Concatenated Coding Systems Employing a Unit-Memory Convolutional Code and a Byte-Oriented Decoding Algorithm

LIN-NAN LEE, MEMBER, IEEE

Abstract—Concatenated coding systems utilizing a convolutional code as the inner code and a Reed-Solomon code as the outer code are considered. In order to obtain very reliable communications over a very noisy channel with relatively modest coding complexity, it is proposed to concatenate a byte-oriented unit-memory convolutional code with an RS outer code whose symbol size is one byte. It is further proposed to utilize a real-time minimal-byte-error probability decoding algorithm, together with feedback from the outer decoder, in the decoder for the inner convolutional code. The performance of the proposed concatenated coding system is studied, and the improvement over conventional concatenated systems due to each additional feature is isolated.

I. INTRODUCTION

THE COMPLEXITY of conventional coding systems grows exponentially with the block length for block codes (or with the constraint length for convolutional codes). To overcome the complexity of very long codes, the idea of cascading two or more codes of lesser complexity to achieve highly reliable communications was considered first by Elias [1], and later by Forney [2]. Forney's technique of using two or more block codes over different alphabets to obtain a very low error rate is known as concatenated coding.

Guided by the premise that a convolutional code generally performs better than a block code of the same complexity, Falconer [3], and later Jelinek and Cocke [4], considered cascading an outer block code with an inner convolutional code. Figure 1 shows a general representation of such a block-convolutional concatenated coding system. In both the Falconer and Jelinek-Cocke schemes, sequential decoding was used for the inner decoder; and the outer block coding system was used only to intervene when the sequential decoder experienced computational overflow. Therefore, these systems can be regarded as primarily sequentially decoded convolutional coding systems.

Maximum likelihood (i.e., Viterbi [5]) decoding of convolutional codes with a moderate constraint length can provide an error rate of less than 10^{-2} at a rate slightly higher than

Paper approved by the Editor for Communication Theory of the IEEE Communications Society for publication after presentation in part at the IEEE International Symposium on Information Theory, Ronneby, Sweden, June 21-24, 1976. Manuscript received October 19, 1976; revised May 6, 1977. This work was supported by the National Aeronautics and Space Administration under NASA Grant NSG 5025 at the University of Notre Dame in liaison with the Goddard Space Flight Center. This paper forms part of a dissertation submitted to the Graduate School of the University of Notre Dame in partial fulfillment of the requirements for the Ph.D. degree.

The author was with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556. He is now with the LINKABIT Corporation, San Diego, CA 92121.

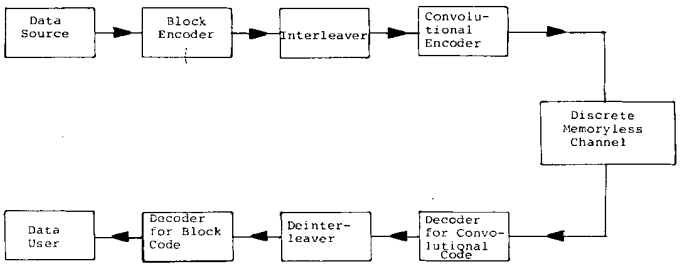


Fig. 1. A concatenated coding system employing a convolutional code as the inner code and a block code as the outer code.

R_{comp} of the noisy channel. Forney's work [2] suggested that a concatenated coding system with a powerful outer code can perform reasonably well when its inner decoder is operated with a probability of error in the range between 10^{-2} and 10^{-3}. It was natural then for Odenwalder [6] to choose a Viterbi decoder for the inner coding system in his block-convolutional concatenated coding system.

Because the output error patterns of Viterbi-type decoders for convolutional codes are bursty, block codes over a large alphabet, such that many bits of the inner code form one symbol of the outer code, appear very attractive for the outer coding system. The Reed-Solomon (RS) block codes are particularly appealing because they can be decoded by relatively simple procedures (such as the Berlekamp-Massey [7], [8] algorithm) and have optimum distance properties [17]. Because the lengths of the bursts of output errors made by Viterbi decoders are widely distributed, it is generally necessary to interleave the inner convolutional code so that errors in the individual RS-symbols of one block are independent; otherwise, a very long block code would be required to operate the system efficiently. Because the most likely length of the output error patterns made by the inner decoder are on the order of the constraint length, K, of the convolutional code, Odenwalder chose the RS symbol alphabet to be $GF(2^K)$.

In a block-convolutional concatenanted coding system such as Odenwalder's employing a Viterbi decoder with conventional convolutional codes, it is very unlikely that the beginning of a decoding error burst is always aligned with the boundary between two RS symbols; in fact, such a burst only two bits long may affect two RS symbols. This fact led us to consider using good convolutional codes which are symbol-oriented rather than bit-oriented. In [9], we reported a class of unit-memory convolutional codes for which k_0-bit information segments are encoded into n_0-bit encoded segments. It was shown there that an (n_0, k_0) convolutional

code with unit memory always achieves the largest free distance possible for codes of the same rate k_0/n_0 and the same number 2^{Mk_0} of encoder states, where M is the encoder memory. The unit-memory codes are naturally byte-oriented with byte size equal to k_0 information bits. It will be shown that the improved free distance and the symbol-oriented nature of these codes provides an improvement of approximately 0.3 dB in the overall performance of the concatenated coding system when these codes replace bit-oriented convolutional codes.

Another improvement is to modify the decoder for the convolutional code so that the decoder emits not only the most-likely estimated symbol, but also reliability information about the estimated symbol. The outer decoder may then use this reliability information to perform either "erasures-and-errors" decoding or "generalized-minimum-distance" (GMD) decoding as suggested by Forney [2]. Zeoli [10] and Jelinek [11] proposed to extract reliability information by annexing a long tail to the original convolutional code and using this added tail to provide an error detection capability for the estimate made by the Viterbi decoder for the original shorter convolutional code. This approach requires the feedback of symbols previously decoded by the Viterbi decoder and, more importantly, uses the output of the outer decoder to restart the inner Viterbi decoder whenever an error is corrected by the outer decoder. It will be shown that the error detecting capability used with an "erasures-and-errors" outer decoder provides an improvement of 0.2 dB and that the feedback from the outer decoder further improves the performance by 0.3 dB.

An alternative approach to extracting reliability information from the inner decoder is to compute the *a posteriori* probability of correctness for each decoded symbol from the decoder for the short constraint length convolutional code and then use this probability as the reliability information provided to the outer coding system. It will be shown that, when used with an errors-and-erasures outer decoder, this scheme improves performance by only 0.05 dB to 0.1 dB compared to hard-decision decoding and hence is less powerful than Zeoli's tail annexation scheme; yet its performance is undoubtedly optimal among all schemes employing only the short constraint length convolutional code (with no annexed tail). However, it will be shown that, in conjection with the use of feedback from the outer decoder, the *a posteriori* probability inner decoder provides about 0.2 dB more improvement than does the Viterbi decoder aided by feedback. In fact, the *a posteriori* inner decoder, used with feedback from the outer decoder, offers a slight improvement over Zeoli's scheme; moreover the inner encoder and the inner decoder have the same constraint length so that the inner decoder generally and automatically returns to normal operation only a few branches after making an error.

The plan of this paper is as follows. In Section II, a "real time" decoding algorithm for unit-memory convolutional codes is developed which calculates the *a posteriori* probability for each value of the byte being decoded. In Sections III, IV, and V, the performances of several block-convolutional concatenated coding systems having unit-memory convolutional inner codes are compared with similar systems having conventional bit-oriented convolutional inner codes. In each case, we chose the (18, 6) unit-memory convolutional code as the inner code because it has practically minimum complexity in terms of decoder implementation, and because of its reasonably large free distance ($d_{\text{free}} = 16$). We chose the Reed-Solomon codes over $GF(2^6)$, with block length 63 symbols, as the outer codes so that the symbol size of the RS codes would be matched to the byte-size (six bits) of the unit-memory code. In Section VI, the degradation of performance, when the rate 1/3 inner convolutional code is replaced by a rate 1/2 convolutional code, is considered in order to demonstrate the tradeoff between bandwidth expansion and signal-energy-to-noise ratio. In Section VII, the 95% confidence intervals for the simulation results are obtained and interpreted.

II. REAL-TIME MINIMAL-BYTE-ERROR PROBABILITY DECODING OF UNIT-MEMORY CODES

We now develop an alogrithm for real-time minimal-byte-error probability decoding of the unit-memory convolutional codes described in [9].

Let $\boldsymbol{a}_t$ ($t = 1, 2, \cdots$) denotes the byte (or subblock) of k_0 information bits to be encoded at time t, and let $\boldsymbol{b}_t$ ($t = 1, 2, \cdots$) be the corresponding encoded subblock of n_0 bits. For a unit-memory code,

$$\boldsymbol{b}_t = \boldsymbol{a}_t G_0 + \boldsymbol{a}_{t-1} G_1 \tag{1}$$

where G_0 and G_1 are $k_0 \times n_0$ matrices and where, by way of convention, $\boldsymbol{a}_0 = \boldsymbol{0}$. We assume that the sequence $\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots$ has been transmitted over a discrete memoryless channel and that $\boldsymbol{r}_t$ ($t = 1, 2, \cdots$) is the received subblock corresponding to the transmitted subblock $\boldsymbol{b}_t$. We shall write $\boldsymbol{a}_{[t,t']}$ to denote $[\boldsymbol{a}_t, \boldsymbol{a}_{t+1}, \cdots, \boldsymbol{a}_{t'}]$; similarly for $\boldsymbol{b}_{[t,t']}$ and $\boldsymbol{r}_{[t,t']}$.

By *real-time decoding with delay* Δ, we mean that the decoding decision for $\boldsymbol{a}_t$ is made from the observation of $\boldsymbol{r}_{[1,t+\Delta]}$. The real-time minimal-byte-error probability (RTMBEP) decoding rule then is that which chooses its estimate $\hat{\boldsymbol{a}}_t$ as that value of $\boldsymbol{a}_t$ which maximizes $P(\boldsymbol{a}_t \mid \boldsymbol{r}_{[1,t+\Delta]})$ for $t = 1, 2, \cdots$. To find a recursive algorithm for this decoding rule, we begin by noting that

$$P(\boldsymbol{a}_t \mid \boldsymbol{r}_{[1,t+\Delta]}) = P(\boldsymbol{a}_t, \boldsymbol{r}_{[1,t+\Delta]}) \Big/ \sum_{\alpha} P(\alpha, \boldsymbol{r}_{[1,t+\Delta]}) \tag{2}$$

where we have used α to denote a running variable for $\boldsymbol{a}_t$. It suffices then to find a recursive method for calculating $P(\boldsymbol{a}_t, \boldsymbol{r}_{[1,t+\Delta]})$.

We next observe that

$$\begin{aligned} P(\boldsymbol{a}_t, \boldsymbol{r}_{[1,t+\Delta]}) &= P(\boldsymbol{a}_t, \boldsymbol{r}_{[1,t]}) P(\boldsymbol{r}_{[t+1,t+\Delta]} \mid \boldsymbol{a}_t, \boldsymbol{r}_{[1,t]}) \\ &= P(\boldsymbol{a}_t, \boldsymbol{r}_{[1,t]}) P(\boldsymbol{r}_{[t+1,t+\Delta]} \mid \boldsymbol{a}_t) \end{aligned} \tag{3}$$

where we have used the facts that the channel is memoryless and that the code has unit memory. It remains to find recursive rules for obtaining the two probabilities on the right in (3).

Obtaining the recursion for $P(\boldsymbol{a}_t, \boldsymbol{r}_{[1,t]})$ is quite standard [12] - [14];

$$P(a_t, r_{[1,t]}) = \sum_{a_{t-1}} P(a_{[t-1,t]}, r_{[1,t]})$$
$$= \sum_{a_{t-1}} P(a_{t-1}, r_{[1,t-1]}) \cdot P(a_t, r_t \mid a_{t-1}, r_{[1,t-1]}). \quad (4)$$

But also

$$P(a_t, r_t \mid a_{t-1}, r_{[1,t-1]}) = P(a_t \mid a_{t-1}, r_{t-1}) P(r_t \mid a_{[t-1,t]}, r_{[1,t-1]}) = 2^{-k_0} P(r_t \mid b(a_{[t-1,t]})) \quad (5)$$

where we have written $b(a_{[t-1,t]})$ for the value of b_t determined by (1) from $a_{[t-1,t]}$, and where we assume here and hereafter that all information sequences are equally likely (as corresponds to maximum-likelihood decoding.) Substituting (5) into (4), we have our desired recursion

$$P(a_t, r_{[1,t]}) = 2^{-k_0} \sum_{a_{t-1}} P(a_{t-1}, r_{[1,t-1]}) \cdot P(r_t \mid b(a_{[t-1,t]})). \quad (6)$$

We now turn to the quantity $P(r_{[t+1,t+\Delta]} \mid a_t)$ which we note is the $i = 1$ value of

$$P(r_{[t+i,t+\Delta]} \mid a_{t+i-1}) = \sum_{a_{t+i}} P(a_{t+i}, r_{[t+i,t+\Delta]} \mid a_{t+i-1}). \quad (7)$$

Proceeding in the same manner that (6) was obtained from (4), we find the desired recursion

$$P(r_{[t+i,t+\Delta]} \mid a_{t+i-1}) = 2^{-k_0} \sum_{a_{t+i}} P(r_{[t+i+1,t+\Delta]} \mid a_{t+i}) \cdot P(r_{t+i} \mid b(a_{[t+i-1,t+i]})). \quad (8)$$

This recursion in initialized with its $i = \Delta$ value

$$P(r_{t+\Delta} \mid a_{t+\Delta-1}) = \sum_{a_{t+\Delta}} P(a_{t+\Delta}, r_{t+\Delta} \mid a_{t+\Delta-1}) = 2^{-k_0} \sum_{a_{t+\Delta}} P(r_{t+\Delta} \mid b(a_{[t+\Delta-1,t+\Delta]})), \quad (9)$$

and evaluated with $i = \Delta - 1, \Delta - 2, \cdots, 1$. It should be noted that the recursion (8) requires less multiplication than the RTMBEP decoding previously reported in [14]. The reduction is particularly evident for unit-memory codes due to their fully connected trellis structure.

An algorithm to carry out the recursive rules given by (6) and (8) requires, for each byte (or "state" in the usual Viterbi decoding terminology) α, the storage of two real numbers $f(\alpha)$ and $h(\alpha)$; namely,

$$f(\alpha) = P(a_t = \alpha, r_{[1,t]}) \quad (10)$$

and

$$h(\alpha) = P(r_{[t+i,t+\Delta]} \mid a_{t+i-1} = \alpha) \quad (11)$$

where i will be decremented from Δ to 1 as the algorithm progresses. (Of course, the received segment $r_{[t+1,t+\Delta]}$ must also be stored so that $P(r_{t+i} \mid b(a_{[t+i-1,t+i]}))$ can also be found for $i = \Delta, \Delta - 1, \cdots, 1$.) We may now state:

The RTMBEP Decoding Algorithm for Unit-Memory Codes

Step 0: Set $f(0) = 2^{k_0}$ and set $f(\alpha) = 0$ for $\alpha \neq \mathbf{0}$. Set $t = 1$.

Step 1: Make the replacement, for all states α,

$$f(\alpha) \leftarrow 2^{-k_0} \sum_{\alpha'} f(\alpha') P(r_t \mid b(\alpha', \alpha)).$$

Step 2: Set $i = \Delta$ and, for all states α, set

$$h(\alpha) = 2^{-k_0} \sum_{\alpha'} P(r_{t+\Delta} \mid b(\alpha, \alpha')).$$

Step 3: Decrease i by 1 and make the replacement, for all states α,

$$h(\alpha) \leftarrow 2^{-k_0} \sum_{\alpha'} h(\alpha') P(r_{t+i} \mid b(\alpha, \alpha')).$$

If now $i = 1$, go to Step. 4. Otherwise, return to Step 3.

Step 4: Emit, as the estimate of a_t, that byte α_0 which maximizes $f(\alpha)h(\alpha)$, and emit, as the reliability indicator, the probability

$$P(a_t = \alpha_0 \mid r_{[1,t+\Delta]}) = f(\alpha_0)h(\alpha_0) \Big/ \sum_{\alpha} f(\alpha)h(\alpha).$$

Increase t by 1 and return to Step 1.

The only feature of the algorithm that should require any comment is the initialization of $f(\mathbf{0})$ at 2^{k_0}. This is required so that the first time step 1 is performed one obtains the correct initial value $f(\alpha) = P(r_1 \mid b(\mathbf{0}, \alpha))$. In fact, however, it makes no difference in the output from the algorithm if the f and h values are scaled by fixed positive constants, so that $f(\mathbf{0}) = 1$ is permissible in Step 0 and the factors 2^{-k_0} can be removed in Steps 1, 2, and 3.

Note that Step 3 of the algorithm, which has the same complexity as Step 1, is performed $\Delta - 1$ times for each time that Step 1 is performed. It is clearly desirable then to keep Δ as small as possible. Table I shows the variation of the decoding byte-error probability, P_{BE}, with the decoding delay, Δ, for the $(n_0 = 18, k_0 = 6)$ unit-memory code of [9] used on a simulated three-bit-quantized additive white Gaussian noise (AWGN) channel. We see that $\Delta = 8$ gives virtually the same P_{BE} as the "optimum" choice $\Delta = \infty$.

We now point out, however, that one can reduce the ratio of Step 3 operations to Step 1 operations to as close to unity as desired without any degradation in performance but at the cost of additional storage. The "trick" is to use a *variable decoding delay* Δ. Each a_t is decoded from $P(a_t \mid r_{[1,t+\Delta]})$ but Δ, depending on the value of t, takes some value in the

TABLE I
VARIATION OF DECODING BYTE-ERROR PROBABILITY p WITH DECODING DELAY Δ FOR RTMBEP DECODING OF THE (18, 6) UNIT-MEMORY CODE ON A SIMULATED AWGN CHANNEL WITH A SIGNAL ENERGY PER INFORMATION BIT TO ONE-SIDE NOISE POWER SPECTRAL DENSITY RATIO OF 1.25 DB (4000 BYTES DECODED FOR EACH Δ)

E_b/N_o	1.00 db	1.25 db	1.50 db	1.75 db
p(95% confidence) (18,6) unit-memory code	.0305 (±.0053)	.0200 (±.0044)	.0118 (±.0033)	.0065 (+.0025
p(95% confidence) M=6, (3,1) code	.0488 (±.0068)	.0325 (±.0056)	.0233 (±.0048)	.0128 (±.0035
p(95% confidence) M=7, (3,1) code	.0400 (±.0062)	.0225 (±.0047)	.0140 (±.0037)	.0103 (±.0032

range $\Delta_m \leqslant \Delta \leqslant \Delta_M$. The minimum decoding delay, Δ_m, is chosen large enough to ensure negligible degradation, say $\Delta_m = 8$, while the maximum decoding delay, Δ_M, is chosen small enough to make the increased memory tolerable as will soon become apparent.

