Performance and Energy Consumption Analysis of 802.11 with FEC Codes over Wireless Sensor Networks

Jong-Suk Ahn, Jong-Hyuk Yoon, and Kang-Woo Lee

Abstract: **This paper expands an analytical performance model of 802.11 to accurately estimate throughput and energy demand of 802.11-based wireless sensor network (WSN) when sensor nodes employ Reed-Solomon (RS) codes, one of block forward error correction (FEC) techniques. This model evaluates these two metrics as a function of the channel bit error rate (BER) and the RS symbol size. Since the basic recovery unit of RS codes is a symbol not a bit, the symbol size affects the WSN performance even if each packet carries the same amount of FEC check bits. The larger size is more effective to recover long-lasting error bursts although it increases the computational complexity of encoding and decoding RS codes. For applying the extended model to WSNs, this paper collects traffic traces from a WSN consisting of two TIP50CM sensor nodes and measures its energy consumption for processing RS codes. Based on traces, it approximates WSN channels with Gilbert models. The computational analyses confirm that the adoption of RS codes in 802.11 significantly improves its throughput and energy efficiency of WSNs with a high BER. They also predict that the choice of an appropriate RS symbol size causes a lot of difference in throughput and power waste over short-term durations while the symbol size rarely affects the long-term average of these metrics.**

Index Terms: **802.11, forward error correction (FEC) algorithm, performance analysis, Reed-Solomon (RS), wireless sensor networks**

I. INTRODUCTION

Recently, wireless sensor network (WSN) researchers have proposed various power-conserving and throughput-efficient algorithms [1], [2] to lengthen sensor nodes' lifetime to overcome the difficulty of replacing their battery. Among them, forward error correction (FEC) techniques have been actively employed over WSNs to avoid retransmissions which are quite expensive in terms of energy and throughput metrics [3–5]. Since long-lasting error bursts are predominant in WSNs, the retransmitted packets are likely to be re-corrupted. Note that WSN channels are noisy due to their low transmission power, random deployments of sensor nodes ignoring the underlying geographical constraints, and moving intermediate obstacles [6].

The performance of FEC codes, however, can widely fluctuate depending on the types of FEC algorithms and the amount of FEC check bits to employ. These FEC attributes should be appropriately determined according to the underlying WSN channel characteristics such as the average bit error rate (BER) and the degree of error burstiness. Note that FEC check bits mean redundant data to deduce correct data when data are stale.

Many researches [3–5] have been conducted to measure this FEC performance variation as a function of the WSN channel characteristics, especially when WSNs adopt 802.11 protocols [7] with block FEC codes in their MAC (medium access control) layer. Even though the MAC-layer FEC codes may not be appropriate at 802.11a [8] and 802.11g [9] which have already employed FEC codes at their physical layer such as Viterbi, other 802.11 protocols need to employ them over noisy channels.

The block FEC algorithm, especially, is preferred at the MAC layer of WSNs to other types of FEC algorithms such as the convolutional codes since it more effectively deals with bursty errors. It corrects more tainted bits once all error bits are contained in few symbols since they recover corrupt bits symbol-by-symbol not bit-by-bit. The block FEC algorithm can arbitrarily choose the size of an FEC symbol which is an atomic recovery unit regardless of the number of corrupt bits inside the symbol.

One of these FEC studies [3] proposed a closed joint equation of throughput and power demand of 802.11-based wireless LANs and compared these metrics when 802.11 employed either Reed-Solomon (RS) codes or convolutional codes. Other relevant research [4], [5] measured the performances of their dynamic algorithms to adjust the check-bit amount of RS codes according to the wireless channel state.

As in [3], this paper proposes an extended performance model [10], [11] to estimate throughput and power need of 802.11-based WSNs with RS algorithm. Differently from [3], however, it aims at investigating the performance reliance on the RS symbol size, not the amount of check bits since the RS symbol size significantly impacts the packet correction rate and the time complexity of processing FEC codes.

When the amount of RS check bits to be allocated is fixed, the maximum number of correctable symbols varies according to the symbol size. When a packet consists of n-bit payload and t-bit check-bit with s-bit RS symbol, for example, RS algorithm can recover $t/(2s)$ error symbols at maximum. When s is large and error bits tend to be evenly distributed over several symbols, it has less chance to recover corrupt packets. The large s also requires more energy for decoding operations since their complexity is proportional to the square of the symbol size [12].

For applying our analytical performance model to real WSNs, this paper abstracts real WSN traffic traces with Gilbert models [13]. It differs from [11] in that [11] adopted an analytical channel model, which is inappropriate for representing the WSN burstiness. Based on traffic traces measured from a WSN with two TIP50CM sensor nodes, we build WSN Gilbert model by

Manuscript received August 16, 2006; approved for publication by P. Takis Mathiopoulos, Division II Editor, May 13, 2007.

J.-S. Ahn and J.-H. Yoon are with Computer Engineering Dept., Dongguk University, Jung-Gu Pil-Dong 3-Ga 26 Seoul, Korea, phone: (+82)-2-2260-3811, email: {jahn, ronaldo}@dgu.edu.

K.-W. Lee is with Computer and Communication Engineering Dept., Dongguk University, Jung-Gu Pil-Dong 3-Ga 26 Seoul, Korea, phone: (+82)-2-2260-3843, email: klee@dgu.edu.

This research was supported by Seoul Future Contents Convergence (SFCC) Cluster established by Seoul R&BD Program and the Dongguk University Research Fund of 2006 (DRIMS 2006-2004-0).

approximating the run length complementary cumulative distribution function (CCDF) of Gilbert model to that of real traces. For computing energy demand, we actually measure energy dissipation of encoding and decoding various-sized FEC codes on TIP50CM node and use theoretical values [14] for delivering packets.

The analytical computations based on real 3-hour WSN packet traces and actual power consumption measurements foretell that 802.11 with RS codes significantly outperforms the legacy 802.11 in throughput efficiency and energy consumption. They also indicate that the FEC symbol size seldom influences the two metrics when they are averaged over 3-hour long-term intervals. The FEC size, however, severely affects throughput efficiency and energy consumption by up to 24 % and 740 % respectively over 10-minute short periods. This result implies that if we dynamically adjust the FEC symbol size to the underlying channel status as like the amount of check bits [3], [5], the performance of 802.11-based WSNs would significantly increase.

This paper is organized as follows. Section II illustrates 802.11 payload formats when 802.11 adopts RS codes with various symbol sizes and Section III introduces two extended 802.11 performance models for throughput and power need respectively to embrace the effect of WSN channel errors and RS codes. Section IV evaluates the two performance models over WSN channels abstracted with Gilbert model by applying various FEC symbol sizes. Section V finally summarizes this paper's contributions and lists future research items.

