A Class of Adaptive Hybrid ARQ Schemes for Wireless Links

Sunghyun Choi, *Member, IEEE,* and Kang G. Shin, *Fellow, IEEE*

***Abstract*—Wireless links are known to suffer location-dependent, time-varying, and bursty errors. This paper considers a class of adaptive error-control schemes in the data link layer for reliable communication over wireless links in which the error-control code and the frame length are chosen adaptively, based on the estimated channel state/condition. Three error-control schemes are considered according to: 1) the number of code segments a packet is divided into and 2) the way a lost packet is retransmitted. Through throughput performance and computation complexity analyses, these three schemes are compared, and then one of them is claimed to be the most attractive in terms of computation complexity and practicality even though its throughput performance is not the best. The simulation results also verify that this scheme works well over a time-varying fading channel. Error control for the medium access control (MAC) header and its effect on the performance of each error-control scheme are also considered since, without proper error protection for the header, it would be futile to exercise error control on the user data.**

***Index Terms*—Adaptive error control, hybrid automatic repeat request (ARQ), link layer protocol, Reed–Solomon (RS) codes, wireless networks.**

I. Introduction

RECENTLY, there has been an impetus in the design and deployment of wireless/mobile networks due mainly to a rapidly growing number of applications such as intelligent transportation systems and ubiquitous computing. As a result, wireless systems such as cellular networks, personal communication systems (PCS), and wireless LANs are becoming widely accepted and deployed. A key requirement of these wireless networks is reliable transmission service. Unfortunately, wireless links are known to be error-prone and suffer location-dependent, time-varying, and bursty errors [4], [15], [16]. Physical factors contributing to this unreliability include: 1) signal degradation due to the distance between sender and receiver; 2) multipath propagation of a transmitted signal; 3) movement of sender, receiver, environment, or all of them; and 4) interferences by the signals in the same frequency band. Most wireless systems have, therefore, adopted various error-combating schemes in both the physical layer (e.g., error-correction coding, spread spectrum, equalizer, and diversity techniques) and the data link layer (e.g., frame length control, error correction, and error detection and retransmission).

This paper considers the problem of using error-control coding in the data link layer to achieve reliable communication over a wireless link. Broadly speaking, there are two types of error control: forward error correction (FEC) and automatic repeat request (ARQ). ARQ is efficient when the channel condition is good or moderately good, but as the channel condition gets deteriorated, ARQs throughput performance becomes unacceptably poor. We consider: 1) an adaptive hybrid of FEC and ARQ using Reed–Solomon (RS) code and 2) adaptive frame-length control. RS codes are known to provide excellent error-correction capability, especially, in terms of bursty error suppression. The RS code rate and frame length are chosen adaptively based on the estimated channel condition to maximize the throughput performance. We also consider error control on the medium access control (MAC) header since: 1) without proper error protection of the header, it would be futile to apply any error control on the user data and 2) the MAC header is also part of the data link layer.

Three error-control schemes (referred to as **RS-I**, **RS-II**, and **RS-III**) are considered depending on: 1) how many RS code segments are used for each packet and 2) how a packet with uncorrectable errors is retransmitted. The computation complexity of an error-control scheme is as important as its throughput performance, as this is closely related to battery power consumption, which is one of the limiting factors in wireless/mobile devices. Encoding/decoding processes with RS codes are known to consume substantial battery power [10]. We use the central processing unit (CPU) time for encoding/decoding of an RS codec implementation as the measure of computational complexity. While the actual complexity of the RS codec will depend on a particular implementation, we believe that the CPU time measurement can be a good reference. The three schemes are compared in terms of throughput performance and computation complexity. **RS-II** is shown to be the most attractive in terms of computation complexity and practicality even though its throughput performance is a little worse than that of **RS-III**.

The paper is organized as follows. Section II describes the specification of error-control codes and wireless systems under consideration. Error-control schemes are described in Section III. Section IV analyzes the error probabilities of the RS codes for user data and Bose–Chaudhuri–Hocquenghem (BCH) codes for the MAC header. The throughput efficiency and computational complexity of the three error-control schemes are analyzed in Sections V and VI, respectively. Section VII comparatively evaluates these three schemes, and

Manuscript received November 15, 1999; revised October 10, 2000. This work was supported in part by the AFOSR under MURI Grant F49620-00-1-0327.

S. Choi was with the University of Michigan, Ann Arbor, MI 48109 USA. He is currently with Philips Research, Briarcliff Manor, NY 10510 USA (e-mail: sunghyun.choi@philips.com).

K. G. Shin is with Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: kgshin@eecs.umich.edu).

Publisher Item Identifier S 0018-9545(01)03955-X.

determines **RS-II** to be the most preferred. The adaptation rule of adaptive **RS-II** is presented in Section VIII, then Section IX evaluates its performance in time-varying fading channels. Section X puts our results in a comparative perspective relative to other related work, then the paper concludes with Section XI.

II. System Specifications

We now describe the RS codes used in our error-control schemes for user data, and the physical layer (PHY) and MAC frame structures necessary for our throughput evaluation.

A. RS Codes

For the error control of user data, we adopt (N, K, q) RS codes over GF(q), in which the codeword size N $(\leq q-1)$ and the number of information symbols K $(<N)$. A q-ary symbol is mapped to b bits, so $q = 2^b$. RS codes are known to have the maximum error-correction capability for given redundancy, i.e., a maximum distance separable (MDS) code. For an (N, K) MDS code, the minimum distance $d_{\min}$ is determined as $d_{\min} = N - K + 1$, where the error correction capability $t = (d_{\min} - 1)/2 = (N-K)/2$, i.e., any combination of t symbol errors out of N symbols can be corrected. The code rate r_c is defined as $r_c = K/N$. One can easily see that the more parity symbols (i.e., larger $N-K$), the better error-correction capability. RS codes are also known to be efficient for handling bursty errors. For example, with $(N, K, 2^b)$ RS code with the error correction capability t, as many as $b \cdot t$ bit errors can be corrected in the best case when all of b bits in each of tb-bit symbols are erroneous (i.e., bursty errors). But only t bit errors can be corrected in the worst case when only one bit in each of tb-bit symbols is erroneous (i.e., nonbursty errors).

Originally, the codeword size of (N, K, q) RS code is determined to be $q-1$. However, a shorter codeword can be obtained via *code shortening*. For example, given an (N, K, q) code, $K - s$ information symbols are appended by s zero symbols. These K symbols are then encoded to make an N symbol-long codeword. By deleting all s zero symbols from the codeword, we can obtain $(N-s, K-s)$ code. For decoding this shortened code, the original (N, K) decoder can still be used by appending zero symbols between $K - s$ information symbols and $N - K$ parity symbols. Shortened RS codes are also MDS codes. Code shortening is useful especially for transmitting information with less than K symbols.

B. PHY and MAC Frame Structures

Our study draws on the specification of the popular WaveLAN modem. PHY and MAC overheads of the WaveLAN are shown in Fig. 1 [11]. In WaveLAN, no error-correction coding is implemented; only the cyclic redundancy check (CRC) code for error detection is implemented. We modified the MAC frame structure as shown in Fig. 2 for error-control schemes. First, the MAC header size is increased by one byte. This additional byte is used to give the error-control scheme information like the RS-code rate used for the subsequent user data, which can be adaptively changed. For the MAC frame header, we adopt the $(255, 139)$ binary BCH code, so 116 $(=255 - 139)$ parity bits (equivalently, 14.5 bytes) are

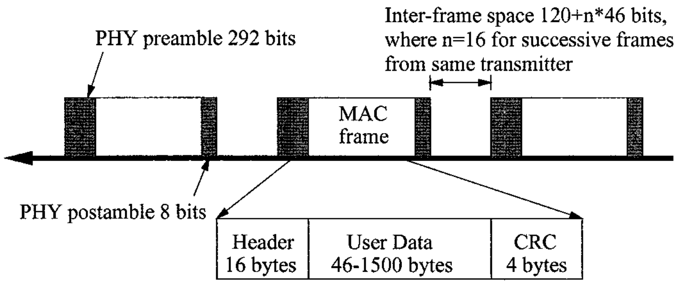


Fig. 1. PHY and MAC overheads for WaveLAN.

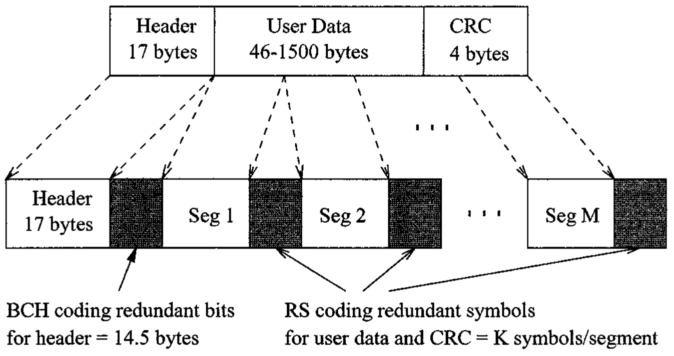

Header 17 bytes
User Data 46-1500 bytes
CRC 4 bytes
Header 17 bytes
Seg 1
Seg 2
Seg M
BCH coding redundant bits for header = 14.5 bytes
RS coding redundant symbols for user data and CRC = K symbols/segment

Fig. 2. The newly modified MAC frame structure.

appended to the header. Using this code, up to $t = 15$ bit errors can be corrected. This coding will be compared with other choices in Section IV-B. Note that this BCH coding is fixed for every MAC frame so that the receiver can receive/decode it without any extra information to check if the received MAC frame is destined for itself.

The user data plus CRC is divided into M segments, and each segment is encoded by an RS code. The segmentation and encoding procedures are detailed in the next section. The user data is assumed to be an IP packet with a 20-byte-long header and, throughout this paper, the terms "user data" and "packet" are used interchangeably. In order to examine the performance of error-control schemes, we specify various overheads of interest as follows.

- $H =$ the length of MAC header overhead including BCH redundancy $= 31.5$ $(=17 + 14.5)$ bytes $= 252$ bits.
- $O =$ the length of PHY overhead $= 420$ $(=120+292+8)$ bits (assuming $n = 0$).
- C $=$ the length of CRC and IP header overheads $=$ 24 $(=4 + 20)$ bytes $= 192$ bits.

