Energy-Efficient MAC-Layer Error Recovery for Mobile Multimedia Applications in 3GPP2 BCMCS

Kyungtae Kang, Yongwoo Cho, and Heonshik Shin

***Abstract*—Combining broadcast and mobile phone technologies, 3GPP2 has introduced the Broadcast and Multicast Services (BCMCS) architecture to deliver multimedia content over cdma2000 1xEV-DO wireless networks. In designing a mobile device to support multimedia broadcast services, it is important to reduce delay and energy consumption, while maintaining a tolerable level of data loss. We analyse the energy consumed by a mobile device receiving broadcast services, focusing on error recovery using the Reed-Solomon (RS) MAC-layer coding scheme. Our model is based on the energy consumed by each computational component of the RS decoder, which we determined by running the decoding process on a realistic ARM7TDMI testbed with experimentally justified cache sizes, and characterizing the energy consumption accurately with the SNU Energy Explorer. By varying the radio channel conditions, RS coding scheme and other system parameters, we determined energy-efficient cache configurations and operating ranges for each RS encoding scheme, corresponding to given levels of service quality in a video application. We have also found that significant energy can be saved by selecting a size of error control block that is appropriate to the target video quality and the channel conditions at the mobile.**

***Index Terms*—Broadcasting, code division multiaccess, energy conservation, multimedia communication, Reed-Solomon codes.**

I. Introduction

WORK has begun, in both the Third Generation Partnership Project (3GPP) and the 3GPP2, on enhancing 3G cellular networks to support multimedia broadcast and multicast services, called MBMS [1], [2] and BCMCS [3], [4] respectively. The 3GPP2 has recently produced the specification for the cdma2000 high-rate broadcast packet-data air interface [5], [6]. In designing a mobile phone device to support multimedia in this broadcast service environment, energy conservation is more critical than the performance of a particular application program. High data-rate packet data services supported by third-generation systems are expected to consume more energy than conventional circuit-switched voice services, while battery life remains constrained by strict limits on the size and weight of mobiles.

A battery charge can be made to last longer by reducing the energy consumption of hardware, which is in part caused by the execution of programs on components including the CPU, memory and I/O devices. But the implementation of energy-efficient software requires the modeling, analysis, measurement, and development of energy-saving techniques. There has not yet been any energy analysis at the software level for devices operating in a 3G cellular broadcast environment.

The objective of this paper is to characterize the energy consumed by a mobile device that is receiving broadcast services, especially during error recovery. Since a wireless radio network is prone to errors (i.e. has a high bit error-rate) in burst mode, error control is an essential component of broadcast services. Forward error correction (FEC) has been widely adopted for video broadcast applications because of the strict delay requirements and semi-reliable nature of video streams. In cdma2000 1xEV-DO BCMCS, Reed-Solomon (RS) coding is used for FEC in the MAC layer [5], [6]. That makes the RS decoding process a major target in reducing energy consumption, because all multimedia data transmitted to a mobile device needs to undergo RS decoding.

We will go on to analyse the execution time and energy consumption of RS decoding with different sizes of cache in the mobile device, under varying radio channel conditions and with varying RS coding parameters. From this analysis, we will characterize the energy consumed by the RS decoder and propose an appropriate energy model. We will assume that, for real-time applications such as voice and video streaming, a certain level of packet loss is tolerable in return for reduced energy consumption and latency. We have found that the energy efficiency of these applications can be improved by selecting appropriate caching, RS coding schemes and error control block (ECB) dimensions, based on the energy implications of varying I-cache and D-cache sizes and different RS codes.

We will determine the most suitable cache settings for energy-efficient RS coding, and investigate the tradeoffs between performance and the energy used in error recovery. This will allow us to determine the most energy-efficient operating range for each RS code and suitable ECB sizes for different quality of service (QoS) requirements. Our experiments have been conducted on an ARM testbed using a measurement tool called SEE [7] which has proved to be reliable and accurate in previous work.

The remainder of this paper is organized as follows: In Section II, we present the background to our study and, in Section III, we introduce the energy model of the RS decoding process which we will use in Section IV to derive and analyse the total energy consumption of the RS decoder during error recovery under a range of simulated radio channel conditions. The verification of the energy model by experiment is also described in Section IV. We determine the most energy-efficient RS code and ECB size for a target bit error-rate under realistic radio channel conditions in Section V, and in Section VI we draw conclusions.

Manuscript received July 1, 2006; revised December 5, 2006.

The authors are with the School of Electrical Engineering and Computer Science, Seoul National University, Seoul 151-744, Korea (e-mail: ktkang@cslab.snu.ac.kr; xtg05@cslab.snu.ac.kr; shinhs@cslab.snu.ac.kr).

Digital Object Identifier 10.1109/TBC.2007.891699

II. Error Recovery in cdma2000 Broadcast Networks

A. Error Recovery to Maintain Multimedia Service Quality

The cdma2000 1xEV-DO wireless standard supports the delivery of broadcast and multicast services (BCMCS). By transmitting multimedia content from a single source to many users simultaneously, BCMCS complement unicast services, which provide video content to subscribers individually (i.e., video-on-demand).

BCMCS support the delivery of high data-rate multimedia content over wireless networks. In unicast services, a subscriber's forward-link data-rate depends on the local RF conditions [8], [9], but BCMCS enable service providers to use a common data-rate to send video to all subscribers in the cell-coverage area. Under these conditions, delivery of a consistent high-quality video stream relies on the Reed-Solomon error correction scheme.

In practice, wireless video communications face several obstacles, such as high error-rates, energy restrictions, bandwidth variations and limitations, and the finite processing capabilities of handheld devices; but the unreliable and error-prone nature of the wireless channel is one of the most serious challenges. Wireless channels are afflicted by time-varying fading and interference conditions, which may lead to bursts of corrupted packets. In BCMCS, RS coding is applied to the layers above the underlying turbo code, and is particularly effective in correcting long error bursts.

Real-time traffic, such as voice and video streaming, is very sensitive to delay but can stand a certain level of data loss, while it is important for mobile devices to be frugal users of system resource and energy. Thus a mobile broadcast system should be designed to be energy-efficient while satisfying the QoS requirement by maintaining an allowable bit error-rate (BER) at the input to the video decoder. As a video application usually has a degree of error-resilience, allowing it to mitigate or compensate for errors, a certain error-rate can be tolerated. We will denote the highest acceptable bit error-rate as the target BER (T_{BER}). An overview of video error-resilient techniques is presented by Wang and Zhu [10] and the support for error-resilience provided by the H.264 and MPEG-4 standards is reviewed by Chung-How and Bull [11], while Praveenkumar *et al.* [12] analyse the error-resilience techniques that are applicable within the context of the MBMS standard.

We recognize that the bit-level QoS does not directly reflect the user's view of service quality, which is better represented by application-layer QoS parameters such as the peak signal to noise ratio (PSNR) [13]. However, since the PSNR can be easily derived from the bit error-rate in video applications (MPEG-2, MPEG-4, etc.) [14], [15], the bit-level QoS is a useful measure of quality.

B. Reed-Solomon Coding in BCMCS

A detailed description of the Reed-Solomon code used in BCMCS, and the structure of the error control block (ECB) in the MAC layer, is given elsewhere [5], [6]. The base station (BS) creates an ECB for each logical channel. During this process, data is filled into the ECB in rows. The BS then applies RS coding along the columns of the ECB, and the data is transferred

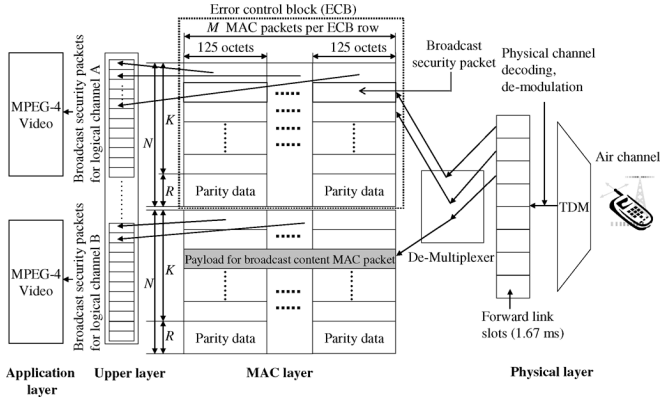


Fig. 1. Error recovery structure at the receiver in BCMCS.

row by row to the physical slot, where it forms one or more physical-layer packets. The RS code used in this process is specified by the tuple (N, K, R), where N is the number of octets, K is the number of data octets, and R is the number of parity octets in each RS codeword. An RS decoder can correct up to R corrupted octets if their positions are known (erasure code), or detect and correct up to $R/2$ octets if the positions of the errors are unknown (error correcting code). In cdma2000 1xEV-DO BCMCS, the RS code is used as an erasure code and an RS codeword consists of a sequence of 8-bit octets, containing a configurable number of parity octets.