In this variable real-time minimal-byte-error probability (VRTMBEP) decoding, one stores $\Delta_M - \Delta_m + 2$ real numbers for each state α, namely: $f_i(\alpha)$ for $i = 1, 2, \cdots, \Delta_M - \Delta_m + 1$ and $h(\alpha)$ where

$$f_i(\alpha) = P(a_{t+i-1} = \alpha, r_{[1,t+i-1]}) \tag{12}$$

and where $h(\alpha)$ is as in (11) with Δ replaced by Δ_M.

Observe now that, in the process of executing Step 2 of the RTMBEP algorithm with $\Delta = \Delta_M$, one would obtain sequentially the quantities

$$P(r_{[t+i,t+\Delta_M]} \mid a_{t+i-1} = \alpha) \tag{13}$$

for $i = \Delta_M - 1, \Delta_M - 2, \cdots, 1$. But the product of the quantity in (13) with $f_i(\alpha)$ as in (12) is, according to (3), equal to $P(a_{t+i-1} = \alpha \mid r_{[1,t+\Delta_M]})$; this is precisely the statistic needed to estimate a_{t+i-1} with a decoding delay of $\Delta = \Delta_M - i + 1$. Hence, if we had the foresight to perform Step 1 of the RTMBEP algorithm $\Delta_M - \Delta_m + 1$ times and to store the resulting $f_i(\alpha)$, then we could make $\Delta_M - \Delta_m + 1$ decoding decisions during the $\Delta_M - 1$ times that Step 3 is performed. Thus, for each block of $\Delta_M - \Delta_m + 1$ forward recursions of step 1, the backward recursion of step 3 would be performed $\Delta_m - 1$ times. The average ratio of step 3 to step 1 would be $(\Delta_M - 1)/(\Delta_M - \Delta_m + 1)$. For instance, with $\Delta_m = 8$ and $\Delta_M = 13$, we would perform Step 3 only twice for each time we performed Step 1; and we would be storing only $\Delta_M - \Delta_m + 2 = 7$ real numbers per state rather than 2 as in the original RTMBEP algorithm in which Step 3 is performed $\Delta - 1 = 7$ times for each time that Step 1 is performed.

It should now be obvious that the following algorithm is the necessary modification to the RTMBEP decoding algorithm for obtaining reduced computation at the price of additional storage as has just been described.

The VRTMBEP Decoding Algorithm for Unit-Memory Codes

Step 0: Set $f_{\Delta_M - \Delta_m + 1}(\mathbf{0}) = 2^{k_0}$ and set $f_{\Delta_M - \Delta_m + 1}(\alpha) = 0$ for $\alpha \neq \mathbf{0}$. Set $t = 1$.

Step 1: Set

$$f_1(\alpha) = 2^{-k_0} \sum_{\alpha'} f_{\Delta_M - \Delta_m + 1}(\alpha') P(r_t \mid b(\alpha', \alpha)),$$

and set

$$f_{i+1}(\alpha) = 2^{-k_0} \sum_{\alpha'} f_i(\alpha) P(r_{t+i} \mid b(\alpha', \alpha))$$

for $i = 1, 2, \cdots, \Delta_M - \Delta_m$ in order.

Step 2: Set $i = \Delta_M$ and, for all states α, set

$$h(\alpha) = 2^{-k_0} \sum_{\alpha'} P(r_{t+\Delta_M} \mid b(\alpha, \alpha')).$$

Step 3: Decrease i by 1 and make the replacement, for all states α,

$$h(\alpha) \leftarrow 2^{-k_0} \sum_{\alpha'} h(\alpha') P(r_{t+i} \mid b(\alpha, \alpha')).$$

If now $i \leqslant \Delta_M - \Delta_m + 1$, go to Step 4. Otherwise, return to Step 3.

Step 4: Emit, as the estimate of a_{t+i-1}, that byte α_0 which maximizes $f_i(\alpha)h(\alpha)$, and emit, as the reliability indicator, the probability

$$P(a_{t+i-1} = \alpha_0 \mid r_{[1,t+\Delta_M]}) = f_i(\alpha_0)h(\alpha_0) \Big/ \sum_{\alpha} f(\alpha)h(\alpha).$$

If $i = 1$, increase t by $\Delta_M - \Delta_m + 1$ and return to Step 1. Otherwise, decrease i by 1 and return to Step 3.

It is satisfying to note that VRTMBEP decoding algorithm reduces to the RTMBEP algorithm when $\Delta_M = \Delta_m$. It should be pointed out that when only a finite number, L, of information bytes are encoded and one takes $\Delta_M = L$, the largest possible choice, then the VRTMBEP algorithm reduces to that given by Bahl *et al.* [12] (when the later algorithm is specialized to unit-memory codes) and does about twice the computation of the usual Viterbi decoder; but this case also maximizes the memory requirements. The chief advantage which both RTMBEP and VRTMBEP decoding of unit-memory codes have over Viterbi decoding is in their providing reliability information about the decoding decisions; information of considerable value to the outer decoder in a concatenated coding system. One may argue that Viterbi algorithm can be implemented in logarithm domain, thus resulting much simpler implementation. However, it is interesting to note that both RTMBEP and VRTMBEP can also be implemented in the logarithm domain with the assistance of a ROM storing the operation in the logarithm domain corresponding to normal addition [18].

Because the resulting performance of the RTMBEP and VRTMBEP algorithms are indistinguishable when $\Delta = \Delta_m$ is chosen large enough for negligible degradation compared to $\Delta = \infty$, say $\Delta_m = 8$, we will not hereafter distinguish between the two algorithms in our discussion of concatenated coding systems.

III. ODENWALDER'S CONCATENATED CODING SYSTEM AND SOFT-DECISION MODIFICATION WITH THE RTMBEP DECODING ALGORITHM

The concatenated coding system proposed by Odenwalder [6], which we shall call System I, is as shown in Figure 1 where the inner decoder is a hard-decision Viterbi decoder and

TABLE II
BYTE-ERROR PROBABILITY, p, FOR VITERBI DECODING OF THREE $R = k_0/n_0 = 1/3$ CONVOLUTIONAL CODES ON A SIMULATED AWGN CHANNEL (8000 BYTES DECODED FOR EACH POINT SHOWN, DECODING DELAY Δ IN BITS OF 48 IN ALL CASES)

Δ (bytes)	4	6	8	16
p	.0285	.0248	.0193	.0193

where the outer decoder is a t-error correcting decoder for the RS outer block code. Here and hereafter, we assume that the interleaving is "perfect," i.e., that the symbols in each RS block at the output of the interleaver have been independently decoded by the inner Viterbi decoder. Thus, we can then upperbound the probability of a decoding error in an RS block, P_{ERS}, as

$$P_{ERS} = \sum_{i=t+1} \binom{n}{i} p^i (1-p)^{n-i}, \tag{14}$$

where n is the RS block length (in bytes) and p is the byte-error probability at the Viterbi decoder output. Further, since almost all the incorrectly decoded RS codewords are $d_{\min} = 2t + 1$ symbols away from the correct codeword (where $d_{\min}$ is the minimum distance of the RS code), the byte-error probability, P_{BE}, of the concatenated coding system is given closely by

$$P_{BE} = \frac{2t+1}{n} P_{ERS}. \tag{15}$$

For a byte size of 6 bits, as will be assumed hereafter, the RS code has length $n = 2^6 - 1 = 63$ bytes. For convenient reference, we give in Table II the byte-error probability of a Viterbi decoder for the three different convolutional codes of rate $R_{CON} = k_0/n_0 = 1/3$ that will be used in our subsequent comparisons when used on four different AWGN channels; these data are taken from [9]. The AWGN channels are specified by the ratio of channel energy per encoder input bit to one-side noise power spectral density, E_b'/N_0. Note that the energy per channel input bit (decoder output bit), E_s, is given by $E_s = R_{CON}E_b'$. But also $E_b' = R_{RS}E_b$ where R_{RS} is the rate of the RS code and E_b is the channel energy per information bit entering the RS encoder. Thus, the channel energy per information bit to one-sided noise power spectral density ratio for the overall concatenated coding system, E_b/N_0 is given by

$$E_b/N_0 = \frac{1}{R_{RS}R_{CON}} (E_s/N_0). \tag{16}$$

Using the results of Table II together with (14) and (15), we can calculate the byte-error probability for Odenwalder's System I for various RS outer codes. The results of this calculation are shown in Fig. 2 for the three different $R_{CON} = 1/3$ convolutional codes, namely (i) the conventional (3, 1) code with $M = 6$, i.e., $K = 7$; (ii) the conventional (3, 1) code with $M = 7$, i.e., $K = 8$; and (iii) the (18, 6) unit-memory code. Codes (i), (ii) and (iii) have free distances of 15, 16 and 16, respectively, and their corresponding Viterbi decoders have 64, 128 and 64 states, respectively. We see, from Fig. 2, that

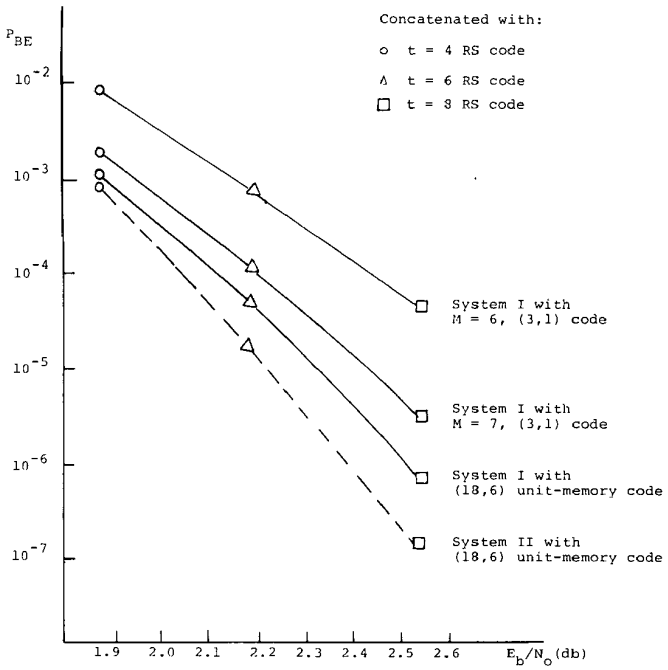


Fig. 2. The performance of Concatenated Coding Systems I and II with RS codes over $GF(2^6)$ on a simulated AWGN channel with $E_b'/N_0 = 1.25$.

the use of the unit-memory code provides an advantage of about 0.3 dB over the conventional code with the same state complexity, part of which gain is attributable to the larger free distance of the unit-memory code. But the unit-memory code is also about 0.1 dB superior to the conventional code with the same free distance (and doubled number of decoder states); this gain is attributable entirely to the byte-oriented structure of the unit-memory code.

It should be mentioned that gains of 0.1 dB are not insignificant in concatenated coding systems. As can be seen from Fig. 2 a gain of 0.1 dB corresponds to a reduction of P_{BE} by nearly an order of magnitude, such steepness of the P_{BE} vs. E_b/N_0 curves being characteristic of well-designed concatenated coding systems.

The inner decoder, i.e., the Viterbi decoder, in System I makes "hard decisions" on the decoded bytes. The system performance can be improved by using a "soft decision" decoder which passes along to the outer decoder a reliability indicator for each decoded byte. Such a system, in which the inner decoder is RTMBEP decoder and the outer decoder is an errors-and-erasures decoder for the RS code, will be called *System II*. (For ease of reference, we summarize in Table III the characteristics of each of the six concatenated coding systems that will be considered in this paper.) When the reliability indicator, $P(a_t \mid r_{[1,t+\Delta]})$ for a decoded byte is less than some specified T, the outer decoder treats the byte as having been "erased." The erasures-and-errors decoder for the RS code can correct t errors and e erasures, whenever $2t + e < d_{\min}$. Thus, the block error probability for the outer decoder is upperbounded by

$$P_{ERS} = \sum_{d=d_{\min}}^{n} \sum_{\substack{t=0 \\ e=d-2t}}^{d/2} \binom{n}{t,\, e} p^t q^e (k-p-q)^{n-t-e} \tag{17}$$

TABLE III
THE SIX BLOCK-CONVOLUTIONAL CONCATENATED CODING SYSTEMS STUDIED (EO = ERRORS ONLY DECODER, E + E = ERRORS AND ERASURES DECODER, FBTID = FEEDBACK TO INNER DECODER)

	Inner Decoder Type	Outer Decoder Type	Inner Code Tail Annexation
System I	Viterbi hard-decision	EO	NO
System II	RTMBEP soft-decision	E+E	NO
System III	Viterbi hard-decision	EO with FBTID	NO
System IV	RTMBEP hard-decision	EO with FBTID	NO
System V	RTMBEP soft-decision	E+E with FBTID	NO
System VI	Viterbi soft-decision	E+E with FBTID	YES

where p is again the byte-error probability for the inner decoder, where q is the byte-erasure probability for the inner decoder, and where

$$\binom{n}{t, e} = \frac{n!}{t!\, e!(n-t-e)!}.$$

The byte-error probability of the overall system is again obtained from (15).

The byte-error-probability, p, and the erasure probability, q, depend on the particular threshold, T, specified. The optimal threshold is a function of E_b'/N_0 and the minimum distance, $d_{\min}$, of the Reed-Solomon code. Roughly speaking, for a given block length n, as $d_{\min}$ gets larger, the overall block error probability is minimized at a higher erasure rate. We have found no simple way to determine the optimal threshold analytically. Instead, we have found p and q for $T = 0.5, 0.7$ and 0.8 by simulation and have used these values of p and q to calculate the byte-error probability of the coding system. In Table IV, we show the result of this calculation. We see, for E_b'/N_0 in the range form 1.25 dB to 1.75 dB, that $T = 0.7$ is the best threshold among the three candidates.

The performance of System II with $T = 0.7$ is also plotted in Figure 2. The inprovement over System I of the performance due to the erasure scheme, as observed from Figure 2, is dependent on the error correcting capability of the outer coding system as well as on E_b'/N_0 and is approximately 0.1 dB. This slight improvement is probably not significant enough to justify the increased complexity of the RTMBEP decoder over the Viterbi decoder. However, as we shall soon see, the RTMBEP decoder coupled with an "erasures-and-error" block decoder performs much better than the Viterbi decoder *when feedback from the outer decoder is utilized.*

IV. FEEDBACK FROM THE OUTER DECODER TO THE INNER DECODER

Because of the nature of convolutional code and the Viterbi decoding algorithm, once an "error event" occurs the decoder often makes a number of closely spaced erroneous estimations before it recovers to correct operation. Since the outer decoder of a concatenated coding system is designed in such a way that it is able to detect and correct *almost all* of the errors made by the inner decoder, it is then of significant advantage if the corrected estimates of the outer decoder are fed back to restart the inner decoder from the point where it first erred in order to eliminate the "burst" of errors. Figure 3 illustrates

TABLE IV
VARIATION OF INNER DECODER BYTE-ERROR PROBABILITY p AND BYTE ERASURE PROBABILITY q AND OF OUTER DECODER BYTE-ERROR PROBABILITY P_{BE} WITH THE ERASURE THRESHOLD T FOR THE (18, 6) UNIT-MEMORY CODE ON A SIMULATED AWGN CHANNEL AND WITH THE MINIMUM DISTANCE $d_{\min}$ OF THE OUTER RS CODE

E_b'/N_0 in db	T	p	q	P_{BE} for $d_{min} = 9$	P_{BE} for $d_{min} = 13$	P_{BE} for $d_{min} = 17$
1.00	0.80	.01000	.05000	$.735 \times 10^{-2}$	$.407 \times 10^{-3}$	$.877 \times 10^{-5}$
	.70	.01325	.04150	$.740 \times 10^{-2}$	$.477 \times 10^{-3}$	$.128 \times 10^{-4}$
	.50	.02100	.01950	$.677 \times 10^{-2}$	$.555 \times 10^{-3}$	$.213 \times 10^{-4}$
1.25	.80	.00675	.03400	$.123 \times 10^{-2}$	$.244 \times 10^{-4}$	$.193 \times 10^{-6}$
	.70	.00800	.02650	$.902 \times 10^{-3}$	$.179 \times 10^{-4}$	$.149 \times 10^{-6}$
	.50	.01350	.01125	$.107 \times 10^{-2}$	$.332 \times 10^{-4}$	$.481 \times 10^{-6}$
1.50	.80	.00425	.02125	$.112 \times 10^{-3}$	$.691 \times 10^{-6}$	$.173 \times 10^{-8}$
	.70	.00525	.01625	$.981 \times 10^{-4}$	$.684 \times 10^{-6}$	$.204 \times 10^{-8}$
	.50	.00900	.00400	$.113 \times 10^{-3}$	$.136 \times 10^{-5}$	$.774 \times 10^{-8}$
1.75	.80	.00250	.01050	$.416 \times 10^{-5}$	$.636 \times 10^{-8}$	$.416 \times 10^{-11}$
	.70	.00250	.00825	$.249 \times 10^{-5}$	$.334 \times 10^{-8}$	$.196 \times 10^{-11}$
	.50	.00400	.00250	$.336 \times 10^{-5}$	$.816 \times 10^{-8}$	$.927 \times 10^{-11}$

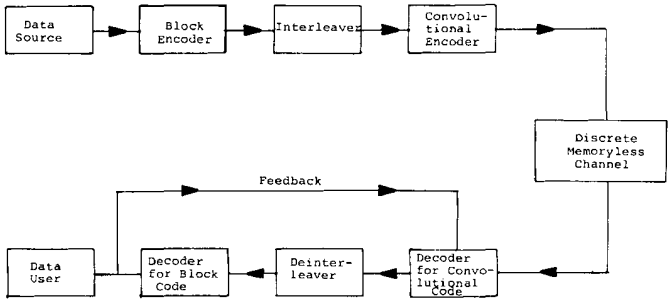


Fig. 3. A block/convolutional concatenated coding system with feedback from the outer decoder to the inner decoder.

the general concept of such a block-convolutional concatenated coding system.

