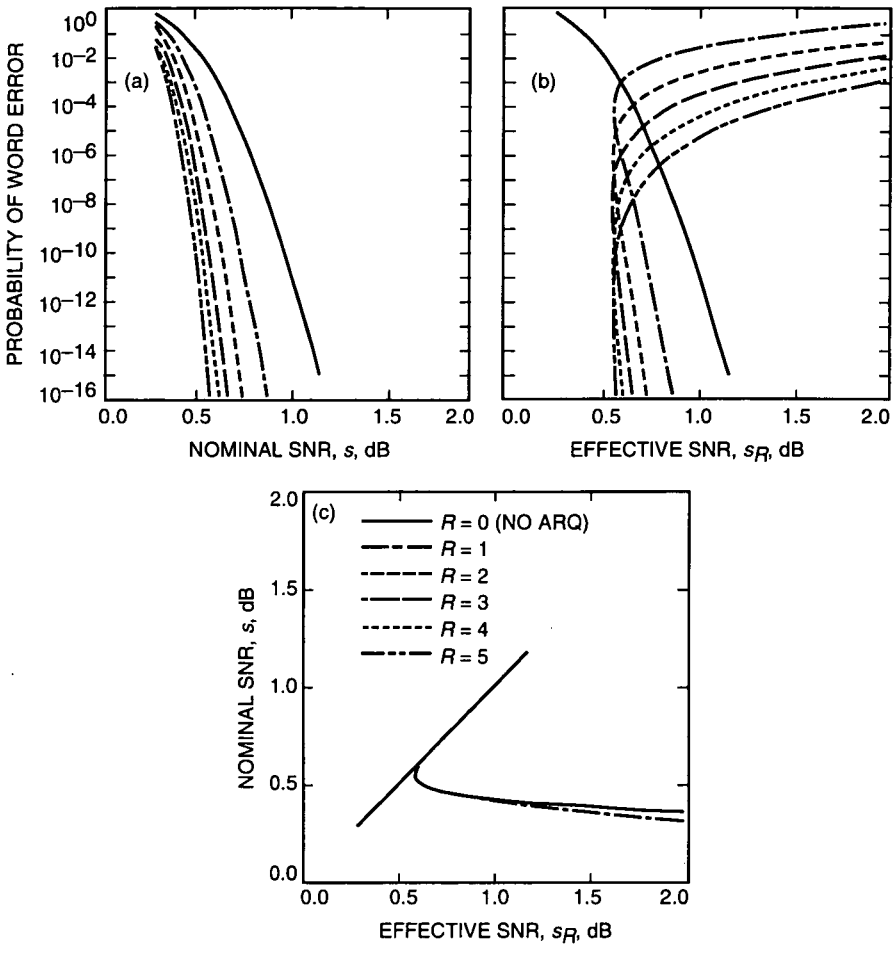

Fig. 12. Pairwise relationships between (a) nominal SNR s, (b) effective SNR $s_R(s)$, and (c) probability of word error $P_R(s)$ for the (15,1/6) + (255,223) convolutional code.

of the total data volume that is accepted after R retransmissions, $1 - P^{R+1}$, versus the amount of time for R retransmissions is shown in Fig. 14.

For the concatenated system considered in this section, we have $s^*_\infty = 0.574$ dB, where the minimum is achieved at $s^* = 0.536$ dB. This limiting SNR value of s^*_∞ is still significantly higher than the Shannon limit $(2^{2r} - 1)/(2r) = -1.15$ dB, where $r = (1/6) \cdot (223/255)$. We can compute the limiting time and rate at the channel SNR s^* to be

$$t^*_\infty = t_\infty(s^*) = 1.009$$

$$r^*_\infty = r_\infty(s^*) = 0.991$$

The limiting performance curve is shown in Fig. 15 as the envelope of the parametric curves of $s_R(s)$ and $P_R(s)$ for finite R.

