Reliable Data Transmission Based on Erasure-resilient Code in Wireless Sensor Networks

Jian-Jun Lei and Gu-In Kwon
Department of Computer and Information Engineering
Inha University, Incheon, South Korea
[e-mail: dannylei@hotmail.com, gikwon@inha.ac.kr]
*Corresponding author: Gu-In Kwon

Received October 27, 2009; revised December 21, 2009; accepted January 4, 2010; published February 27, 2010

Abstract

Emerging applications with high data rates will need to transport bulk data reliably in wireless sensor networks. ARQ (Automatic Repeat request) or Forward Error Correction (FEC) code schemes can be used to provide reliable transmission in a sensor network. However, the naive ARQ approach drops the whole frame, even though there is a bit error in the frame and the FEC at the bit level scheme may require a highly complex method to adjust the amount of FEC redundancy. We propose a bulk data transmission scheme based on erasure-resilient code in this paper to overcome these inefficiencies. The sender fragments bulk data into many small blocks, encodes the blocks with LT codes and packages several such blocks into a frame. The receiver only drops the corrupted blocks (compared to the entire frame) and the original data can be reconstructed if sufficient error-free blocks are received. An incidental benefit is that the frame error rate (FER) becomes irrelevant to frame size (error recovery). A frame can therefore be sufficiently large to provide high utilization of the wireless channel bandwidth without sacrificing the effectiveness of error recovery. The scheme has been implemented as a new data link layer in TinyOS, and evaluated through experiments in a testbed of Zigbex motes. Results show single hop transmission throughput can be improved by at least 20% under typical wireless channel conditions. It also reduces the transmission time of a reasonable range of size files by more than 30%, compared to a frame ARQ scheme. The total number of bytes sent by all nodes in the multi-hop communication is reduced by more than 60% compared to the frame ARQ scheme.

Keywords: Error recovery, erasure-resilient code, wireless sensor network, wireless channel, reliable data transmission

This work was supported by INHA UNIVERSITY Research Grant (INHA-3611201).

DOI: 10.3837/tiis.2010.01.004

1. Introduction

Wireless sensor network applications have gained the attention of researchers in the last few years due to the rapid growth of wireless technology [1]. As software and hardware of sensor networks mature, it becomes increasingly possible to conceive of a class of applications that require the transport of bulk data. Sources for bulk data include images and voice data [2]. In certain scenarios of interest (e.g. precision agriculture, forest fire prevention, and military target tracking), bulk data gathered by monitoring devices are required to be transmitted reliably to a monitor center through sensor network nodes. While other types of reliability in WSN, such as *event-driven* reliability, have been studied, we consider the provision of *packet-driven* reliability (all packets sent by the source node should be delivered to the sink) in this paper.

The frame size has to be considered carefully due to the low quality wireless channel in sensor networks. In an ARQ (Automatic Repeat request) based scheme, the small frame size would be desired to reduce the frame error rate (FER) with a given bit error rate (BER). However, there is a constant overhead per frame. This overhead includes the physical preamble, the frame header, and the frame CRC. The per-frame overhead ratio has to be minimized to achieve high wireless channel utilization [3]. Thus, a large frame size would be desired. However, if there is a bit error in the large frame, the entire frame has to be retransmitted in the ARQ scheme. The larger the frame size to maximize channel utilization, the greater the increase of FER [4][5]. Careful framing needs to consider both high utilization of the wireless channel and low frame error rate simultaneously. Traditional framing is unable to provide the optimal frame size to solve the above conflict.