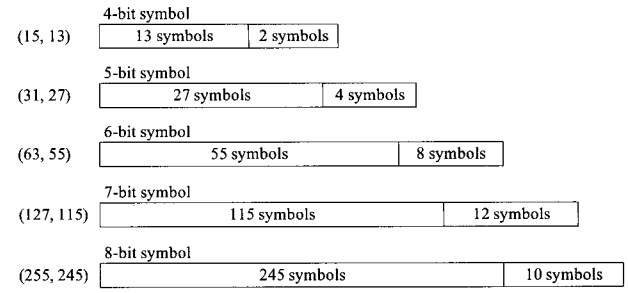


Fig. 1. Codeword configurations for five different FEC symbol sizes.

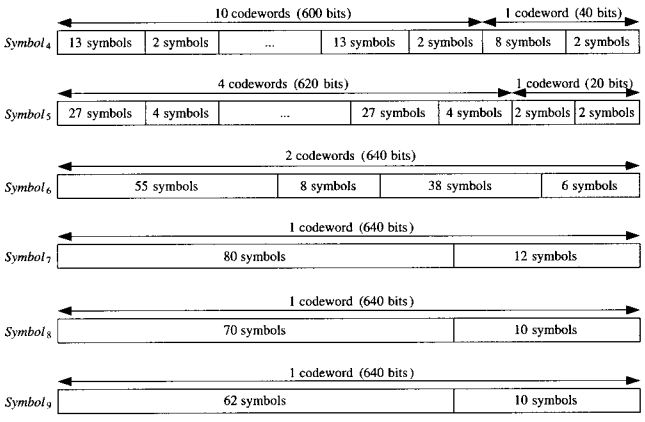


Fig. 2. Packet configurations for six different FEC symbol sizes.

II. 802.11 PACKET FORMATS WITH DIFFERENT FEC SYMBOL SIZES

This section introduces a hypothetical 802.11 payload format to organize the payload with various RS symbol sizes. We assume that the legacy 802.11 will add some FEC-related header fields to indicate what format it adopts since it currently specifies no payload structure in terms of FEC codes. The reason why different RS symbol sizes require different payload formats is that the maximum length of data to cover with a given symbol size called a codeword is fixed. When the symbol size is s bits, for example, the maximum size n of its codeword that it builds is $(2^s - 1)$ symbols or $(2^s - 1)s$ bits. The codeword consisting of a pair of given data to protect and their corresponding FEC check bits is represented as (n, u) where u is the size of user data. The FEC code of $(n - u)$ symbols corrects as many as $(n - u)/2$ error symbols regardless of whether the corrupt part is either data or FEC code since RS algorithm needs two FEC symbols to restore one error symbol. Fig. 1 shows five exemplary codeword configurations when the symbol size is 4, 5, 6, 7, 8 bits respectively.

Fig. 2 illuminates six possible packet configurations to partition the fixed-size user data into some number of codewords when the symbol is 4, 5, 6, 7, 8, 9 bits. Note that $Symbol_i$ in Fig. 2 denotes i-bit FEC symbol size. In these packets, FEC code represented by gray boxes and user payload depicted by white boxes are adjusted to around 11 bytes and 69 bytes respectively. Since codeword sizes vary depending on the FEC symbol size, a packet contains different number of codewords when the FEC symbol size changes. When s is small, a packet would be larger than one codeword so that the packet concatenates several codewords to carry a given amount of user data as in $Symbol_4$, $Symbol_5$, and $Symbol_6$ of Fig. 2.

When one codeword is larger than the packet as in $Symbol_7$, $Symbol_8$, and $Symbol_9$, the packet's FEC code is computed by stuffing as many null data as the difference of the two sizes. Since null data are not actually transmitted, the FEC decoder at the receiver verifies the transferred FEC code after padding the same amount of null data. In $Symbol_7$, for example, it attaches null data of 35 symbols $(= ((27 - 1) - 80 - 12))$ before confirming the 12-symbol FEC code that arrived at the receiver. Note also that since the packet size, 80 bytes in this example, is not the multiple of codewords of every symbol size, the total amount of check bits and user data in Fig. 2 is not the same. The check-bit size of $Symbol_8$ is 80 bits while that of $Symbol_7$ is 84 bits. We believe that this minor difference would rarely affect the validness of our performance and energy comparison of 802.11 with different RS symbol sizes.

III. AN EXTENDED PERFORMANCE MODEL

This section is divided into two subsections such as throughput model and energy model for 802.11 with RS codes.

A. Throughput Model of 802.11 with RS Codes

This section extends an analytical model of 802.11 throughputs in [11] to include the effect of employing RS codes over WSN channels. The original 802.11 model [10] evaluates the contention window operated by BEB (Binary Exponential Back-

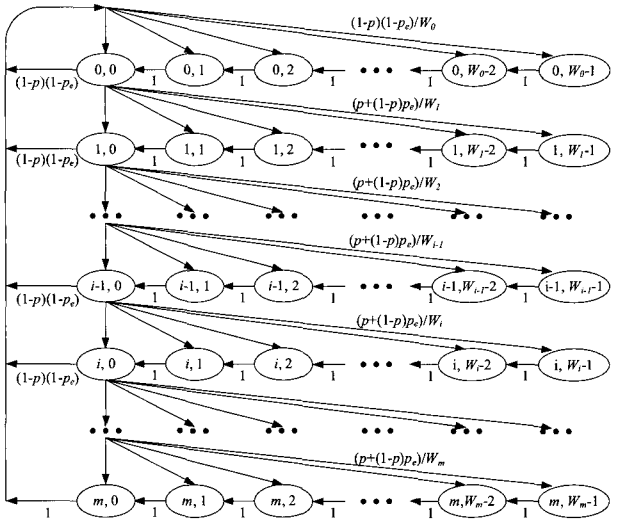


Fig. 3. Modified Markov chain model of back-off window size.

off) algorithm with a multi-state Markov chain shown in Fig. 3 where each circle labeled with (i, j) represents a state that previously experienced i collisions and needs to wait j time slots before transmission. The steady-state probability that the 802.11 contention window stays in each circle is denoted as $b_{i,j}$ in (1).