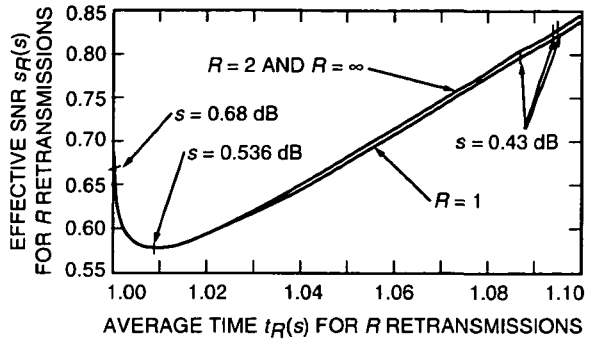

Fig. 13. Effective SNR, s_5, plotted with the average time to transmit, t_5, for five allowed retransmissions using the (15,1/6) + (255,223) code. This represents the performance for nominal SNRs above 0.480 dB.

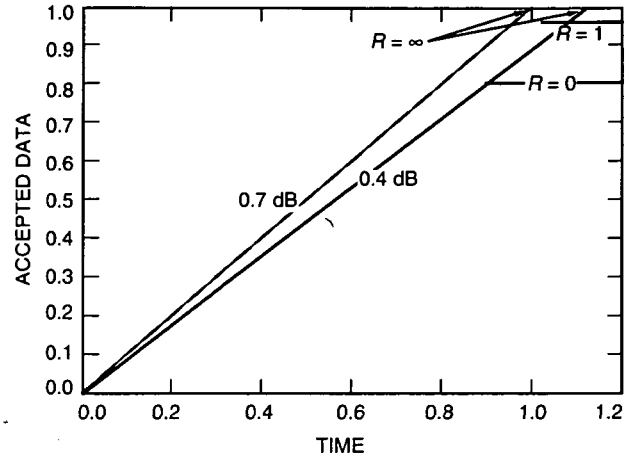

Fig. 14. Decoded data volume versus time for the (15,1/6) + (255,223) code.

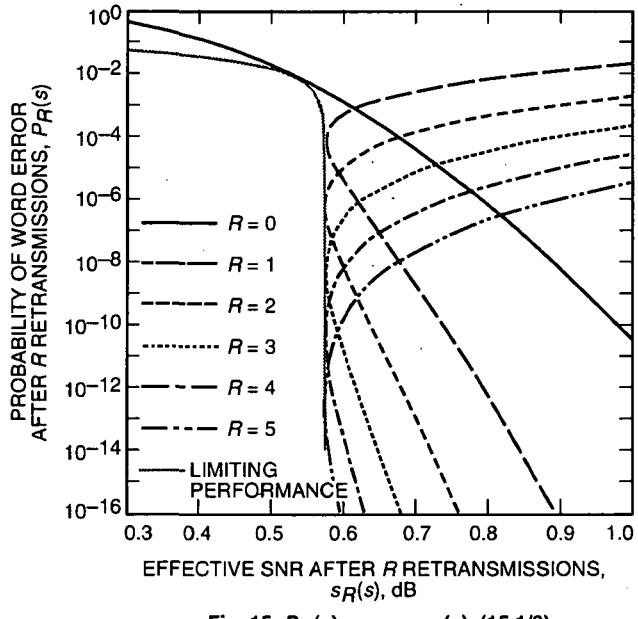

Fig. 15. $P_R(s)$ versus $s_R(s)$, (15,1/6) + (255,223) code.

IV. Conclusions

We have compared the performance of three reference concatenated coded systems used in actual deep-space missions to that obtainable by ARQ methods using the same codes, in terms of required power, time to transmit with a given number of retransmissions, and achievable probability of word error. We have established the ultimate limits of ARQ with an arbitrary number of retransmissions.