To study the gain provided by feedback from the outer decoder, we first implemented a software Viterbi decoder and a software RTMBEP decoder which can be restarted with feedback. Assuming that the outer decoder always makes correct decisions, a justifiable assumption since the probability of byte-error at the outer decoder output is at least several orders of magnitude less than that at the inner decoder's output, we obtained the results shown in Table V for the (18, 6) unit-memory convolutional code on a simulated AWGN channel with an E_b'/N_0 of 1.25 dB. From Table V, we see that the RTMBEP decoder receives a considerably greater benefit from the feedback than does the Viterbi decoder. We then considered the following block-convolutional concatenated coding systems.

System III: A hard-decision Viterbi inner decoder with feedback from the errors-only RS outer decoder, i.e., System I with feedback.

System IV: A hard-decision RTMBEP inner decoder with feedback from the errors-only RS outer decoder.

System V: A soft-decision RTMBEP inner decoder with feedback from the erasures-and-errors RS outer decoder, i.e., System II with feedback.

TABLE V
THE EFFECT OF FEEDBACK FROM THE OUTER DECODER ON THE BYTE-ERROR PROBABILITY FOR A VITERBI DECODER AND AN RTMBEP DECODER ON A SIMULATED AWGN CHANNEL WITH AN E_b'/N_0 OF 1.25 DB (8000 BYTES DECODED FOR EACH POINT SHOWN, DECODING DELAY Δ OF 48 BITS IN EACH CASE)

	p for (18,6) unit memory code (95% confidence)		p for M = 7, (3,1) code (95% confidence)	
	No feedback	With feedback	No feedback	With feedback
Viterbi Decoder	.0200 (±.0032)	.0110 (±.0023)	.0225 (±.0034)	.0133 (±.0025)
RTMBEP Decoder	.0193 (±.0031)	.0075 (±.0019)		

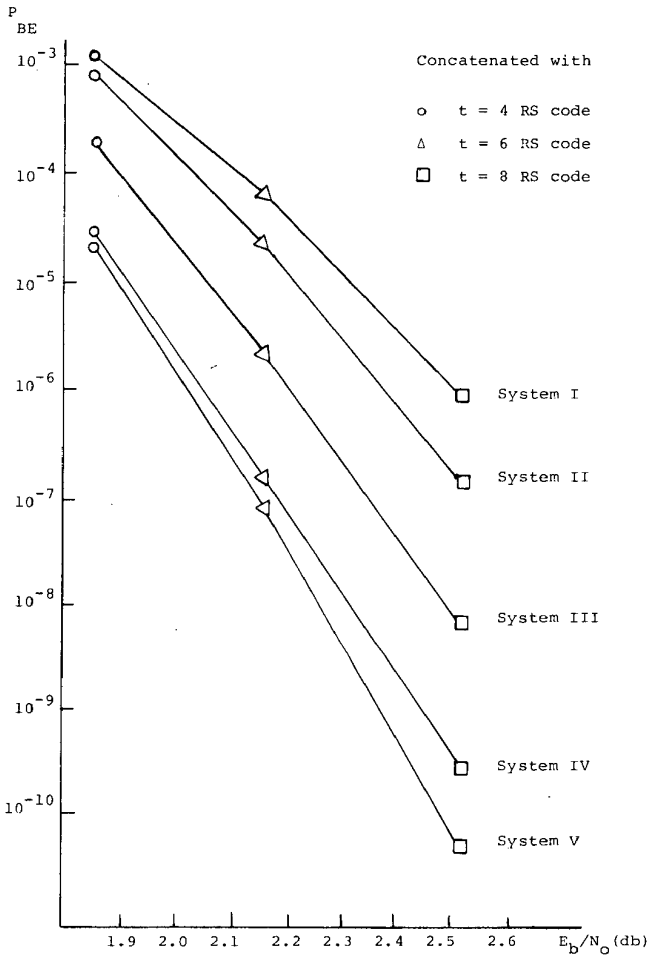


Fig. 4. Performance of Concatenated Coding Systems I-V employing the (18, 6) unit-memory convolutional code and RS codes over $GF(2^6)$ on a simulated AWGN channel with $E_b'/N_0 = 1.25$ dB.

The performance of Systems III, IV and V when used with the (18, 6) unit-memory code on the AWGN channel are shown in Fig. 4. For ease of comparison, the corresponding performances of Systems I and II, given in Fig. 2, are repeated in Fig. 4. By comparing performances between Systems I and III, we see from Fig. 4 that feedback from the outer decoder improves the system by about 0.3 dB for a hard-decision Viterbi inner decoder. As can be seen from Table V, the performance of a *hard-decision* RTMBEP inner decoder is virtually indistinguishable from that of a Viterbi inner decoder for a unit-memory code; thus, the performance of System I in Fig. 4 is also the performance of the system with an RTMBEP inner decoder without feedback from an errors-only RS outer decoder. Hence, by comparing the performances of Systems I and IV in Fig. 4, we can conclude that feedback from the outer decoder improves the system by a full 0.5 dB for a hard-decision RTMBEP inner decoder. By comparing the performances of Systems IV and V in Fig. 4, we can further conclude that, when feedback from the outer decoder is used, an additional 0.1 dB improvement can be gained by using a soft-decision RTMBEP inner decoder rather than a hard-decision one—the same improvement as was observed in the previous section when there was no feedback from the outer decoder.

V. ZEOLI'S TAIL ANNEXATION SCHEME APPLIED TO A UNIT-MEMORY CONVOLUTIONAL CODE

In [10], Zeoli proposed a concatenated coding system that employed a rather long constraint length ($K = 32$, i.e., $M = 31$) convolutional code obtained by annexing a long tail to the $M = 7$, (3, 1) convolutional code. The longer code is then decoded by the *same* Viterbi decoder as for the short code with the exception that the information sequence along the best path to each state is treated as correct and used to "cancel" the effect of the longer tail from the encoded sequence. Thus, the decoder state complexity remains the same as that for the original code and the annexed tail has absolutely *no effect on the hard-decision decoding error probability until after an error has been made.* But the tail provides excellent "error-detection" once the Viterbi decoder starts to make mistakes. Because the tail is not canceled when a decoding error is made, the state metrics become extremely ominous after a few decoded branches and can be used as the basis for excellent erasure rules for the output of the inner decoder. However, feedback from the outer decoder is no longer an option, but now a necessity in order to reset the decoder to the correct state and thus to terminate the very "error propagation" used to trigger the erasure alarm.

To study the improvement resulting from Zeoli's scheme, we annexed, to the (18, 6) unit-memory convolutional code, a three-branch-long "random tail" such that the resultant code is actually an $M = 4$, (18, 6) convolutional code. The encoding matrices of this latter convolutional code are shown in Table VI. The length of the tail was chosen to be comparable in memory to the $M = 31$, (3, 1) code used in [10]. (Because the decoder is intended to made mistakes continually after its first error, it makes no difference whether the annexed $M = 4$, (18, 6) code is catastrophic [15] or not.) The last of the systems to be considered in this paper, *System VI,* is that of Zeoli [10], namely a soft-decision Viterbi inner decoder with feedback from an errors-and-erasures RS outer decoder, with the $M = 4$, (18, 6) code replacing his conventional $M = 31$, (3, 1) code.

The state metric used in the "real time Viterbi decoder" [14] of System VI, namely $\mu(t + \Delta) = \log P(a_{[1,t+\Delta]} \mid r_{[1,t+\Delta]})$ when $\hat{a}_{[1,t+\Delta]}$ is the "best path" at time $t + \Delta$, can be used as the basis for an effective erasure rule as follows. The difference, $\mu(t + \Delta) - \mu(t)$, is, along the correct encoded path, the sum of Δn_0 is statistically independent random variables, each corresponding to one encoded bit. Note that, for System VI, $\Delta n_0 = 8(18) = 144$. The central-limit-theorem can

TABLE VI
THE ENCODING MATRICES OF THE $M = 4$ (18, 6) CONVOLUTIONAL CODE OBTAINED BY ANNEXING A RANDOMLY CHOSEN TAIL TO THE (18, 6) UNIT-MEMORY CODE

$$G_0 = \begin{bmatrix} 111000 & 110100 & 110001 \\ 011100 & 011010 & 011000 \\ 001110 & 001101 & 001100 \\ 000111 & 100110 & 000110 \\ 100011 & 010011 & 000011 \\ 110001 & 101001 & 100001 \end{bmatrix} \quad G_1 = \begin{bmatrix} 000011 & 000111 & 001011 \\ 000110 & 001110 & 010110 \\ 001100 & 011100 & 101100 \\ 011000 & 111000 & 011001 \\ 110000 & 110001 & 110010 \\ 100001 & 100011 & 100101 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 000110 & 000001 & 101111 \\ 100011 & 000011 & 010011 \\ 110001 & 100110 & 100001 \\ 111000 & 110101 & 001000 \\ 011000 & 011010 & 011100 \\ 001100 & 011100 & 110110 \end{bmatrix} \quad G_3 = \begin{bmatrix} 011000 & 111001 & 011000 \\ 110001 & 110010 & 110000 \\ 100011 & 100101 & 100001 \\ 000111 & 000011 & 001011 \\ 000110 & 011100 & 010110 \\ 001100 & 011100 & 101100 \end{bmatrix}$$

$$G_4 = \begin{bmatrix} 111111 & 010100 & 000000 \\ 000111 & 111010 & 100000 \\ 000000 & 111111 & 010100 \\ 100000 & 000111 & 111010 \\ 010100 & 000000 & 111111 \\ 111010 & 100000 & 000111 \end{bmatrix}$$

thus be invoked to assert that $\mu(t + \Delta) - \mu(t)$ is approximately Gaussian. Letting m and σ be the (easily calculable) mean and standard deviation of $\mu(t + \Delta) - \mu(t)$, it is natural to use the *erasure rule*: Erase a_t whenever $\mu(t + \Delta) - \mu(t)$ is more than λ standard deviations above m. In Table VII, we give the performance of System VI using this erasure rule for $\lambda = 1.5$, 1.8 and 2.0; the value 1.8 is seen to give the best performance. Note that if $\mu(t + \Delta) - \mu(t)$ were truly Gaussian, the probability that it would exceed $m + 1.8\sigma$ (i.e., the probability of an erasure in the Viterbi decoder output) would be .036; the observed value of 0.21 given in Table VII is rough confirmation of the appropriateness of the Gaussian approximation.

The performance of System VI on the AWGN channel is shown in Fig. 5; for comparison, the performance of Zeoli's original system, taken from [10], is also shown. The performance of Systems III and V, given in Fig. 4, are also repeated in Fig. 5 to indicate how System VI compares to the systems previously considered. By comparing the performance of Systems III and VI, we see that Zeoli's tail annexation scheme (and the resulting erasure capability) has improved the performance of the feedback system with a Viterbi inner decoder by about 0.2 dB.

VI. DEGRADATION OF PERFORMANCE FOR EMPLOYING HIGHER RATE INNER CODES

We have studied, rather extensively, block-convolutional concatenated coding systems employing rate 1/3 convolutional codes and Reed-Solomon codes over $GF(2^6)$. However, it is sometimes desired in practice to operate the inner convolutional codes at a higher rate (i.e., narrower bandwidth), rate 1/2 in particular, in order to ease the burden imposed on the phase-lock loops in the receiver. We now describe an heuristic approach to estimate the performance of similar concatenated coding systems with rate 1/2 coding systems from the rate 1/3 results.

From past experience [16], it has been observed that the performance of a rate 1/2 convolutional coding system is about 0.5 dB inferior to that of a rate 1/3 convolutional coding system of the same complexity. To verify the general applicability of this rule-of-thumb, we used a hard-decision

TABLE VII
VARIATION OF INNER DECODER BYTE-ERROR PROBABILITY p AND BYTE-ERASURE PROBABILITY q AND OF THE OUTER DECODER BYTE-ERROR PROBABILITY P_{BE} WITH THE ERASURE PARAMETER FOR THE $M = 4$ (18, 6) CODE OBTAINED BY ANNEXING A TAIL TO THE (18, 6) UNIT-MEMORY CODE ON A SIMULATED AWGN CHANNEL WITH AN E_b'/N_0 OF 1.25 DB AND WITH THE MINIMUM DISTANCE d_{min} OF THE OUTER RS CODE

λ	p	q	P_{BE} for $d_{min} = 9$	P_{BE} for $d_{min} = 13$	P_{BE} for $d_{min} = 17$
1.50	.00125	.03788	2.095×10^{-4}	8.175×10^{-7}	9.465×10^{-10}
1.80	.00263	.02088	3.602×10^{-5}	1.074×10^{-7}	1.245×10^{-10}
2.00	.00425	.01450	4.168×10^{-5}	1.899×10^{-7}	3.708×10^{-10}

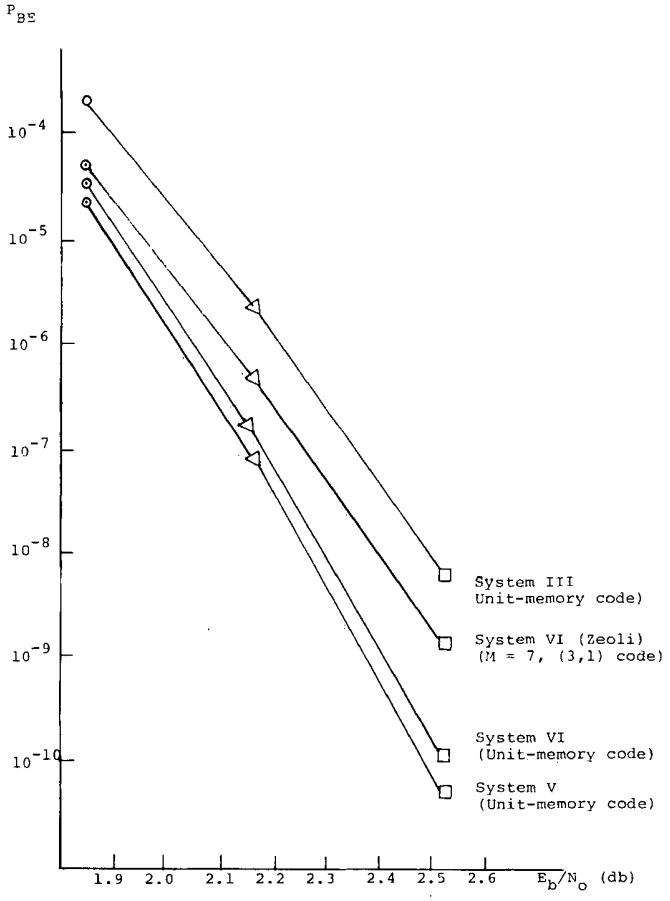


Fig. 5. Performance of Zeoli's tail annexation scheme (System VI) on a simulated AWGN channel with $E_b'/N_0 = 1.25$ dB, and comparison with other concatenated coding systems.

Viterbi decoder (without feedback) for an $M = 6$, (2, 1) convolutional code on a simulated AWGN channel at $E_b'/N_o = 1.75$ dB, or, equivalently, $E_s/N_0 = -1.25$ dB. The results of this simulation and the calculated overall byte-error-probability when this decoder is used with an errors only RS outer decoder concatenated with Reed-Solomon codes are given in Figure 6. For comparison, the performance of the similar $R = 1/3$ system employing the $M = 6$, (3, 1) code is also shown. We see from Fig. 6 that the latter system is about 0.5 dB superior to the former. It seems reasonable then to conclude

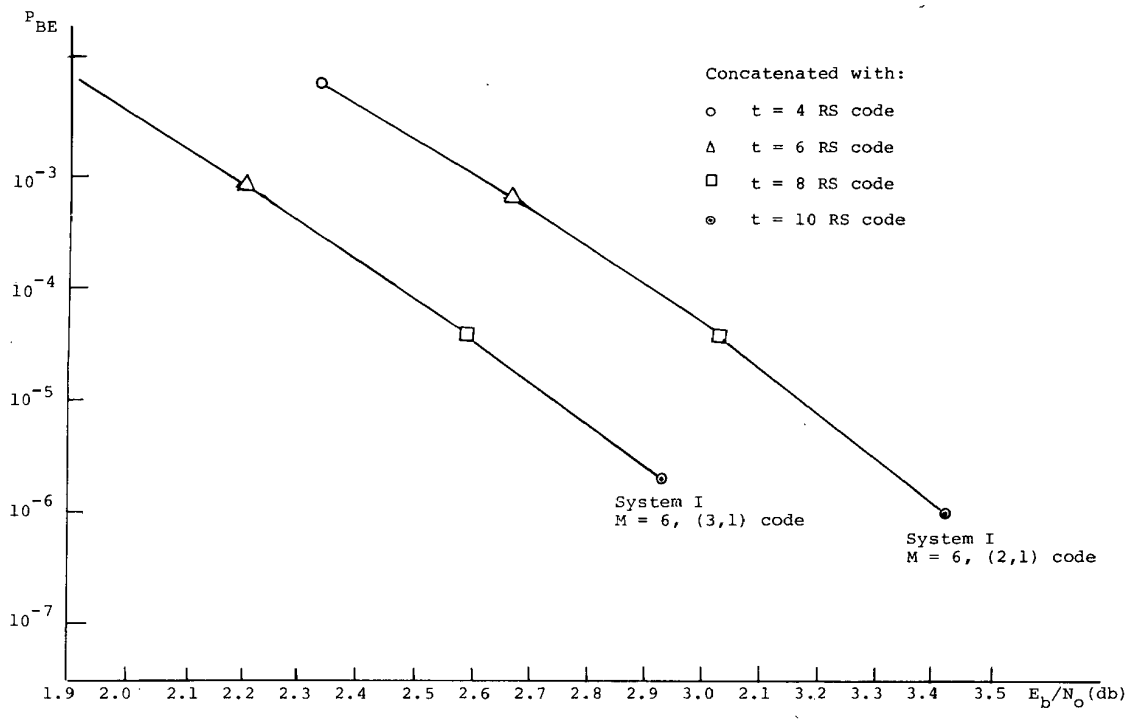


Figure 6. Performance of Concatenated Coding System I on a simulated AWGN channel with $E_b'/N_0 = 1.75$ dB when a rate 1/2, $M = 6$, (2, 1) convolutional code is used, and with $E_b'/N_0 = 1.25$ dB when a rate 1/3, $M = 6$, (3, 1) convolutional code is used.

that a concatenated block-convolutional coding system with a rate 1/2 inner code may be about 0.5 dB inferior to that with a rate 1/3 inner code for the same number of decoder states for the Viterbi inner decoder, though we should be a little bit cautious that performance of rate 1/2 and rate 1/3 inner coding system with soft-decision and errors and erasures outer decoders have not been compared.