III. Error-Control Schemes

For the proposed error-control schemes, we consider nine RS codes as shown in Table I. The reason for limiting the number of codes is that we need a different encoder-decoder pair for each (N, K, q) RS code. Described below are error-control schemes for the link layer.

A. A Hybrid of FEC and ARQ

We adopt a hybrid of FEC and ARQ, i.e., the receiver attempts to correct errors first and, if the errors are uncorrectable,

TABLE I
NINE RS CODES CONSIDERED FOR ADAPTATION

N	255			511			1023		
K	229	153	77	459	307	153	921	613	307
q	256			512			1024		
r_c	0.9	0.6	0.3	0.9	0.6	0.3	0.9	0.6	0.3
t	13	51	89	26	102	179	51	205	358

retransmission of the packet is requested. When errors are successfully corrected, an acknowledgment (ACK) is transmitted to the sender and when errors are detected but not correctable, a *negative acknowledgment* (NAK) is sent. We use the *selective-repeat* (SR) ARQ in this study. The sender keeps transmitting packets without waiting for ACK/NAK of those packets already transmitted. If NAK of a transmitted packet is received, or if neither ACK nor NAK of a packet is received within a timeout interval, the packet is retransmitted. The timeout interval is determined based on the roundtrip time. As will be clear below, the schemes considered here do not depend on ARQ, and hence, they can be used in conjunction with any other ARQ schemes, such as stop-and-wait (SW) and go-back-N (GBN). But throughput performance and complexity may depend on the underlying ARQ scheme. The SR-ARQ scheme is known to achieve better throughput performance than SW and GBN-ARQ at the expense of higher complexity.

An ACK/NAK packet consists of four bytes, in which the first two bytes are for the frame number of the packet associated with ACK/NAK, the third byte is to inform (1) whether it is an ACK or NAK, and (2) adapted code rate; and the last byte is a checksum. As will be clear later, the code rate is mainly adapted by the receiver depending on the estimated channel state/condition, the code-rate information is fed back to the sender within each ACK/NAK packet. These four bytes are encoded by $(148, 32)$ BCH code, which is a shortened code from $(255\,139)$ BCH code adopted for the MAC header error protection.

B. Sender Side

An IP packet is fragmented into a number of segments—which we call *user data*—depending on the maximum user data size that can be accommodated in a MAC frame, or the maximum transmission unit (MTU). The MTU of the WaveLAN is originally 1.5 Kbytes. With our schemes, MTU is adapted dynamically as required by the underlying error-control scheme. The CRC is calculated for both the user data and MAC header. The combined user data and CRC is then divided into M segments, where each of the first $M-1$ segments is Kb bits long and the length of the last segment is $\leq Kb$ bits. Note that MTU can be determined as $MKb - 32$ bits, where -32 represents the CRC overhead.

Each segment is encoded with $(N, K, 2^b)$ RS code. For the last segment, the code shortening might be needed. The MAC header is then encoded by $(255\,139)$ BCH code. Note that one byte of the header is used to represent the information of the error-control code such as N, K, and M. For the proposed schemes, only a set of codes is used, so one byte suffices to specify all the necessary information. Finally, the MAC frame forms the structure, as shown in Fig. 2.

C. Receiver Side

The error control at the receiver works as follows.

1) The header of the received frame is decoded at the MAC sublayer. If the header is successfully decoded and, if the frame is found to be destined for itself, the user data with M RS-coded segments as well as the RS code information from the header are sent upward to the link layer.
2) At the link layer, each of M RS-coded segments is first decoded by the RS decoder, then the entire frame (including the header and user data) is checked by the CRC decoder.

As described in the next section, the RS decoder at the receiver first attempts to correct errors. If the errors are uncorrectable, only their presence will be detected. Note that errors can be detected in three different places in the receiver: the MAC sublayer by the BCH decoder, the link layer by the RS decoder, and finally by the CRC decoder. This three-level error detection detects virtually all uncorrected errors.

D. Three Error-Control Schemes

We consider the following three possible error-control schemes using the above error-control facilities.

RS-I: $M = 1$, i.e., when a long RS code with a large N is used for each packet.
RS-II: $M \neq 1$ and the entire packet is retransmitted if any RS code segment has uncorrectable errors.
RS-III: $M \neq 1$ and each code segment with uncorrectable errors is requested to be retransmitted. Then, a set of those code segments received in error are retransmitted by the sender in one MAC frame.

Note that for **RS-III**, a simple form of ACK and NAK is not enough; more on this issue will be discussed later. A packet decoded without any RS decoding failure can have undetected errors as explained in the next section. These errors are most likely to be detected by the CRC. In this case, the sender is requested to retransmit the entire packet for all three schemes.

IV. ERROR PERFORMANCE ANALYSIS

We analyze the error probabilities of RS codes for user data and BCH codes for the MAC header in this section.

A. RS Code Performance

We use a bounded-distance RS decoder with the error-correction capability t. The decoder looks for a codeword within distance t of the received word; if there is such a codeword, the decoder will find it and, if not, the decoder will declare "decoding failure." Any RS-code segment with smaller than or equal to t errors can be corrected while code segments with larger than t errors result in either a decoding error (i.e., decoding into a wrong codeword) or decoding failure (i.e., inability to correct errors).

Let P_s denote the symbol error rate at the input of the RS decoder, then the probability P_T of the RS decoder anomaly, which includes both decoding errors and failures is given by

$$P_T = \sum_{i=t+1}^{N} \binom{N}{i} P_s^i (1-P_s)^{N-i} \tag{1}$$

where $t = \lfloor (N-K)/2 \rfloor$ is the RS error-correction capability and P_s is represented by

$$P_s = 1 - (1-P_b)^b \tag{2}$$

for a binary symmetric channel (BSC) with the channel bit error rate (BER) P_b. Now, the probability P_E of RS decoding error and the probability P_F of RS decoding failure are given by [13]

$$P_E \le P_T \sum_{i=0}^{t} \binom{N}{i} (2^b-1)^{i-(N-K)}, \tag{3}$$

$$P_F = P_T - P_E. \tag{4}$$

B. MAC Frame Header Performance

The MAC frame with an error-free or error-corrected header carries its contents [i.e., user data plus cyclic redundancy check (CRC)] to the RS decoder. This means that error protection of the user data is of no use unless the MAC frame header is also protected by an equally powerful error-control scheme. We adopt the $(255\,139)$ binary BCH code for the header error control. The header-error probability for a given BER P_b can be expressed as

$$P_h = \sum_{e=t+1}^{n} \binom{n}{e} P_b^e (1-P_b)^{n-e} \tag{5}$$

where the BCH codeword size $n = 255$ and error-correction capability $t = 15$. Note that the header errors can be classified as a decoding failure/error. In case of a decoding failure, the receiver just ignores the packet and the sender will retransmit the packet upon expiration of the retransmission timeout if the receiver was the real intended receiver (case 1). In case of a decoding error, the receiver might send the packet to the RS decoder when the decoded header points to the receiver's address (case 2) or it might otherwise ignore the packet, thus triggering retransmission via the expiration of the retransmission timeout (case 3). In case 2, the error in the decoded header will eventually be detected by CRC decoder after RS decoding.

Fig. 3 shows the header error probability as BER increases for the following four different cases of header error control: 1) without any coding for the header, which is the case with WaveLAN; 2) with $(255\,139)$ BCH coding as used in our schemes; 3) with the shortened $(31, 17, 256)$ RS coding with $t = 7$; and 4) with the shortened $(42, 23, 64)$ RS coding with $t = 9$. All the three codes compared have roughly the same code rate (i.e., $r_c \approx 0.55$), while accommodating a 17-byte header within them. Note that RS codes of code-rate 0.55 with symbol size <64 cannot accommodate the header. The header error probability for RS codes can be obtained directly from (1).

We first observed that the header without coding is too vulnerable to be useful. The BCH code is found to be the best

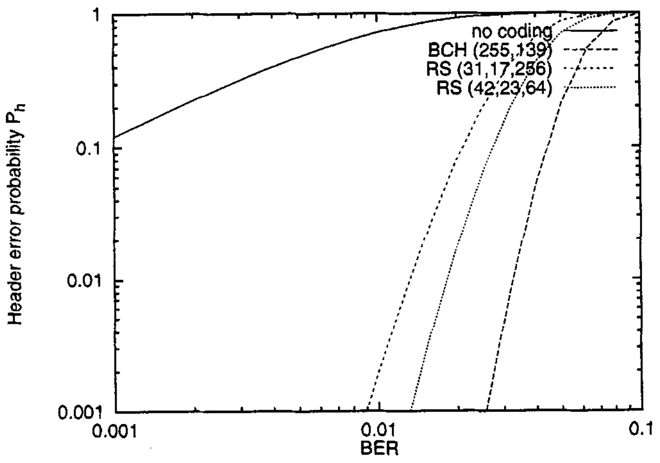


Fig. 3. Header error probability versus BER.

among the cases compared. The reason why RS codes perform worse than the BCH code is that they are nonbinary codes, so an error in an 8-bit symbol (e.g., for $(31, 17, 256)$ RS code) results in a symbol error. This is why $(42, 23, 64)$ code outperforms $(31, 17, 256)$ code. On the other hand, RS codes work better in dealing with bursty errors and this is why the RS codes are adopted for user data.

V. Throughput Efficiency

Throughput efficiency is defined as the portion of "useful" information bits (or the payload of an IP packet excluding the IP header) in each packet transmitted considering both PHY and MAC overheads. Note that the throughput efficiency corresponds to so-called *goodput* since only successful (or useful) information bits are accounted for.

For the analysis, we 1) ignore ACK/NAK overheads and 2) assume an ideal packet error-detection code (or CRC), i.e., all the errors are detected eventually. The second assumption is reasonable since errors are detected by BCH, RS, and CRC decoders. For example, even with the $(255, 229, 256)$ RS code, of which code rate is 0.9, the decoding error probability $P_E \approx 1.18 \times 10^{-10}$ for $P_b = 0.1$, i.e., most errors are detected by the RS decoder, then most of the remaining errors will be captured by the CRC decoder.