The possible values of N are 16 or 32. As the value of N increases, the time diversity also increases, but at the cost of additional memory requirements and decoding complexity at the mobile station, leading to an increase in energy consumption. The possible values of K are 28, 26 or 24 for $N = 32$ (and 14, 13 or 12 for $N = 16$), which correspond to RS code-rates of 7/8, 13/16 and 3/4 respectively. Using a smaller value of K provides an improved error correction capability at the expense of effective data-rate.

The ECB structure, shown in Fig. 1, is designed to allow efficient recovery from bursts of errors, because of the way that such bursts are interleaved spatially within the ECB. Let M be the number of MAC packets in each row of the ECB. As M increases, the time-diversity also increases and thus a mobile which is in a time-varying shadow environment is still able to recover a substantial amount of corrupted data. The organization of BCMCS is such that the value of M for a given ECB has to be less than or equal to 16.

C. The Reed-Solomon Decoding Process

Reed-Solomon (RS) is an algebraic error-correcting code, belonging to the large class of BCH (Bose-Chaudry-Hocquehen) multiple-burst-correcting cyclic codes, which operate on bytes of fixed length.

An understanding of the encoding part of the RS algorithm is not necessary here because we are going to focus on the evaluation of receiver performance; but decoding will be described briefly. An RS decoder generates four syndrome bytes, which will all be zero if the message has no errors. These syndromes can be computed relatively simply, as follows [16]: Let $\nu(x) = \nu_0 + \nu_1 x + \nu_2 x^2 + \ldots \nu_{n-1} x^{n-1}$ be the transmitted

code vector and let $\gamma(x) = \gamma_0 + \gamma_1 x + \gamma_2 x^2 \ldots + \gamma_{n-1} x^{n-1}$ be the corresponding received vector. Then $e(x) = \gamma(x) - \nu(x) = e_0 + e_1 x + e_2 x^2 \ldots + e_{n-1} x^{n-1}$ is the error pattern added by the channel, where $e_i = \gamma_i - \nu_i$ is a symbol from $GF2^m$, which is a Galois Field of order 2^m. If we assume that there are ν errors in positions $i_1, i_2, \ldots, i_\nu$, then $e(x) = e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \ldots + e_{i_\nu} x^{i^\nu}$. If R is the error-correcting capability of the code, then $2R$ syndromes can be computed, as follows:

$$\begin{aligned} S_j &= \gamma(\alpha^j) \\ &= e_{i_1}(\alpha^j)^{i_1} + e_{i_2}(\alpha^j)^{i_2} + \cdots + e_{i_\nu}(\alpha^j)^{i_\nu} \\ &= e_{i_1}(\alpha^{i_1})^j + e_{i_2}(\alpha^{i_2})^j + \cdots + e_{i_\nu}(\alpha^{i_\nu})^j \\ &= e_{i_1} X_1^j + e_{i_2} X_2^j + \ldots + e_{i_\nu} X_\nu^j, \end{aligned} \tag{1}$$

where X_ι is defined as $\alpha^{i_\iota} (\iota = 1, 2 \ldots \nu)$.

These syndrome equations can be obtained as a sequence of $2R$ algebraic equations, which can be translated into a series of linear equations by defining the error locator polynomial $\Lambda(x)$ as follows:

$$\begin{aligned} \Lambda(x) &= \prod_{\iota=1}^{\nu} (1 - X_\iota x) \\ &= \Lambda_\nu x^\nu + \Lambda_{\nu-1} x^{\nu-1} + \ldots + \Lambda_1 x + 1. \end{aligned} \tag{2}$$

If it is assumed that R errors ($\nu = R$) have occurred, then we obtain the following matrix equation:

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_R \\ S_2 & S_3 & \cdots & S_{R+1} \\ S_3 & S_4 & \cdots & S_{R+2} \\ \vdots & \vdots & \ddots & \vdots \\ S_R & S_{R+1} & \cdots & S_{2R-1} \end{bmatrix} \begin{bmatrix} \Lambda_R \\ \Lambda_{R-1} \\ \Lambda_{R-2} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{R+1} \\ -S_{R+2} \\ -S_{R+3} \\ \vdots \\ -S_{2R} \end{bmatrix}. \tag{3}$$

Once the syndromes have been computed, there are a number of ways to find the error locations. In our RS decoder, we use the Berlekamp iterative algorithm in the following form [16]:

1) For each codeword received, compute the syndrome sequence $S_1, \ldots, S_{2R}$.
2) Initialize the algorithm variables: $\kappa = 0$, $\Lambda^{(0)}(x) = 1$, $L = 0$ and $T(x) = x$.
3) Set $k = k + 1$ and then compute the discrepancy $\Delta^{(k)}$ as follows:

$$\Delta^{(k)} = S_k - \sum_{i=1}^{L} \Lambda_i^{(k-1)} S_{k-1}. \tag{4}$$

4) If $\Delta^{(k)} = 0$, then go to Step 8.
5) Modify the connection polynomial:

$$\Lambda^{(k)}(x) = \Lambda^{(k-1)}(x) - \Delta^{(k)} T(x). \tag{5}$$

6) If $2L \geq k$, then go to Step 8.
7) Set $L = k - L$ and $T(x) = \Lambda^{(k-1)}(x)/\Delta^{(k)}$.
8) Set $T(x) = x \cdot T(x)$.
9) If $k < 2R$, then go to Step 3.
10) Determine the roots of $\Lambda(x) = \Lambda^{(2R)}(x)$. If the roots are distinct and lie in the right field, then determine the error magnitudes, correct the corresponding locations in the codeword, and stop.
11) Otherwise, declare a decoding failure and stop.

Once errors have been detected, they must be deleted and the original data recovered. For erasure decoding, assuming the received codeword has ν errors and f erasures, an erasure locator polynomial is defined as follows:

$$\Gamma(x) = \prod_{\iota=1}^{f} (1 - Y_\iota x) \quad \text{where } Y_\phi = \alpha^{j_\phi}, \quad (\phi = 1, 2 \ldots, f). \tag{6}$$

In summary, decoding is achieved by the following steps:

1) Compute an erasure polynomial $\Gamma(x)$ using the erasure information provided by the receiver.
2) Replace the erased coordinates with zeros and compute the syndrome.
3) Compute the modified syndrome polynomial:

$$\Phi(x) = (\Gamma(x)[1 + S(x)] - 1) \bmod x^{2R+1}. \tag{7}$$

4) Apply the Berlekamp algorithm to find the connection polynomial $\Lambda(x)$, using the modified syndrome coefficients Φ_i, $i = f + 1, \ldots, 2R$.
5) Find the roots of $\Lambda(x)$, and thus the error locations.
6) Determine the magnitudes of the errors and erasures using the error and erasure locator polynomial, which is

$$\Psi(x) = \Lambda(x)\Gamma(x). \tag{8}$$

Then the error and erasure values are

$$e_{i_k} = \frac{-X_k \Omega\left(X_k^{-1}\right)}{\Psi'\left(X_k^{-1}\right)}, \quad f_{i_k} = \frac{-Y_k \Omega\left(Y_k^{-1}\right)}{\Psi'\left(Y_k^{-1}\right)}, \tag{9}$$

where

$$\begin{aligned} \Lambda(x)[1 + \Phi(x)] &= \Omega(x) \bmod x^{2R+1}, \\ X_\phi &= \alpha^{i_\phi}, \quad (\phi = 1, 2 \ldots, \nu). \end{aligned}$$

In summary, an RS decoder operates as shown in Fig. 2. When a data stream is received, syndrome bytes are created. Their number is proportional to the number of parity octets, regardless of the quality of the data stream. So the time required to generate a syndrome depends only on the number of parity octets, and will be stable for a data stream with a static bit-rate. However, if there is an error in the data stream, additional work is required for error detection, location and correction. The locations of errors in a codeword are found using the Berlekamp algorithm [16], and the original data can then be recovered by erasure decoding. The number of times that these procedures need to be executed is proportional to the number of errors (ν), and so the time and energy required for RS decoding is proportional to the octet error-rate input to the decoder.

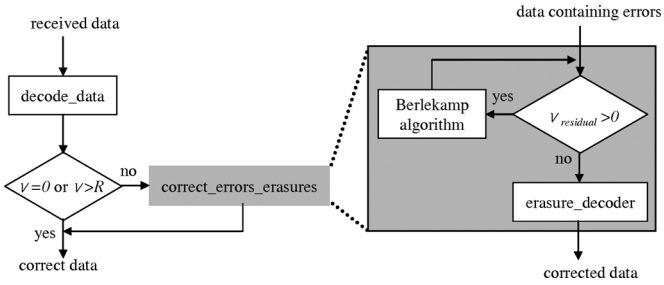


Fig. 2. Flow diagram of the RS decoding process.