Other approaches using FEC (Forward Error Correction) have been proposed to provide reliable transmission. In these approaches, the original frames are encoded using FEC and this redundant data are added to the original frame. The sender transmits these frames to the next node and the redundant data would be used to recover the bit errors inside the frame. In a bad channel with high BER, the amount of redundancy data would be insufficient to perform decoding. Thus, the redundancy data are useless and the original frame should be retransmitted. In contrast, in a good channel with low BER, many frames will be received without a bit error. The redundant data will seldom be used, thus the transmission of redundant data wastes bandwidth. The sender must know the current BER and decide the amount of FEC coding per frame to apply FEC effectively [6]. However, measurements have shown that the wireless channel quality changes quickly and unpredictably [7][8][9][10]. It will be very difficult to gauge the FEC encoding redundancy to match the current channel error rate.

In this paper, we propose a novel transmission scheme, which removes the technical complexity of the proper amount of FEC coding, to match the current channel error rate, whilst providing high link utilization without worrying about FER. The basic idea is to use the packet level erasure-resilient code[1] with a small size of input symbols.

Performing packet level FEC takes the original contents as a series of symbols for the input [11]. The encoder generates the encoded symbols and the sender transmits the original symbols with the encoded symbols to the receiver. The original symbols and the encoded symbols are treated equally in terms of the decoding process. The receiver does not have to receive all symbols, but it only needs any k symbols to perform the decoding process, where

[1] Often referred to as forward error-correcting (FEC) codes.

the number of original symbols is k. In the general FEC approach [12], a symbol (either the original or the encoded symbol) will be encapsulated into a frame (i.e. a frame consists of only one symbol). While this general approach would be applied to WSN, this approach has the same conflict problem as the ARQ scheme in terms of effective error recovery and high channel utilization for deciding the appropriate frame size.

We apply this packet level FEC; however, dividing the original content into many small sizes of blocks and encoding these blocks using LT code [13]. The sender node packages these encoded blocks into a frame and transmits the frames on the wireless sensor network. If there is an erroneous block inside a frame, the receiver just drops this block (as opposed the the entire erroneous frame) and does not request retransmission of this block. An error detection field is added at the end of each block. A frame consists of many of these blocks. The receiver continues to receive any error free k blocks to decode, where k is the number of original blocks.

Our approach also resolves the conflict between effective error recovery and high channel utilization in the wireless sensor network. As described above, increasing the frame size usually induces a high FER in the ARQ approach. However, in our approach, only erroneous blocks are identified and dropped. The remaining blocks in the frame will be used for another transmission to a next node or for decoding process at a sink. Thus, our approach can use a large frame size and provide high channel utilization without worrying about FER. The frame size will only be constrained by the amount of buffer space available in the wireless transceiver firmware. Our approach has been implemented as a new data link layer in TinyOS, and evaluated through experiments in Zigbex motes. Our results show that significant channel capacity can be achieved. The remainder of this paper is organized as follows: we briefly describe some related work and an efficient erasure-resilient code in section 2. Section 3 presents the implementation details and protocol design. Section 4 provides an analysis based on models derived from real measurements of wireless channel errors. Section 5 describes the results of the evaluation through experiments. Finally, we conclude this paper in section 6.

2. Related Work

2.1 Erasure-resilient Codes

While error-correcting codes provide resilience to bit errors, erasure-resilient codes provide resilience to packet-level losses. We use the following terminology. The content being sent by the encoder is a sequence of symbols $\{X_1,...., X_k\}$, where each is called an *input* symbol. An encoder produces a sequence of *encoding symbols* $y_1, y_1,...$ from the set of input symbols. For the family of erasure-resilient codes we use, parity-check codes, each encoding symbol is simply the bitwise XOR of a specific subset of the input symbols. A decoder attempts to recover the original content from the encoding symbols. Well-designed codes require recovery of a few percent (less than 5%) of symbols beyond k, the minimum needed for decoding. The *decoding overhead* of a code is defined to be e_d if $(1+e_d)k$ encoding symbols are needed on average to recover the original content. Provably good degree distributions for sparse parity check codes were first developed and analyzed in [12]. However, these codes are fixed-rate, meaning that only a pre-determined number of encoding symbols are generated, for example only *ck*, where *c* is a small constant > 1. In our application, this can lead to inefficiencies, as the origin server will eventually be forced to retransmit symbols. Newer codes, called *rateless* codes, avoid this limitation and allow unbounded numbers of encoding symbols to be generated on demand. Two examples of rateless codes, along with further discussion of the merits of ratelessness, may be found in [13], [14]. Both of these codes also have strong probabilistic decoding guarantees, along with low decoding overhead and average degrees. In our experiments, we simulate the use of LT codes [13], and assume a fixed decoding overhead of 5% based on the simulations and importance sampling described in [15].