A. RS-I

For **RS-I**, the average number $E[N_{\text{tr}}^{\text{I}}]$ of bits transmitted till the successful reception of a packet can be obtained as

$$E\left[N_{\text{tr}}^{\text{I}}\right] = L_{\text{pkt}}(1) + P_R^{\text{I}} E\left[N_{\text{tr}}^{\text{I}}\right] \tag{6}$$

where the packet retransmission probability is given by

$$P_R^{\text{I}} = 1 - (1-P_h)(1-P_T) \tag{7}$$

and the number $L_{\text{pkt}}(M)$ of bits needed to transmit a frame with $M(N, K, 2^b)$ RS code segments including PHY overhead, MAC overhead, coding redundancies, and information bits is represented by

$$L_{\text{pkt}}(M) = MNb + H + O. \tag{8}$$

This reduces to

$$E\left[N_{\text{tr}}^{\text{I}}\right] = \frac{L_{\text{pkt}}(1)}{1-P_R^{\text{I}}}. \tag{9}$$

Finally, the throughput efficiency η^{I} of **RS-I** can be obtained by the number of information bits in a packet divided by $E[N_{\text{tr}}^{\text{I}}]$

$$\eta^{\text{I}} = \frac{Kb-C}{E\left[N_{\text{tr}}^{\text{I}}\right]} = \frac{(Kb-C)\left(1-P_R^{\text{I}}\right)}{L_{\text{pkt}}(1)}. \tag{10}$$

B. RS-II

To analyze the throughput efficiencies for **RS-II** and **RS-III** with $M \neq 1$, we first define the probability $p_{e,f}(M)$ that e and f code segments out of M code segments result in decoding errors and decoding failures, respectively, as

$$p_{e,f}(M) = \binom{M}{e,f} P_E^e P_F^f (1-P_E-P_F)^{(M-e-f)}. \tag{11}$$

Then, for **RS-II**, we can easily see that P_T for **RS-I** is equivalent to $1-p_{0,0}(M)$ for **RS-II**. So, following (9)

$$E\left[N_{\text{tr}}^{\text{II}}\right] = \frac{L_{\text{pkt}}(M)}{1-P_R^{\text{II}}} \tag{12}$$

where

$$P_R^{\text{II}} = 1-(1-P_h)p_{0,0}(M). \tag{13}$$

Finally, the throughput efficiency can be expressed as

$$\eta^{\text{II}} = \frac{(MKb-C)}{E\left[N_{\text{tr}}^{\text{II}}\right]} = \frac{(NKb-C)\left(1-P_R^{\text{II}}\right)}{L_{\text{pkt}}(M)}. \tag{14}$$

C. RS-III

In case of **RS-III**, an entire packet will be retransmitted upon detection of a MAC header error or CRC error. However, only part of the packet will be retransmitted upon RS error detection. Basically, there are five cases to consider.

1) A MAC header error with probability P_h results in retransmission of the entire packet.
2) No error with probability $(1-P_h)p_{0,0}(M)$ requires no retransmission at all.
3) No decoding failure and some decoding errors with probability $(1-P_h)p_{e,0}(M)$ for $e \neq 0$ result in retransmission of the entire packet.
4) Some decoding failures and no decoding error with probability $(1-P_h)p_{0,f}(M)$ for $f \neq 0$ result in retransmission of a number of code segments until none of these code segments results in a decoding failure. This might require eventual retransmission of the entire packet if some of the retransmitted RS code segments contain undetected errors.
5) Some decoding failures and some decoding errors with probability $(1-P_h)p_{e,f}(M)$ for $e \neq 0$ and $f \neq 0$ result in retransmission of a number of code segments. This requires retransmission of the entire packet after successful reception of all those RS code segments originally with decoding failures.

Considering all of these cases, $E[N_{\text{tr}}^{\text{III}}]$ has the relationship in (15), shown at the bottom of the page, where the average number $A(f)$ of bits transmitted until all of f RS code segments in a packet pass through the RS decoder (after retransmitting some code segments) without any decoding failure is represented by

$$A(f) = L_{\text{pkt}}(f) + (1-P_h)\sum_{f'=1}^{f}\sum_{e=0}^{f-f'} p_{e,f'}(f)A(f') + P_hA(f) \tag{16}$$

and the probability $B(f)$ that any of these f retransmitted code segments contains undetected errors is given by

$$B(f) = 1-\left(\frac{P_S}{P_S+P_E}\right)^f. \tag{17}$$

By solving (15), we can obtain (18), shown at the bottom of the page.

For the special case of $M=1$, i.e., one RS code segment for a packet, (18) reduces to

$$E\left[N_{\text{tr}}^{\text{III}}\right] = \frac{L_{\text{pkt}}(1)}{(1-P_h)(1-P_T)} = E\left[N_{\text{tr}}^{\text{I}}\right]. \tag{19}$$

Finally, the throughput efficiency can be represented by

$$\eta^{\text{III}} = \frac{(MKb-C)}{E\left[N_{\text{tr}}^{\text{III}}\right]}. \tag{20}$$

$$E\left[N_{tr}^{\text{III}}\right] = L_{\text{pkt}}(M) + \left(P_h + (1-P_h)\sum_{e=1}^{M} p_{e,0}(M)\right)E\left[N_{\text{tr}}^{\text{III}}\right] + (1-P_h)\left(\sum_{f=1}^{M} p_{0,f}(M)\left(A(f)+B(f)\cdot E\left[N_{\text{tr}}^{\text{III}}\right]\right) + \sum_{f=1}^{M-1}\sum_{e=1}^{M-f} p_{e,f}(M)\left(A(f)+E\left[N_{tr}^{III}\right]\right)\right) \tag{15}$$

$$E\left[N_{\text{tr}}^{\text{III}}\right] = \frac{L_{\text{pkt}}(M) + (1-P_h)\sum_{f=1}^{M}\sum_{e=0}^{M-f} p_{e,f}(M)A(f)}{(1-P_h)\left(1-\sum_{f=1}^{M} p_{0,f}(M)B(f) - \sum_{f=0}^{M}\sum_{e=1}^{M-f} p_{e,f}(M)\right)}. \tag{18}$$

TABLE II
ENCODING TIMES T_e AND DECODING TIMES $T_d(e)$ FOR NINE RS CODES: THE VALUES IN $(\cdot)$ ARE NORMALIZED TIMES PER INFORMATION BIT. ALL TIMES ARE MEASURED IN μS

N	K	r_c	T_e	$T_d(0)$	$T_d(1)$	$T_d(t)$	$T_d(t+1)$
255	229	0.9	339 (0.185)	364 (0.199)	408 (0.223)	620 (0.338)	553 (0.302)
	153	0.6	870 (0.711)	1489 (1.217)	1761 (1.439)	4138 (3.381)	3181 (2.599)
	77	0.3	823 (1.336)	2839 (4.609)	3670 (5.958)	9655 (15.674)	6937 (11.262)
511	459	0.9	1462 (0.354)	1603 (0.388)	1672 (0.405)	2637 (0.638)	2382 (0.577)
	307	0.6	4187 (1.515)	6309 (2.283)	7452 (2.697)	17586 (6.365)	13293 (4.811)
	153	0.3	3603 (2.617)	11818 (8.582)	14846 (10.781)	38399 (27.886)	27413 (19.908)
1023	921	0.9	5272 (0.572)	6709 (0.728)	7075 (0.768)	10887 (1.182)	10010 (1.087)
	613	0.6	13685 (2.232)	27189 (4.435)	31036 (5.063)	67533 (11.017)	54091 (8.824)
	307	0.3	14082 (4.587)	44417 (14.468)	55222 (17.988)	162082 (52.795)	122036 (39.751)

D. No Coding

We now consider the throughput efficiency in case of no RS coding for user data. Note that the MAC header is still encoded by the BCH code even without RS coding. The average number $E[N_{\rm tr}^{\rm No}]$ of bits transmitted until successful reception of a packet with the maximum transmission unit *MTU* is given by

$$E\left[N_{\rm tr}^{\rm No}\right] = \frac{L_{\rm pkt}^{\rm No}}{1 - P_R^{\rm No}} \tag{21}$$

where the packet-retransmission probability

$$P_R^{\rm No} = 1 - (1 - P_h)\left(1 - (1 - P_b)^{(MTU+32)}\right) \tag{22}$$

and the number $L_{\rm pkt}^{\rm No}$ of bits needed to transmit a frame is given by

$$L_{\rm pkt}^{\rm No} = MTU + 32 + H + O. \tag{23}$$

Finally, the throughput efficiency can be expressed as

$$\eta^{\rm No} = \frac{MTU + 32 - C}{E\left[N_{\rm tr}^{\rm No}\right]} = \frac{(MTU + 32 - C)\left(1 - P_R^{\rm No}\right)}{L_{\rm pkt}^{\rm No}}. \tag{24}$$

VI. COMPUTATIONAL COMPLEXITY ANALYSIS

We analyze the computational complexity associated with the process from the generation of a packet to its successful reception by the receiver. RS encoding and decoding complexities are considered to account for packet generation and reception overheads, assuming that these two are the most dominant factors compared to the others including packetization, segmentation, BCH encoding and decoding, and CRC encoding and decoding overheads.

A. Measured Encoding and Decoding Times

As the complexity measure, we use the CPU time, which is obtained from 100 000 to 500 000 runs of encoding randomly generated information bits and decoding code segments with errors in randomly chosen locations on a Sun 300 MHz Ultra 10 with 128 Mbytes of RAM running Solaris 2.6. As mentioned in the introduction, this measure can work as a good informative reference while not an absolute measure since the computational complexity really depends on how the codec is implemented. The C source code used is based on a code written by Phil Karn and available free for noncommercial use at his homepage [8].