VII. CONFIDENCE INTERVALS FOR THE SIMULATION RESULTS

In the preceding, we have reported the performances of numerous block-convolutional concated coding systems. The overall byte-error rate was calculated from the byte-error rate of the inner decoder as obtained by simulation. The rather large values of P_{BE} for the inner decoding imply that the simulations require only a modest sample size. The predominate single-byte error events indicate a quite independent byte-decision. Assuming that the decoder makes an error with probability P_{BE} independently for each byte-decision, the number of byte errors for L decisions is a binomial random variable with parameters L and P_{BE}. The mean value of this random variable is LP_{BE}, and the standard deviation is $\sqrt{LP_{BE}(1 - P_{BE})}$. L is sufficiently large for this binomial random variable to be well approximated by a Gaussian random variable with the same mean and variance. Since 95.4% of the samples of a Gaussian random variable are within the interval specified by the mean plus and minus twice the standard deviation, we can be 95% confident that the actual byte-error rate for the inner decoder is in the interval $P_{BE} \pm 2\sqrt{LP_{BE}(1 - P_{BE})}$. Such 95% confidence intervals are indicated in Tables II and V.

The performances of System I for the $M = 6$, (3, 1) inner code and for the (18, 6) unit-memory inner code are shown in Fig. 7 together with their corresponding confidence intervals. We conclude that we may be 95% confident that the actual performance of the concatenated coding system deviates no more than about 0.1 dB from our simulation results. Moreover, since all the simulation results are obtained through the same pseudorandom number sequence, the *relative* differences in performance among various systems are, in fact, much more accurate than the 0.1 dB confidence interval alone would indicate.

VIII. SUMMARY AND CONCLUSIONS

We have extensively studied block-convolutional concatenated coding systems with various modifications. We have found that employing unit-memory convolutional codes rather than conventional codes can improve the performance by nearly 0.3 dB. Feedback from the outer decoder to restart a Viterbi inner decoder also contributes an improvement of about 0.3 dB. But, surprisingly, feedback from the outer decoder to restart an RTMBEP inner decoder provides an approximately 0.5 dB advantage; this might be the principal occasion where the use of RTMBEP decoding rather than Viterbi decoding is justified. Another unexpected result is that soft-decisions by the inner decoder in conjunction with an erasures-and-errors outer decoder improve the overall performance by only about 0.1 dB for RTMBEP decoding. Even with

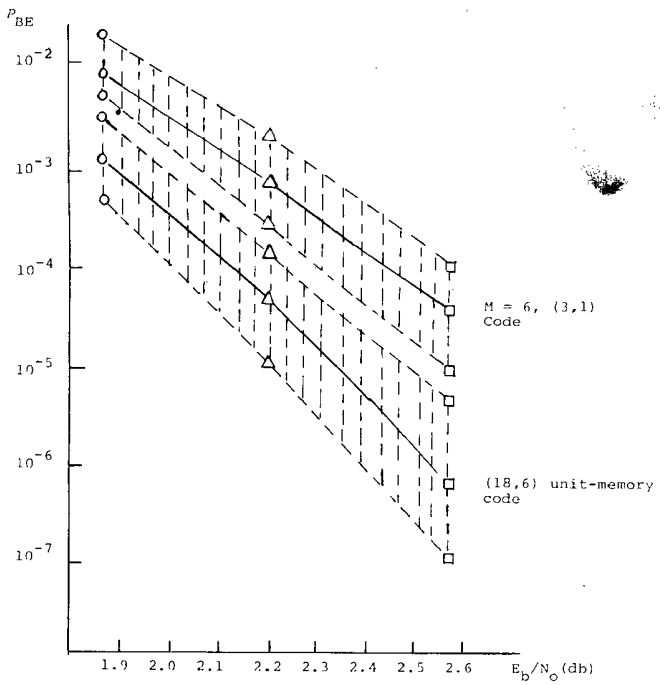


Fig. 7. 95% Confidence Intervals for the performance of System I with the $M = 6$, (3, 1) convolutional code and with the (18, 6) unit-memory convolutional code.

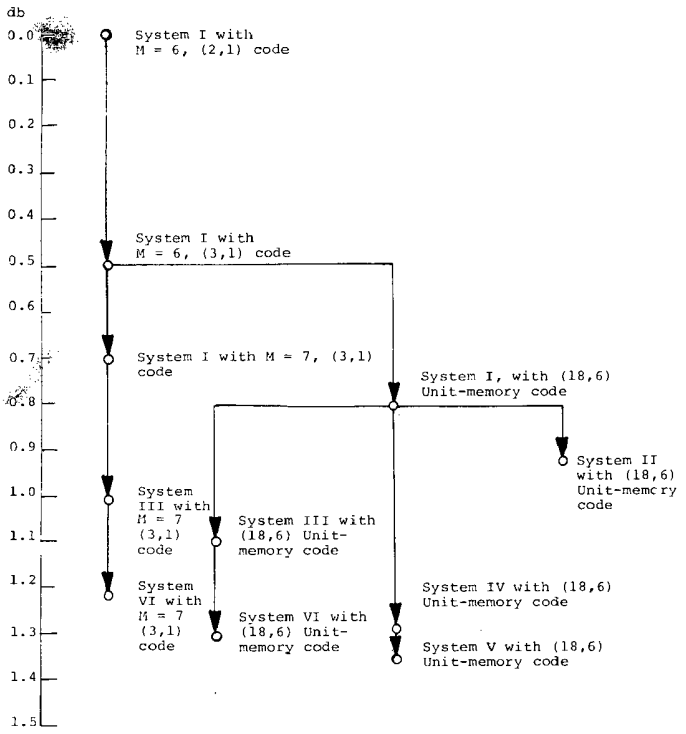


Fig. 8. The relative dB gains among the concatenated coding systems studied.

Zeoli's modification, which provides an excellent erasure capability, soft-decisions in conjection with an erasures-and-errors outer decoder improve performance by about only 0.2 dB.

In Fig. 8, we summarize the effects of each feature discussed above on the performance of block-convolutional concatenated coding systems. The figure is drawn in terms of a dB scale. As a communications engineer starts to choose a concatenated coding system, the first question he faces is whether he is willing to trade the increased cost of modems to operate at lower signal energy per channel bit for coding gains, if he decides to choose a rate 1/3 convolutional inner code instead of a rate 1/2 code, he gains about 0.5 dB. Then, he decides which inner code to employ; to choose the $M = 7$, (3, 1) code gives a 0.2 dB advantage over the $M = 6$ (3, 1) code but requires twice the number of states in the decoder, whereas to choose the $M = 1$, (18, 6) code gives a 0.3 dB advantage with same number of states, but more branch connections required in the inner decoder. The third question is whether he will allow the decisions of the outer decoder to be fed back to the inner decoder; if not, the obvious choice is Viterbi decoding. Otherwise, he can gain 0.3 dB or 0.5 dB, depending on whether a Viterbi decoder or an RTMBEP decoder is utilized. And finally, if a soft-decision inner decoder is used, he can gain 0.2 dB through Zeoli's erasure scheme if he uses a Viterbi decoder, or gain about 0.05 dB if an RTMBEP decoder is employed.

The leading contenders for a good concatenated system are Zeoli's annexation scheme with the unit-memory code (System VI), or either hard decision (system IV) or soft-decision (System V) RTMBEP decoding of the unit-memory code with feedback from the outer decoder. Among them, the soft-decision RTMBEP decoder with feedback performs the best. In terms of hardware implementation, Zeoli's modification with the unit-memory code and the hard-decision RTMBEP decoder are of approximately the same complexity. However, since the operation of the Viterbi decoder for Zeoli's system depends on the correct feedback from the outer decoder, there is always a slim chance that the outer decoder fails to provide correct decisions to the Viterbi decoder. Since the encoder constraint length is much larger than the decoder constraint length, this can cause endless errors as if a catastrophic convolutional code were used. Thus, it is necessary to send synchronization signals in the Zeoli scheme periodically to reset the Viterbi decoder to guarantee restoration of normal operation. the RTMBEP decoder has the same constraint length as that of the encoder; therefore, the decoder is able to recover from errors in a few branches by itself without feedback. The feedback from the outer decoder only speeds this process up therefore, when an error is fed back, the most damage it can cause is for the RTMBEP decoder to make a few more errors before it recovers by itself. This is certainly a very desirable advantage for a concatenated coding system. Moreover, because the decoder can restore its normal operation quickly, the degree of interleaving required for this scheme is considerably less than the full Reed-Solomon block length interleaving required for the Zeoli's scheme.

Finally, as a remark to information theorists, we note that for System III (the RTMBEP inner decoder for the rate 1/3 (18, 6) unit-memory code concatenated with the (63, 51), 6-error-correcting RS code with feedback from the RS errors-only decoder) we can achieve a byte-error-probability of

10^{-6} at E_b/N_0 of 2.17 dB, or, equivalently, at E_s/N_0 of 3.52 dB. The cut-off rate, R_{comp}, of this 8-level quantized AWGN channel is 0.275 whereas its channel capacity is 0.44. The overall rate of the concatenated coding system is 0.27. It seems that the cut-off rate, rather than the channel capacity, is still the practical limit of rate for reliable communications, even for a very sophisticated concatenated coding system, just as it is in a conventional convolutional coding system employing sequential decoding [16]. The advantage of the concatenated coding system resides only in the elimination of "deleted data" such as is always present in a sequential decoding system because of the latter's highly variable computation.

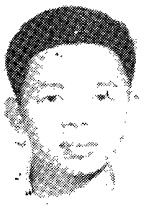
ACKNOWLEDGMENT

The author would like to express his gratitude to his Dissertation Advisor, Professor James L. Massey for his patient and generous guidance of this research and for his suggestions to match byte-oriented convolutional codes with Reed-Solomon codes and to develop the "real-time minimal-byte-error probability (RTMBEP) decoding algorithm."

REFERENCES

[1] P. Elias, "Error-Free Coding," *IRE Transactions on Information Theory,* Vol. IT-4, pp. 29-37, Sept. 1954.
[2] G. D. Forney, Jr., *Concatenated Codes.* Cambridge, Mass.: M.I.T. Press, 1966.
[3] D. D. Falconer, "A Hybrid Sequential and Algebraic Decoding Scheme," Ph.D. Dissertation, Dept. of Electrical Engineering, M.I.T., Cambridge, Mass., 1966.
[4] F. Jelinek and J. Cocke, "Bootstrap Hybrid Decoding for Symmetrical Binary Input Channels," *Information and Control,* Vol. 18, pp. 261-298, March 1971.
[5] A. J. Viterbi, "Error Bounds for Convolutional Codes and An Asymptotically Optimal Decoding Algorithm," *IEEE Transactions on Information Theory,* Vol. IT-13, pp. 260-269, April 1967.
[6] J. P. Odenwalder, "Optimal Decoding of Convolutional Codes," Ph. D. Dissertation, School of Engineering and Applied Sciences, University of California, Los Angeles, 1970.
[7] E. R. Berlekamp, *Algebraic Coding Theory.* New York: McGraw-Hill, 1968.
[8] J. L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Transactions on Information Theory,* Vol. IT-15, pp. 122-125, Jan. 1969.
[9] L. N. Lee, "Short, Unit-Memory, Byte-Oriented, Binary Convolutional Codes Having Maximal Free Distance," *IEEE Transactions on Information Theory,* Vol. IT-22, No. 3, pp. 349-352, May 1976.
[10] G. W. Zeoli, "Coupled Decoding of Block-Convolutional Concatenated Codes," *IEEE Transactions on Communications,* Vol. COM-21, pp. 219-226, March 1973.
[11] F. Jelinek, "Bootstrap Trellis Decoding," *IEEE Transactions on Information Theory,* Vol. IT-21, pp. 318-325, May 1975.
[12] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory,* Vol. IT-20, pp. 284-288, March 1974.
[13] P. L. McAdam, L. R. Welch and C. L. Weber, "M.A.P. Bit Decoding of Convolutional Codes," presented at 1972 International Symposium on Information Theory, Asilomar, California, January, 1972.
[14] L. N. Lee, "Real-Time Minimal-Bit-Error Probability Decoding of Convolutional Codes," *IEEE Transactions on Communications,* Vol. COM-22, pp. 146-151, Feb. 1974.
[15] J. L. Massey and M. K. Sain, "Inverses of Linear Sequential Circuits," *IEEE Transactions on Computers,* Vol. C-17, pp. 330-337, April 1968.
[16] I. M. Jacobs, "Sequential Decoding for Efficient Communications from Deep Space," *IEEE Transactions on Communication Technology,* Vol. COM-15, pp. 492-501, Aug. 1967.
[17] W. W. Peterson and E. J. Weldon, *Error Correcting Codes,* Second Edition MIT Press, Cambridge, Mass., 1972.
[18] L. N. Lee, "Concatenated Coding Systems Employing Unit-Memory Convolutional Codes and Byte-Oriented Decoding Algorithms", Ph.D. Dissertation, Department of Electrical Engineering, University of Notre Dame, June 1976.

★



Lin-nan Lee (S'73-M'76) was born in Kaohsiung, Taiwan, China, on February 24, 1949. He received the B.S. degree from National Taiwan University, Taipei, Taiwan, China, in 1970, and the M.S. and the Ph. D. degrees from the University of Notre Dame, Notre Dame, IN, in 1973 and 1976, respectively, all in Electrical Engineering.

From 1970 to 1971, he was a Communications Officer in the Chinese Air Force. Between 1971 and 1975, he was a research assistant at the University of Notre Dame engaged in graduate studies on digital communications with emphasis on information and coding theories. In 1975, he joined the LINKABIT Corporation where he is presently engaged in study, research and development of coding, modulation and multiple access techniques for satellite communication systems.

Exhibit 7

GSM ENHANCED FULL RATE SPEECH CODEC

K. Järvinen, J. Vainio, P. Kapanen, T. Honkanen, P. Haavisto
Nokia Research Center, Tampere, Finland
kari.jarvinen@research.nokia.com

R. Salami, C. Laflamme, J-P. Adoul
University of Sherbrooke, Sherbrooke, Canada

ABSTRACT

This paper describes the GSM enhanced full rate (EFR) speech codec that has been standardised for the GSM mobile communication system. The GSM EFR codec has been jointly developed by Nokia and University of Sherbrooke. It provides speech quality at least equivalent to that of a wireline telephony reference (32 kbit/s ADPCM). The EFR codec uses 12.2 kbit/s for speech coding and 10.6 kbit/s for error protection. Speech coding is based on the ACELP algorithm (Algebraic Code Excited Linear Prediction). The codec provides substantial quality improvement compared to the existing GSM full rate and half rate codecs. The old GSM codecs lack behind wireline quality even in error-free channel conditions, while the EFR codec provides wireline quality not only for error-free conditions but also for the most typical error conditions. With the EFR codec, wireline quality is also sustained in the presence of background noise and in tandem connections (mobile to mobile calls).

1. INTRODUCTION

The background for introducing wireline speech quality to GSM is the increasing use of the GSM system in communications environments where it competes with fixed or cordless systems. To be competitive also with respect to speech quality, GSM must provide wireline speech quality which is robust to typical usage conditions such as background noise and transmission errors.

The standardisation of an enhanced full rate (EFR) codec for GSM started in European Telecommunications Standards Institute (ETSI) in 1994 with a pre-study phase. The pre-study phase was undertaken to set essential requirements for the EFR codec and to assess the technical feasibility of meeting them. During the pre-study phase, wireline speech quality was set as a development target for the EFR codec [1]. Wireline quality (with ITU-T G.728 16 kbit/s LD-CELP as a reference codec) was required not only for error-free transmission, but also in low error-rate conditions (C/I=13 dB) as well as in background noise (error-free conditions). Wireline performance was required also for speaker independence and for speaker recognisability. For more severe error conditions (C/I=10 dB and C/I=7 dB) significant improvement to the existing GSM full rate (FR) codec was required. In extreme error conditions (C/I<7 dB) the requirement was to provide graceful degradation without annoying effects. In error-free self-tandem (mobile to mobile calls) the EFR codec should perform equal to G.728 in tandem. In erroneous tandem at C/I=10 dB, the EFR codec was required to perform significantly better than the FR codec.

In addition, essential requirements were set for bit-rate, complexity, and delay. The same channel bit-rate of 22.8 kbit/s was required as used in the existing FR codec. The complexity was not to exceed the complexity of the GSM half rate codec. The requirement for one way end-to-end delay was to be no more than in the FR channel.

A competitive EFR codec selection process was launched in ETSI in 1995. Altogether six EFR candidate codecs were submitted into the first phase of testing (pre-selection tests) which started in August 1995. Based on the pre-selection test results and also results from complementing verification tests the EFR candidate codec jointly developed by Nokia and University of Sherbrooke (USH) was selected for GSM in October 1995. A few months earlier the same codec had been chosen as the EFR codec for the US PCS 1900 system which is based on the GSM technology. The advantage of using the same codec in PCS 1900 and in GSM was one more factor in favour of this particular solution. During 1996, verification tests for the EFR codec have been completed and the GSM specifications have been finalised.