2.2 Other FEC Schemes

We briefly review the related work on FEC schemes to provide reliable transmission.

Ahn et al. [16] propose an adaptive FEC algorithm which dynamically adjusts the amount of FEC coding per packet based on the presence or absence of receiver acknowledgements. Kyle et al. [17] take a different scheme called PP-ARQ. Instead of using stronger codes for the entire packet on retransmit, it uses hints from the physical layer about which data are more likely to be in error, and retransmits just those data.

Hybrid ARQ is a way of combining FEC and ARQ. The Type-I hybrid ARQ scheme retransmits the same data with a stronger FEC code. Chase's code combining [18] and Dubois-Ferriere's packet combining [8] improve on this strategy by storing corrupted packets and feed them all to the decoder. While Type-II hybrid ARQ schemes retransmit only stronger FEC code or some incremental FEC code when the channel is quiet. Metzner [19] and later Lin and Yu [20] developed type II hybrid ARQ schemes based on incremental redundancy. Type-I performs more poorly, since it wastes the bandwidth by repeatedly resending the same data, while Type-II becomes inefficient when either the data packet or some of the previous incremental FEC packets cannot reach the receiver [16].

3. System Design

In this section, we discuss the system design architecture, information format, and how the upper layer file is split into blocks at the sender and reassembled into the upper layer file at the receiver.

3.1 Overview

In this section, we describe single path system architecture, since we focus on error recovery, rather than routing or traffic dispersion. **Fig. 1** depicts a transmission route with a sender (sensor node), intermediate relays, and a receiver (or sink). Relays are the traditional power constrained sensor network nodes.

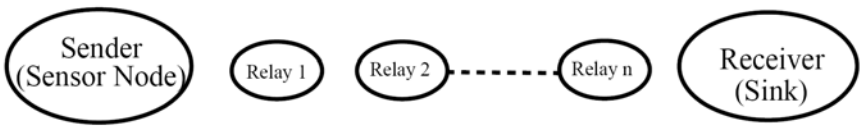

Fig. 1. System architecture

The basic procedure works as follows. Assume a sender transmits a file to a receiver via some intermediate relays. In the sender, a file from the application layer is divided into *k* blocks, and these blocks are encoded to *k* blocks using erasure-resilient code (LT code). Each encoded block is augmented with a 1-byte sequence number and a 1-byte CRC-8 for error detection. These encoded blocks are then packaged into a frame and this frame is transmitted to the receiver. When an intermediate receiver receives this frame, it identifies and drops only the corrupted block using the 1-byte CRC (Cyclic Redundancy Check) in each block. The remaining error free blocks in the frame are supposed to be transmitted to the next relay node. If there are no erroneous blocks in the frame, the same frame sent by the sender will be delivered to the next receiver. After $(1+e_d)k$ correct encoded blocks have been received by the sink receiver, these received blocks are decoded, then reassembled and handed off to the upper layer.