For accommodating packet errors ignored in [11] although they frequently occur over noisy WSN channels, we replace $(1-p)$ with $(1-p)(1-p_e)$ on the left-hand side in Fig. 3. This substitution is made to model the operation that the contention window resets only when corruption should not happen in addition to collision. Note that p, p_e, and τ stand for the collision probability of a transmitted packet, the packet error probability and the probability to send a packet at a given time slot respectively. Based on this modified Markov model, (1)–(3) compute both p and τ where m, m', and W specify the maximum number of allowable retransmissions, the maximum number of contention window's back-offs, and the minimum contention window size respectively. Please refer to [10], [11] for more details of the derivation of (1)–(3).

$$\tau = \sum_{m}^{i=0} b_{i,0} = \frac{1-(p+(1-p)p_e)^{m+1}}{1-(p+(1-p)p_e)} b_{0,0} \tag{1}$$

where

$$b_{0,0} = \begin{cases} \frac{2\{1-2(p+(1-p)p_e)\}\{1-p(p+(1-p)p_e)\}}{Z}, & \text{when } m \le m' \\ \frac{2\{1-2(p+(1-p)p_e)\}\{1-p(p+(1-p)p_e)\}}{Z'}, & \text{when } m > m' \end{cases} \tag{2}$$

for

$$\begin{aligned} Z = {} & W\{1-(2(p+(1-p)p_e))^{m+1}\}\{1-(p+(1-p)p_e)\} \\ & +\{1-2(p+(1-p)p_e)\}\{1-(p+(1-p)p_e)^{m+1}\}, \end{aligned}$$

$$\begin{aligned} Z' = {} & W\{1-(2(p+(1-p)p_e))^{m+1}\}\{1-(p+(1-p)p_e)\} \\ & +\{1-2(p+(1-p)p_e)\}\{1-(p+(1-p)p_e)^{m+1}\} \\ & +W2^{m'}p^{m'+1}\{1-2(p+(1-p)p_e)\} \\ & \cdot\{1-(p+(1-p)p_e)^{m-m'}\}, \end{aligned}$$

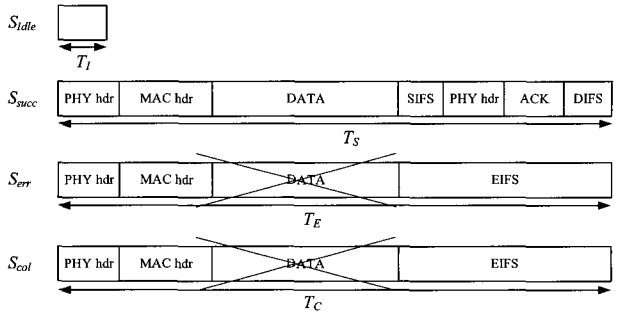


Fig. 4. Four states' time block diagrams of 802.11.

and

$$p = 1-(1-\tau)^{n-1}. \tag{3}$$

As an extension of this modified model, (4) changes the throughput efficiency over noisy channels by adding average error duration term $P_E T_E$ in the denominator. Equation (4) defines the throughput efficiency Th_{eff} when n nodes constantly try to transmit packets with $L_{pld}(1-\alpha)$ payload according to 802.11 protocol. Th_{eff} is the average length of data bits to successfully arrive at the receiver $E[L_{pld}]$ divided by the average slot time spent $E[T]$. Note that α represents the ratio of FEC code size to the total payload size.

$$Th_{eff} = \frac{E[L_{pld}]}{E[T]} = \frac{P_S L_f (1-\alpha)}{P_I T_I + P_S T_S + P_E T_E + P_C T_C}. \tag{4}$$

Equation (5) enumerates the time duration for the four states described in Fig. 4 such as idle period T_I, successful transmission time T_S, unsuccessful transmission time due to propagation errors T_E, and collision time T_C. To send a packet, 802.11 undergoes the four states such as idle state S_{idle}, successful transmission state S_{succ}, error state S_{err}, and collision state S_{col} whose time diagrams are depicted in Fig. 4. T_I is equal to one slot time σ while T_S is comprised of seven delay components; T_{PHYhdr}, T_{MAChdr}, T_{DATA} of sending the physical header, MAC header, L_{pld} payload, T_{SIFS} delay of SIFS (Short Inter-Frame Space), T_{ACK} transmission delay of an acknowledgement, round-trip propagation delay (2δ) that is not explicitly described in Fig. 4, and finally T_{DIFS} trailing delay of DIFS (Distributed Inter-Frame Space). Note that one slot time of 802.11 using FHSS (Frequency Hopping Spread Spectrum) is defined as $50\mu s$ [7]. Differently from T_S, T_E and T_C need EIFS (Extended Inter-Frame Space) before starting another contention period.

$$\begin{aligned} T_I = {} & \sigma, \\ T_S = {} & 2T_{PHYhdr} + T_{MAChdr} + T_{DATA} + T_{SIFS} + \\ & T_{ACK} + 2\delta + T_{DIFS}, \\ T_E = {} & T_{PHYhdr} + T_{MAChdr} + T_{DATA} + \delta + T_{EIFS}, \\ T_C = {} & T_{PHYhdr} + T_{MAChdr} + T_{DATA} + \delta + T_{EIFS}. \end{aligned} \tag{5}$$

Equation (6) lists the probability of each state's occurrence under the assumption that n is the total number of nodes to compete. In (6), P_S is the probability that only one node out of n nodes delivers a packet without error where p_e is the probability

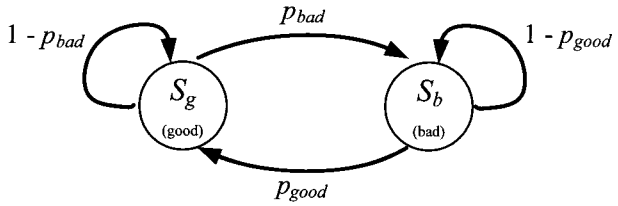


Fig. 5. Gilbert model.

of a packet's corruption. p_e will be derived later in (8). Differently from [11], our model contains P_E and multiplies $(1-p_e)$ to $n\tau(1-\tau)^{n-1}$ for P_S.

$$\begin{aligned} P_I &= (1-\tau)^n, \\ P_S &= n(1-\tau)^{n-1}(1-p_e), \\ P_E &= n(1-\tau)^{n-1}p_e, \\ P_C &= 1-P_I-P_S-P_E. \end{aligned} \tag{6}$$

For p_e with FEC codes, at first we need to compute the FEC symbol error probability $p_{fec\_sym\_err}$ shown in (7) and the packet recovery probability p_{pkt_rec} or $(1-p_e)$ in (6). For $p_{fec\_sym\_err}$, we abstract the WSN channel behavior as Gilbert model shown in Fig. 5 as a two-state Markov model consisting of two states named good state S_g and bad one S_b. Gilbert model is known to be one of the common models to appropriately represent the burstiness of wireless channel errors by suitably selecting its two transition probabilities, p_{bad} and p_{good}.