2. SPEECH CODEC

The GSM EFR speech codec is based on the ACELP algorithm (Algebraic Code Excited Linear Prediction) [2]. The speech coding (source coding) bit-rate is 12.2 kbit/s. For channel coding (error protection) 10.6 kbit/s is used resulting in 22.8 kbit/s channel bit-rate. The EFR codec employs 0.8 kbit/s more error protection than the FR codec where speech coding bit-rate is 13.0 kbit/s. The bit allocation of the GSM EFR codec is shown in Table I. A block diagram of the encoder is shown in Figure 1.

Parameter	1st and 3rd subframe	2nd and 4th subframe	Total per frame
2 LSP sets			38
ACB index (lag)	9	6	30
ACB gain	4	4	16
FCB pulses	35	35	140
FCB gain	5	5	20
Total			244

Table I: The parameter bit-allocation of the GSM EFR codec.

The EFR codec operates on 20 ms speech frames which are divided into four 5 ms subframes. In the encoder, the speech signal is analysed and the parameters of the CELP speech synthesis model are extracted. Two sets of linear prediction filter coefficients are calculated for each frame. The indices for the adaptive (ACB) and fixed codebooks (FCB) as well as their gains are searched for each subframe. In the decoder, a spectral post-filter is used for quality enhancement.

0-8186-7919-0/97 $10.00

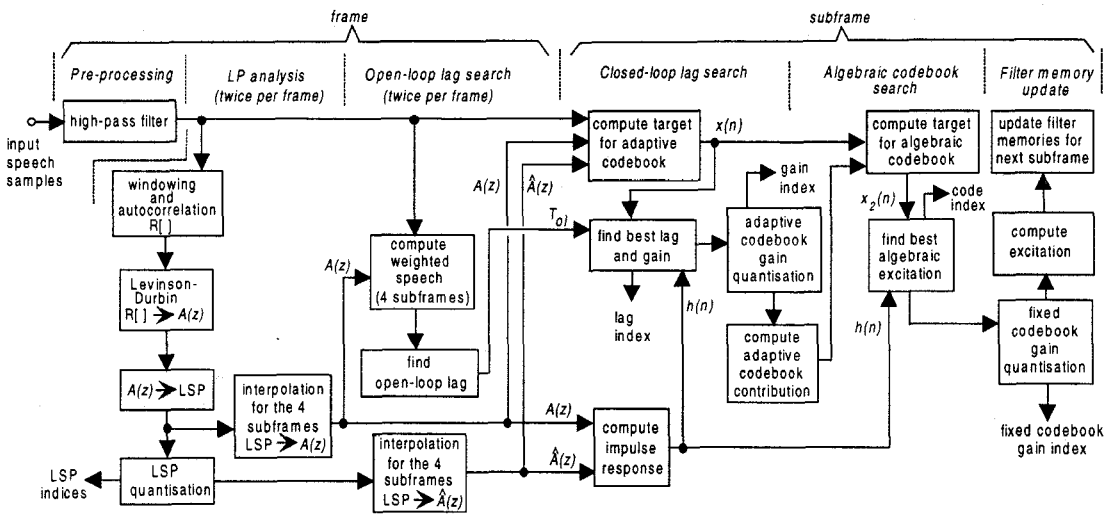


Figure 1: Block diagram of the GSM EFR encoder.

2.1 Linear Prediction

A 10th order linear prediction (LP) analysis is carried out twice for each 20 ms frame using two different asymmetric windows of length 30 ms. Both LP analyses are performed for the same set of speech samples without using any samples from future frames (no lookahead). The two sets of LP parameters are converted into line spectrum pairs (LSP). First order moving average prediction is used for both LSP sets. The LSP residual vectors are jointly quantised using split matrix quantisation (SMQ) with 5 submatrices of dimension 2x2 (two elements from both sets). The submatrices are quantised with 7, 8, 9, 8, and 6 bits, respectively. A total of 38 bits are used for LSP quantisation. For the 1st and 3rd subframes, LP parameters interpolated from the adjacent subframes are used in the codec.

2.2 Pitch Analysis

The adaptive codebook is searched for the lag range [17 3/6, 143] with a combined open-loop/closed-loop search [2]. A fractional lag with 1/6th resolution is used for lag values below 95 in the 1st and 3rd subframes and for all lag values in the 2nd and 4th subframes. The codebook search consists of the following steps:

- An open-loop search for integer lag values is carried out once every 10 ms from the weighted original speech. Small lag values are preferred to avoid pitch multiples.
- A closed-loop search for integer lag values is performed on subframe basis. For the 1st and 3rd subframe the search is carried out in the vicinity of the found open-loop lag [T_{ol} - 3, T_{ol} + 3] and for the 2nd and 4th subframe in the vicinity of the lag found for the previous subframe [T_{ps} - 5, T_{ps} + 4].
- Fractions are searched around the closed-loop lag if it is less than 95 (and always in the 2nd and 4th subframes).

The lag is quantised with 9 bits for the 1st and 3rd subframes and with 6 bits for the other two subframes where the lags are coded differentially. The codebook gain is quantised to 4 bits.

2.3 Fixed Codebook

An algebraic codebook with 35 bits is used as the fixed codebook. Each excitation vector contains 10 non-zero pulses, with amplitudes +1 or -1. The 40 positions in each subframe are divided into 5 tracks where each track contains two pulses. In the design, the two pulses for each track may overlap resulting in a single pulse with amplitude +2 or -2. The allowed positions for pulses are shown in Table II.

Track	Pulses	Positions
1	*p0, p1*	0, 5, 10, 15, 20, 25, 30, 35
2	*p2, p3*	1, 6, 11, 16, 21, 26, 31, 36
3	*p4, p5*	2, 7, 12, 17, 22, 27, 32, 37
4	*p6, p7*	3, 8, 13, 18, 23, 28, 33, 38
5	*p8, p9*	4, 9, 14, 19, 24, 29, 34, 39

Table II: Allowed pulse positions for each track.

The optimal pulse positions are determined using a non-exhaustive analysis-by-synthesis search:

- For each of the five tracks, the pulse positions with maximum absolute values of the sum of normalised backward filtered target and normalised long-term prediction residual are searched. From these the global maximum value for all the pulse positions is selected. The first pulse *p0* is always set in the position corresponding to the global maximum value.
- Five iterations are carried out in which the position of pulse *p1* is set to one of the five track maxima. The rest of the pulses are searched in pairs by sequentially searching each of the pulse pairs {*p2, p3*}, {*p4, p5*}, {*p6, p7*}, {*p8, p9*} in nested loops. For each iteration, all 9 initial pulse positions are cyclically shifted so that the pulse pairs are changed and the pulse *p1* is placed at a local maximum of a different track. The rest of the pulses are searched also for the other positions in the tracks. In the search, at least one pulse position is located corresponding to the global maximum and one pulse is located in a position corresponding to one of the 5 local maxima.

For subframes with a lag less than the subframe length, adaptive pitch sharpening filter is used. The two pulse positions in each track are encoded with 6 bits and the sign of the first pulse in each track is encoded with one bit. The fixed codebook gain is coded using moving average prediction and quantised to 5 bits.

2.4 Error Concealment

Error concealment in the EFR codec is based on partially replacing the parameters of the received bad frame with values extrapolated from the previous good frames:

- The LSPs of the previous good frame are used but shifted towards their mean values.
- The codebook gains are replaced by attenuated values derived from the previous subframes using median filtering. The amount of attenuation is different for the two codebooks and depends on which state the error concealment is in. The lag values are replaced by the lag value from the 4th subframe of the previous good frame.
- The received excitation pulses of the fixed codebook are used as such.

In case a good frame is preceded by a bad frame, the codebook gains for the good frame are limited below values used for the last good subframe.

3. CHANNEL CODEC

The EFR channel codec is almost the same as the FR channel codec because the design aim was to keep it as unchanged as possible. During the GSM EFR codec standardisation, the use of the existing FR channel codec (or any existing GSM generator polynomials) was encouraged since this minimises hardware changes in the GSM base stations and speeds up the introduction of the EFR codec. In the PCS 1900 EFR codec standardisation, the use of the existing FR channel codec was an essential requirement. Therefore, the FR channel codec was included in the EFR channel codec as a module together with additional error protection. The additional error protection consists of an 8-bit CRC parity check and a repetition code. The FR channel codec module protects the 182 most important bits with 1/2-rate convolution code and it uses a 3-bit CRC that covers the 50 most important bits. The bits in the EFR codec are divided into protected and unprotected bits according to their subjective importance to speech quality. Only 66 bits are left unprotected. These consist of the least significant bits of pulse positions in the algebraic code. The 8-bit CRC covers the 65 most important bits. It was included in the EFR codec to achieve reliable bad frame detection and, consequently, to reduce the number of undetected bad frames. These are a major source of audible degradations in current digital cellular systems.

4. VAD/DTX

The EFR codec contains also the functions of discontinuous transmission (DTX) and voice activity detection (VAD). DTX allows the radio transmitter to be switched off during speech pauses in order to save power in the mobile station and also to reduce the overall interference level over the air interface. VAD is used on the transmit side to detect speech pauses, during which characteristic parameters of the background acoustic noise are transmitted to the receive side, where similar noise, referred to as comfort noise, is then generated. The comfort noise parameters in the EFR codec consist of averaged LSP parameters and an averaged fixed codebook gain. Locally generated random numbers are used on the receive side as excitation pulses. During comfort noise generation, the adaptive codebook is switched off. The estimated average speech channel activity for the EFR codec is 64% [3].

5. COMPLEXITY AND DELAY

The complexity of the EFR codec has been estimated from a C-code implemented with a fixed point function library in which each operation has been assigned a weight representative for performing the operation on a typical DSP [3]. The theoretical worst case complexity of the EFR codec has been estimated during ETSI EFR verification phase to be 18.1 WMOPS (weighted million operations per second) [3]. This is below that of the GSM half rate codec (21.2 WMOPS) [3], [4]. Memory consumption estimated for data RAM (4.7k 16-bit words), data ROM (5.9k 16-bit words) and program ROM are each below those of the GSM half rate codec.

The delay of the EFR codec is approximately the same as that of the FR codec. Both codecs have a buffering delay of 20 ms without any lookahead. The round-trip delay (uplink delay + downlink delay) for EFR taking into account all system and processing delays of the GSM network is 191.0 ms while for the FR codec it is estimated to be 188.5 ms [3]. The difference is unnoticeable.

6. SUBJECTIVE TEST RESULTS

The most extensive subjective tests of the performance of the EFR codec are from the PCS 1900 EFR codec validation tests [5]. These were carried out to characterise the performance of the EFR codec after selection for PCS 1900. The standardisation process in ETSI also included pre-selection tests which were carried out in six laboratories, but no common analysis averaging the scores over all laboratories exists [3]. The results from both tests are well in line with each other. Both show that the EFR codec has basic speech quality at least equal to that of a wireline reference (G.721 32 kbit/s ADPCM in PCS 1900 tests and G.728 16 kbit/s LD-CELP in ETSI tests).

The PCS 1900 EFR codec validation test results are discussed first. The EFR codec was tested in three channel error conditions with C/I-ratios 13, 10 and 7 dB. The channel bit error-rates for these are approximately 2%, 5% and 8%, respectively. The two lowest error-rate conditions correspond to operating well inside a cell while the 7 dB C/I condition corresponds to operating at a cell boundary. Figure 2 shows the results from channel error test. These show that the EFR codec performs much better than the FR codec in the error-free case and in all the tested error conditions. In error-free and low error-rate conditions (C/I=13 dB), the performance of the EFR codec is statistically better (based upon 95% confidence intervals) than the performance of the error-free reference ADPCM codec. At medium error-rate conditions (C/I=10 dB) the EFR codec performs equally well to (error-free) ADPCM. Only at medium to high error-rates (C/I<10 dB) the EFR performance falls below wireline quality. Figure 3 shows the results from background noise test (home noise at 20 dB, car noise at 15 dB and 25 dB, street noise at 10 dB, and office noise at 20 dB). For all of these, the EFR codec performs clearly better than the FR codec and equal to or better than ADPCM. For scores averaged over all background noise conditions, the EFR codec performs statistically better than ADPCM. Figure 4 shows test results from tandem test for self-tandem and for tandeming with either FR or ADPCM codec. The EFR codec performs statistically equivalent to ADPCM in tandems with FR and ADPCM and statistically better than ADPCM for self-tandem. Figure 5 shows the results from talker dependency test (with 12 talkers). The

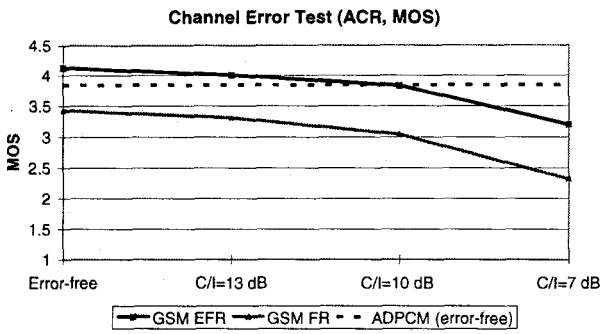


Figure 2: Results from channel error test.

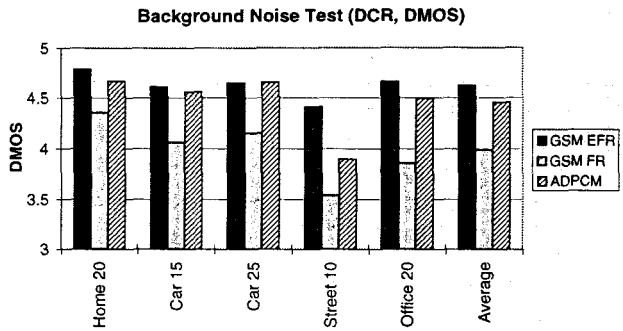


Figure 3: Results from background noise test.

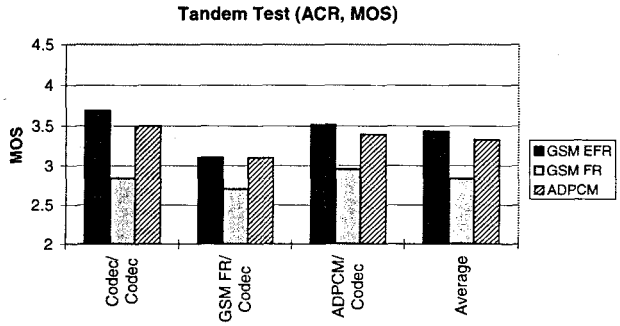


Figure 4: Results from tandem test.

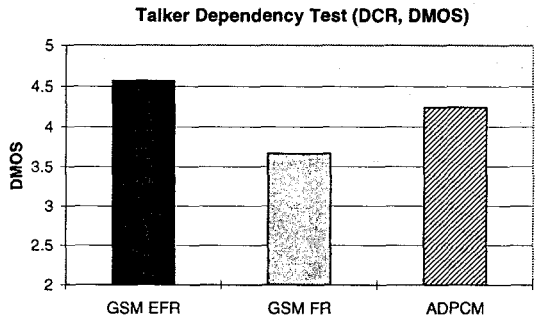


Figure 5: Results from talker dependency test.

EFR codec was found statistically better than ADPCM.

The ETSI pre-selection tests consisted of five experiments: transmission errors (C/I=10 and 7 dB), tandeming (C/I=10 dB), background noise (music 20 dB and vehicle 10 dB), talker dependency and high error conditions (C/I=4 dB). The performance of the EFR codec was found equal to G.728 for error-free transmission, speech in background noise (for both noise types) and talker dependency. No testing was carried out for the low error-rate condition C/I=13 dB. In erroneous transmission at C/I=10 dB and 7 dB, the EFR codec was found clearly better than the FR codec. At C/I=10 dB, the EFR codec performed equal to or better than MNRU 24 dB in half of the tests. For the outside-a-cell error condition of C/I=4 dB, the results show that the EFR codec has approximately the same performance as the FR codec. The EFR codec was tested in error-free self-tandem in one laboratory and was found equivalent to G.728. In self-tandem at C/I=10 dB, the EFR codec performed clearly better than the FR codec and equal performance to the FR codec in single encoding at C/I=10 dB was demonstrated in all laboratories except one.

Verification tests complementing the pre-selection tests were carried out in ETSI for such items as DTMF and network information tones, frequency response, complexity, delay, different languages, music and special input signals (e.g., different input levels, sine waves, noise signals etc.). In all the verification tests, the EFR codec performed well [3]. For DTMF-tones, the EFR codec was found 100% transparent.

7. CONCLUSIONS

The GSM EFR codec has met and even exceeded the essential requirements set for the development of the EFR codec. It provides substantial improvement in speech quality compared to the existing GSM full rate codec and brings high speech quality associated previously only with fixed networks to the end users of mobile communication systems. The GSM EFR specifications have been completed in 1996 and the codec is expected to be deployed in GSM, DCS 1800 and PCS 1900 networks in 1997-1998.

ACKNOWLEDGEMENTS

The authors acknowledge the efforts of M. Fatini and J. Hagqvist in PCS 1900 EFR codec standardisation, the algorithmic contribution of B. Bessette and the efforts of members of ETSI SMG2 SEG, especially those who have carried out pre-selection and verification tests for the EFR codec.

REFERENCES

[1] "Enhanced Full-Rate Speech Codec - Study Phase Report," Report of the ETSI SMG1 sub-group on the EFR codec, Version 0.5.3, December 19, 1994.

[2] R. Salami, C. Laflamme, J-P. Adoul, and D. Massaloux, "A toll quality 8 kb/s speech codec for the personal communications system (PCS)," IEEE Trans. Veh. Technol., vol 43, no. 3, pp. 808-816, Aug. 1994.

[3] "Digital cellular telecommunications system; Performance characterization of the GSM enhanced full rate speech codec (GSM 06.55)," ETSI Technical Report (in preparation), Draft ETR 305, version 1.0.0.

[4] Complexity Evaluation Subgroup of ETSI TCH-HS (Alcatel Radiotelephone, Analog Devices, Italtel-SIT, Nokia), "Complexity Evaluation of the Full-Rate, ANT and Motorola Codecs for the Selection of the GSM Half-Rate Codec," Proceedings of the 1994 DSPx Exposition & Symposium, June 13-15, San Fransisco (USA), 1994.