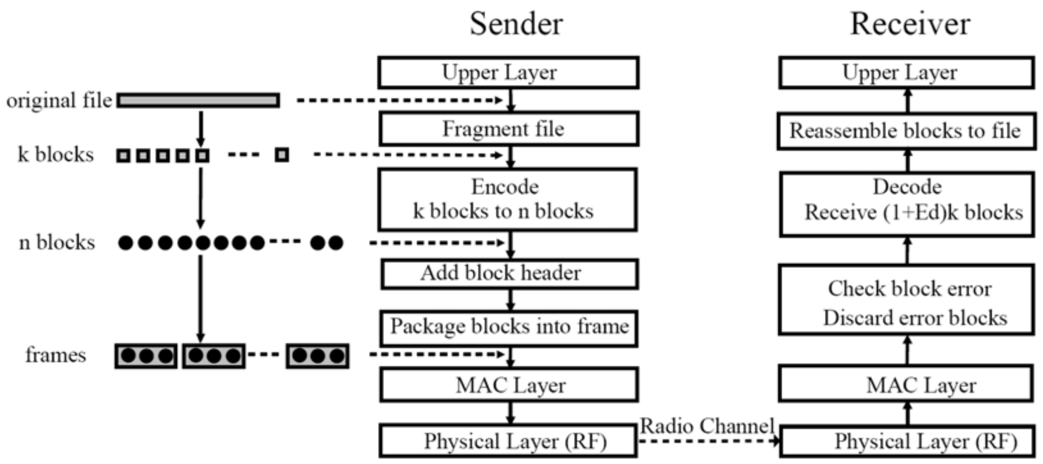

Fig. 2. Communication protocol flow

Fig. 2 presents the functions at both the sender and the receiver side. We enhance link utilization by using the error free blocks in the erroneous frame. The naive ARQ-based scheme should retransmit the entire frame, even with a single bit error in the frame. In our scheme, we only drop the corrupted block and transmit error free blocks to the next node, thus we avoid the retransmission of error free parts and increase the link utilization. We also improve link utilization by increasing the frame size. In the wireless sensor network, each frame has the physical layer and MAC layer headers that include the physical preamble, the frame header, and the frame CRC. A large frame size is desired to minimize this per-frame overhead. However, a larger frame size increases the FER. Under a high FER wireless channel, a single bit error in the frame induces the retransmission of the entire frame. Thus, the sender cannot increase the frame size just to maximize the wireless channel throughput. Due to this limitation of large frame size, a small frame size would be desired to maximize the error recovery at the cost of low link utilization. We resolve this conflict between high utilization and effective error recovery by using packet level FEC with a small block sizes for the input. The sender can increase the frame size for high utilization without compromising the effective error recovery. There should be large amounts of error-free bits in the corrupted frames, since the erroneous bits are highly clustered [21]. The frame consists of many small blocks in our approach and the error bits would be in few blocks in the frame.

Our approach with the rateless LT [13] code eliminates the complexity of deciding on the appropriate amount of encoding redundancy. The fixed-rate FEC like [12] has to carefully determine the appropriate amount of encoding blocks at the sender, since if the sink does not receive $(1+e_d)k$ blocks after the sender sends all encoding blocks, the sender has to transmit only missing blocks from the already sent blocks to the sink to avoid duplicated block reception. The sender generates $(1+e_d)k$ blocks first, since the LT code can generate new encoding blocks on demand. Even though transmitting all these blocks, the sink may not receive $(1+e_d)k$ error free blocks. The sink then transmits the number of missing packets up to this point, and the sender generates fresh encoding symbols based on information from the sink.

While the approach using the packet aggregation scheme has been proposed in [21][22], our scheme does not require the retransmission of corrupted blocks and does not have to store the sent frame in the sender or intermediate nodes. In Seda [21], a receiver requests the retransmission of corrupted blocks every four frames. This method avoids a small retransmission frame compared to sending the retransmission request for every frame. However, the approach induces the sender node to keep four sent frames in the buffer, since any block in the four frames could be requested for the retransmission. This buffering could occur at the receiver also to complete the reception of four frames. Each intermediate relay node along the path from the sender to a sink could buffer eight frames (four frames for the receiving side and four frames for the sending side of the relay node). This buffering is required for each sender-to-sink connection and a relay node that is close to the sink would relay packets from a variety of senders. Thus, Seda would make the buffer overflow at the relay node on a large scale WSN.