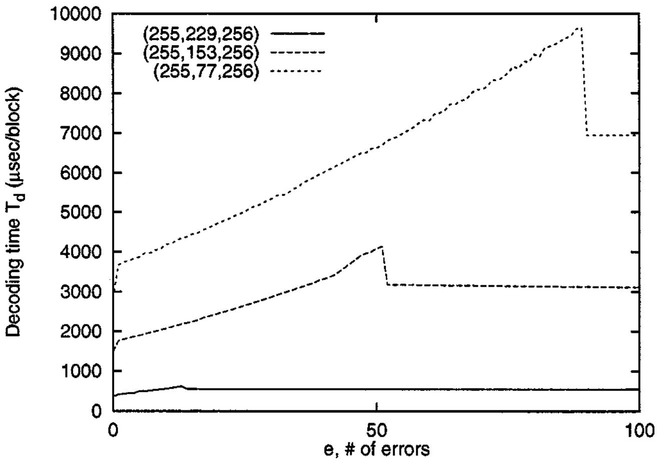


Fig. 4. Decoding time versus the number of errors in a $(255, K, 256)$ RS code segment.

The decoder is implemented following the standard implementation found in [1], using the Massey–Berlekamp algorithm for finding the error-locator polynomial and the Chien search for finding its roots. First, the *encoding complexity* per RS code segment in terms of the CPU time for a given (N, K, q) RS code can be represented by

$$C_e = T_e \tag{25}$$

where T_e is the CPU time required for RS encoding. The fourth column in Table II shows the encoding times of nine RS codes. The values in the parentheses are the normalized encoding time per information bit.

Fig. 4 shows the decoding times $T_d(e)$ as the number e of symbol errors in a code segment increases for three $(255, K, 256)$ RS codes. We observe that $T_d(e)$ increases abruptly from $e = 0$ to 1 and increases approximately linearly until $e = t$, then drops abruptly from $e = t$ to $t + 1$. This dropoff happens since the decoding process is stopped without attempting to correct errors when the number of errors is found to be greater than t. $T_d(e)$ gets saturated beginning at $e = t + 1$. The fifth to eighth columns in Table II show $T_d(e)$ for four critical e values: $e = 0, 1, t$, and $t+1$. The values in the parentheses are again the normalized values per information bit. Note that for a number of errors $e > t$, when an RS code segment is decoded into a wrong codeword due to a decoding error, $T_d(e)$ can be totally different from the saturated value of $T_d(e)$ for $e > t$. However, since the probability P_E of a decoding error is very small compared to that of a decoding failure (i.e., P_F), this effect is not reflected in the average values from many runs. Note that the encoding and decoding times are the same for all the shortened codes (i.e., $(N-s, K-s, q)$ RS codes where $s < K$), since the same encoder and decoder are used for the shortened codes.

According to our error-control schemes described thus far, decoding may be done multiple times depending on the number of times the packet is retransmitted. Now, we can easily see that the decoding complexity will depend on both the error-control scheme and channel error rate.

B. RS-I

For **RS-I**, the average decoding complexity per code segment in terms of the CPU time for a given (N, K, q) RS code and channel BER P_b can be represented by

$$\hat{C}_d^{\mathrm{I}} = E[T_d(E)] = \sum_{e=0}^{N} P_E(e) T_d(e) \tag{26}$$

where

$$P_E(e) = \binom{N}{e} P_s^e (1-P_s)^{N-e} \tag{27}$$

and

$$P_s = 1-(1-P_b)^b. \tag{28}$$

One fact we didn't consider in the above equation is that a packet with uncorrectable errors should be retransmitted, and hence the retransmitted packet should also be decoded again.

Considering retransmissions, the average *decoding complexity* per "successfully transmitted" code segment (or equivalently, packet for **RS-I**) is given by

$$C_d^{\mathrm{I}} = \hat{C}_d^{\mathrm{I}} + P_T \cdot C_d^{\mathrm{I}} \tag{29}$$

and this reduces to

$$C_d^{\mathrm{I}} = \frac{\hat{C}_d^{\mathrm{I}}}{1-P_T}. \tag{30}$$

Finally, the computational complexity per successfully transmitted "information bit" for both encoding and decoding can be represented by

$$C_t^{\mathrm{I}} = \frac{C_e + C_d^{\mathrm{I}}}{Kb}. \tag{31}$$

C. RS-II

Since there are M RS code segments, the average decoding complexity without considering retransmissions is given by

$$\hat{C}_d^{\mathrm{II}}(M) = M\hat{C}_d^{\mathrm{I}} \tag{32}$$

respectively. Then, since a packet is retransmitted with the probability $1-p_{0,0}$, the average decoding complexity per successfully transmitted code segment is given by

$$C_d^{\mathrm{II}} = \frac{\hat{C}_d^{\mathrm{II}}(M)}{p_{0,0}(M)}. \tag{33}$$

Finally, we obtain the average total complexity

$$C_t^{\mathrm{II}} = \frac{MC_e + C_d^{\mathrm{II}}}{MKb}. \tag{34}$$

D. RS-III

For **RS-III**, the decoding complexity can be derived by following a similar step taken for $E[N_{\mathrm{tr}}^{\mathrm{III}}]$ in Section V-C. Then, we can derive the following relationship:

$$\begin{aligned} C_d^{\mathrm{III}} &= \hat{C}_d^{\mathrm{II}}(M) + \sum_{e=1}^{M} p_{e,0}(M) C_d^{\mathrm{III}} \\ &\quad + \sum_{f=1}^{M} p_{0,f}(M)\left(\hat{C}_d^{\mathrm{III}}(f) + B(f)\cdot C_d^{\mathrm{III}}\right) \\ &\quad + \sum_{f=1}^{M-1}\sum_{e=1}^{M-f} p_{e,f}(M)\left(\hat{C}_d^{\mathrm{III}}(f) + C_d^{\mathrm{III}}\right) \end{aligned} \tag{35}$$

where

$$\hat{C}_d^{\mathrm{III}}(f) = \hat{C}_d^{\mathrm{II}}(f) + \sum_{f'=1}^{f}\sum_{e=0}^{f-f'} p_{e,f'}(f)\hat{C}_d^{\mathrm{III}}(f'). \tag{36}$$

Then, this reduces to

$$\begin{aligned} &C_d^{\mathrm{III}} \\ &= \frac{\hat{C}_d^{\mathrm{II}}(M) + \sum_{f=1}^{M}\sum_{e=0}^{M-f} p_{e,f}(M)\hat{C}_d^{\mathrm{III}}(f)}{1-\sum_{f=1}^{M} p_{0,f}(M)B(f) - \sum_{f=0}^{M}\sum_{e=1}^{M-f} p_{e,f}(M)}. \end{aligned} \tag{37}$$

Finally, we obtain

$$C_t^{\mathrm{III}} = \frac{MC_e + C_d^{\mathrm{III}}}{MKb}. \tag{38}$$

We can easily see that the equation of the average decoding complexity (e.g., C_d^{III}) resembles that of the average number of transmitted bits till the successful reception of a packet (e.g., $E[N_{\mathrm{tr}}^{\mathrm{III}}]$) for each error-control scheme. One difference is that the header error probability has nothing to do with the decoding complexity since the packet with uncorrectable errors in the MAC header will be detected by the BCH decoder in most cases, so no RS decoding is needed at all.

VII. Comparative Evaluation and Discussion

In this section, the three error-control schemes discussed so far are comparatively evaluated in terms of throughput efficiency and computational complexity.

A. Comparison of No Coding and RS-I

Fig. 5 plots the throughput efficiency η^{No} as BER increases when no RS coding is used for user data. The η^{No} value of a larger (smaller) MTU is observed to be larger for a smaller (larger) BER. With a larger MTU, PHY, and MAC overheads are relatively small, but η^{No} decreases rapidly as BER increases since the larger user data, the more likely it is corrupted. This

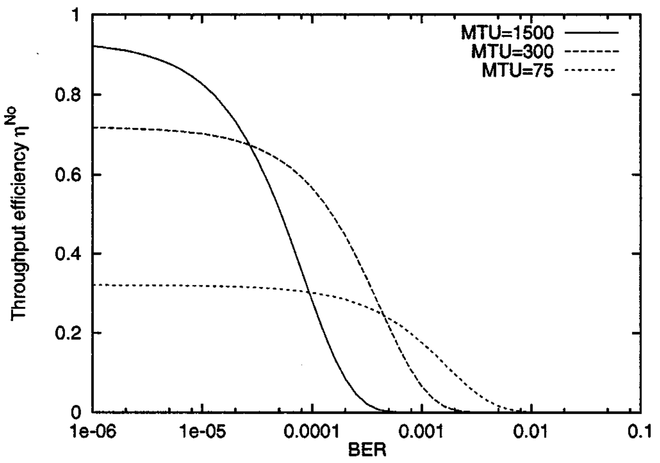


Fig. 5. Throughput efficiency $\eta^{\rm No}$ versus BER without RS coding for user data.