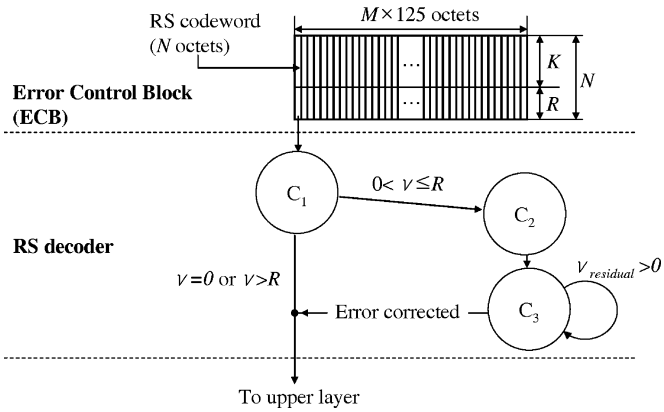


Fig. 3. Computational components of the RS decoding process.

III. Energy Model of the RS Decoding Process

A Reed-Solomon decoder can be logically separated into three computational components, as shown in Fig. 3. C_1 is the computational component that decodes each codeword and decides whether to try erasing the errors or not. C_2 is the computational component that builds the corresponding syndromes, and C_3 performs the erasure loops. Thus C_1 operates on every codeword received, while C_2 is called once for each codeword which contains any errors. Finally, C_3 is repeatedly invoked to correct the errors until no more remain in that codeword (i.e. $\nu_{residual} = 0$).

From a consideration of this whole process, the total energy consumption $(E_{codeword})$ during the decoding of an RS codeword can be predicted by summing the expected energy required by each component to deal with the number of errors (ν) contained in that codeword:

$$E_{codeword}(\nu) = \begin{cases} E_{C_1} & (\text{if } \nu = 0 \text{ or } \nu > R) \\ E_{C_1} + E_{C_2} + \nu E_{C_3} & (\text{if } 0 < \nu \leq R), \end{cases} \tag{10}$$

where E_{C_i} is the expected energy consumption of computational component C_i.

A. Measurement of Energy Consumption

1) Testbed for RS Decoding: The execution time and energy consumption of RS decoding were measured using the SEE (SNU Energy Explorer) [7]. In the following experiments, both the ARM7TDMI core and the SEC 128 Mbit SDRAM array (K4S280832A) were operated at a clock speed of 100 MHz, and the cache we used has 4-way associativity. This organization

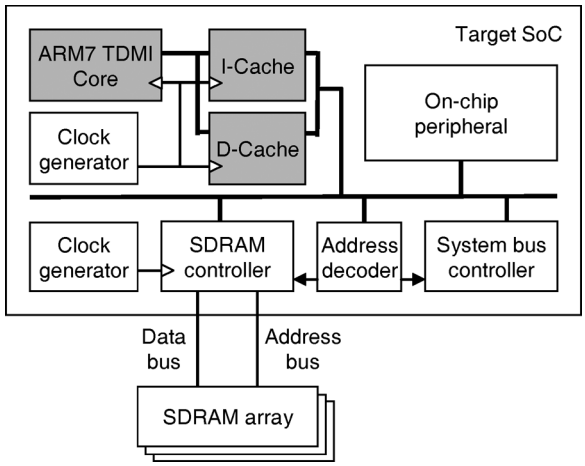


Fig. 4. Testbed for RS decoding.

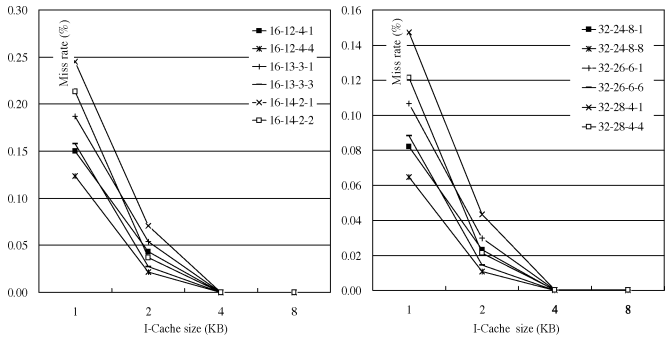


Fig. 5. I-cache miss-rate against I-cache size.

is abstracted from the general ARM7-based embedded system. Fig. 4 shows the block diagram of our RS decoding testbed.

Fig. 5 shows how the instruction cache (I-cache) miss-rate varies with cache size, while the decoder is handling an ECB which consists of codewords containing one octet error or the maximum number of octet errors that can be dealt with by each coding scheme. There may be 16 or 32 octets in an RS codeword, according to the BCMCS specification, and we will call these 16 and 32-series codes. In the figures, 32-24-8-ν denotes a (32,24,8) code with ν octet errors in each codeword, and similarly for other codes. As the number of errors increases, the I-cache miss-rate also increases. Then the Berlekamp algorithm runs more often but, because it is compact, it achieves many cache hits, so the absolute miss-rate is low in all cases. The cache miss-rate stabilizes when the size of the I-cache is 4KB, so we use an I-cache of this size in our testbed. Similarly, the miss-rate of the data cache (D-cache) stabilizes around 8KB, which is also apparent from Fig. 6.

As the number of errors increases, the number of accesses to the data block also increases. Having a cache large enough to exploit spatial and temporal locality is the major factor that determines the miss-rate. We have counted the number of load and store instructions to determine the effect of D-cache size. The results are shown in Fig. 7. As the number of errors increases, the number of load and store instructions also increases. We found similar ratios of between 8% and 14% for other 16 and 32-series

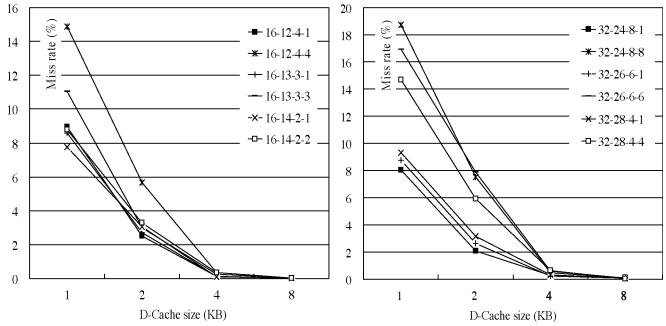


Fig. 6. D-cache miss-rate against D-cache size.

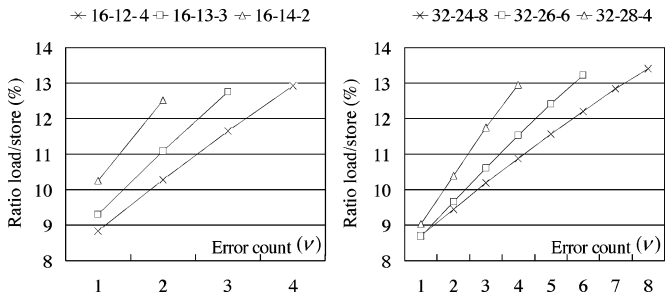


Fig. 7. Load/store instruction ratio against error count for different coding schemes.

TABLE I
EXPECTED EXECUTION TIME OF EACH COMPUTATIONAL COMPONENT

	Execution time (μs)		
RS code	C_1	C_2	C_3
(16,12,4)	11.9	823.5	54.8
(16,13,3)	8.1	658.6	52.3
(16,14,2)	4.3	501.9	49.1
(32,24,8)	60.9	1559.4	66.3
(32,26,6)	45.1	1179.0	60.0
(32,28,4)	29.5	828.2	53.7

TABLE II
EXPECTED ENERGY CONSUMPTION OF EACH COMPUTATIONAL COMPONENT

	Energy consumption (μJ)		
RS code	C_1	C_2	C_3
(16,12,4)	4.2	266.2	17.0
(16,13,3)	3.0	212.4	16.2
(16,14,2)	1.7	161.7	15.2
(32,24,8)	19.8	501.8	20.7
(32,26,6)	14.8	380.0	18.3
(32,28,4)	9.8	266.1	16.8

coding schemes with different parameters. The Berlekamp algorithm references the polynomial and Galois coefficients, and the exponential tables, which together take up 5KB. This is too much data for the registers, and load and store instructions are frequently used to save register values. As the number of errors increases, accesses to the data block grow. Thus the number of load and store instructions rises and the number of D-cache misses increases dramatically.