[5] "COMSAT Test Report on the PCS1900 EFR Codec," Source: Nokia, Temporary Document 47/95 of ETSI SMG2-SEG, June 1995.

Exhibit 8

A GOOD JOB WELL DONE: THE LATEST WIRELESS CODECS DELIVER WIRELINE QUALITY

Leigh A. Thorpe & Paul V. Coverdale

Nortel, Ottawa, Canada

Abstract

The results of a listening test conducted at Nortel on behalf of the CDMA Development Group (CDG) suggest that the new crop of wireless codecs are all able to deliver voice quality comparable to wireline with single encoding over a clean wireless transmission channel.

The codecs were tested with clean speech, varying input level, background noise, and tandem encoding. Given these findings, we hope to provoke discussion among the audience about the next challenges facing speech coding researchers and codec designers. Potential issues: codec robustness to errors, noise reduction, and very-low bit rate codecs for a variety of applications.

Introduction

How good is the general quality of the latest codecs intended for the three major wireless technologies? How does that quality compare to the quality of wireline equipment? Listening tests were conducted to evaluate the performance of the current standard codecs for the three major wireless technologies. The codecs tested are the latest standards for CDMA, North American TDMA, and GSM (PCS1900). The tests examined performance with clean speech and robustness to input impairments and tandem encoding.

Method

Four codecs intended for wireless and two codecs used in wireline networks were evaluated. Clean source speech and speech with input impairments was processed through each codec in a non-real-time software simulation. Ratings were gathered from listeners in an absolute category rating (ACR) procedure[1].

Test and reference codecs

The following codecs listed below in the tests. Mu-law PCM and ADPCM were included to represent the range of wireline quality. ADPCM at 32 kb/s is generally taken to represent the lower boundary of wireline quality.

Test codecs:

TDMA EFRC: 8 kb/s enhanced full-rate codec (EFRC) for TDMA (IS-641).

CMDA EVRC: the new 8 kb/s variable-rate coding standard for CDMA systems. (This codec includes a noise reduction algorithm applied to the input speech signal.)

CDMA Q13: The 13 kb/s variable-rate codec proposed by the CDG for CDMA systems.

GSM ERFC: 13 kb/s enhanced full-rate (EFR) for GSM and PCS 1900.

Reference codecs:

μ-law PCM: G.711 at 64 kb/s. Used in North American digital wireline switching and transmission.

ADPCM: G.726 at 32 kb/s. Used on many international wireline calls and private networks; also in CT2 and DECT wireless standards.

Test cases

The test cases examined the contribution of varying input level, background noise level and degradation from asynchronous tandem operation. (Because of the difficulty of defining comparable channel degradation for the various wireless systems, we did not try to compare the effects of transmission impairments across wireless platforms.) The test cases included the following:

Clean speech: SNR > 45 dB; -20 dBm0 input level

Input levels: SNR > 45 dB; -10 dBm0 and -30 dBm0

Background noise: car interior noise at 10 and 20 dB SNR; street noise at 15 dB SNR; babble at 20 dB SNR; and Hoth noise at 20 dB SNR (white noise filtered to model long-term average room noise).

Asynchronous tandem: clean speech input; nominal level. The source file is encoded, decoded, converted to analog, re-digitized, encoded and decoded a second time.

Additional reference cases processed through the modulated noise reference unit [2] were also included in the session. Samples with dB Q values of 5, 10, 15, 20, 25, 30 and 35 were used.

Preparation of source files and test samples

Source speech from four talkers was prepared from high-quality 16-bit linear PCM recordings from each of four talkers (two men and two women). To maintain equivalent speech levels for each sample, the filtered source files were equalized for speech power as determined by the P.56 algorithm (SV6) [3]. The samples were then filtered with the modified IRS transmit filter [1] and processed through μ-law PCM before processing through the test codecs to simulate the effects of the wireline portion of a connection.

For speech-plus-noise samples, the filtered, equalized speech was mixed with filtered noise scaled to the appropriate level to achieve the intended signal-to-noise ratio.

0-7803-4073-6/97/$10.00©1997IEEE.

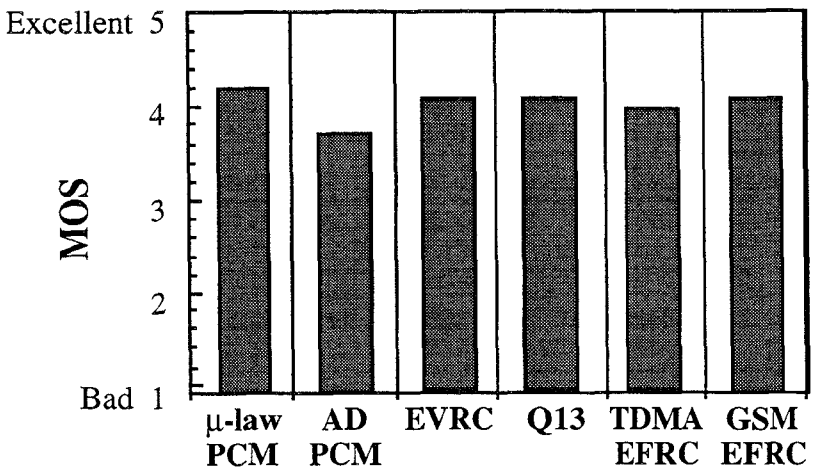


Figure 1.

Mean ratings given to the clean speech case for each of the test and reference codecs. The results for all of the wireless codecs fall between those for μ-law PCM and ADPCM, often taken as defining the range of wireline quality.

Source files were processed through each codec in a non-real-time software simulation. Asynchronous tandem processing was done in digital simulation using up-sampling, applying an all-pass filter, and downsampling again [4].

Listening test procedure

Speech samples for each test case were rated by 60 typical telephone users in a carefully controlled listening test. Listeners rated the overall audio quality of the sample on a 5-point scale. The rating scale was defined as 1–bad, 2–poor, 3–fair, 4–good, and 5–excellent. All listeners rated every test sample once in a classic repeated-measures design. Such a design allows variation due to differences in subject rating criteria to be partialled out in an analysis of variance.

The test samples were played back to listeners over one channel of high-fidelity headphones. The playback signal was filtered to simulate an ideal handset receiver response, and was presented at 79 dB SPL. Listeners were told to wear the headphones with the live speaker at their telephone ear.

Listeners were run in groups of three. Each group received a different randomization of the test samples. Randomization was done using a randomized block design, with each test case presented once in each of four blocks. This controls both for effects of order of presentation and for criterion shifts due to effects of familiarization and fatigue. The randomization was also constrained so that samples from the same talker were never presented consecutively.

Listeners were given written instructions to read at the beginning of the listening session, and completed a short practice session before the test session began. The whole session took about one hour to complete.

Results

Listener ratings for each test case were averaged to obtain the Mean Opinion Score (MOS). Data were collapsed over talkers in computing these means. In an experiment with 60 listeners and 240 judgments per test case, differences of greater than about 0.1 MOS are statistically reliable.

The chart shows the results for the clean speech/clean transmission test case. All of these codecs were found to provide voice quality better than ADPCM. In addition, Q13, EVRC, and GSM EFRC showed voice quality equivalent to or nearly equivalent to μ-law PCM with clean input speech.

Results for cases with background noise showed that all the test codecs performed as well or better than ADPCM with all the noise types tested. Because of its on-board noise reduction algorithm, the EVRC performed better than any other codec in the noise cases. Finally, voice quality remains acceptable even with asynchronous tandem processing.

These data demonstrate clearly that the latest codecs for each of the major wireless standards (Q13 and IS-127 for CDMA, IS-641 for TDMA, and EFRC for GSM) can deliver voice quality equivalent to wireline under good transmission conditions. Given these findings, we hope to provoke discussion among the audience about the next challenges facing speech coding researchers and codec designers. These perhaps include: robustness to transmission impairments, noise reduction, and the development of half-rate codecs with wireline equivalent quality for wireless and other applications.

References

[1] Recommendation P.830 (1996). Subjective performance assessment of telephone-band and wideband digital codecs. Geneva: International Telecommunications Union.

[2] Recommendation P.81 (1993). Modulated Noise Reference Unit (MNRU). Geneva: International Telecommunications Union.

[3] Recommendation G.191 (1993). Software tools for speech and audio coding standardization. Geneva: International Telecommunications Union.

[4] Recommendation P.56 (1993). Objective measures of active speech level. Geneva: International Telecommunications Union.

Exhibit 9

IEEE 802.11
Wireless Access Method and Physical Specifications

Title:	A Compromise MAC Protocol Concept
Date:	May 10, 1993
Author:	Jim Schuessler — National Semiconductor

Abstract

An effective compromise is offered using major elements of both the Reservation Based protocol proposed by IBM (doc 91/74, 92/39 and others) and the Hybrid CSMA/CA Based protocol proposed by Xircom (doc. 92/14, 93/13, 93/40 and others). Three major changes are made to bring both protocols together:

1. CSMA/CA using RTS/CTS messages is used in the 'C' or contention area of the Reservation Based protocol.
2. HDLC-like framing structure is used for all data transmission
3. A mapping of functions to the 802.11 reference model is suggested that pushes low level framing into the Media Convergence Layer. The choices admittedly do not follow convention for the purpose of facilitating progress toward publishing a standard.

Issues Addressed

Due to the broad scope of this submission the number of issues effected in the Issues Log (doc. 92/64) is quite large. Especially relevant are sections 9 (Performance), 10 (Coordination Function), 12 (Interfaces), 14 (Connection Type), 15 (Services), and 17 (Addressing).

Introduction

For quite some time now two of the prominent protocols proposed to our plenary have been viewed as having "irreconcilable differences". The protocols need not be viewed that way. It is in the entire committee's interest to facilitate progress toward one MAC protocol providing interoperability with the first PHY Medium Dependent Layer (FHSS, 2.4GHz ISM).

This document is intended more as a "concept" document as opposed to having all the details and end cases nailed down. Admittedly there are numerous details yet to be resolved, but this document should provide the guidelines with which to make those decisions.

The description of this blend can be approached from a number of different angles. I have chosen to describe the differences to the Hybrid Asynchronous / Time-bounded Protocol first.

Hybrid Protocol Operation in the Absence of Reservation Protocol Cycles

This is the easiest starting point since there are no differences to the Coordination Function (CF); only differences to the frame structure. In this mode, a node would power up and attempt to acquire a potential existing FHSS hop sequence. If found, the Station (STA) must listen for Reservation Protocol cycles. Finding none, the STA has the option of initiating the cycles itself, or attempting communication using the Hybrid CSMA/CA CF as presently described in 93/40 - leaving the CF unchanged.

The major difference to the Hybrid Protocol exists in the frame structure. In all cases, the Hybrid Protocol uses the HDLC-like frame structure described in "Wireless LAN Medium Access Control Protocol: Description of the Air Interface, doc. 93/__". It is repeated here for clarity.

Table 1. HDLC packet frame structure

FIELD	BYTE LENGTH	VALUE	MEANING
F	1	0x7E	Start Frame Delimiter
DA	1	variable	Destination Address
SA	1	variable	Source Address
C	4	variable	Control Field
L	1	variable	Data Field Length
Data	variable	variable	Information Data
FCS	2	variable	Frame Check Sequence
F	1	0x7E	End Frame Delimiter

In addition to the frame structure, the definition of the control fields, the addressing and other elements of this document will be conformed to. Since the Hybrid Protocol uses additional CTS/RTS messages to avoid collisions, the format of these messages will also conform to the above general structure. The specific control field codes for CTS and RTS are TBD..

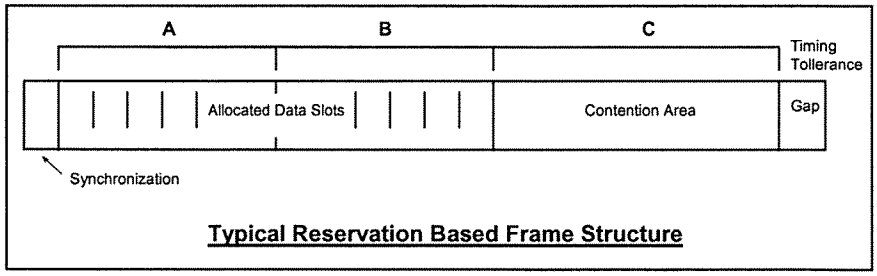
In summary, all transmission will utilize the same general frame structure. If no Reservation cycles are present or desired, the CF may operate according to the CSMA/CA (RTS/CTS) method for asynchronous (non-Time-bounded) data. Time-bounded data transmission is required to utilize the Reservation Based cycles.

Hybrid Protocol Operation in the Presence of Reservation Protocol Cycles

Similarly, a STA would awaken or power up and attempt to acquire an existing hop sequence. Assuming it is found, Reservation cycles are listened for and (in this case) found or initiated. Asynchronous data is now conveyed according to the bandwidth allocation of the CF, or within the contention area of the cycle.

If data is conveyed within the contention area (Area 'C'), it does so using the Hybrid CSMA/CA protocol. This is a major departure from the Slotted ALOHA protocol previously proposed for this area.

Data occupying any slot in the A or B intervals, or the C area, will use the HDLC frame structure described above.



Typical Reservation Based Frame Structure

The boundaries A/B and B/C are dynamic and a system could be designed with all data conveyed in the 'C' interval, however to conform to the proposed standard, the same system (STA) must operate with as little as 20% of the frame reserved for the 'C' interval.

Data conveyed in the 'A' and 'B' intervals and the allocation of bandwidth between all intervals does not change from the current Reservation Based proposal (doc. 92/39, etc.).

Functional Partitioning in the Reference Model

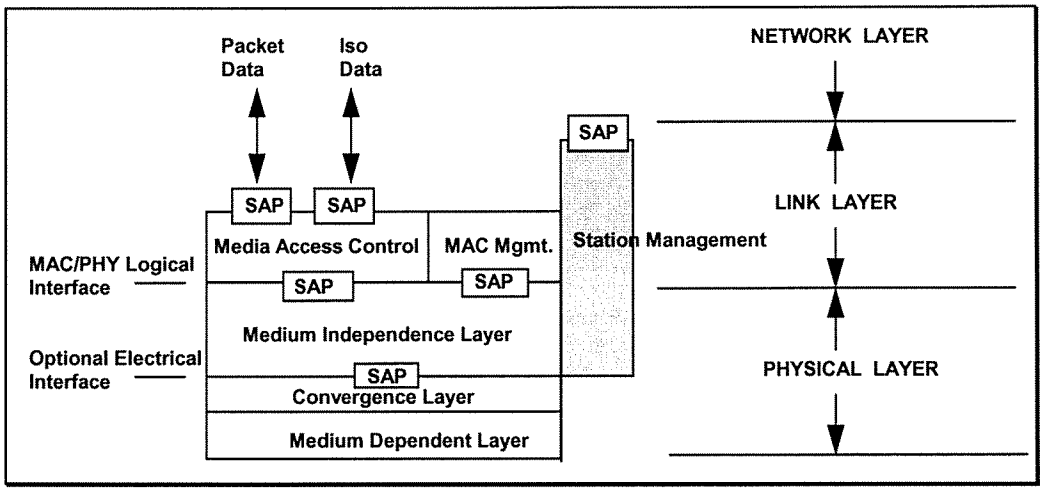


Figure #1. 802.11 Reference Model

The most unconventional aspect of our proposal is to locate the HDLC-like sub frame structure in the PHY Convergence Layer. Although perhaps not architecturally pure by a strict OSI interpretation, it is a practical and expedient approach to couple the FHSS PHY (which is closest to standardization) to certain elements of the blended protocol. We believe that if the proposed framing remains solely in the MAC layer, further delays will be incurred while additional development of the protocol was done for many other possible PHY MDL.

By coupling the framing and low-level timing to a particular MDL, design trade-offs such as performance, simplicity and cost are allowed to be naturally optimized.

Specifically we propose the following split of functions:

Layer	*Function*
Medium Access Control	• Association / Disassociation / Reassociation • Authentication • Security interface • Interface to Distribution System
Medium Independent Layer	• Hybrid Mux: Time-bounded / Asynchronous selection • Bandwidth allocation • Segmentation and Reassembly • Low-level packet retransmission
Convergence Layer	• Assemble bits into low-level frames • Access Method: hybrid TDMA / CSMA • Low-level packet framing: Preamble, address, check field • Hop Timing: Acquisition and tracking
Medium Dependent Layer	• Bit Transmission / Reception • Activity Monitoring (Carrier Sense) • Signal Strength (RSSI) • Clock Recovery • Signal Acquisition and Antenna Selection

These functions are, of course, under the control of the Station Management entity. We agreed that document 92/98 formed the basis of our SMT implementation and further work on defining the function of the blocks was done and voted on at the January, 1993 meeting. A diagram or mapping of how SMT fit into the approved Reference Model has never been done, and seems to point up an error.

Our current model shows SMT descending only to the Medium Independence Layer, when Steve Chen's doc. 92/98 more correctly shows connections all the way down to the Medium Dependent Layer (a.k.a. Physical Medium Dependent or PMD)

Should we alter the Reference Model?