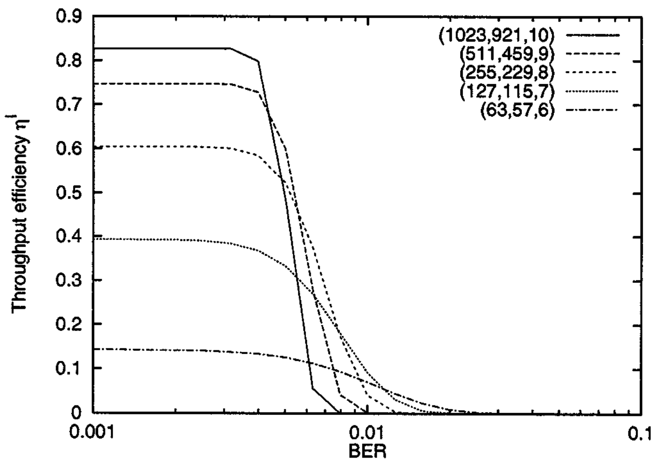
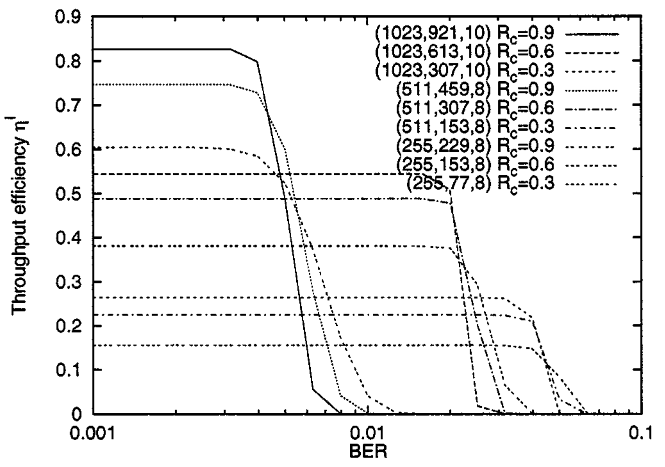
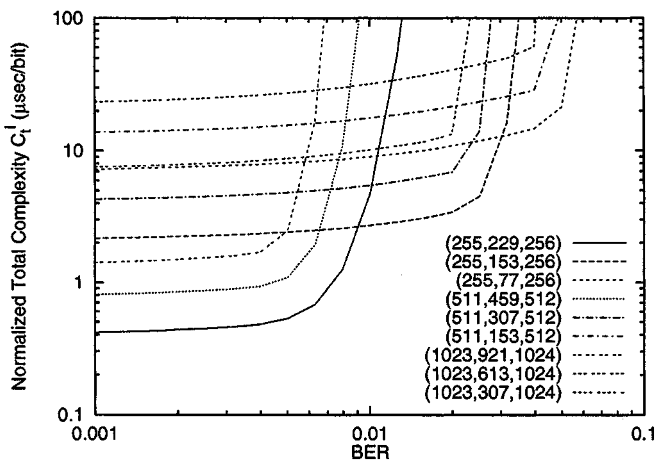


Fig. 6. Throughput efficiency $\eta^{\rm I}$ versus BER with $r_c = 0.9$ RS codes for **RS-I**.



(a)



(b)

Fig. 7. (a) Throughput efficiency $\eta^{\rm I}$ versus BER and (b) computational complexity $C_t^{\rm I}$ versus BER with nine RS codes for **RS-I**.

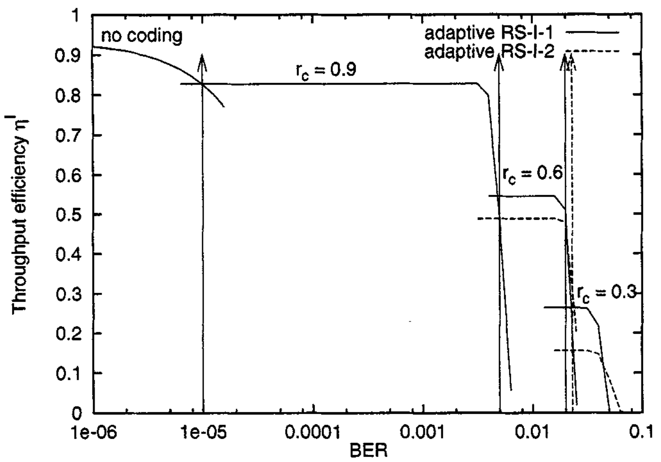
phenomenon was also observed in [11], where an adaptive frame length control for WaveLAN was considered.

Fig. 6 plots the throughput efficiency of **RS-I** for RS codes with the code rate $r_c = 0.9$. Note that as N decreases, the MTU (and, hence, the frame length) gets shorter. The general trend with RS coding is observed to be the same as that with no coding, that is, it is better to use a shorter frame as a channel gets more error-prone. However, thanks to RS coding, the throughput starts to decrease at pretty higher BERs than those with no coding. More importantly, with RS coding, throughputs for all codes drop to zero very rapidly after staying saturated until reaching some threshold BERs. Compared to the case of no coding, it seems very difficult to adapt the codes with the same code rate (i.e., size N or equivalently, the frame length) to the channel state (or BER). The observation on block length of error-control coding here was also described in [2].

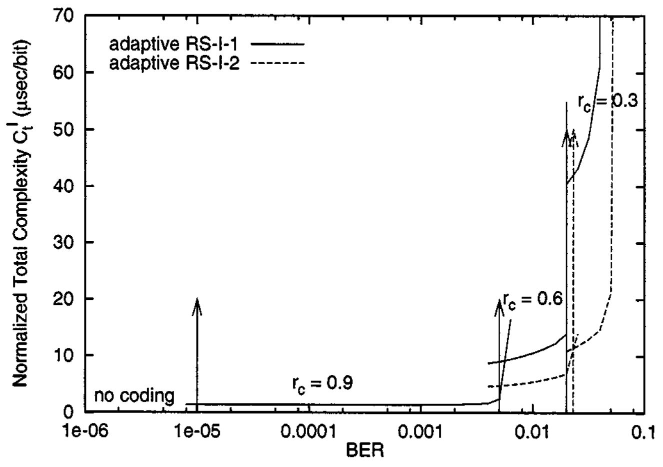
B. Performance of RS-I

Fig. 7 shows throughput efficiencies and computational complexities for nine RS codes with **RS-I**. As shown in Table I, the nine codes can be categorized into three different sizes (i.e., $N = 255, 511, 1023$), or three different code rates (i.e., $r_c = 0.9, 0.6, 0.3$). One can observe that the lower the code rate, the larger the dropping points of throughput and the larger the rising points of computational complexity. We again find that adaptation among different N values with the same code rate is not feasible. From the complexity curve, we find that the complexity increases approximately linearly with N when BER is small. The BER at which the complexity starts to increase is observed to be smaller for a larger N with the same code rate.

Since it is not reasonable to adapt among codes with the same code rate and different N values, the problem is: 1) which code to select for each code rate and 2) how to adapt the code rate based on the channel state. In other words, we choose three RS codes, and then use them adaptively. Depending on the adaptation goal, two strategies are considered. First, when achieving the maximum throughput efficiency for each channel state is the goal, we choose three $N = 1023$ codes for adaptation. The throughput and complexity are plotted in Fig. 8 with the label "adaptive **RS-I-1**." No coding with $MTU = 1500$ is also used when BER is very small. Arrow marks are the adaptation points between two code rates, i.e., BER $= 1 \times 10^{-5}, 5 \times 10^{-3}, 2 \times 10^{-2}$. The adaptation points are determined based on both throughput and complexity, i.e., the smaller of: 1) BER at which the throughput of the higher-rate code starts to get lower

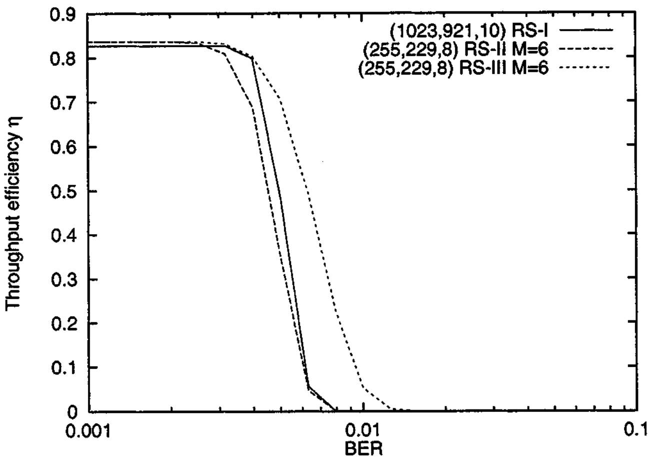


(a)

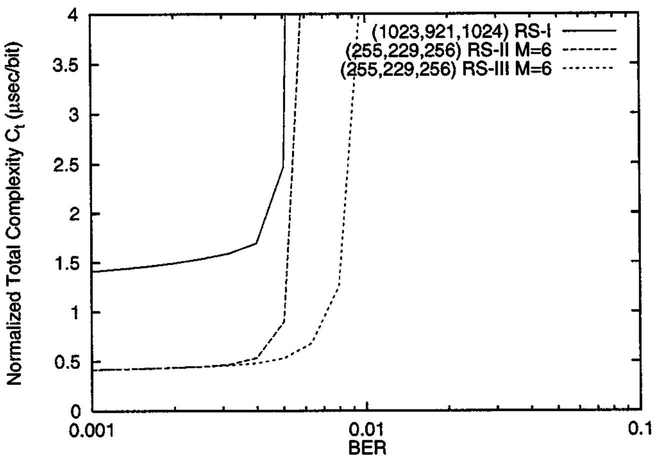


(b)

Fig. 8. (a) Throughput efficiency η^{I} versus BER and (b) computational complexity C_t^{I} versus BER with adaptive coding for **RS-I**.



(a)



(b)

Fig. 9. (a) Throughput efficiency η versus BER and (b) computational complexity C_t for three schemes with $r_c = 0.9$.

than that of the lower-rate code and 2) BER at which the complexity of the higher rate code starts to get higher than that of the lower rate code.

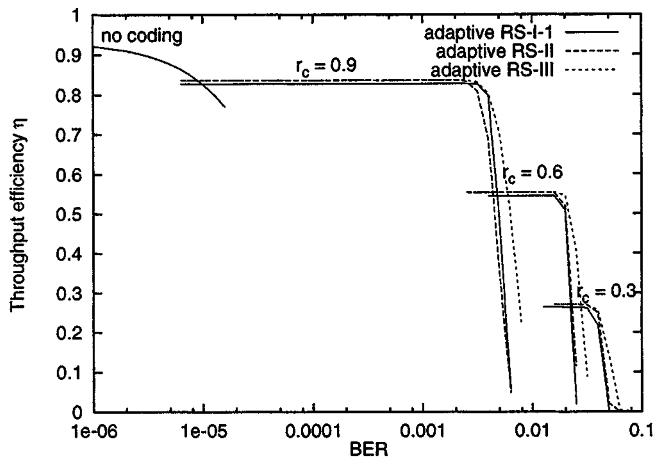
Second, when achieving a reasonably good throughput efficiency with a reasonable computational complexity is the goal, we choose a set of $(1023, 921, 1024)$, $(511\,307\,512)$, and $(255, 77, 256)$ codes for adaptation. The term "reasonable" can be interpreted as "reasonable relative to adaptive **RS-I-1**." The performance of this strategy is also plotted in Fig. 8 with the label 'adaptive **RS-I-2**.' The adaptation points are BER $= 1 \times 10^{-5}, 5 \times 10^{-3}, 2.3 \times 10^{-2}$. One can easily see that adaptive **RS-I-1** outperforms adaptive **RS-I-2** in terms of throughput efficiency, while adaptive **RS-I-2** outperforms adaptive **RS-I-1** in terms of computational complexity. If battery power consumption is not a concern, adaptive **RS-I-1** might be preferable, but for most wireless systems, adaptive **RS-I-2** is expected to be more attractive.