For an additional 0.035 to 0.057 dB of average power, the retransmission scheme can deliver virtually error-free data to the user. This is important when just a few residual errors may disrupt the operation of subsequent data processing stages, e.g., the decompression of source encoded data. ARQ schemes are also attractive when the available SNR is poorly predictable, since they allow one to trade time for link margin. The disadvantages are the need for large onboard storage and for heavier usage of the uplink channel.

V. Topics for Future Investigation

Frame Identification: If the identification headers are not in every frame, there may be questions about whether the frame requested for repeat can be accurately identified in the request. This may depend on the usage of onboard memory or on how consistent the round-trip delay is. If there is a sequential log of data that was coded and transmitted, and the periodic header information is associated with the right frames, the repeat request should be able to accurately identify the data to be re-sent.

Packet Combining: Traditionally, ARQ systems decode retransmitted frames without consideration of the information contained in earlier noisy versions of frames already received. By combining subsequent transmissions of the same word/packet, the effect is that the probability of correctly decoding increases. There are two ways to do this. One is to concatenate the many received versions of the word, forming a longer, lower-rate codeword. This is called code combining. The other is to combine symbols of the received versions of the word, forming a word of the same length but with stronger symbols. This is called diversity combining.

Kallel and Leung [7] describe the performance, in terms of throughput, of several schemes with different techniques for combining different numbers and combinations of copies of received words. They run the comparisons for a binary symmetric channel, a nonfading and a Rayleigh fading channel with additive white Gaussian noise, as well as for schemes with and without forward error correction (in the form of rate 1/2 and rate 7/8 convolutional coding). Their focus is on systems with finite receiver buffers. Kallel [6] shows some significant performance improvements for systems using code combining.

Either of these techniques may be able to be translated into an improved SNR. Let s_i denote the improved SNR for a word transmitted i times. The effect of considering this new SNR is that the lines in Fig. 6 would have a steeper slope with each retransmission. One can view this as a multiple rate adaptive coding system, since the system runs at the highest rate, while the channel is relatively noiseless and effectively decreases the rate when the channel quality drops. The fraction of words that were retransmitted i times that must be retransmitted again is on average $P(s_R)$. The fraction of data that is still in error after i retransmissions is

$$P_i(s) = \prod_{j=0}^{i} P(s_j)$$

It follows that the amount of time it takes for i retransmissions is

$$t_i(s) = 1 + \sum_{m=0}^{i-1} P_m(s) = 1 + \sum_{m=0}^{i-1} \prod_{j=0}^{m} P(s_j)$$

This leads to an effective SNR of

$$s_i(s) = s\left(1 + \sum_{m=0}^{i-1} \prod_{j=0}^{m} P(s_j)\right)$$

Incremental Redundancy: An ARQ system using incremental redundancy is one in which the transmitter sends more parity when requested by the receiver. While packet combining is strictly a strategy for improving the receiver without altering the transmission process, it is also equivalent to the incremental redundancy strategy when a repetition code is used. Incremental redundancy methods have the potential

for achieving the best possible ARQ performance, but their implementation is more complicated. Parallel concatenated codes (turbo codes) have been recently proposed for deep-space applications [4] and seem to be particularly suited for incremental redundancy ARQ schemes.

Acknowledgments

The authors are grateful to A. Kiely and L. Swanson for several helpful comments during the preparation of this article.

References

[1] S. A. Butman, L. J. Deutsch, and R. L. Miller, "Performance of Concatenated Codes for Deep Space Missions," *The Telecommunications and Data Acquisition Progress Report 42-63, March–April 1981*, Jet Propulsion Laboratory, Pasadena, California, pp. 33–39, June 15, 1981.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

[3] D. Divsalar and J. H. Yuen, "Performance of Concatenated Reed–Solomon/ Viterbi Channel Coding," *The Telecommunications and Data Acquisition Progress Report 42-71, July–September 1982*, Jet Propulsion Laboratory, Pasadena, California, pp. 81–94, November 15, 1982.