2) Energy Consumption by the Decoding Components: Tables I and II show experimental values of the average execution time and energy consumption for each computational component during RS decoding of one codeword when there is a single octet error ($\nu = 1$). These results were obtained by averaging the execution time and energy consumption over 10 trials ($\kappa = 10$), while varying the location of the error as follows:

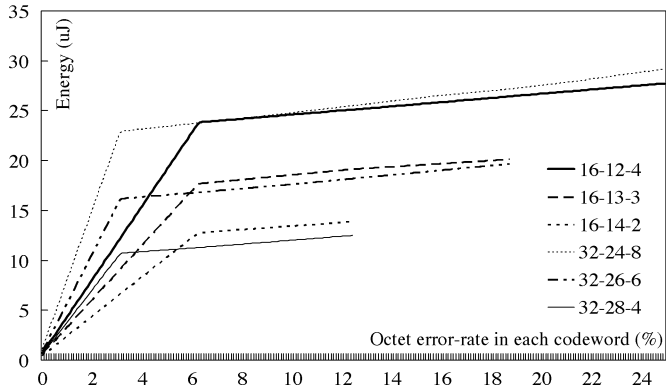


Fig. 8. Average energy consumption for transferring a payload of 1 byte.

$$T_{C_i} = \text{Avg}\left[\frac{1}{N}\sum_{j=1}^{N} T_{C_i}(j)\right]_{\text{trials}=\kappa}, \tag{11}$$

$$E_{C_i} = \text{Avg}\left[\frac{1}{N}\sum_{j=1}^{N} E_{C_i}(j)\right]_{\text{trials}=\kappa}, \tag{12}$$

where $T_{C_i}(j)$ and $E_{C_i}(j)$ are the execution time and energy consumption respectively, when an error occurs in the jth octet of a codeword. Fig. 8 shows the energy required to transfer 1 byte, excluding parity data. We see that the performance of RS depends more on the encoding scheme than on the number of errors encountered during decoding. The number of loops performed by the Berlekamp algorithm while generating the polynomial, lambda and the syndrome bytes required for error detection and correction is proportional to the amount of parity information. Generating the polynomial used to erase or evaluate the errors takes a significant amount of time: with the maximum number of errors execution takes about 15% longer than it does when there is only one error in each codeword, and energy consumption grows similarly. Also, there is a huge difference between the performance with no errors and with one error in each codeword, as we see in Fig. 8.

A coding scheme with fewer parity octets requires less time and energy, regardless of the number of errors, so it is important to use the most appropriate RS code for a particular channel condition.

In summary, each computational component has a characteristic effect on the execution time and energy consumption. The performance of all the components is dependent on the encoding scheme. The number of parity bits has a linear relation to the execution time and the energy consumption in computational components C_1 and C_2, whereas the time and energy used by C_3 are proportional to the number of parity octets and the number of errors.

IV. Energy Consumption by 32-Series RS Decoding in a Simulated Channel

In this section, we examine the energy consumed during the decoding of RS codes which have 32 octets in a codeword (32-series codes). We will set the number of packets in each ECB row (M) to 16, so as to maximize error recovery capacity (each packet contains 125 octets, so there are 125×16 codewords per ECB). We then derive the average total energy required to decode a data stream (t seconds of a b_i kb/s video clip), with varying bit error-rates in the traffic channel (BER) and mobile speeds (f_dT).

A. Channel Model

In this study, we used the Gilbert channel model [17]–[19] to simulate the behavior of data errors which arise in transmission over fading channels. Fading in the radio channel is assumed to have a Rayleigh distribution. A first-order two-state Markov process can simulate the error sequences generated by data transmission over a correlated Rayleigh fading channel: these errors occur in clusters or bursts with relatively long error-free intervals between them.

By choosing different values for the input bit error-rate and for f_dT (which is the Doppler frequency normalized to the data-rate, where f_d is the Doppler frequency, equal to the mobile velocity divided by the carrier wavelength [20]), we can model different degrees of correlation in the fading process. The value of f_dT determines the correlation properties, which are related to the mobile speed for a given carrier frequency. When f_dT is small, the fading process is strongly auto-correlated, which means long bursts of errors (slow fading). Conversely, the errors are weakly auto-correlated for large values of f_dT (fast fading). In the following experiments, we used values of 0.0001 (s_1) and 0.00005 (s_2) for f_dT, which correspond to fast and moderate fading conditions respectively, with a reference channel data-rate of 409.6 kb/s, and a carrier frequency of 900 MHz. It has been suggested [5] that a data-rate of 409.6 kb/s can be supported using a (16,14,2) RS code over more than 90% of network coverage with a dual receiver. Thus we assumed the use of QPSK modulation with a 409.6 kb/s data-rate forward channel.

In the equations that are to follow, α is the probability that the ith bit is corrupted, given that the $(i-1)$th bit is transmitted successfully, and β is the probability that the ith bit is successful, given that the $(i-1)$th bit was unsuccessful. The steady-state BER (ε) is then obtained as follows:

$$\varepsilon = \frac{\alpha}{\alpha + \beta}. \tag{13}$$

If the Rayleigh fading margin is F, the average bit error-rate can be expressed as

$$\varepsilon = 1 - e^{-\frac{1}{F}}. \tag{14}$$

Using (13) and the equations which follow, we can now derive values for α and β. The average number of consecutive bit errors is given by $1/\beta$, where

$$\beta = \frac{Q(\theta, \rho\theta) - Q(\rho\theta, \theta)}{e^{\frac{1}{F}} - 1}, \quad \text{and} \quad \theta = \sqrt{\frac{2/F}{1-\rho^2}}. \tag{15}$$

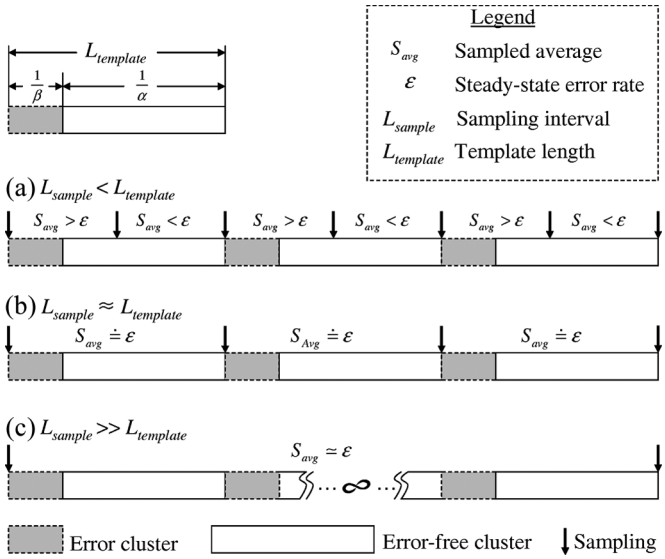


Fig. 9. The effect of sampling interval. (a) $L_{sample} < L_{template}$; (b) $L_{sample} \approx L_{template}$; (c) $L_{sample} \gg L_{template}$.

The term ρ is the correlation coefficient of two samples of the complex Gaussian fading process, and is expressed as $\rho = J_0(2\pi f_d T)$, where $j_0(\cdot)$ is a Bessel function of the first kind and of zeroth order. Additionally,

$$Q(x,y) = \int_y^\infty e^{-\frac{x^2+w^2}{2}} I_0(xw) w\, dw \tag{16}$$

is the Marcum-Q function. Thus, the relationship between bit error-rate and the Markov parameter can be represented as

$$\beta = \frac{1-\varepsilon}{\varepsilon}\left[Q(\theta, \rho\theta) - Q(\rho\theta, \theta)\right], \tag{17}$$

where

$$\theta = \sqrt{\frac{-2\log(1-\varepsilon)}{1 - J_0^2(2\pi f_d T)}}.$$

1) Sufficient Interleaving by the RS ECB: The BCMCS system varies the size of the ECB to scatter error clusters into a sparse pattern so as to maximize the error recovery performance of the Reed-Solomon decoder. We suggest that, if sufficient interleaving space is provided by the RS code, the Rayleigh distribution of an error cluster is converted into a random one. We will now present a model to support this contention.