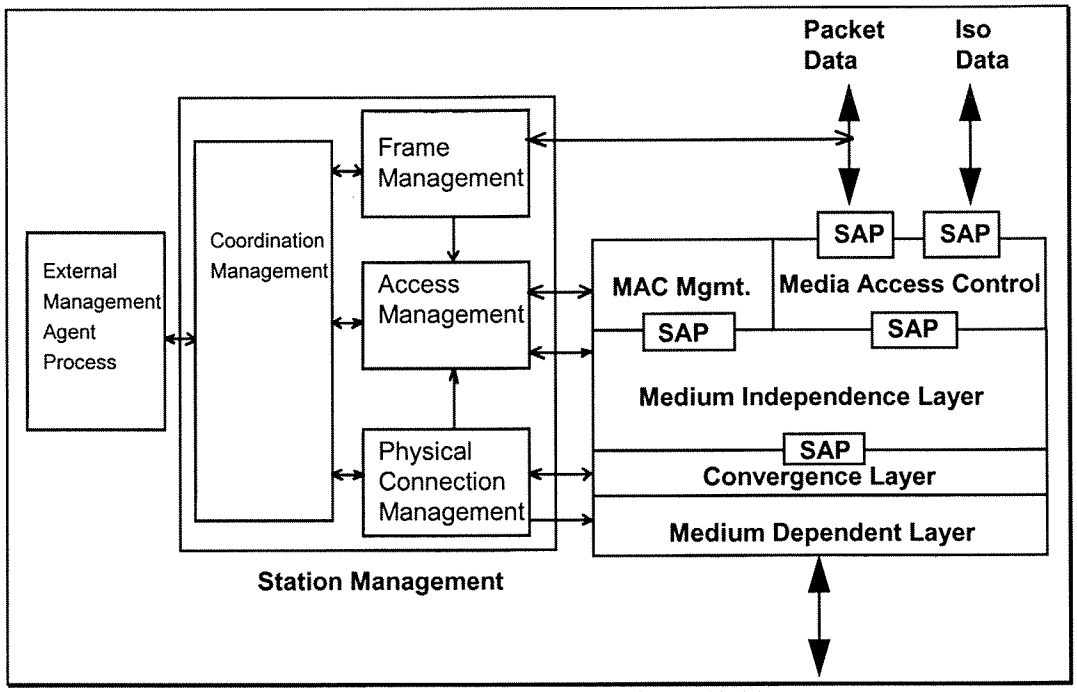


Figure #2. Connection of SMT to Reference Model

Furthermore, a number of committee members desires for more intelligence being located in the PHY layer can be accommodated, making an exposed DTE/DCE interface easier to implement. A Command / Status / Data protocol over this interface is easily imagined working within the proposed functional partitioning. For those of us interested in a "lowest cost" approach; the option of not exposing this interface does not increase complexity (cost).

Summary

A possible compromise position has been presented between two of the proposed MAC protocols. My intention of this paper is to lobby for a convergence of the protocols and further progress toward an approved standard. Three techniques have been described which ease this convergence.

Exhibit 10

IEEE 802.11
Wireless Access Method and Physical Layer Specifications

Title: The CODIAC Protocol

Centralized Or Distributed Integrated Access Control (CODIAC), A Wireless MAC Protocol

Author: Carolyn Heide
Spectrix Corporation
906 University Place
Evanston, Illinois 60201
Voice: (708) 491-4534
FAX: (708) 467-1094
EMail: 71041.3262@compuserve.com

Abstract:

This paper proposes a media access control (MAC) protocol to address the diverse requirements of the IEEE 802.11 standard effort.

The goals of the proposed protocol are: (1) to take advantage of the contention avoidance, power efficiency and time-bounded service support characteristics of a deterministic MAC protocol; (2) to operate with efficiency and fairness in the absence of infrastructure; and (3) to provide maximum flexibility, allowing the protocol to be tailored to varying implementations without loosing compatibility across implementations.

Related Documents:

IEEE 802.11-93/40 The WHAT MAC Protocol
IEEE 802.11-93/59 Evaluation of CODIAC Protocol
IEEE 802.11-93/65 Performance of CODIAC Protocol

1. Introduction

This protocol stems from a merging of two very different protocols, each of which is ideally suited for a particular wireless LAN scenario, but not for all scenarios.

The first protocol of influence is the Spectrix Reservation/Polling Protocol[1], which is currently implemented in high density, fixed population networks - specifically, 1500 mobile stations per Basic Service Area (BSA). This implementation of the protocol is described briefly in Appendix A. It is a deterministic collision avoidance, master/slave protocol. No station may access the medium without permission, implicit or explicit, of the master and so the protocol is collision free despite the large number of stations and the possibility of hidden stations. However, the strict deterministic nature of this protocol makes the issues of adhoc networks and BSA overlap difficult to handle.

The second protocol is the WHAT protocol proposed by XIRCOM in several IEEE 802.11 submissions[2], described in detail in document IEEE P802.11-93/40. This protocol excellently handles wireless issues, such as hidden stations, overlapping service areas, and ad-hoc networks. It is simple and elegant, and has been implemented and simulated with success for small station populations with office LAN traffic patterns. However 802.11 must address scenarios of more varying traffic patterns and much greater station density, and the ability of an Enhanced Listen-Before-Talk protocol to do so without major performance degradation is questionable.

The protocol proposed here endeavors to combine the best features of the these two protocols, resulting in a protocol with the flexibility to support the diverse requirements of the 802.11 standard.

[1]Patent Pending, US Serial No. 07/643,875

[2]IEEE P802.11/91-92 A Hybrid Wireless MAC Protocol; IEEE P802.11/92-14 An Update to the Hybrid Wireless MAC Protocol; IEEE P802.11/92-49 A Review of Some Properties of the Hybrid Protocol; and, IEEE P802.11/93-40 The Wireless Hybrid Asynchronous Time-bounded MAC protocol.

2. The Concept

Take the WHAT protocol without the T - the asynchronous support only. All data transfer is via a four step frame sequence which is issued when the Net Allocation Vector (NAV) shows a gap in the bandwidth allocation:

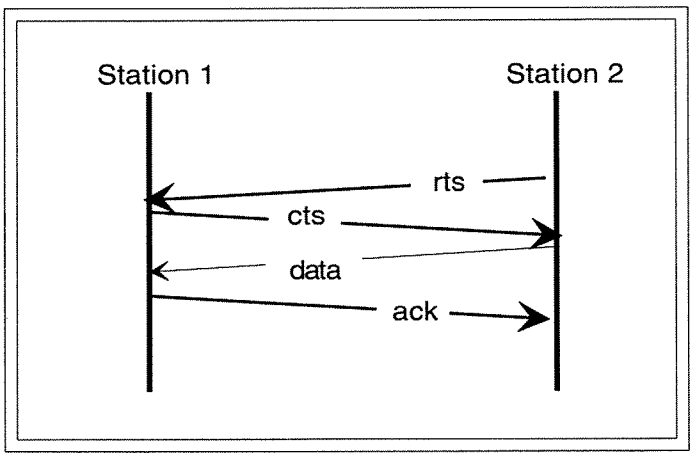


Figure 2.1

Then take a Reservation/Polling Protocol (RPP) where all data transfer is via the four step frame sequence that is initiated by a synchronization frame from a controller:

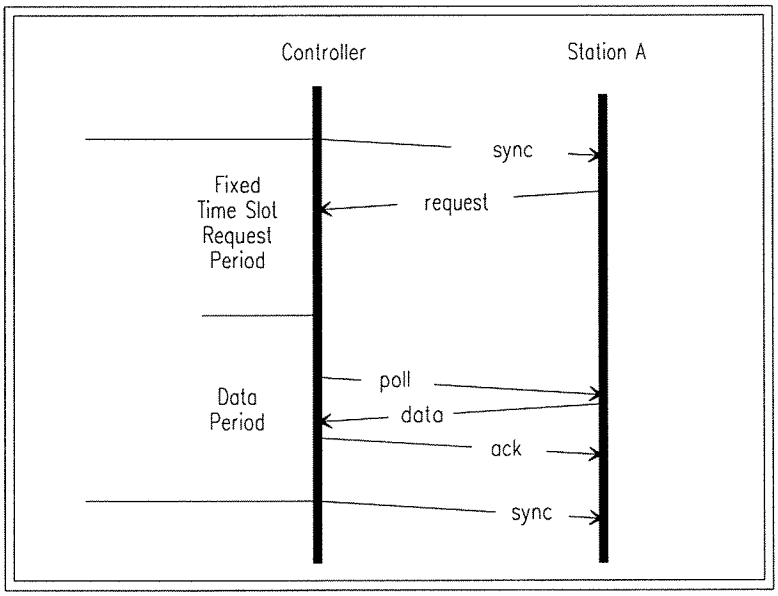


Figure 2.2

Combine the processes:

- Change the REQ frame to an RTS frame - a request for bandwidth. The request specifies the destination of data. If there is a controller present, send the request to it in response to the SYNC frame and let it allocate the bandwidth. If not, broadcast the request according to the NAV;

- Change the POLL to a CTS - the controller is not asking for data from the station, but clearing the bandwidth requested for the station to send it's data;
- The DATA and ACK follow the CTS just as in both the WHAT and the RPP.

To a station the only difference between the two protocols is when the RTS is issued - either according to the NAV, for following a SYNC frame from a controller. Stations could easily have two modes of operation according whether or not they are in the BSA of a controller, and these modes would differ very little.

3. Theory of Operation

The CODIAC protocol has two modes of operation:

(1) Centralized: In the presence of a controller station, which is issuing a periodic, but not necessarily regular, synchronization frame. In centralized mode access to the medium is highly deterministic, and completely controlled by the controller station. This mode of operation supports both 'asynchronous' and 'time-bounded' service;

(2) Distributed: In the absence of a controller station, when a station cannot 'hear' sync frames it operates in distributed mode, enhanced Listen-Before-Talk (LBT). This mode of operation does not support time-bounded service.

It is not required that the controller station be an Access Point (AP), or that an AP must be a controller station. Nor is either mode associated with an adhoc or an infrastructure network - an adhoc network could be created by bringing a portable controller station into a room, or an infrastructure network with no requirement for time-bounded services could function in distributed mode.

The startup procedure for a station, when it is powered up or comes 'awake' is:

(1) Listen for x amount of time for a sync frame (where x is an known maximum time);

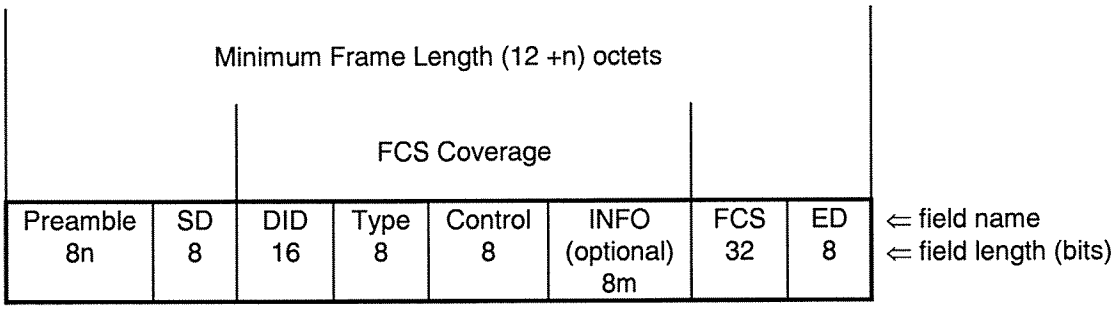
(2) If sync heard, operate in centralized mode;

(3) If no sync heard, operate in distributed mode.

In either mode of operation the transfer of data is accomplished by a four step transaction - the exchange of RTS, CTS, DATA and ACK frames. The originating station sends a Request To Send (RTS) frame. In response to the RTS, a Clear To Send (CTS) frame is sent. On receipt of the CTS frame the originating station sends the DATA frame, and the destination responds with an ACK frame. The timing of the transmission of these frames differs between centralized and distributed mode, but not their format or meaning.

4. Frame Format

All frames have the following format:



Preamble = Preamble (n to be determined)
SD = Start Delimiter
DID = Destination Identifier
Type = Frame Type
Control = Control Flags: AP, sequence, out-of-sequence, retry, hierarchical
INFO = Information (0 <= m <= to be determined)
FCS = Frame Check Sequence, CRC-32
ED = End Delimiter

4.1. Destination Identifier

The purpose of the destination identifier (DID) field in the frame is to allow the MAC to determine as quickly as possible whether this frame needs to be received by this station. That does not mean that it contains the same value for a specific station at all times.

If a station is registered with a controller, the controller has assigned an ID to that station. The controller will use this ID as DID in all frames it sends to that station, and it will ensure that all stations registered have exclusive IDs.

When stations are not registered with a controller (i.e. operating in distributed mode), they choose any value for their ID, and may vary it with each transaction (this is equivalent to the WHAT protocol's MSDUID, which is incremented with each MSDU sent).

Destination identifier values:

(1) FFFFh = broadcast;
(2) 8000 - FFFEh = controller stations;
(3) 0 - 7FFFh = non-controller station.

4.2. Control Flags

AP - indicates frame originated from an Access Point.
Sequence - alternation bit, one bit sequence number (see section 8).
Out-of-sequence - indicates invalidity of sequence bit (see section 8).
Retry - indicate re-transmission (see section 8).
Hierarchical - specifies frame destination must be Access Point only.

5. Distributed Mode Operation

Distributed mode of operation is the asynchronous operation of the WHAT protocol. The CODIAC protocol uses slightly different frame formats, but the operational rules are the same.

When operating in distributed mode all stations must have their receivers on at all times if: (1) there is any possibility of them receiving data; or (2) they wish to send anything.

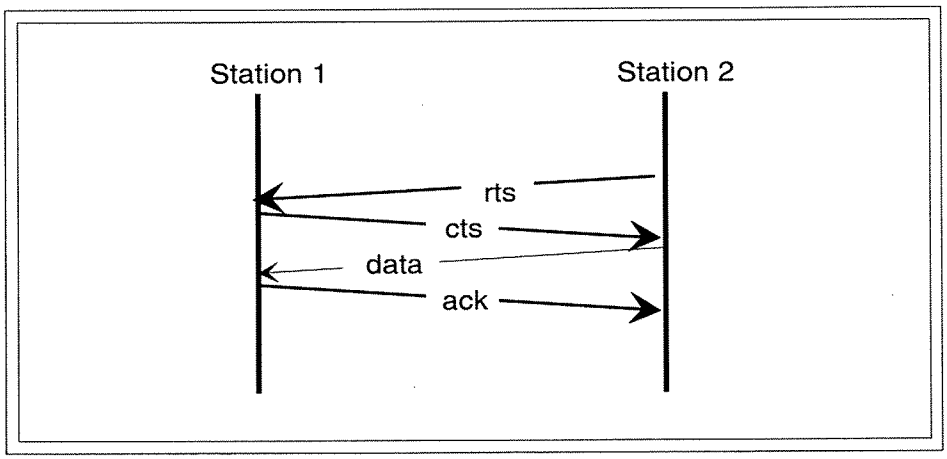


Figure 5.1 - Distributed Mode: Data from Station 2 to Station 1

6. Centralized Mode Operation

6.1. Introduction

The characteristics of the centralized mode of operation allow a lot of flexibility for individual implementations. The following sections cover the most basic possible method of operation. In later sections some possible enhancements (i.e. complexities!), and the reasoning behind them, are introduced.

6.2. The Superframe

Stations operate in centralized mode when they know they are in the presence of a controller station because they receive synchronization frames from it.

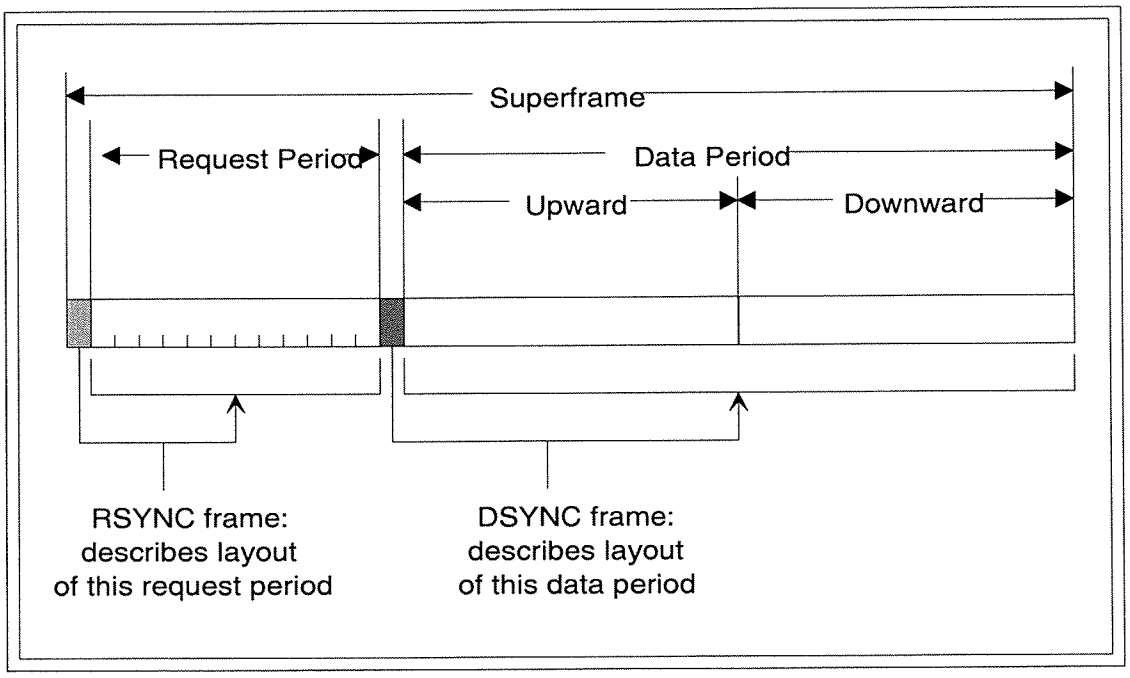


Figure 7.1 - Superframe Layout

Centralized mode operation takes place in quantums of time called superframes. Each superframe has two periods - the request period and the data period. The length of each of these periods is dynamic and controlled by the controller station. Each period is initiated by transmission of a synchronization frame by the controller - an RSYNC frame marks the start of the request period, and a DSYNC the start of the data period.

The request period is divided into fixed length time slots which are used by stations to register with the controller, and to request bandwidth to transmit after they have become registered. During this period stations will transmit only, nothing will ever be sent to them, so stations may turn off their receivers for this period. The purpose the RSYNC frame at the start of the request period is to facilitate synchronization of the time slots, and to specify the length of time for which stations may turn off their receivers.

The data period is itself divided into two parts: upward (data sent from stations to the controller) and downward (data sent to stations, either from the controller, or from other stations). The purpose of dividing

the data period is to allow stations which have no data to send to the controller - those which have not issued requests in the request period - to turn off their receivers during the upward data period. The purpose of the DSYNC frame sent by the controller at the start of the data period is to specify the length of the upward data period.

During the upward data period stations which requested to send data to the controller will be allocated bandwidth by the controller to do so. Stations which did not request to send to the controller during the request period may keep their receivers off, they will not be sent anything during this period.