[4] D. Divsalar and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications," *The Telecommunications and Data Acquisition Progress Report 42-121, January–March 1995*, Jet Propulsion Laboratory, Pasadena, California, pp. 66–77, May 15, 1995.

[5] S. Dolinar, L. Ekroot, and F. Pollara, "Improvements on Probability of Error Bounds for Block Codes on the Gaussian Channel," *1994 IEEE International Symposium on Information Theory*, Trondheim, Norway, 1994.

[6] S. Kallel, "Analysis of a Type II Hybrid ARQ Scheme With Code Combining," *IEEE Transactions on Communications*, vol. 38, no. 8, pp. 1133–1137, August 1992.

[7] S. Kallel and C. Leung, "Efficient ARQ Schemes With Multiple Copy Decoding," *IEEE Transactions on Communications*, vol. 40, no. 3, pp. 642–650, March 1992.

[8] R. J. McEliece and L. Swanson, "On the Decoder Error Probability for Reed–Solomon Codes," *The Telecommunications and Data Acquisition Progress Report 42-84, October–December 1985*, Jet Propulsion Laboratory, Pasadena, California, February 15, 1986.

[9] F. Pollara, and K.-M. Cheung, "Performance of Concatenated Codes Using 8-Bit and 10-Bit Reed-Solomon Codes," *The Telecommunications and Data Acquisition Progress Report 42-97, January–March 1989*, Jet Propulsion Laboratory, Pasadena, California, pp. 194–201, May 15, 1989.

[10] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, New Jersey, Prentice Hall: 1995.

Appendix
Low Nominal SNRs

When an ARQ system is limited in the number of retransmissions R, at low SNRs, the effects of R being finite are apparent. This appendix describes the effects at low SNR although, as we shall see, the system should not be designed to operate in this region. To illustrate the effects, we will consider the system with concatenated RS and (7,1/2) codes, though the results hold for all three concatenated systems described in this article.

We describe an ARQ system as having three performance regions that depend on the nominal SNR. At high SNRs the system behaves as if no ARQ is present; at mid SNRs, the system behaves like an ARQ system with an unlimited number of repeat requests; and at low SNRs, the system behaves like $R+1$ repeats are being used. The transitions between these regions are smooth and follow the equations in Section III.C. The plot of effective SNR versus time in Fig. A-1 illustrates the three regions for three different values of R. The system performance for SNRs ranging from high through mid range is the focus of the main portion of this article. Here they will be discussed at an intuitive level for comparison to the low-SNR regions.

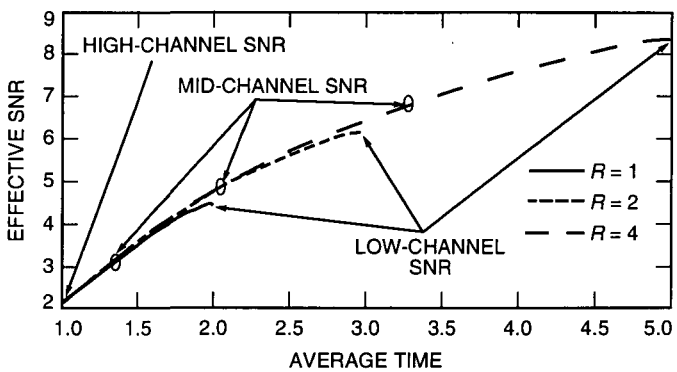

Fig. A-1. Average time versus effective SNR for the (7,1/2) + (255,223) code and varying nominal SNR.