According to this model, the probability of any one transmission symbol out of c symbols being in error is $(1-(p_{good}/(p_{bad}+p_{good}))(1-p_{bad})^{c-1})$ since the probability of being in S_g and the probability of consecutively revisiting $(c-1)$-time S_g are $p_{good}/(p_{bad}+p_{good})$ and $(1-p_{bad})^{c-1}$ respectively. Note that most wireless networks modulate some number of bits with one transmission symbol for speedup. In TIP50CM [14] used for building our experimental WSN, for example, the size of one transmission symbol is 2 bits since it employs offset-DQPSK (Differential Quadrature Phase Shift Keying). Equation (7) computes $p_{fec\_sym\_err}$ when one FEC symbol consists of c transmission symbols. For computing (7), p_{bad} and p_{good} would be computed by matching CCDF of real traces' burst lengths with that of Gilbert model in Section 4.

$$p_{fec\_sym\_err} = 1 - \frac{p_{good}}{p_{bad}+p_{good}}(1-p_{bad})^{c-1}. \tag{7}$$

Equation (8) computes p_{pkt_rec} or $(1-p_e)$ of successfully recovering a packet by RS algorithm. Here, n and t represent the size of one packet and the size of FEC codes in the unit of FEC symbols while the size of one FEC symbol is m-bit. When $n \le (2^m-1)$, p_{pkt_rec} is the probability that the number of corrupt symbols is less than or equal to $t/2$ out of n symbols. When $n > (2^m-1)$, p_{pkt_rec} is the probability that the number of contaminated symbols is not greater than $t/2d$ in each codeword since user data are assumed to be packed into $d(=n/(2^m-1))$ codewords and $t/2$ symbols are assumed to be equally distributed over d codewords. When d is not an integer, the remaining FEC code, $t-(d-1)\lfloor t/d \rfloor$ is allocated to the last codeword. For instance, 2 symbols assigned to the last codeword at Symbol 5 in Fig. 2.

Note that $\lfloor \cdot \rfloor$ is the floor operator. In this case, the probability of correcting the last codeword should be separately computed and multiplied with the recovery probabilities of the previous codewords. p_{pkt_rec} of $Symbol_5$ in Fig. 2, for example, would be 0.995784×0.99975 since the recovery probabilities of the first 4 codewords and the last one are 0.99578 and 0.99975 when p_e is 1.7×10^{-2}.

$$p_{pkt\_rec} = \begin{cases} \displaystyle\sum_{k=0}^{t/2} \binom{n}{k} (p_{fec\_sym\_err})^k \cdot \\ \qquad (1-(p_{fec\_sym\_err})^{n-k}), \\ \quad \text{when } n \le (2^m-1), \\ \left(\displaystyle\sum_{k=0}^{t/2d} \binom{d/k}{k} (p_{fec\_sym\_err})^k \cdot \right. \\ \qquad \left. (1-(p_{fec\_sym\_err})^{\frac{n}{d}-k}) \right)^d, \\ \quad \text{when } n > (2^m-1). \end{cases} \tag{8}$$

B. Energy Model of 802.11 with RS Codes

This section estimates the amount of power that sensor nodes require to encode and decode RS FEC codes. The power demand by RS algorithm is known to be determined by the total codeword length n and the FEC code length t as shown in (9) [12] where E_{add} and E_{mult} represent the energy spent for executing one addition and one multiplication instruction respectively. Note that (9) evaluates only the energy consumed at the FEC decoder since the energy spent at the FEC encoder is assumed to be negligible in [12].

$$E_{dec} = (2n+2t^2)(E_{add}+E_{mult}). \tag{9}$$

Table 1 presents the actual power consumption of TIP50CM exhibiting the same performance as Moteiv Telos [15] when it executes a RS program [16] to encode and decode five different RS codewords depicted in Fig. 1. TIP50CM employs TI's MSP430f149 low-power processor requiring $3V$ and $240\mu A$ for processing one instruction. Note that in Table 1, we omit 9-bit symbol codeword case due to the lack of memory in TIP50CM for programming RS codes with 9-bit symbol.

The energy in Table 1 for decoding an RS codeword relies on the number of its error symbols since the program code lengths to process the different number of corrupt symbols differ. To obtain the actual power in Table 1, we measure the encoding and decoding time t_{proc} of the RS program [16] for each different number of errors by counting the internal clock ticks of TIP50CM. In addition, we compute the total energy spent using (10) under the assumption that each instruction's execution consumes almost the same amount of voltages and currents regardless of the instruction types. Especially, this assumption is known to be valid even when processors like TIP50CM adopt a pipeline technique.

$$E = VIt_{proc}. \tag{10}$$

Table 1. Energy consumption for encoding and decoding RS code

RS code	The number of error symbols	Encoding time (sec)	Decoding time (sec)	Encoding energy (mJ)	Decoding energy (mJ)
(15, 13)	0	0.26	0.33	0.18	0.24
	1	0.26	0.46	0.18	0.33
	Uncorrectable	0.26	0.32	0.18	0.23
(31, 27)	0	0.49	0.59	0.35	0.43
	1	0.49	0.72	0.35	0.53
	2	0.49	0.84	0.35	0.61
	Uncorrectable	0.49	0.67	0.35	0.48
(63, 55)	0	0.78	0.95	0.56	0.69
	1	0.78	1.08	0.56	0.77
	2	0.78	1.19	0.56	0.85
	3	0.78	1.30	0.56	0.94
	4	0.78	1.43	0.56	1.03
	Uncorrectable	0.7820	1.28	0.56	0.92
(127, 115)	0	1.22	1.44	0.88	1.04
	1	1.22	1.59	0.88	1.14
	2	1.22	1.72	0.88	1.24
	3	1.22	1.86	0.88	1.34
	4	1.22	2.01	0.88	1.45
	5	1.22	2.16	0.88	1.55
	6	1.22	2.31	0.88	1.66
	Uncorrectable	1.22	2.22	0.88	1.60
(255, 245)	0	2.16	2.44	1.56	1.75
	1	2.16	3.43	1.56	2.46
	2	2.16	3.42	1.56	2.46
	3	2.16	3.60	1.56	2.59
	4	2.16	3.60	1.56	2.59
	5	2.16	3.60	1.56	2.59
	Uncorrectable	2.16	3.39	1.56	2.44

Table 1 indicates that the encoding energy is not negligible differently from the assumption in [12]. The encoding energy for this RS program, for example, is almost half of the decoding energy in all error cases. Table 1 also shows the total decoding energy monotonously increases as the number of error symbols and the size of codewords grow even though the decoding energy for uncorrectable errors is slightly less than that for the maximum number of correctable errors.