Fig. 9 shows the relation between the sampling interval (L_{sample}) and the sampled average value (S_{avg}). As stated in the previous section, the average lengths of sequences of uncorrupted and error bits are $1/\beta$ and $1/\alpha$ respectively. The pattern of fluctuation in the channel condition is made up of a repetition of these sequences of normal and error bits, and we will call the average length of one sequence of normal bits and error bits a template $(L_{template})$, which can be expressed as follows:

$$L_{template} = \frac{1}{\alpha} + \frac{1}{\beta} = \frac{1}{(1-\varepsilon)\left[Q(\theta, \rho\theta) - Q(\rho\theta, \theta)\right]}. \tag{18}$$

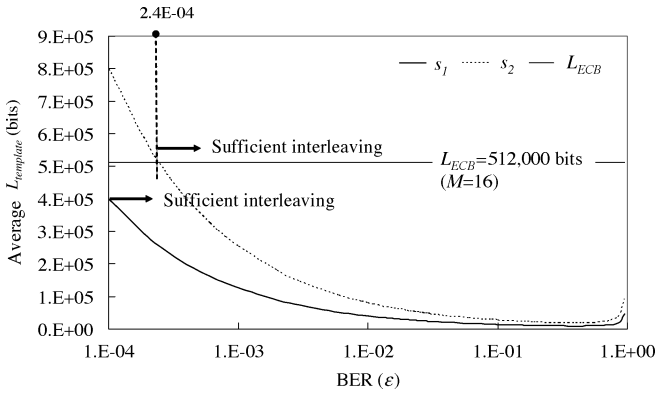


Fig. 10. Average $L_{template}$ and L_{ECB} against BER.

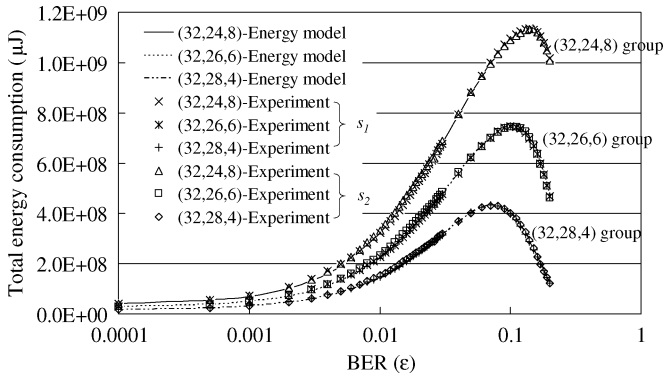


Fig. 11. Total energy consumption to decode a 1-hour 100 kb/s video clip.

We now consider the relative lengths of L_{sample} and $L_{template}$. If L_{sample} is smaller than $L_{template}$, then the composition of the samples is not homogenous, and the value of S_{avg} will vary dramatically. Most errors will be localized into a few samples and S_{avg} for those samples will be exaggerated. The other samples will contain relatively few errors, and for these S_{avg} will naturally be underestimated.

As L_{sample} converges to $L_{template}$, the fluctuation of S_{avg} will stabilize. After this point, the distribution of errors in the samples will approach more and more closely to the steady-state error-rate as L_{sample} increases further; and finally S_{avg} saturates to ε as L_{sample} grows towards ∞.

In BCMCS, the errors in each channel are interleaved by making the ECB larger (which also increases the value of M). The size of the ECB (L_{ECB}) can be considered as equivalent to L_{sample}, and is defined as follows:

$$L_{ECB} = (M \times 125) \times N \times (\text{bits/octet}), \tag{19}$$

where (bits/octet) is the bit-count of the transmission octet of the specific communication system.

Fig. 10 shows the theoretical variation of $L_{template}$ with the BER (ε), and the corresponding curve for L_{ECB} when M is 16 and N is 32. The value of $L_{template}$ for a mobile moving with a speed s_1 is smaller than the size of the ECB (L_{ECB}) for BERs in the range 1.0×10^{-4} to 1.0. Within that range, we can expect the channel to be in a situation similar to that shown in Fig. 9(b) or (9c), which is the result of sufficient interleaving. However, when the BER (ε) is below 2.4×10^{-4}, the value of $L_{template}$ for a mobile moving at a speed of s_2 is larger than L_{ECB}. This corresponds to a situation similar to that shown in Fig. 9(a). But, at both speeds, the ECB interleaves the error cluster sufficiently when the BER (ε) is more than 2.4×10^{-4} and M is 16.

From these results, we can see that, under most channel conditions, the BCMCS system can select an ECB size that will achieve sufficient interleaving to randomize the influence of the channel on the error distribution and from now on we will assume that sufficient interleaving is indeed being achieved.

B. Modeling Recovery Rate and Energy Metrics Under Sufficient Interleaving

Using (N, K, R) RS codes, $M = 16$, and assuming sufficient interleaving, the probability that a codeword contains ν octet errors, and the residual error-rate after RS error recovery ($\varepsilon_{residual}$) can be expressed as follows:

$$P(\nu|\varepsilon) = \binom{N}{\nu} (\varepsilon)^{\nu} (1-\varepsilon)^{N-\nu} \tag{20}$$

$$\varepsilon_{residual|\varepsilon, M=16} = \frac{1}{N} \times \sum_{\nu=R+1}^{N} \nu \times P(\nu|\varepsilon). \tag{21}$$

Thus the error recovery rate, which is an indicator of performance, can also be derived:

$$recovery\ rate = 1 - \frac{\varepsilon_{residual|\varepsilon, M=16}}{\varepsilon}. \tag{22}$$

If a mobile receives a t-seconds video clip at a data-rate of b_i kb/s, the expected energy consumption of RS decoding for a given steady-state error-rate of ε can be expressed as follows:

$$E[energy|\varepsilon, b_i, t] = \sum_{\nu=0}^{N} E_{codeword}(\nu) \times N(\nu|\varepsilon, b_i, t), \tag{23}$$

where $N(\nu)$ is the number of codewords that contain ν octet errors, and is given by

$$N(\nu|\varepsilon, b_i, t) = P(\nu|\varepsilon) \times \frac{b_i(\text{kb/s}) \times t(\text{seconds})}{K \times (\text{bits/octet})}. \tag{24}$$

If there is sufficient interleaving, we can use (23) to predict the total energy required by the RS decoder to process a given data payload, for different values of ε.

C. Measurement of Energy Consumption With a Simulated Channel

1) Verification of Our Energy Model: Fig. 11 shows theoretical and simulated results for the total energy required by the RS decoder to service a 1-hour video clip arriving at 100 kb/s, under BER (ε) conditions ranges from 0.0001 to 0.2. Experimental values of energy consumption ($E_{exp.}$) follow the theoretical

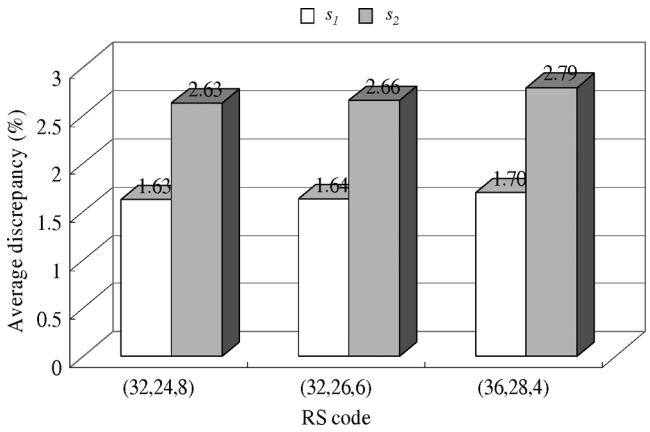


Fig. 12. Average discrepancy between the results from the energy model and from experiment.

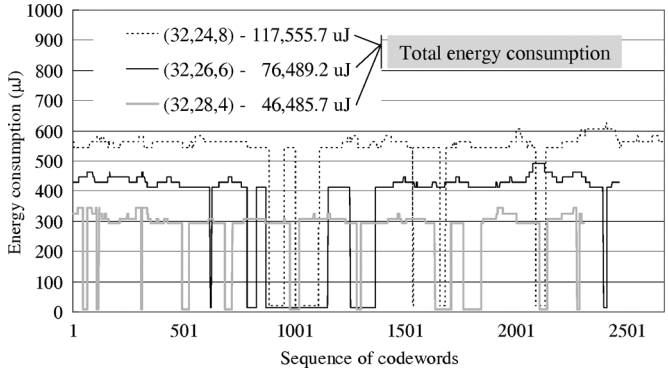


Fig. 13. Total energy consumption required to decode a sequence of codewords using the three RS codes shown. The mobile speed is s_1 and the BER (ε) is 5%.

curves closely: the agreement for a mobile moving quickly (at speed s_1) is excellent, and results for the slower mobile (moving at speed s_2) are also very close. The average discrepancy of our energy model can be expressed as follows:

$$\text{Average discrepancy} = 100 \times \text{Avg}\left[\frac{E_{exp.|\varepsilon,b_i,t} - E[energy|\varepsilon,b_i,t]}{E_{exp.|\varepsilon,b_i,t}}\right]_{\varepsilon_{run}}, \quad (25)$$

where the experimental run of values of ε is

$$\varepsilon_{run} = \{0.0001, 0.0005, 0.001, 0.002, \ldots, 0.03, 0.04, \ldots, 0.2\}.$$

From this equation we obtain errors of less than 3%, as shown in Fig. 12.