C. Comparison of the Three Schemes

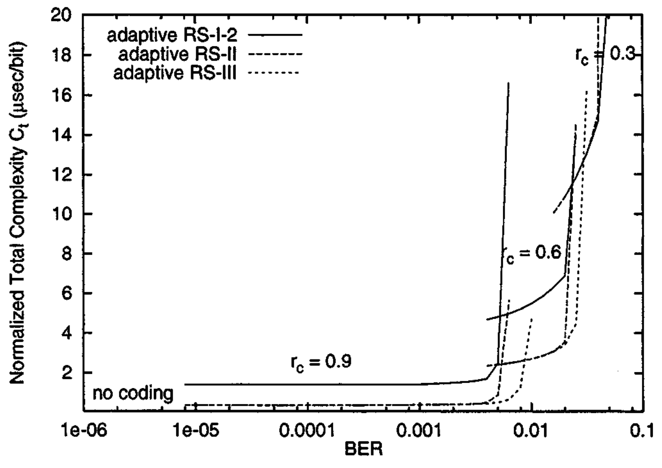
Now, we consider the performance of **RS-II** and **RS-III**, and compare them with **RS-I**. Fig. 9 shows the throughput and complexity of three error-control schemes using the RS codes with $r_c = 0.9$. The number, $M = 6$, of segments with $(255\,229\,256)$ RS codes is used for **RS-II** and **RS-III** since: 1) this corresponds to 1530 bytes for both user data and RS code redundancy, which is roughly the same as the original MTU of WaveLAN and 2) the larger MTU, the better as was with **RS-I**. We first observe that the throughput of **RS-I** lies between those of **RS-II** and **RS-III** in the transition region. On the other hand, when BER is low, the throughput of **RS-I** is a little worse than those of the other two.

The complexity of **RS-I** is about three times worse than those of the others when BER is low and it starts to rise at a lower BER. From the figures, we can easily determine that **RS-III** is the best among the three while it is difficult to determine which of **RS-I** and **RS-II** is better. One interesting finding (not shown in the figure) is that the complexity of **RS-III** is independent of the value of M. This is because with **RS-III**, the segments with uncorrectable errors only need to be retransmitted so, in terms of the encoding/decoding complexity, **RS-III** with an arbitrary number of segments works the same as **RS-I** (which is **RS-III** with $M = 1$) using the same RS code.

Fig. 10 shows the performance of adaptive schemes. For throughput efficiency, we compare adaptive **RS-I-1** with adaptive **RS-II** and **RS-III** while for complexity, adaptive **RS-I-2** is compared with adaptive **RS-II** and **RS-III**. We observe that in terms of throughput efficiency, all three schemes work quite similarly, but in terms of complexity, **RS-I-2** is much

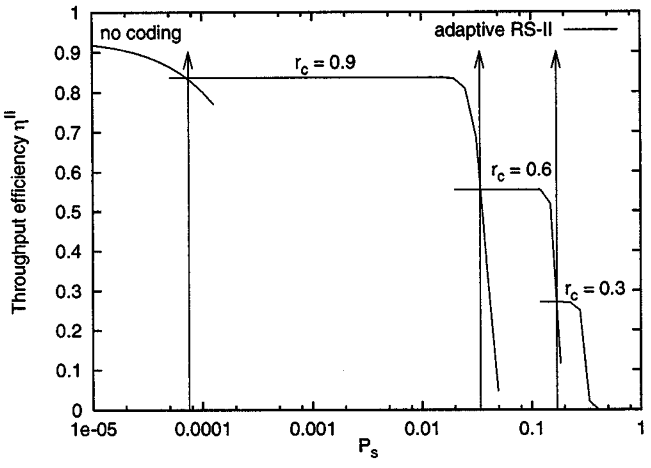


(a)

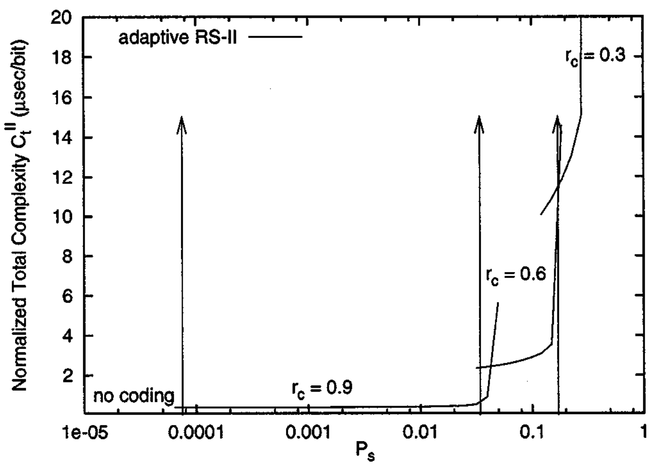


(b)

Fig. 10. (a) Throughput efficiency η versus BER and (b) computational complexity C_t versus BER for three schemes with adaptive coding.



(a)



(b)

Fig. 11. (a) Throughput efficiency η^{II} versus P_s; and (b) computational complexity C_t^{II} versus P_s for adaptive **RS-II**.

worse than the other two. Interesting findings include: 1) the throughput of adaptive **RS-I-1** is worse than the other two except for the region of $r_c = 0.9$ and 2) the complexity of adaptive **RS-III** is hidden by that of **RS-I-2** in the region of $r_c = 0.3$ since the same $(255, 77)$ RS code is used in this region for both schemes. However, recall that the throughput (complexity) of **RS-I-2** (**RS-I-1**) is worse than that of **RS-I-1** (**RS-I-2**). By considering all of these, one can conclude that **RS-I** is worse than the other two. Moreover, **RS-III** is superior to **RS-II** in terms of the throughput and complexity we examined. However, note that **RS-III** requires additional bookkeeping not included in our complexity analysis. That is, selective requests and retransmissions of some segments in a once-formed MAC frame would not be easy nor acceptable. Moreover, the performance of adaptive **RS-II** is not much worse than that of adaptive **RS-III** as can be seen from Fig. 10.

So, **RS-III** might not be an attractive choice in terms of practicality. Considering all these, we choose adaptive **RS-II** for our further study. Fig. 11 shows the throughput and complexity of adaptive **RS-II** as the symbol error probability P_s increases with the adaptation points (or symbol error probabilities at which the code rate would be adapted), which correspond to $P_s = 7.5 \times 10^{-5}, 0.034, 0.17$. The reason why P_s is used instead of BER is that P_s is used as a reference of adaptation as will be discussed in the next section. In the rest of this paper, we consider how this adaptation really works and the performance of adaptive **RS-II** in an environment with time-varying errors will be discussed.

VIII. Adaptation Rule

Here we consider how to adapt the error-control code based on adaptive **RS-II**. First, we define the set of codes for adaptive use as c_0, c_1, c_2, c_3, where c_0 is without RS coding, c_1 with $(255\,229\,256)$ code, c_2 with $(255\,153\,256)$ code and c_3 with $(255, 77, 256)$ code. According to Fig. 11, adaptation points (or symbol error probabilities at which the code rate would be adapted) are determined by $P_s = 7.5 \times 10^{-5}, 0.034, 0.17$. We use a modified set of the adaptation points as $P_s^{0,1} = 6.5 \times 10^{-5}$, $P_s^{1,2} = 0.025$, and $P_s^{2,3} = 0.15$. The reasons for choosing values lower than the original ones include: 1) adaptation at a higher P_s than the original adaptation point can result in a very low throughput while adaptation at a lower P_s would not and 2) an adaptation rule based on the original probabilities might result in too frequent adaptations (due to the first reason).

To develop an adaptation rule, we need to divide time-varying error patterns into two categories. One is the long-term variation in which error characteristics vary over several seconds to ten seconds. A time-varying distance between the sender and receiver, and shadowing can cause this type of variation. The other is the short-term variation in which error characteristics vary over several milliseconds or less. The multipath fading environment resulting from multipath propagation of the transmitted signal and the mobile's relative movement can cause this type of variation.

Our error-control code adaptation is based on the channel state estimation, which is done using the decoding results at the receiver. That is, the receiver determines the desired code rate based on the decoding result of a received packet, then informs it to the sender through the ACK/NAK of each received packet. Because our adaptation is not based on any prediction but is based on the observation and feedback, it might not be able to handle the short-term variation of the channel condition well depending on the actual channel behavior. However, it is effective for long-term variations. The code currently set at the receiver is denoted by c_{curr}.

A. Increase of Code Rate

When a packet is successfully received (i.e., without any decoding failure), the receiver considers if it is time to increase the code rate. Based on the currently set code c_{curr} at the receiver, the code rate is increased adaptively as follows.

- When $c_{\text{curr}} = c_i$ for $i = 2, 3$, the code is adapted to c_{i-1} if the number e_M of symbol errors out of $N_{M,L}$ RS symbols in the last L frames is smaller than $P_s^{(i-1),i} N_{M,L}$ for the smallest L such that $P_s^{(i-1),i} N_{M,L} > 1$.
- When $c_{\text{curr}} = c_1$, the code is adapted to c_0 if there was only a single or no symbol error during the last L frames received for the smallest L such that $P_s^{0,1} \cdot N_{M,L} > 1$, where $N_{M,L}$ is the total number of RS symbols in the last L frames.

Note that the number of symbol errors in a RS code segment is readily available from the decoder unless the decoding fails.