For computing t_{proc} for decoding packets whose formats are described in Fig. 2, two equations for $t_{dec\_cor\_err}$ and $t_{dec\_unc\_err}$ in (11) and (12) average the decoding times of a packet with recoverable errors and fatal ones, respectively. The total energy consumed for encoding and decoding a packet without errors would be simply the sum of energies needed for encoding and decoding each codeword contained in packets. Since correctable and uncorrectable packets have various error patterns, (11) and (12) compute their average decoding time by summing each pattern's processing time multiplied by each one's occurrence probability.

For (11) and (12), we assume that a packet contains n codewords and each codeword is capable of recovering t corrupt symbols maximally. In this case, the time for decoding a packet with correctable errors is $k_0t_0 + k_1t_1 + \cdots + k_tt_t$ when $k_i (0 \le k_i \le n)$ codewords hold $i (0 \le i \le t)$ corrupt symbols and each codeword needs t_i seconds to process i corrupt symbols. Note that each k_i ranges from 0 to n inclusively while satisfying $k_0 + k_1 + \cdots + k_t = n$. The number of instances belonging to this same error pattern is, which is equivalent to the number of ways to pick up $k_0, k_1, \cdots, k_t$ out of n codewords. Since the total number of all correctable error patterns is $(t+1)^n$, the probability of this error pattern is $({}_nC_{k_0}\,{}_{n-k_0}C_{k_1} \cdots {}_{n-k_0-k_1-\cdots-k_{t-1}}C_{k_t})/(t+1)n$. $t_{dec\_cor\_err}$ averages the packet decoding times for all these possible correctable error patterns.

The average decoding time of a packet with uncorrectable errors, $t_{dec\_unc\_err}$ consists of two parts; the time to correct $j (0 \le j \le n-1)$ recoverable codewords and t_{t+1} to process the uncorrectable $(j+1)$th codeword. Note that the RS program terminates the decoding operation once it meets the first uncorrectable codeword. The first term of $t_{dec\_unc\_err}$ in (12) averages the recovery time of j correctable codewords by multiplying $1/n$ after summing n delays taken by the case of $0 \le j \le n-1$.

$$t_{dec\_cor\_err} = \sum_{k_t=0}^{n-(k_0+k_1+\cdots+k_t)} \cdots \sum_{k_2=0}^{n-(k_0+k_1)} \sum_{k_1=0}^{n-k_0} \sum_{k_0=0}^{n-1} \left(\frac{{}_nC_{k_0}\,{}_{n-k_0}C_{k_1} \cdots {}_{n-k_0-k_1-\cdots-k_{t-1}}C_{k_t}}{(t+1)^n} (k_0t_0 + k_1t_1 + \cdots + k_tt_t) \right) \quad (11)$$

where $k_0 + k_1 + \cdots + k_t = n$.

$$t_{dec\_unc\_err} = \frac{1}{n} \sum_{j=0}^{n-1} \left[\sum_{k_t=0}^{j-(k_0+k_1+\cdots+k_t)} \cdots \sum_{k_2=0}^{j-(k_0+k_1)} \sum_{k_1=0}^{j-k_0} \sum_{k_0=0}^{j} \left(\frac{{}_jC_{k_0}\,{}_{j-k_0}C_{k_1} \cdots {}_{j-k_0-k_1-\cdots-k_{t-1}}C_{k_t}}{(t+1)^j} (k_0t_0 + k_1t_1 + \cdots + k_tt_t) \right) \right] + t_{t+1} \quad (12)$$

where $k_0 + k_1 + \cdots + k_t = j$.

Fig. 6 presents the encoding and decoding energy consumptions of five-format packets in Fig. 2. In Fig. 6, E_{enc} represents the encoding energy while $E_{dec\_no\_err}$, $E_{dec\_cor\_err}$, and $E_{dec\_unc\_err}$ denote the average energies of decoding packets with no error, correctable error, and uncorrectable error respectively. E_{enc} almost linearly increases in proportional to the symbol size, resulting in 88 % difference between 4-bit and 8-bit symbol sizes. It is due to that the codeword size exponentially grows so that the encoding time to process each codeword gets longer as FEC symbol gets larger.

Fig. 6 confirms that all three RS decoding times continue to increase as a function of FEC symbol size like the encoding energy. It also illustrates that $E_{dec\_cor\_err}$ is 30 % larger than $E_{dec\_no\_err}$ for all symbol sizes since the error rectification requires more code lines to run. It finally indicates that $E_{dec\_unc\_err}$ is slightly less than $E_{dec\_no\_err}$ for 4-bit symbol since RS algorithm finishes its recovery operation as soon as it meets an uncorrectable codeword without checking the remaining codewords. As the number of codewords in the packet diminishes, $E_{dec\_unc\_err}$ becomes equal to $E_{dec\_cor\_err}$.

IV. PERFORMANCE EVALUATION

This section consists of two subsections such as throughput and energy consumption evaluations of 802.11 with RS codes over WSNs.

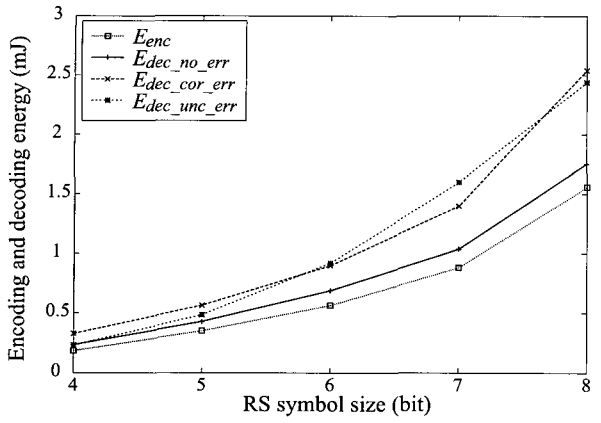


Fig. 6. Energy consumption of encoding and decoding in three states.

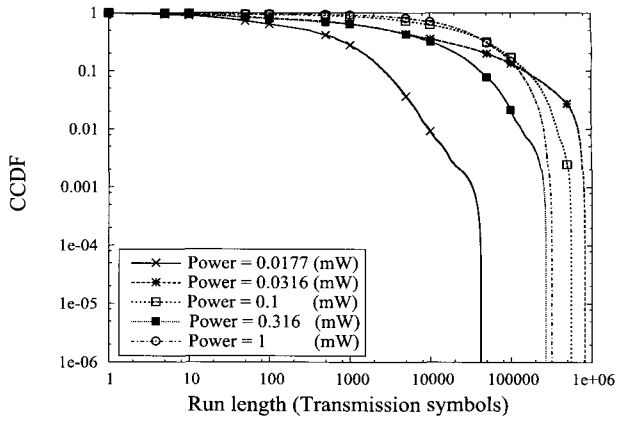


Fig. 8. WSN run length CCDF.