As we have seen in Section III.C, at high SNR the system performs much like a system without ARQ. The data are almost always delivered reliably by the first transmission. So, for high SNRs, the probability of word error, amount of time to transmit, and effective SNR are indistinguishable from a system without ARQ. As such, there also are no effects of a finite R on the performance. For mid-range SNRs, repeat requests are being made with some frequency, but the SNR is sufficiently high that correct decoding almost always occurs within the limited number of retransmissions. In this region, performance is approximately that of a system with an unlimited number of retransmissions. For low SNRs, the system is frequently retransmitting all R times. The amount of time it takes to transmit is nearly $R+1$. The effective SNR, $s_R(s)$, is approximately $10\log_{10}(R+1)$ more than the nominal SNR, s. The probability of word error is near 1, and the system is performing as though it is using a $R+1$ repeat on a very noisy channel (this is, in fact, what it is doing most of the time). The transition from mid-to-low SNRs produces a "hook" in all of the curves. The behavior is similar for different R, so $R=1$ will be used to show the mid-to-low SNR transitions for the different performance curves in Fig. A-2.

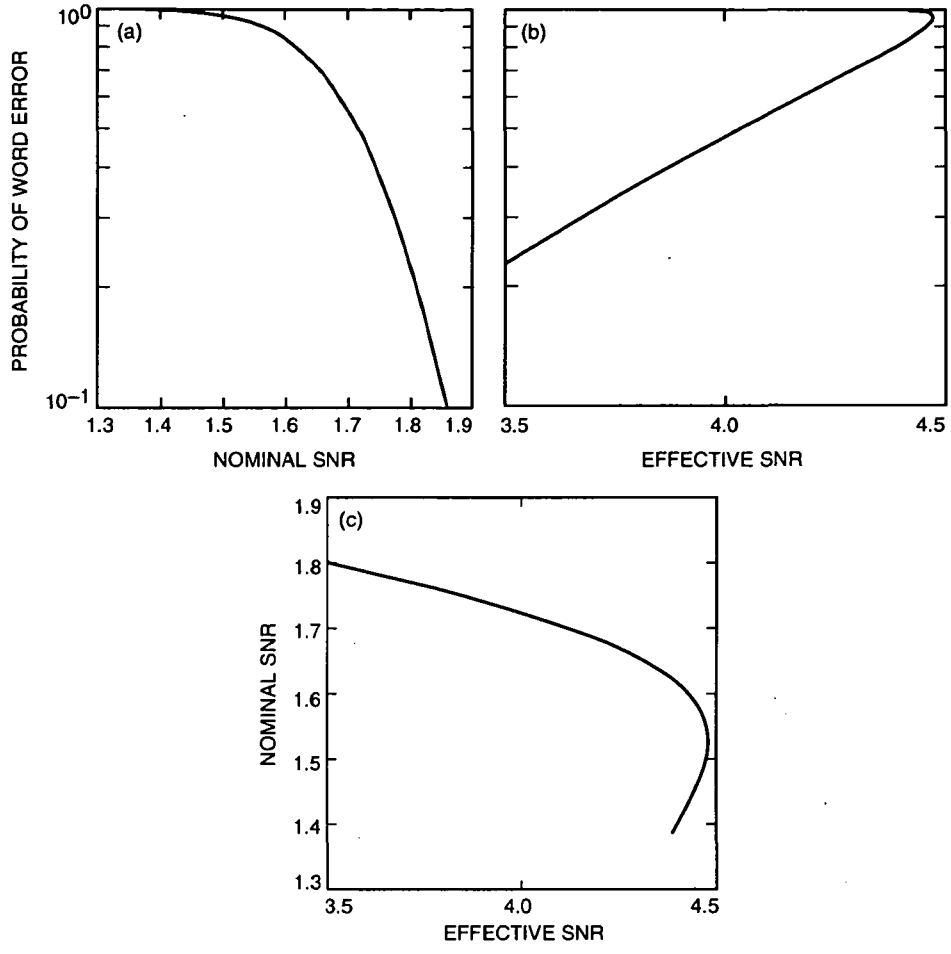

Fig. A-2. Parametric plot of probability of word error versus effective SNR with curves showing relations between (a) nominal SNR and probability of both (b) word error and (c) effective SNR.