B. Decrease of Code Rate

When a packet needs to be retransmitted due to a CRC error for $c_{\text{curr}} \neq c_0$, the code rate is not adapted in order to defer the adaptation decision until the next packet is received since a CRC error is very rare and will not usually happen over consecutive packets. On the other hand, when $c_{\text{curr}} = c_i$ for $i = 1, 2$ and a packet needs to be retransmitted due to RS decoding failures, the code rate is decreased by one step, i.e., c_i to c_{i+1} for $i = 1, 2$, if any of the following conditions occurs.

- When all RS segments of a packet result in decoding failures.
- When more than a half of all RS segments in the last two packets result in decoding failures.
- When three of the last ten received packets result in decoding failures.

The above adaptation rule for code-rate decrease may appear very *ad hoc* but, with the third condition, we try to limit the retransmission probability P_R^{II} under 0.3 since a similar or better throughput efficiency can otherwise be achieved with the next lower rate code from Fig. 11. The first and second conditions, on the other hand, render prompt adaptation for an abrupt change of the channel condition.

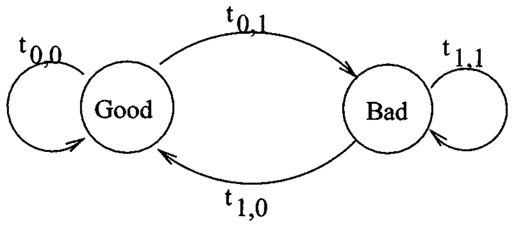


Fig. 12. Two-state Markov chain model for the channel.

Now, when $c_{\text{curr}} = c_0$, the code is adapted to c_1 if there were more than one CRC error detection in the last ten received packets in order to keep the retransmission probability P_R^{No} under 0.2. When the code rate is adapted to decrease, and it is known to the sender, the sender encodes packets with the adapted code rate. The packets which need to be retransmitted are also reencoded. Note that, for example, 229 symbols encoded by $(255\,229\,256)$ RS code cannot be reencoded by $(255\,153\,256)$ RS code. For a packet encoded by code c_i to be reencoded by code c_{i+1}, the user data is divided first into two equal-sized packets (i.e., IP fragmentation) if the user data cannot be accommodated in c_{i+1}, then each encoded by c_{i+1}.

Finally, the code rate can also be decreased by the sender's decision. That is, upon expiration of the retransmission time out, the code is set to c_3. The rationale behind this adaptation is that a time-out happens due to the corruption of the MAC header or the loss of ACK/NAK while both the header and ACK/NAK are protected by very strong codes, thus implying that the channel is very bad.

IX. Performance of Adaptive RS-II

This section presents the simulation results of adaptive **RS-II** in a fading environment with time-varying channel states. We first describe the wireless network used for our simulation.

A. Link Model

We use a very simple wireless network environment to evaluate the proposed adaptive error-control scheme. There is only one sender and one receiver in the network, where the sender is assumed to have an infinite amount of information to send. So, all the packets are transmitted in a full-length MAC frame composed of six RS code segments. Immediately after receiving/decoding a packet, the receiver sends an ACK/NAK for the packet. Upon receiving the ACK/NAK, the sender determines whether to transmit a new packet or retransmit the previously sent packet. When the retransmission time out happens consecutively, the retransmissions are backed off exponentially. That is, for $2, 3, 4, \ldots$ consecutive time outs, the retransmission is deferred for the time of $1, 2, 4, \ldots$ full-length packet transmissions, because consecutive time-out expirations implies that the channel is too bad for a packet to go through successfully.

B. Channel Model

The channel is modeled by a two-state Markov chain, which is widely accepted to model the multipath fading channel, as shown in Fig. 12. The channel state is either *good* or *bad* and can change on bit boundaries, that is, the channel condition stays in a state during one bit duration. Following the widely accepted Rayleigh fading model, which corresponds to the case of no line-of-sight path between the sender and receiver, we can derive the transition probabilities $t_{0,1}$ and $t_{1,0}$ as follows.

First, the author of [16] provides the *level-crossing rate*, defined as the expected rate at which the Rayleigh fading envelope, normalized to the local root mean square (rms) signal level $R_{\rm rms}$, crosses a threshold level R in a positive-going direction

$$N_R = \sqrt{2\pi} f_m \rho e^{-\rho^2} \tag{39}$$

where the maximum Doppler frequency is given by

$$f_m = \frac{v}{\lambda} \tag{40}$$

for the mobile speed v and the wavelength λ of the carrier and the normalized threshold fading envelope is given by

$$\rho = \frac{R}{R_{\rm rms}}. \tag{41}$$

Next, the *average fade duration*, which is defined as the average period of time for which the received signal is below the threshold level R, is given by [16]

$$T_f = \frac{e^{\rho^2} - 1}{\rho f_m \sqrt{2\pi}}. \tag{42}$$

Using the above formulas and assuming steady-state conditions, the probabilities μ_0 and μ_1 that the channel is in good and bad states, respectively, are given by [10]

$$\mu_0 = \frac{1/N_R - T_f}{1/N_R}, \quad \text{and} \quad \mu_1 = \frac{T_f}{1/N_R}. \tag{43}$$

Finally, the state transition probabilities can be approximated by [10]

$$t_{0,1} = \frac{N_R}{R_t^0}, \quad \text{and} \quad t_{1,0} = \frac{N_R}{R_t^1} \tag{44}$$

where $R_t^k = R_t \mu_k$, and R_t is the symbol transmission rate. Now, with the transmission rate $R_t = 1$ Mbps, the mobile speed $v = 2$ km/h (i.e., a pedestrian speed), the carrier frequency $f = 900$ MHz (so, $\lambda = c/f = 1/3$ m), and the normalized threshold fading envelope $\rho = 0.3$, we obtain

$$\mu_0 = 0.914, \quad \mu_1 = 0.0861 \tag{45}$$

and

$$t_{0,1} = 1.253 \times 10^{-6}, \quad t_{0,0} = 1 - t_{0,1} \tag{46}$$

$$t_{1,0} = 1.331 \times 10^{-5}, \quad t_{1,1} = 1 - t_{1,0}. \tag{47}$$

Now, assuming the binary phase shift keying (BPSK) modulation, the BERs $P_{b,0}$ and $P_{b,1}$ when the channel is in *good* and *bad* states, respectively, can be calculated as [9]

$$P_{b,i} = \int_{\gamma_i}^{\gamma_{i-1}} P_{b\,|\,\gamma} f_i(\gamma)\, d\gamma \tag{48}$$

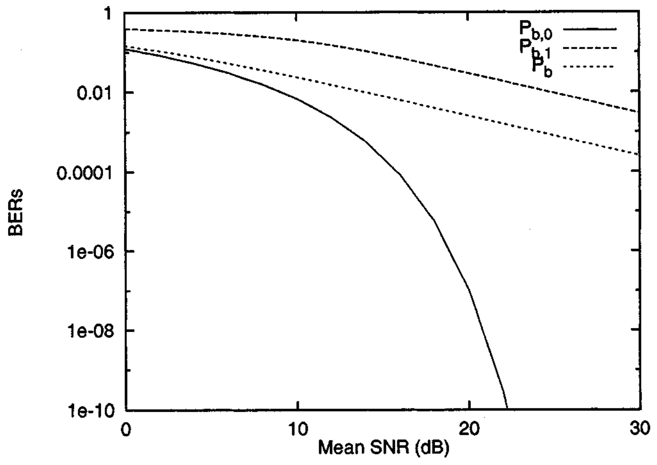


Fig. 13. BERs versus mean SNR for the Rayleigh fading channel.

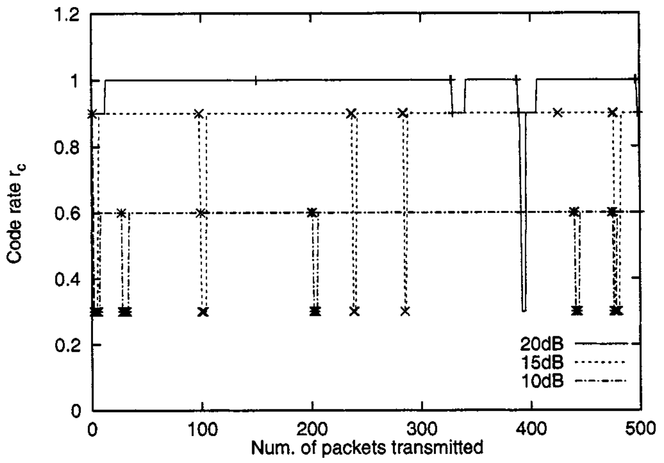


Fig. 14. Code rate versus number of packets transmitted.

where the BER for a given signal-to-noise ratio (SNR) γ is

$$P_{b|\gamma} = Q(\sqrt{2\gamma}) \tag{49}$$

the conditional distribution of the instantaneous SNR γ in a given state with the mean SNR $\bar{\gamma}$ is

$$f_i(\gamma) = \frac{\frac{1}{\bar{\gamma}} e^{-\gamma/\bar{\gamma}}}{e^{-\gamma_i/\bar{\gamma}} - e^{-\gamma_{i-1}/\bar{\gamma}}} \tag{50}$$

for $\gamma_i < \gamma < \gamma_{i-1}$ and the set $\{\gamma_{-1}, \gamma_0, \gamma_1\} = \{\infty, \rho^2\bar{\gamma}, 0\}$. Note that the mean SNR $\bar{\gamma}$ depends on the transmitted power, signal attenuation over the channel, and others. Fig. 13 shows the BERs for two states as the mean SNR $\bar{\gamma}$ increases for $\rho = 0.3$. The BER P_b without considering states, which is obtained by setting $\gamma_i = 0$ and $\gamma_{i-1} = \infty$ in (48), is also plotted as a comparative reference.