3.2 Message Format

The formats of the data frame and block are shown in **Fig. 3 (a)** and **(b)** respectively. Excluding the fixed physical and link layer overhead, each frame contains several data blocks. There are two additional overhead fields in a data block: Sub_Block_Seq and Block_CRC. The frame size is only constrained by the radio transceiver receiving or sending buffer space of the mote; while block size will be optimized based on the wireless channel characteristic in the successive section. The receiver identifies the corrupted block using the block CRC and only the corrupted blocks will be discarded and error-free block will be transmitted to the next node.

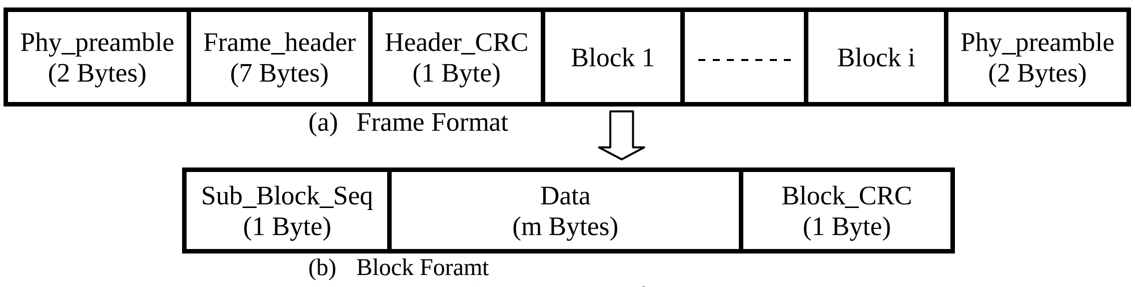

Fig. 3. Message format

4. Wireless Channel Error Characteristics and Utilization Analysis

In this section, we analyze the channel utilization theoretically and choose some design parameters by modeling based on field measurements. We first present our measurements of the BER and FER of the wireless channel between motes with a 250 Kbps CC2420 radio module. We design a simple model that describes the utilization of the wireless channel based on the measurements of the BER and its characteristics. Further, we analyze channel utilization on two typical channel models. The analytical utilization gains and optimal block sizes are verified in the following section with experiments in real sensor networks.

4.1 Channel Measurements

We conducted long-term indoor experiments in our lab to measure wireless channel errors using Zigbex motes. In our experiments, a sender continuously transmits four frames every second. We compute the FER, as shown in **Fig. 4**. It is clear from the figure that FER increases almost linearly with increasing frame size; a frame size of less than 40 bytes provides a superior error-free situation under this channel condition.

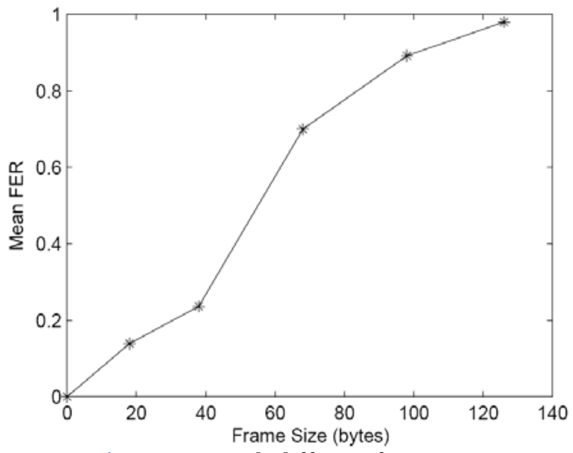

Fig. 4. FER with different frame sizes

Fig. 5 plots the BER with a frame size of 38 bytes (including the physical layer and MAC layer headers). From the figure, we can see that BER changes dramatically between zero and 68%. This clearly demonstrates the burst error of the wireless channel. BER of wireless channels, as bad as our measurements, are reported in [23][24]. We now examine the error characteristics of the same real channel, and observe the distribution of number of error bits in the corrupt frames. The transmitter transmits 35000 frames with the same frame size, 3750 corrupted frames are received. **Fig. 6** shows the cumulative distribution of BER over all corrupted frames. Most corrupted frames have few error bits: the proportion of corrupt frames with BER less than 0.05 is 88%, and with BER beyond 0.1 is less than 5%. This indicates that there should be large amounts of error-free bits in many corrupted frames.