2) Analysis of the Energy Consumed by the RS Decoder in a Simulated Channel: The results presented in Fig. 11 show two significant trends. First, the energy consumption increases with the amount of parity information in the RS codes, as we explained in Section III-A-2. As the parity overhead increases, the effective data-rate is reduced so that more codewords are needed to handle the same data payload. As a result, it takes more time to process the data and more energy is consumed. The observation that RS codes with less parity consume less energy suggests that an appreciable amount of energy can be saved by choosing the slimmest RS code that would still guarantee the bit-level QoS requirement of a video application.

Second, the RS decoding process uses more energy as the value of ε increases. This is a plausible result because the error correction routine has to run more frequently if more errors are input to the RS decoder. However, the energy required for RS decoding actually decreases as the BER (ε) is incremented past a certain value, which depends on the RS code. This occurs because the RS decoder gives up error correction if the BER (ε) of the channel is sufficiently high, and simply forwards the erroneous codeword. When there is little parity information in a codeword, which is the case with a (32,28,4) code for instance, then error correction is abandoned relatively early (at $\varepsilon = 0.07$ for a (32,28,4) code), and thus no further energy is used by the computational components responsible for error correction (C_2 and C_3).

V. Suggestions for Energy-Efficient RS Coding Using 32-Series Codes

A. Selection of RS Codes for Energy-Efficiency

1) Motivation: We can see from the foregoing results that the RS coding scheme and the number of errors in a codeword affect the energy consumption, which increases as the amount of parity information and the number of errors increase. We measured the energy consumed by RS decoding using different codes in simulated but realistic radio channel conditions.

Fig. 13 shows how the energy used in RS decoding of a 5-seconds video clip arriving at 100 kb/s varies as a sequence of codewords (each containing 32 octets) is processed, when the BER (ε) is 0.05 and $f_dT = 0.0001(s_1)$. As the parity overhead increases, the effective data-rate is reduced, requiring more codewords to handle the same payload. As a result, it takes more time to process the data and more energy is consumed. The figures also show that the energy consumed in data decoding without errors is hardly affected by the choice of RS code, but the energy consumed during error correction differs significantly: more energy is needed for error correction if an RS code with more parity information is used. In comparison with a (32,24,8) code, the total energy consumption is reduced by about 60% when using a (32,28,4) code, and by about 35% when using (32,26,6) to decode the same data.

Our observation that RS codes with less parity information achieve a significant energy saving at the cost of error recovery performance suggests that the best choice of RS code for a video application is the one that saves most energy while still guaranteeing the bit-level QoS requirement.

2) Energy-Efficient RS Code for a Target BER: Let the set $\Upsilon = \{r_1, r_2, r_3\}$ contain candidate values of the amount of parity information in an RS code. For 32-series codes ($N = 32$), we can see from the BCMCS specification that $\Upsilon = \{4, 6, 8\}$. Also, let Z be a set which contains parity values that satisfy the target BER (T_{BER}):

$$Z = \{r_i | \forall r_i \in \Upsilon, \varepsilon_{residual|\varepsilon,M=16,K=N-r_i} \leq T_{BER}\}.$$

If we use the RS code with the minimum amount of parity that guarantees the required bit-level quality, i.e. $R = \min(Z)$, instead of the code with the maximum parity that the BCMCS specification allows, i.e. $R = \max(\Upsilon)$, then the average energy

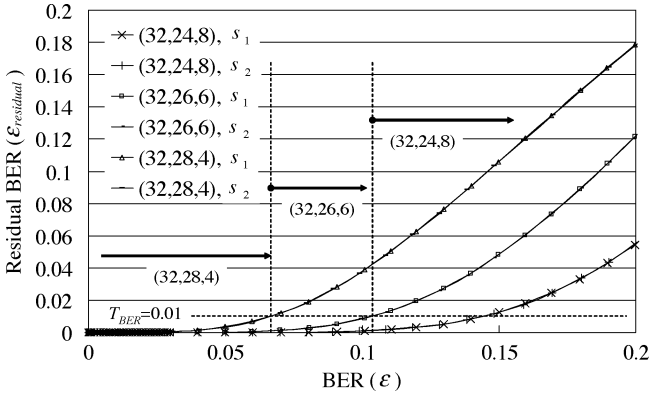


Fig. 14. Error recovery performance against BER (ε).

saving (ES), which is the reduction in total energy consumption for a given bit error-rate (ε) of the channel with a payload (b_i, t), can be approximated as follows assuming there is sufficient interleaving:

$$ES = 100 \times \left[1 - \frac{E\left[energy|\varepsilon, b_i, t, K = N - \min(Z)\right]}{E\left[energy|\varepsilon, b_i, t, K = N - \max(\Upsilon)\right]}\right]. \tag{26}$$

In order to select the RS code that is most efficient under particular operating conditions, and which also satisfies the bit-level QoS requirement, we simulated the error recovery process to determine its performance for given BERs of between 1% and 20%. The results, shown in Fig. 14, can be analysed in terms of two input factors: mobility and the RS coding scheme. First, we observe that the RS decoder achieves better error recovery performance at fast-moving mobiles, because they experience errors in shorter bursts than slow mobiles, even though the effect is not especially clear in Fig. 14. (The performance is almost the same at mobile speeds of s_1 and s_2 because the sufficient interleaving condition is met at both these speeds, and the simulation results follow the analytic results obtained from (21). Second, we see that codes with more parity information have a greater error recovery capacity, which we would expect.

The leftmost marked interval in Fig. 14 contains BERs for which a (32,28,4) code is the most energy-efficient, assuming the required bit-level QoS (T_{BER}) is 0.01; for BERs above 0.07 or so, a (32,26,6) code is the most efficient (middle interval); and when the channel condition is extremely bad and the value of ε exceeds 0.1, it is necessary to select the (32,24,8) code to reduce the residual BER ($\varepsilon_{residual}$) as far as possible (rightmost interval).

We performed a series of simulations to evaluate the energy efficiency of our scheme, and Fig. 15 shows the results. We see that we can make an average energy saving of up to 55% in the best case, by reducing the amount of parity information to a degree which just meets the required bit-level QoS. For example, Fig. 15(a) shows that, when a mobile station is moving at s_1 and the BER (ε) is 0.01, the average energy saving is about 30%, using the (32,26,6) code instead of (32,24,8), when the target BER (T_{BER}) is below 1.83×10^{-6}. In the same situation, we can save about 25% more energy by using the (32,28,4) code when the target BER (T_{BER}) is greater than 1.83×10^{-6}. Similarly, when the BER is 0.03, as shown in Fig. 15(b), a (32,26,6) code is the most energy-efficient, if the target BER is between 5.98×10^{-6} and 4.19×10^{-4}, and the mobile is moving at s_1: under these circumstances (32,26,6) uses about 30% less energy than (32,24,8). If the target BER exceeds 4.19×10^{-4}, then the (32,28,4) code is the most energy-efficient, with an average energy saving of about 55% compared to (32,24,8), at a mobile speed of s_1. At that speed, the only choice is (32,24,8) if the target BER is below 5.98×10^{-6}. Finally, when a mobile is moving at s_1 and the BER is 0.05, which is the result presented in Fig. 15(c), the (32,26,6) code is the most energy-efficient when the target BER is between 1.88×10^{-4} and 3.32×10^{-3}, and the (32,28,4) code is best when the target BER is above 3.32×10^{-3}. Fig. 15 also shows equivalent results for a mobile station moving at s_2.

There are many applications that do not require fastidious control of the BER and, when these are running, our scheme can save a significant amount of energy while providing an entirely adequate service.

B. Selection of ECB Size (M) for Energy-Efficiency

1) Motivation: We have already seen that there is a huge difference between the performance of the RS decoder with no errors and with one error in each codeword, because of the complexity of the Berlekamp algorithm which is used to locate and correct errors. This suggests that it is more energy-efficient to concentrate a burst of errors into a small number of codewords, rather than to disperse it more widely. By making the ECB as small as possible without allowing a significant reduction in error recovery performance, the number of error-free intervals is increased and energy is saved because a smaller proportion of the codewords contain errors.

We investigated the relationship between the average value of the residual BER ($\varepsilon_{residual}$) and the number of MAC packets per ECB (M) in a cdma2000 1xEV-DO broadcast environment. The three RS codes (32,24,8), (32,26,6) and (32,28,4) are used in these experiments. As the change in ECB size has more influence on slow-moving mobiles, which experience longer error bursts, we used values of 0.00001 (s_3) and 0.00002 (s_4) for f_dT, which correspond to slower fading conditions than those in the foregoing experiments (s_1 and s_2).