C. Simulation Results

Fig. 14 shows how the code rate is adapted for each packet transmitted for three different mean SNRs. The dots on the curves represent non-ACKed packets including the cases of NAK, header error, and loss of ACK/NAK. The code rate, at which the system mostly stays, is observed to vary with the mean

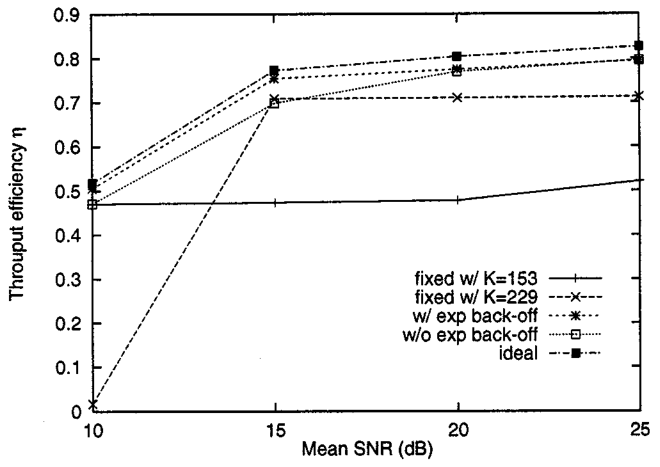


Fig. 15. Throughput efficiency η versus mean SNR.

SNR. One can easily imagine that the moments of code-rate drops correspond to the bad channel state. Note that the time difference between two consecutive packet transmissions is not even due to the retransmission backoff for consecutive time outs.

Fig. 15 shows the throughput efficiencies as the mean SNR increases for five different cases of adaptive **RS-II**: 1) two marked with "fixed w/K" are the nonadaptive versions of **RS-II** with $(255, K, 256)$ codes; 2) marked with "$w/\exp$ backoff" is the adaptive **RS-II** we are considering; 3) marked with "w/o exp backoff" is the adaptive **RS-II** without exponential back-off; and 4) marked with "ideal" is an ideal version of **RS-II**, in which the code rate is determined after seeing errors in the received packet; that is, if these errors are uncorrectable even with $r_c = 0.3$ code, the packet is assumed not to be transmitted at all (i.e., the transmission is deferred), while the code which can correct all these errors with the maximum code rate is selected otherwise. Basically, there is no retransmission in the ideal version. This ideal and unrealistic version can be used as a reference (or an upper bound) of the throughput efficiency.

We observe that the throughput of $K = 153$ fixed one is determined at around 0.5 irrespective of the mean SNR while that of $K = 229$ fixed one is almost zero for mean SNR 10 and also saturated at around 0.7 for mean SNRs larger than 15. Adaptive schemes outperform the nonadaptive ones. The scheme without backoff is worse than our scheme (with backoff) as it should be. However, it gets closer to our scheme as the mean SNR increases since the chance of consecutive time outs gets smaller as the mean SNR increases. We observe that our scheme works pretty close to the ideal version throughout the whole range of the mean SNRs, implying that the code rate is adapted adequately with our scheme as the channel state varies. While the actual performance may vary depending on the underlying channel behavior, the result suggests that the adaptive error control such as the one proposed in this paper can be effective in the real world.

X. Related Work

Error control coding is well-established and widely used for many applications ranging from wireless communications to storage systems [2], [12], [14]. In this paper, we considered how to use error-control coding adaptively by adopting well-known RS codes rather than developing new error-control codes.

The authors of [10] argued for energy efficiency as the most important factor in error-control schemes, and suggested adaptive use of an ARQ or an FEC-ARQ hybrid, depending on the channel condition. However, by 1) considering the scheme of one packet encoded by one RS code like **RS-I** and 2) considering energy efficiency only (and ignoring throughput), they concluded that the FEC-ARQ hybrid with RS codes is so expensive that ARQ without RS coding is preferred for IP packet transmissions even though the throughput of ARQ could be much worse under a poor channel condition. Considering the fact that a wireless link is shared by many users, we should not ignore the throughput performance of any error-control scheme.

Our computation complexity is closely related to energy efficiency; these two would be linearly dependent on each other where energy efficiency encompasses energy consumption for encoding, packet transmission, and decoding. For example, the encoding complexity will be linearly dependent on the energy consumption by encoding, and the decoding complexity will be approximately linearly dependent on the energy consumption by both transmission and decoding. So, our adaptation policy or comparison among the three different error-control schemes won't change even if we consider energy efficiency instead of computation complexity.

The authors of [6] studied adaptive usage of error correcting codes for real-time traffic. With their scheme, one out of two FEC codes and deferment are chosen for packet transmission depending on the estimated channel condition and the required packet-delay bound to minimize the number of data bit transmissions for a given real-time packet. Their scheme is based on FEC only, not ARQ, but has also adopted an RS code for each packet like **RS-I**.

The authors of [5] claimed that the link-layer error control can be very effective for end-to-end transport-layer performance, and evaluated their adaptive error correction for a wireless LAN using measured error traces. A packet under their scheme is composed of a number of RS codes, and the error-control scheme works similarly to **RS-II**. They didn't consider the choice of **RS-I**, which has been investigated by most others. They presented some heuristic coding and packet-length control algorithms based on the observed packet errors. Basically, their adaptation is not geared toward any performance optimization, but is based on a trial-and-error strategy.

The author of [7] proposed a hybrid-II ARQ scheme using concatenated RS/convolutional codes for wireless ATM networks. Hybrid-II ARQ is a different class of ARQ, in which incremental redundancies are added in the transmission in case of uncorrectable errors in the packet received. In this scheme, the redundancy of the convolutional code (instead of the RS code) is incrementally adapted. The authors of [3] attempted to use different FEC, ARQ, and modulation schemes adaptively according to the channel condition in order to maximize the link throughput. They drew a conclusion that FEC is not needed by assuming good channel conditions for most of time, which is not realistic. Moreover, they did not address practical issues, such as how to adapt among the different schemes.

XI. Conclusion

In this paper, we proposed and evaluated adaptive error-control schemes in the data link layer for reliable communication over wireless links which tend to suffer location-dependent, time-varying, and bursty errors. Three error-control schemes were considered according to: 1) the number of RS code segments in a packet and 2) how a lost packet is retransmitted. Through throughput-performance and computational-complexity analyses, these three schemes were compared. While **RS-I**-like schemes (i.e., a packet encoded by a code) have been investigated by many others, **RS-II** was found to be the most attractive in terms of computational complexity and practicality even though its throughput performance is not the best. We then addressed the problem of choosing the error-control code adaptively based on the estimated channel state. Via simulations in the environment of Rayleigh fading, the adaptive error-control scheme is shown to work well.

References

[1] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison Wesley, 1984.

[2] H. O. Burton and D. D. Sullivan, "Errors and error control," *Proc. IEEE*, vol. 60, pp. 1293–1301, Nov. 1972.

[3] F. Cameron, M. Zukerman, and M. Gitlits, "Adaptive transmission parameters optimization in wireless multi-access communication," *Proc. IEEE Int. Conf. Networks (ICON'99)*, Sept. 1999.

[4] D. Eckhardt and P. Steenkiste, "Measurement and analysis of the error characteristics of an in-building wireless network," in *Proc. ACM SIGCOMM'96*, 1996, pp. 243–254.

[5] D. A. Eckhardt and P. Steenkiste, "Improving wireless LAN performance via adaptive local error control," *Proc. IEEE ICNP'98*, pp. 327–338, 1998.

[6] M. Elaoud and P. Ramanathan, "Adaptive use of error-correcting codes for real-time communication in wireless networks," *Proc. IEEE INFOCOM'98*, pp. 548–555, 1998.

[7] I. Joe, "An adaptive hybrid ARQ scheme with concatenated FEC codes for wireless ATM," in *Proc. ACM/IEEE MobiCom'97*, 1997, pp. 131–138.

[8] P. Karn, Digital signal processing for ham radio, in http://people.qualcomm.com/karn/dsp.html.

[9] Y. H. Lee and S. W. Kim, "Adaptive data transmission scheme for DS/SSMA system in bandlimited Rayleigh fading channel," *Proc. IEEE ICUPC'95*, pp. 251–255, 1995.

[10] P. Lettieri, C. Fragouli, and M. B. Srivastava, "Low power error control for wireless links," in *Proc. ACM/IEEE MobiCom'97*, 1997, pp. 139–150.

[11] P. Lettieri and M. B. Srivastava, "Adaptive frame length control for improving wireless link throughput, range, and energy efficiency," *Proc. IEEE INFOCOM'98*, pp. 564–571, 1998.

[12] H. Liu, H. Ma, M. El Zarki, and S. Gupta, "Error control schemes for networks: An overview," *ACM Mobile Networking and Applications (MONET)*, vol. 2, no. 2, pp. 167–182, 1997.

[13] R. J. McEliece and L. Swanson, "On the decoder error probability for Reed–Solomon codes," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 701–703, Sept. 1986.

[14] A. M. Michelson and A. H. Levesque, *Error-Control Techniques For Digital Communication*. New York: Wiley, 1985.

[15] K. Pahlavan and A. H. Levesque, *Wireless Information Networks*. New York: Wiley, 1995.

[16] T. S. Rappaport, *Wireless Communications: Principle and Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Sunghyun Choi (S'96–M'00) received the B.S. (*summa cum laude*) and M.S. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Seoul, in 1992 and 1994, respectively, and the Ph.D. degree from the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, in 1999.

He is a Senior Member Research Staff at Philips Research, Briarcliff Manor, NY. His research interests are in the area of wireless/mobile networks with an emphasis on the QoS guarantee and adaptation, wireless LAN/PAN, in-home multimedia networks, connection and mobility management, and multimedia code division multiple access (CDMA).

Dr. Choi was a recipient of the Korea Foundation for Advanced Studies Scholarship and the Korean Government Overseas Scholarship from 1997–1999 and 1994–1997, respectively. He also received the Humantech Thesis Prize from Samsung Electronics in 1997.

Kang G. Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and the M.S. and Ph.D degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

He is currently a Professor and the Director of the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. He was an Area Editor of *International Journal of Time-Critical Computing Systems*. He has authored and coauthored approximately 600 technical papers and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. He has also coauthored (jointly with C. M. Krishna) *Real-Time Systems* (New York: McGraw Hill, 1997). His current research focuses on quality of service (QoS) sensitive computing and networking with an emphasis on timeliness and dependability. He has also been applying the basic research results to telecommunication and multimedia systems, intelligent transportation systems, embedded systems, and manufacturing applications.

Dr. Shin was a Distinguished Visitor of the Computer Society of the IEEE, an Editor of IEEE Transactions on Parallel and Distributed Computing.