During the downward data period data is sent to stations - either from the controller, or from other stations which requested direct station-to-station transmission during the request period. During this period all stations must keep their receivers on (unless they have some way of guaranteeing that no one is going to send them data).

6.3. Request Period

The request period is divided into fixed time slots. The length (in bit times) of each slot is fixed, while the number of slots for each sync period is specified by the controller and placed into the RSYNC frame. There are two types of time slots, registration slots and owned slots. Once a station has registered with the controller, the controller assigns it a time slot, which it then owns.

A station looses ownership of a time slot when:

(1) The station cancels its registration with the controller;

(2) The lifetime of the ownership, which is a time length known to both the controller and station, expires without the slot having been used; or

(3) The controller cancels the registration of the station.

To register with a controller, a station generates a random number and uses this to determine in which registration slot to send its registration request. Should it collide with another registering station, it's request will not be answered by the controller, and it will repeat the process next superframe.

While the registration request is a special frame type, all other registration related information is passed in MAC management data frames (MDATA). This is so that registration acceptance, rejection and cancellation information can take advantage of the acknowledgment that goes along with data transactions. Information about time slot ownership must be protected against loss so no confusion over ownership can arise.

After a station has issued a registration request in a registration time slot, the controller responds with a Registration Accept contained in an MDATA frame. The MDATA frame is sent to the ID specified by the requesting station in the registration request. The Accept specifies the number of the time slot which is now owned by the station. From this point onwards, the ID of the registered station is it's time slot number.

The total number of registration slots available in each superframe is determined by the controller and specified in the RSYNC frame.

The owned time slots are the first time slots following the RSYNC frame, and the registration slots follow these. For example, if the RSYNC specifies 1024 total slots with 100 registration slots, slots 1 through 924 must be owned, and slots 925 through 1024 are available for registration requests. If a new station registered the distribution of slots in the next request period is at the discretion of the controller - there can to 1025

total slots with 100 registration slots, or 1024 total slots with 99 registration slots, or any other combination as long as there are enough slots for the stations which own them.

When an owning station gives up, or looses, its time slot, this slot is now available to the controller to assign to the next station which registers. Thus the 'holes' in the owned time slots will be filled up, and no need is foreseen to have to re-pack the slots. Should some extreme condition cause a very poor distribution of owned time slots, the controller can de-register stations, and when they register again assign them to the empty time slots.

It should be noted that registration with a controller is not equivalent to association with an Access Point. The purpose of registration is to facilitate coordinated use of the medium by all stations within range a controller. For instance, two stations may be registered with a controller/AP so that they can converse with each other without disrupting communication in the BSA, but they may not be associated with the AP and the infrastructure at all. They are forming a small adhoc network in the presence of the AP's infrastructure network, but both networks are using a common point coordination function under control of the controller.

Special Cases:

At the discretion of the implementer, the owned slots and the registration slots could be the same. If the RSYNC frame specifies this, then requesting stations are sharing their request slots with registering stations. Any time a station has data to send it has the possibility of having its request collide with a registering station - this may be a risk some implementations find acceptable. Since the registering station chooses its slot based on a random number, collisions it causes should not seriously delay transfer of data from any particular station, and the odds of registering in a slot that is not about to be used by its owner can be very high, depending on the application.

6.4. Registration Transaction

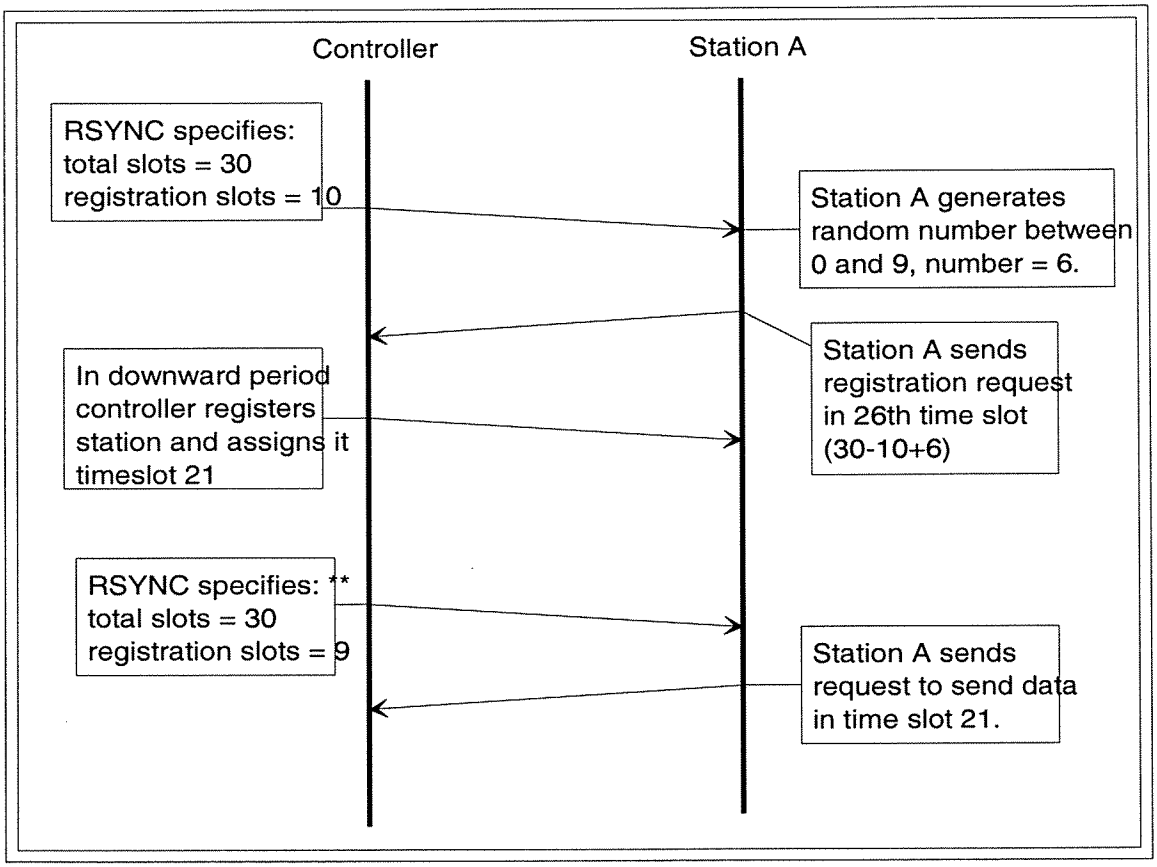


Figure 7.2 - Station A registration. ** the next RSYNC after station A registers could contain total slots 31, registration slots 10, if the controller wants to keep the number of registration slots fixed.

6.5. Upward Data Period

At the end of the request period the controller has a list of all stations which want to transmit, and the destination and length of each transmission. The controller then initiates the data period by broadcasting a DSYNC frame. The DSYNC frame begins the upward data period, the length of which is specified in the DSYNC frame. The controller allocates the bandwidth within the upward data period by sending a CTS frame to, and receiving a DATA frame from, each station from which it wishes to receive data.

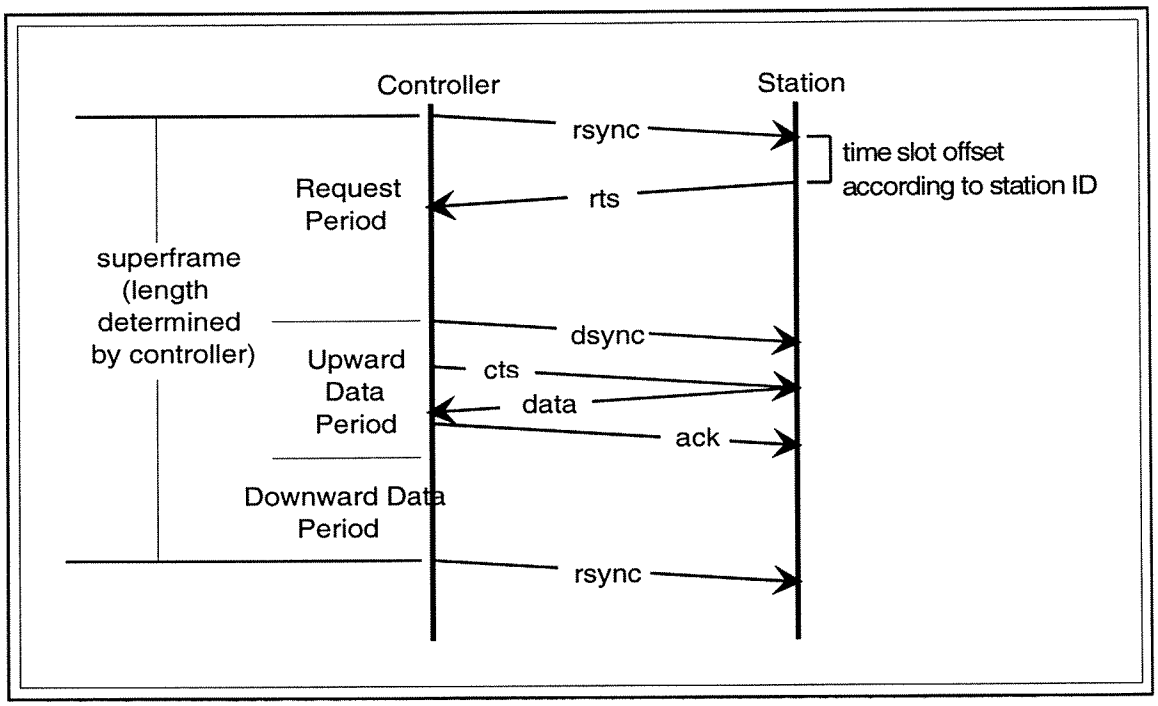


Figure 7.3 - Centralized Mode: Data from Registered Station to Controller

The station's original RTS, although sent to the controller, may not have specified the controller as the destination of the data to be sent. This destination was specified in the RTS by the unique 48-bit address of the destination station. If the controller knows that the destination station is a registered wireless station, the CTS frame will not be sent to the requesting station in the upward data period. It will be sent in the downward data period because that is the only time when the destination station is guaranteed to have its receiver enabled.

There could exist a case when a station's RTS specified a destination which is a registered wireless station, but although both stations are in range of the controller, they are not in range of each other. For this reason, the sending station can specify that the controller should send the data indirectly, by using an RTSI frame rather than an RTS frame. When the controller sees an RTSI frame, it specifies in the CTS that the data should be sent to the controller even if the data destination station is registered. Once the controller receives the data it queues it to be sent as downward data to the destination station, much as if it had originated outside the wireless network.

6.6. Downward Data Period

During the downward data period data is sent to non-controller stations, from the controller and from other stations. This activity is restricted to this period to aid in power consumption, because it requires that all stations keep their receivers on.

In this period the controller sends to stations data which it has for them from other stations in the BSA which requested indirect station-to-station data transfer. If the controller is an AP, it also sends to stations data it has received for them from the distribution system.

There are two possible methods by which the controller can send data to a station during this period, and implementations of the protocol may choose to use either or both:

1. The controller may just send a DATA frame to the station. In this case the controller risks wasting bandwidth if the station is not present. It also risks having stations which overlap with the BSA and are functioning in distributed mode cause loss of the data due to collision;
2. The controller can send an RTS first, get a CTS back from the station and then send the DATA frame. This will save bandwidth in the two cases described above, but cost bandwidth in most cases.

An effective implementation may be to send the DATA frame alone the first time, and if a retry is required then to use the full handshake method. Or to revert to the full handshake based on an overall failure rate of some kind.

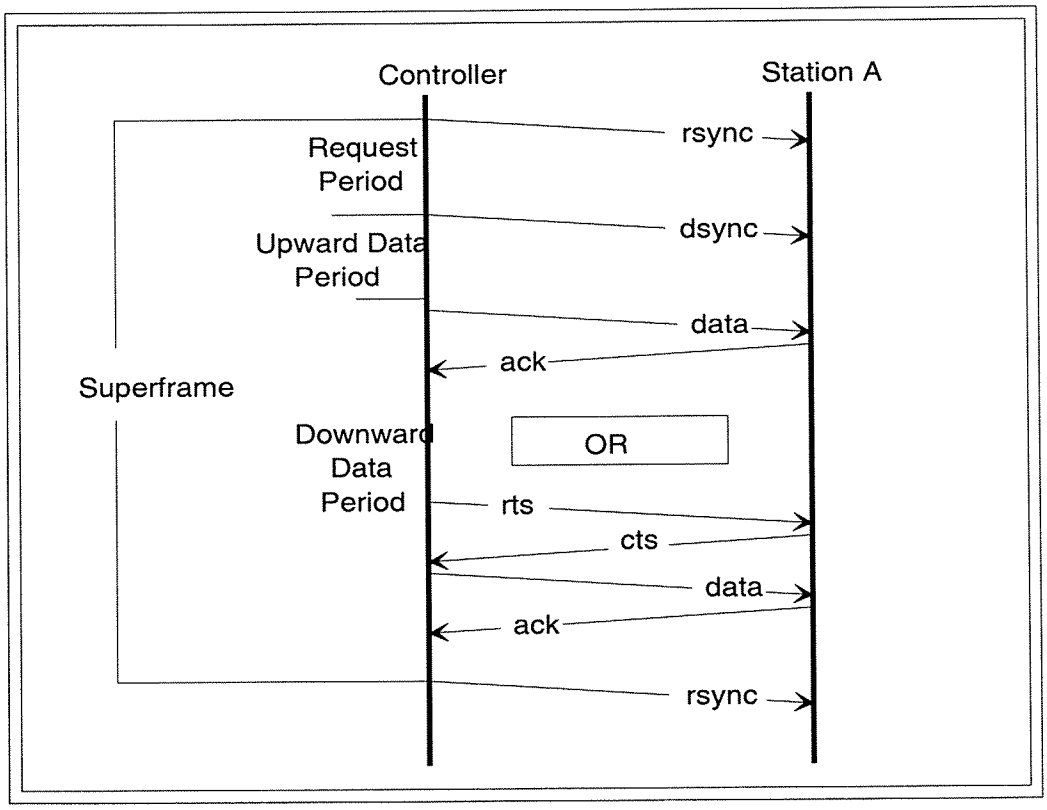


Figure 7.4 - Centralized Mode: Data from Controller to Station A

Any station that issued a request, during the previous request period, to send data to another station is allocated bandwidth by the controller to do so in this period. The controller sends a CTS to the requesting station indicating the ID of the destination station specified in the request. The requesting station may then send data to the destination station.

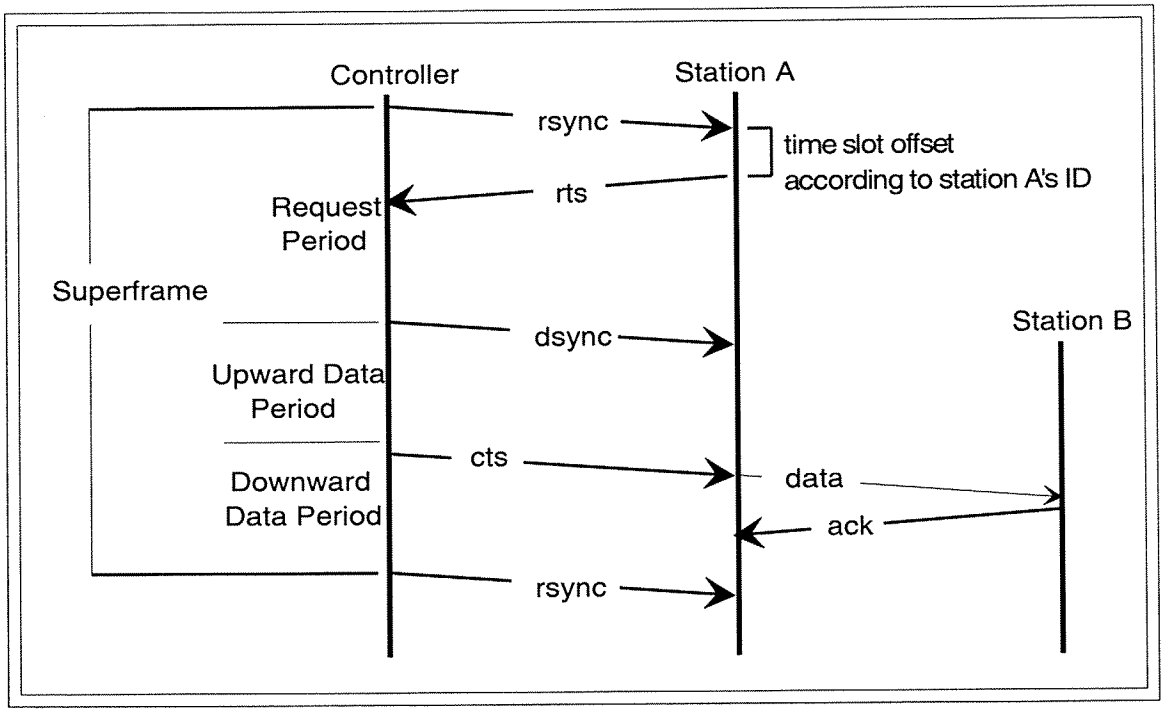


Figure 7. 5 - Centralized Mode: Data from Station A to Station B

7. Default Mode and Changing Modes

Distributed mode is the default operating mode of all non-controller stations. In distributed mode stations always listen prior to accessing the medium, for at least long enough to get an accurate Net Allocation Vector (NAV) built, so they will receive any synchronization frames which a controller within their range is issuing. When a station determines it is in a controller's BSA, it switches to centralized mode operation, the first step of which is to register with the controller.

There are two types of controllers, which have different default operating modes:

1. Dedicated controllers - stations with operating mode always centralized;
2. Potential controllers - stations with default operating mode distributed, which are capable of becoming centralized mode controllers.

The motivation for a potential controller to switch from distributed mode to becoming a controller is an implementation decision. One anticipated use of a potential controller is an AP which functions in distributed mode until it receives a Request To Send Time-bounded (RTST) frame. At that point this AP becomes a controller because centralized mode is required to support time-bounded service. Another might be a smart distributed mode AP which detects too high a rate of collisions due to a growing population in it's BSA, and switches to centralized mode to attempt to alleviate the situation.