For each f_dT, the average residual BER ($\varepsilon_{residual}$) using Reed-Solomon coding is inversely proportional to M, as shown in Fig. 16. The reduction in residual BER for increasing M is more dramatic with RS codes which have more parity information, such as the (32,24,8) code. Using (32,24,8), the RS decoder recovers almost all errors if the value of M reaches 9, even when conditions are unfavorable for error recovery: a BER of 3% and a slow-moving mobile experiencing long error bursts, as shown in Fig. 16(a). Additionally, the average residual BER drops more quickly as the value of M increases when fading is fast, because the error bursts are shorter than those that occur when fading is slow. Short error bursts can be adequately interleaved in an ECB, and thus the average residual BER drops significantly in response to even a small increase of M. The value of M is linked to energy consumption and memory requirement, so that both energy and storage have to be sacrificed to reduce the average

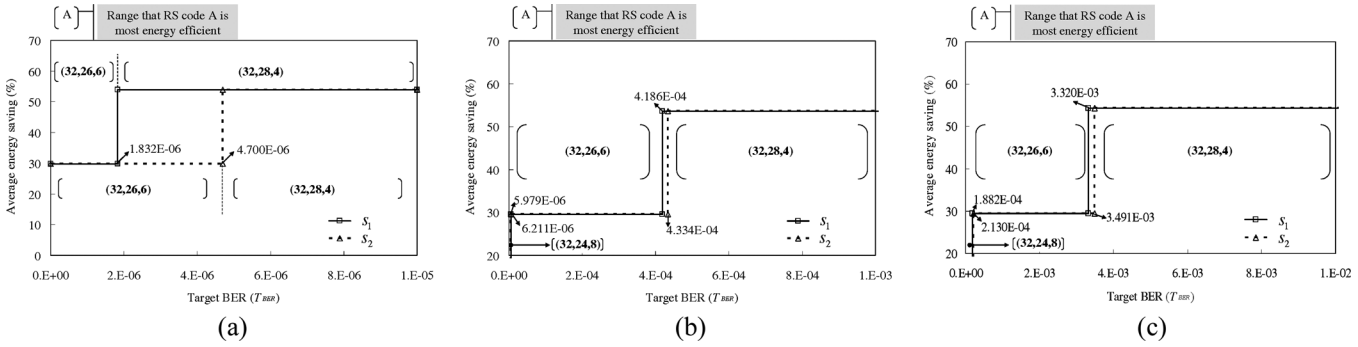


Fig. 15. Energy-efficient operating ranges of 32-series RS codes. (a) BER (ε) of 0.01. (b) BER (ε) of 0.03 (c) BER (ε) of 0.05.

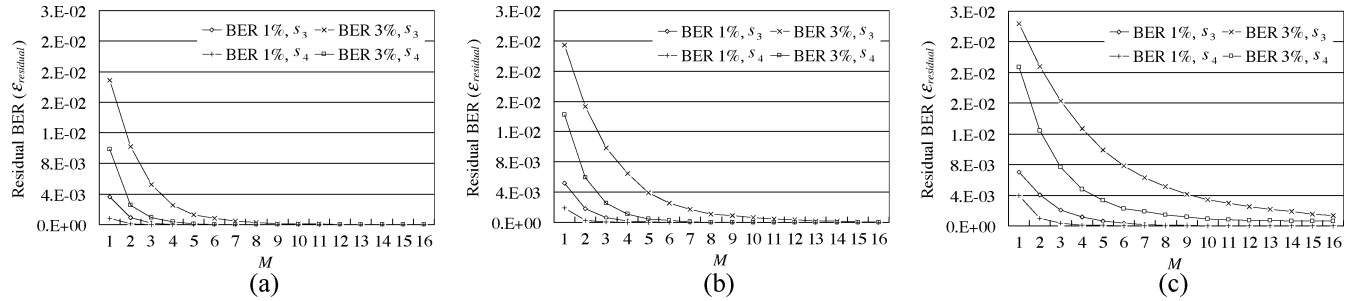


Fig. 16. Average residual BER $(\varepsilon_{residual})$ for different sizes of ECB (M). (a) (32,24,8) code. (b) (32,26,6) code. (c) (32,28,4) code.

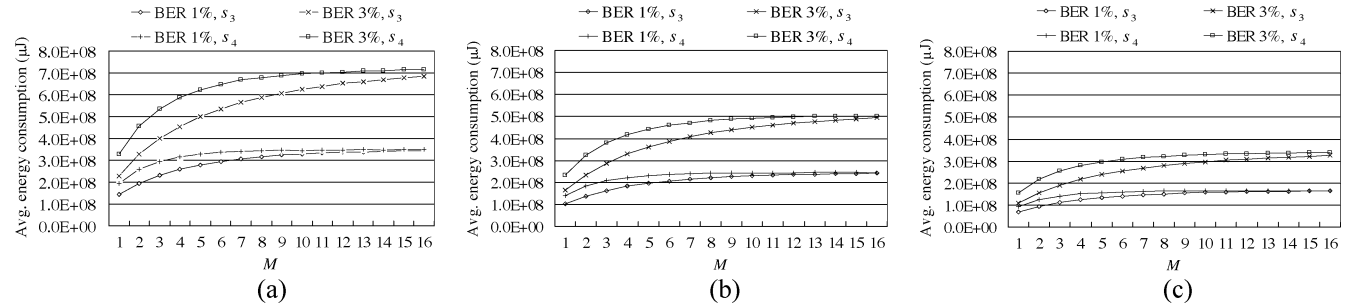


Fig. 17. Average energy consumption for different sizes of ECB (M). (a) (32,24,8) code. (b) (32,26,6) code. (c) (32,28,4) code.

residual BER. The energy consumption increases with the size of the ECB, as shown in Fig. 17. In essence, both Figs. 16 and 17 show aspects of the necessary tradeoff between error recovery performance and energy consumption.

Fig. 17 shows the average total energy required to decode a payload ($b_i = 100$ kb/s, $t = 3{,}600$ seconds) with varying BERs, at speeds of s_3 and s_4. These results can be analysed in terms of three factors: BER (ε), RS code and ECB size (the interleaving factor M). What is immediately clear is that the RS decoding process uses more energy as the BER (ε) increases. This is a plausible result because the error correction routine has to run more frequently if more errors are input to the RS decoder. The amount of parity information also affects the energy consumption as explained in Section III-A-2. Further, it is apparent that the energy consumption increases as the value of M increases because error bursts are dispersed over more codewords, making the error correction process run more frequently: this behavior is the main focus of this section, and suggests that it should be more energy-efficient to concentrate error bursts in a smaller number of codewords and to increase the number of error-free intervals, assuming that the required level of residual BER (T_{BER}) can still be achieved.

2) Energy-Efficient ECB Size: We can see from the way that energy consumption varies with the size of the ECB and the BER (ε) of the radio channel that energy can be saved if we choose the smallest ECB that still provides an acceptable level of performance. To select this adequate ECB size we could use a method similar to the one which we used in the previous section to select the best RS code to satisfy a target BER (T_{BER}). However, we will actually propose an alternative approach, which uses the following energy-efficiency metric to find the best ECB size:

$$E_{efficiency}(M=\delta)=\frac{(\varepsilon_{residual|M=\delta-1})-(\varepsilon_{residual|M=\delta})}{(energy|_{M=\delta})-(energy|_{M=\delta-1})}, \tag{27}$$

where $energy|_{M=\delta}$ is the energy consumption when M is δ. Thus $E_{efficiency}$ is the ratio between the drop in residual BER $(\varepsilon_{residual})$ and the extra energy required as M increases by 1.

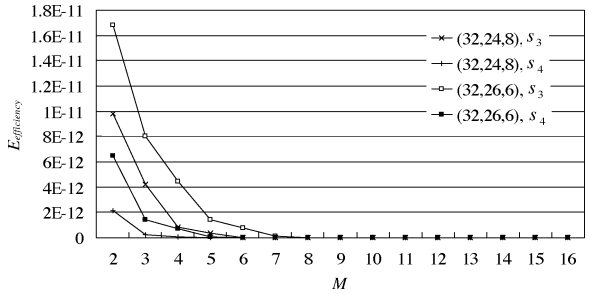


Fig. 18. Energy efficiency when the BER (ε) is 0.01.