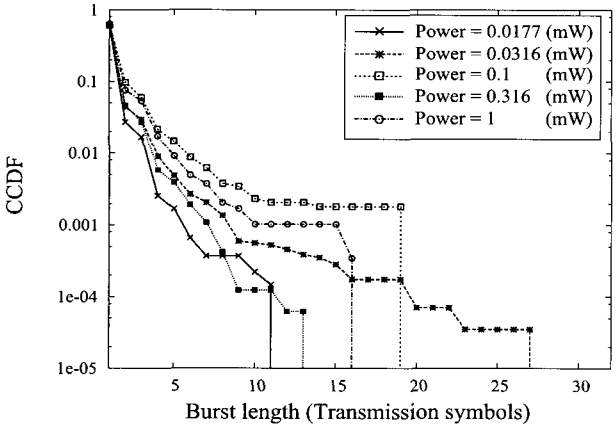


Fig. 7. WSN burst length CCDF.

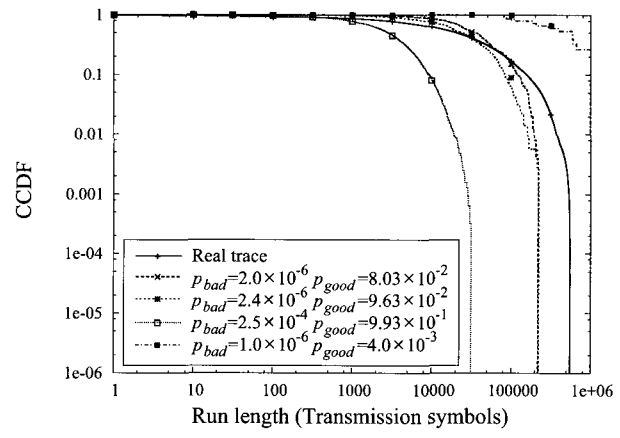


Fig. 9. Run length CCDF comparison.

A. *Throughput Evaluation of 802.11 with RS Codes over WSNs*

This section numerically calculates Th_{eff}, throughput efficiency of 802.11 with RS codes based on Gilbert model. For choosing an appropriate Gilbert model, we match CCDF of WSN channel run lengths or error burst lengths with that of Gilbert model. Note that burst and run lengths represent the number of successive bits and the number of uncorrupt bits respectively. For deriving the analytical channel model, traffic traces are collected from sensor networks where a sender continuously transmits 80-byte packets with the speed of 9.6 Kbps to its receiver which is 10 meters apart on line-of-sight. Figs. 7 and 8 depict the average burst length and run length CCDF of 3 traces at five different signal powers each of which is measured for 3 hours from 1 p.m. to 4 p.m. in the corridor where pedestrian traffic tends to be heavy. Figs. 7 and 8 indicate that these two metrics are not inversely proportional to signal power in this network as predicted in [17] whose channel behaviors are rather heavily affected by the small-scale fading effect [6] caused by human traffic. The burst lengths at 0.316 mW signal power, for instance, are less than that at 1 mW. They also show that the lengths of 95% bursts and 90% runs are shorter than 4 bits and 1000 bits.

Based on these traces, we determine (p_{bad}, p_{good}) of Gilbert models to equalize their average BER and the run length CCDFs to those of the corresponding real traces. After picking up some sets of (p_{bad}, p_{good}) to produce the same average BER, we choose one out of them to generate the most similar CCDF distribution to that of the corresponding real trace. The similarity of two CCDFs is evaluated by LMS (Least Mean Square) of their differences. Fig. 9 shows an example where Gilbert models with four different sets of (p_{bad}, p_{good}) can produce different CCDFs even though their average BERs are the same as the real trace's average. From Fig. 9, we can see that the Gilbert model with (p_{bad}, p_{good}) equal to $(2.0 \times 10^{-6}, 8.03 \times 10^{-2})$ is most similar to the real one.

Fig. 10 depicts Th_{eff} of a hypothetical WSN, where 10 nodes equipped with 802.11 constantly contend to send fixed-size packets, as a function of the FEC symbol size and the average BER. Th_{eff} is normalized by 256 Kbps and the assumed WSN's operational parameters are specified in Table 2. Fig. 10 first confirms that 802.11 with RS codes accomplishes better Th_{eff} by up to 14% over WSN channels than the legacy 802.11 regardless of symbol sizes when BER is greater than 10^{-4}. It plots Th_{eff} of the original 802.11 at 0 of the FEC symbol size axis. It also indicates that the RS symbol size and WSN channel BER rarely influence Th_{eff} of WSNs. Th_{eff} of 8-bit symbol only achieves better by 1% than any other sizes. This independence implies that these long-term average WSN BER behaviors alternately exhibit either uniform or burst error distributions so that any specific size is not preferred to others.

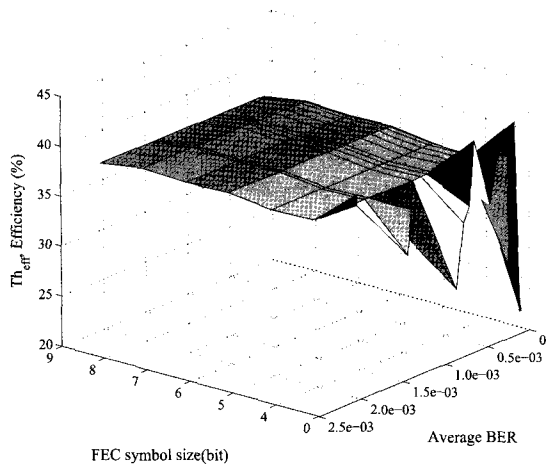


Fig. 10. Throughput efficiency as a function of RS symbol size and average BER.

Table 2. Parameters for TIP50CM's PHY and 802.11 using FHSS.

Parameter	Value
Frame payload size	80 bytes
Slot time (σ)	50 μs
PHY header	16 bytes
SIFS	28 μs
MAC header	10 bytes
DIFS	128 μs
ACK size	14 bytes
EIFS	1 ms
PHY data rate	256 Kbps
Number of node (n)	10
Propagation delay (δ)	1 μs
τ	0.0373

For evaluating WSN's channel burstiness on the fine-grain time scale, Fig. 11 plots variations of p_{bad} and p_{good} in 10-miniute traces. For the 10-minute traces, we divide one of 15 3-hour traces into 10-minute time slots and calculate their (p_{bad}, p_{good}) of each 10-minute slots. In the fine-grain evaluation, we observe that these two parameters widely fluctuate up to 500 times at maximum. Note that the more similar pbad and pgood are, the more uniformly the propagation errors are distributed.