TABLE III
ENERGY SAVINGS ACHIEVED BY RECOMMENDED VALUES OF M (M') COMPARED TO THE MAXIMUM VALUE OF M

RS code	Mobility	M'	BER (ε)	
			0.01	0.03
(32,24,8)	s_3	6	14.65%	11.69%
	s_4	4	9.50%	5.91%
(32,26,6)	s_3	8	13.69%	11.10%
	s_4	6	8.58%	4.03%
(32,28,4)	s_3	10	8.24%	8.09%
	s_4	8	5.31%	3.84%

TABLE IV
INCREASES IN RESIDUAL BER $(\varepsilon_{residual})$ ACHIEVED BY RECOMMENDED VALUES OF M (M')

RS code	Mobility	M'	BER (ε)	
			0.01	0.03
(32,24,8)	s_3	6	0	0
	s_4	4	0	0
(32,26,6)	s_3	8	0	4.73×10^{-5}
	s_4	6	0	8.07×10^{-5}
(32,28,4)	s_3	10	2.21×10^{-5}	5.96×10^{-4}
	s_4	8	9.09×10^{-5}	8.01×10^{-4}

Fig. 18 shows the values of this metric for a BER (ε) of 0.01, with (32,24,8) and (32,26,6) codes. In this example the energy-efficiency of a mobile which uses a (32,24,8) code and moves with a speed of s_3 drops almost to zero when the value of M is 6, which means that there is hardly any extra performance to be gained by using more energy. But when a mobile moves faster (s_4), $E_{efficiency}$ saturates earlier, when M reaches 4. Thus 6 is the most energy-efficient value at s_3, and 4 is best at s_4. Similar selections can be made for a (32,26,6) code, but in this case the saturation point is delayed to a larger size of ECB, because the performance of error recovery is reduced when there is less parity information, and thus gains in performance extend to higher values of M.

By selecting the size of the ECB in this way, a significant amount of energy can be saved in comparison with the maximum size of ECB $(M = 16)$, as shown in Table III. The average energy saving is 14.65% at speed s_3 and 9.5% at speed s_4. When a mobile moves at faster speeds, such as s_1 and s_2, the gain is more dramatic as the error bursts are shorter at these speeds and adjusting the size of the ECB has a greater effect. Because all multimedia data transmitted to a mobile device is subject to RS decoding, using the recommended value of M achieves a significant reduction in total energy consumption with negligible performance degradation. The increases in residual BER $(\varepsilon_{residual})$ are presented in Table IV. As the MPEG decoder has its own error resilience, such modest increases will have little influence on the playback quality of MPEG videos.

VI. CONCLUSION

We have investigated and analysed the energy consumption of mobiles receiving high data-rate broadcast services in a 3G cellular network, focusing on error recovery by the Reed-Solomon decoder that operates in the MAC layer. We have also proposed an analytic energy model which we verified by extensive simulation. The energy consumption of the RS decoding process is mainly determined by three computational components: the first of these decides whether to try correcting any errors, and the other two build the syndromes and erasure loops that are used to perform the required corrections.

We found that the choice of RS code affects the energy consumption, which increases with the amount of parity information in the code. Our analysis of the energy consumed by RS decoding suggests an energy-efficient operating range for each RS coding scheme, while guaranteeing the bit-level QoS under varying channel conditions. Experimental results show that a significant amount of energy can be saved by selecting the RS code appropriate to the channel conditions, while still meeting the target BER (T_{BER}).

The RS error-correction scheme also uses a data interleaving mechanism to increase error-recovery performance. This can be adjusted in current BCMCS by changing the size of the ECB. By increasing the size of the ECB, we can recover from bursty errors more efficiently; however, a larger ECB increases energy consumption, memory requirement, and service delay. We therefore make the ECB as small as we can without incurring significant performance reduction, while allowing for changing channel conditions. This has been shown to reduce the overall average energy consumption of mobile nodes running a video application, with only minor reduction in playback quality. This is a significant improvement on the use of an ECB of the maximum size, without regard to the channel conditions.

REFERENCES

[1] S. Parkvall, E. Englund, M. Lundevall, and J. Torsner, "Evolving 3G mobile systems: broadband and broadcast services in WCDMA," *IEEE Communications Magazine*, vol. 44, no. 2, pp. 30–36, Feb. 2006.

[2] A. Boni, E. Launay, T. Mienville, and P. Stuckmann, "Multimedia broadcast multicast service—Technology overview and service aspects," in *Proc. IEE International Conference on 3G Mobile Communication Technologies*, 2004, pp. 634–638.

[3] J. Wang, R. Sinnarajaj, T. Chen, Y. Wei, and E. Tiedemann, "Broadcast and multicast services in cdma2000," *IEEE Communications Magazine*, vol. 42, no. 2, pp. 76–82, Feb. 2004.

[4] *Broadcast and Multicast Services Framework*, 3GPP2 Std. X.P0019 Rev. 0.1.4, Mar. 2004.

[5] P. Agashe, R. Rezaiifar, and P. Bender, "CDMA2000 high rate broadcast packet data air interface design," *IEEE Communications Magazine*, vol. 42, no. 2, pp. 83–89, Feb. 2004.

[6] *CDMA2000 High Rate Broadcast-Multicast Packet Data Air Interface Specification*, 3GPP2 Std. C.S0054 Rev. 1.0, Feb. 2004.

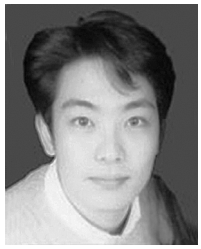
[7] I. Lee, Y. Choi, Y. Cho, Y. Joo, H. Lim, G. Lee, H. Shim, and N. Chang, "Web-based energy exploration tool for embedded systems," *IEEE Design and Test of Computers*, vol. 21, no. 6, pp. 572–586, Nov. 2004.

[8] R. Parry, "cdma2000 1xEV-DO [for 3G communications]," *IEEE Potentials*, vol. 21, no. 4, pp. 10–13, Oct./Nov. 2002.

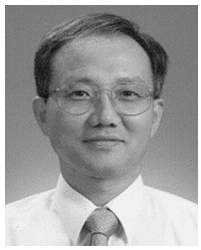
[9] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi, "CDMA/HDR: a bandwidth-efficient high-speed wireless data service for nomadic users," *IEEE Communications Magazine*, vol. 38, no. 7, pp. 70–77, Jul. 2000.

[10] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communications: A review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, May 1998.
[11] J. T. H. Chung-How and D. R. Bull, "Robust H.263+ video for real-time Internet applications," in *Proc. International Conference on Image Processing*, Sep. 2000, vol. 3, pp. 544–547.
[12] S. Praveenkumar, H. Kalva, and B. Furht, "Application of video error resilience techniques for mobile broadcast multicast services (MBMS)," in *Proc. IEEE International Symposium on Multimedia Software Engineering*, Dec. 2004, pp. 507–512.
[13] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Reading, MA: Addison-Wesley, 1992.
[14] P. M. Robert, A. M. Darwish, and J. H. Reed, "MPEG video quality prediction in a wireless system," in *Proc. IEEE Vehicular Technology Conference*, May 1999, vol. 2, pp. 1490–1495.
[15] Y. C. Chang and M. S. Beg, "MPEG-4 video error resilient tools performance evaluation and assessment," in *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Aug. 2003, vol. 1, pp. 70–72.
[16] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.
[17] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journal*, vol. 39, pp. 1235–1266, Sep. 1960.
[18] M. Zorzi and R. R. Rao, "On the statistics of block errors in bursty channels," *IEEE Trans. Communications*, vol. 45, no. 6, pp. 660–667, Jun. 1997.
[19] M. Zorzi, R. R. Rao, and L. B. Milstein, "Error statistics in data transmission over fading channels," *IEEE Trans. Communications*, vol. 46, no. 11, pp. 1468–1477, Nov. 1998.
[20] W. C. Jakes, *Microwave Mobile Communications*. New York: Wiley, 1974.

Kyungtae Kang received B.S. (1999) and M.S. (2001) degrees in computer engineering from Seoul National University, Korea. He is currently working toward a Ph.D. degree in the School of Electrical Engineering and Computer Science at Seoul National University. He is a student member of IEEE and IEICE. His research interests include packet scheduling, error control, QoS provision, and energy minimization issues in next-generation wireless/mobile networks. In particular, he is researching the performance and energy requirements of 3G cellular broadcast services such as BCMCS and MBMS.



Yongwoo Cho received the Premedical Degree from the College of Medicine, University of Ulsan, in 1997, a B.S. degree in Computer Science from Korea National Open University in 2004, while he was in military service, and an M.S. degree in Electrical Engineering and Computer Science from Seoul National University in 2006. He has worked as a researcher in Dooin Corp. and as a general manager in Bluecord Technology, Inc. His primary interests include multimedia systems, digital broadcasting, next-generation wireless/mobile networks, error control, real-time computing, and low-power design. He is currently a Ph.D. student in the School of Electrical Engineering and Computer Science at Seoul National University.



Heonshik Shin received the B.S. degree in applied physics from Seoul National University, Korea, in 1973. Since he received PhD degree in computer engineering from the University of Texas at Austin in 1985, he has actively involved himself in researches of various topics, ranging from real-time computing and distributed computing to mobile systems and software. He is currently a professor of School of Computer Science and Engineering at Seoul National University.