Fig. 12 specifies how rapidly Th_{eff} fluctuates as a function of the time and the FEC symbol size over these relatively short-time intervals whose (p_{bad}, p_{good}) are depicted in Fig. 11. It proves that Th_{eff} widely varies up to 24% over the short periods when the symbol size is inappropriately selected. At time 170 in Fig. 12, for example, Th_{eff} of 9-bit size is 29% while Th_{eff} of 4-bit is nearly 5%. This observation shows that WSNs need to adopt an algorithm dynamically tuning the symbol size to the channel state. Th_{eff} of 802.11 without RS codes finally achieves around 21% at time 0 while it approaches to 0 as time goes by due to the high BER.

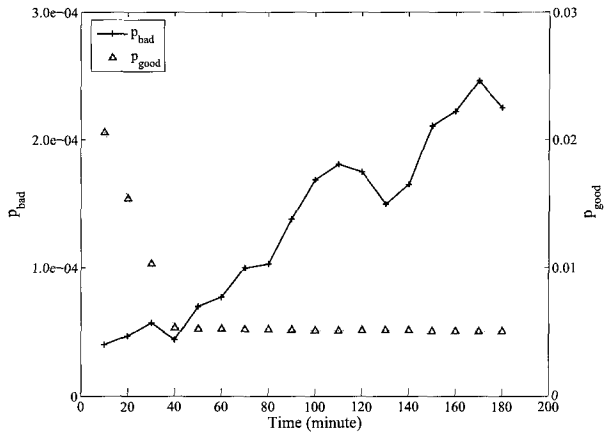


Fig. 11. p_{bad} and p_{good} variations over 10-minute intervals.

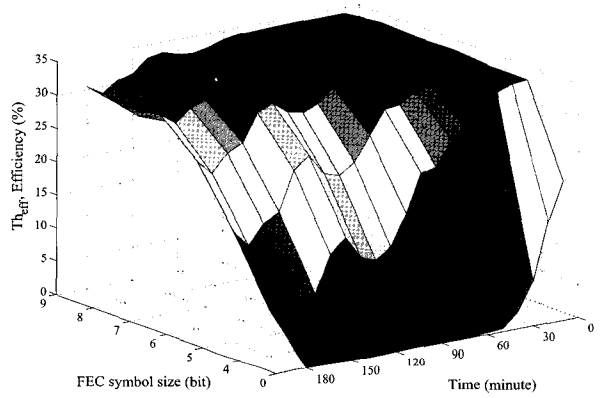


Fig. 12. Transmission efficiency as a function of RS symbol size and time.

B. Energy Consumption Evaluation of 802.11 with RS Codes over WSNs

This section evaluates the energy consumption of 802.11 with RS codes over WSN channels. For energy evaluation, we assume some arbitrary interval to be large enough for the four states of Fig. 2 to repeatedly occur several times. During this duration we measure idle duration, the number of packets to successfully traverse, the number of colliding packets, and the number of erroneous packets based on the ratio of the four states' duration as specified in (5). For decoding and encoding energies, we use data in Table 1 while for idling, transmitting, listening, and receiving energies, we refer to the data sheet of CC2420 [18] for TIP50CM's transceiver.

Equation (13) defines E_{perbit}, the energy for successfully sending a bit, which divides the total energy E_{total} by the total number of successfully transmitted bits Bit_{succ_total} during an arbitrary time interval T_{total}. E_{total} consists of four energies such as E_I, E_S, E_C, and E_E spent during each state. Note that they correspond to the energies required in idle, successful transmission, collision, and erroneous state depicted in Fig. 4 respectively. P_I, P_S, P_C, and P_E are calculated by (6) while E_{idle}, E_{trans}, and E_{recv} denote the idling energy for 1 second, the transmission energy per packet, and the receiving energy per

packet respectively.

The duration of each state is computed by multiplying T_{total} with its probability. To compute the power necessary for processing a packet, (13) divides a state's total duration by a packet's processing time. For instance, the number of colliding packets is obtained by dividing the colliding state duration ($T_{total}P_C$) by a packet's collision duration T_C defined in (5). E_S, furthermore, consists of two terms accounting for E_{S_NOERR}, the transmission energy of packets with no error and E_{S_COR}, the one with correctable errors. In detail, the number of packets without errors is the ratio of $(1 - p_{fec\_sym\_err})^c / p_{pkt\_rec}$ to $(T_{total}P_S)/T_S$ while the number of packets with correctable errors amounts to the remaining ratio. E_C and E_E finally don't include E_{enc} since the FEC code of a packet is not recomputed for its retransmissions after the FEC code is synthesized at the initial transmission.

$$E_{perbit} = \frac{E_{total}}{Bit_{succ\_total}} = \frac{E_I + E_S + E_C + E_E}{T_{total}P_S \frac{1}{T_S} L_{pld}} \tag{13}$$

where

$$\begin{aligned}
E_I &= T_{total}P_I E_{idle}, \\
E_S &= E_{S_NOERR} + E_{S_COR} \\
&= T_{total}P_S \frac{(1 - P_{fec_sym_err})^c}{P_{pkt_rec}} \frac{1}{T_S} \cdot \\
&\quad (E_{enc} + E_{trans} + E_{recv} + E_{dec_no_err} + \\
&\quad E_{SIFS} + E_{DIFS} + E_{ack_trans} + E_{ack_recv}) + \\
&\quad T_{total}P_S \frac{P_{pkt_rec} - (1 - P_{fec_sym_err})^c}{P_{pkt_rec}} \frac{1}{T_S} \cdot \\
&\quad (E_{enc} + E_{trans} + E_{recv} + E_{dec_cor_err} + \\
&\quad E_{SIFS} + E_{DIFS} + E_{ack_trans} + E_{ack_recv}), \\
E_C &= T_{total}P_C \frac{1}{T_C}(E_{trans} + E_{EIFS}), \\
E_E &= T_{total}P_E \frac{1}{T_E}(E_{trans} + E_{recv} + \\
&\quad E_{dec_unc_err} + E_{EIFS}).
\end{aligned} \tag{14}$$

Based on (13) and (14), Fig. 13 depicts E_{perbit} as a function of RS symbol size and T_{total}. Note that Fig. 13 uses the same trace as in Fig. 10. At first, E_{perbit} for 802.11 without RS codes is three order-of-magnitudes larger than E_{perbit} for 802.11 with RS codes regardless of symbol sizes due to the high BER. Note that E_{perbit} of 802.11 without RS codes is not plotted because of the difficulty of placing it in the scale of Fig. 13. It is due to that the original 802.11 can't send any bit when BER is beyond 10^{-3}. It also predicts that the energy demand widely fluctuates depending on the FEC symbol size. E_{perbit} of 8-bit symbol size, for example, is maximally 740% larger than that of 4-bit one. It is due to that the large FEC symbol size demands more encoding and decoding energies than small one even though the former recovers slightly more packets by 24% than the latter as in Fig. 12.

V. CONCLUSIONS

This paper extends the 802.11 evaluation model to analyze throughput and energy demand of 802.11 with RS codes over WSNs. To apply the analytical model to WSNs, it measures traffic traces from a WSN with two TIP50CM sensor nodes and energy consumption of decoding and encoding RS codes. The application of the evaluation model to the abstracted WSNs shows that 802.11 with RS codes outperforms the original 802.11 in terms of these two performance metrics. It also indicates that the RS symbol size can significantly vary throughput and energy saving over short-term durations while the long-term average of these two metrics seldom depends on the symbol size. As future research, we will actually measure the real sensor node's throughput and energy saving after implementing RS algorithm in WSN 802.11 code. Finally we will develop a dynamic algorithm adapting the FEC symbol size to the underlying channel state.

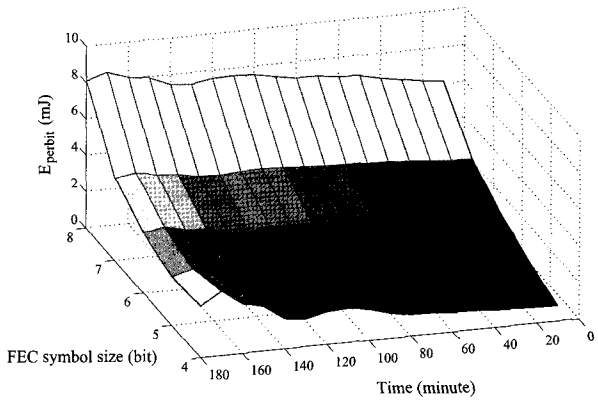


Fig. 13. Energy per bit as a function of RS symbol size and time.

REFERENCES

[1] W. Ye, J. Heidemann, and D. Estrin, "An Energy efficient MAC protocol for wireless sensor networks", in *Proc. IEEE INFOCOM*, 2002, pp. 1567–1576.

[2] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wreless sensor networks", in *Proc. SenSys*, 2004, pp. 95–107.

[3] K. P. Yar, S. Y. Chang, and W. E. Stark, "Adaptive energy scheme for wireless network systems", in *Proc. International Conference on Wireless Networks, Communications and Mobile Computing*, 2005, pp. 163–168.

[4] M. N. Smadi, and B. Szabados, "Error-recovery service for the IEEE 802.11b Protocol", *IEEE Trans. Instrum. Meas.*, vol. 55, pp. 1377–1382, Aug. 2006.

[5] J. S. Ahn, S. W. Hong, and J. Heidemann, "An adaptive FEC code control algorithm for mobile wireless sensor networks", *J. Commun. Networks*, vol. 7, pp. 489–499, Dec. 2005.

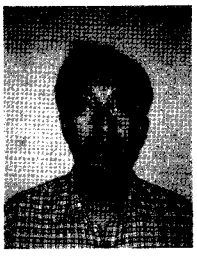
[6] T. S. Rappaport, *Wireless communications: Principles and practice*. Upper Saddle River, NJ: Prentice Hall, 2002.

[7] IEEE Std. 802.11 *IEEE Standard for information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, 1999.

[8] IEEE Std. 802.11a, *IEEE Standard for information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN medium access control (MAC) and physical sayer (PHY) specifications:High-speed physical layer in the 5 GHz Band*, 1999.

[9] IEEE Std 802.11g, *IEEE Standard for information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications—Amendment 4: Further higher data rate xxtension in the 2.4 GHz Band*, 2003.

[10] G. Bianchi, "Performance Analysis of the IEEE 802.11 distributed coordination function", *IEEE J. Sel. Areas Commun.*, vol. 18, pp. 535–547, Mar. 2000.
[11] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, "Performance of reliable transport protocol over IEEE 802.11 wireless LAN: Analysis and enhancement", in *Proc. IEEE INFOCOM*, 2003, pp. 599–607.
[12] Y. Sankarasubramaniam, I. F. Akyildiz, and S. W. McLaughlin, "Energy efficiency based packet size optimization in wireless sensor networks", in *Proc. the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003, pp. 1–8.
[13] E. N. Gilbert. "Capacity of a burst-noise channel", *Bell Syst. Tech. J.*, vol. 39, pp. 1253–1265, Sept. 1960.
[14] MAXFOR, Inc., Wireless sensor network node, http://www.maxfor.co.kr.
[15] Moteiv, Inc., Wireless sensor network node, http://www.moteiv.com.
[16] S. Rockliff, "Reed-solomon (RS) codes" program, 1989, http://www.ecc-page.com.
[17] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "Protocols and architectures for wireless sensor networks", Tech. Rep. UCLA/CSDTR 02-0013, Feb. 2002.
[18] Chipcon, Inc., CC2420 low power O-QPSK trans-ceiver, http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf, Oct. 2005.



Jong-Suk Ahn received the Ph.D degree and the M.S. degree in electrical engineering from the University of Southern California in 1995 and from the Korean Advanced Institute of Science and Technology in 1985, respectively and the B.S. degree from Seoul National University in 1983. He was a visiting researcher in ISI/USC for one year from 2000. He was awarded Gaheon Prize for the best paper in 2003 from the Korean Information Science Society. He is currently a professor at the Dongguk University in Korea. His major interests are in sensor networks, dynamic error control and flow control algorithms over wireless Internet, fast IP routing techniques, and network simulation techniques.

Jong-Hyuk Yoon was born in Yeoju, Korea on September 25, 1980. He received the B.S. degree in Computer Engineering in 2006 from the Dongguk University. He is currently a master student in Computer Engineering department, Dongguk University of Seoul, Korea. His major interests are wireless sensor networks, 802.11 mac protocol, and embedded systems.

Kang-Woo Lee received the M.S. degree and the Ph.D. degree in Electrical Engineering from the University of Southern California in 1991 and in 1997, respectively and the B.S. degree from Yonsei University in Electronic Engineering in 1985. He is currently a associate professor at the Dongguk University in Korea. His major interests are in computer architectures, embedded systems, networking computing, and simulation techniques.