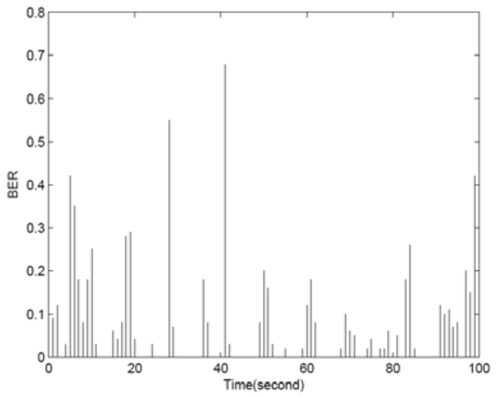

Fig. 5. Measured BER showing the burst error

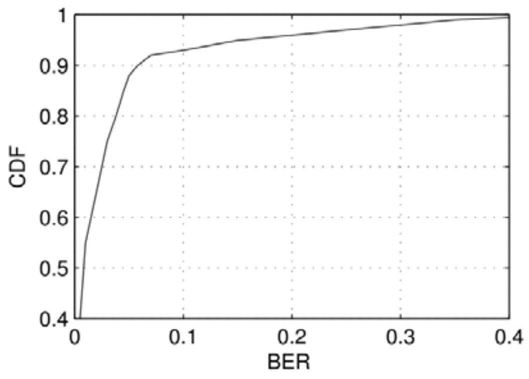

Fig. 6. Empirical CDF of BER

4.2 Channel Utilization Analysis

We first present some analysis of the channel utilization based on some packet-level channel parameters: P_{FER} (fame error rate), P_{FLR} (frame loss rate), P_{FCR} (frame corrupted rate), and P_{KER} (block error rate). Then we derive these parameters from two different bit-level characteristic channel models: BSC (Binary Symmetric Channel) and Gilbert model. We compare the performance of our scheme to the frame-level ARQ scheme by simulation.

It takes an expected $1/(1-P_{FER})$ transmission for the receiver to receive a frame correctly in the frame ARQ scheme. Therefore, the expected channel percentage utilization is:

$$U = \frac{(1-P_{FER})\times S_{data}}{S_{data} + S_{overhead} + S_{ack\_overhead}} \tag{1}$$

Where S_{data}, $S_{overhead}$ and $S_{ack\_overhead}$ are the length of frame data , frame overhead and average ARQ acknowledgement overhead, respectively, in bytes. The frame overhead from the physical and MAC layer is 13 bytes in the TinyOS framing scheme. The average frame acknowledgment overhead can be obtained by the equation: $S_{ack\_overhead} = P_{FER}\times \mathrm{S_{ack\_frame}}$, where, $\mathrm{S_{ack\_frame}}$ is the size of acknowledgement frame, typically 17 bytes.

For our scheme, when an error occurs at the physical or MAC header, the receiver either misses the entire frame, or has to drop the frame, since it cannot be sure of the real sender and the intended receiver of the frame. Therefore, the achievable channel utilization is:

$$U = \frac{(1-P_{FLR})(1-P_{KER})\times n\times S_{b\_data}}{(S_{b\_data} + S_{b\_overhead})\times n + S_{overhead}} \times \frac{1}{1+e_d} \tag{2}$$

Where S_{b_data} and $S_{b\_overhead}$ are block data length and block overhead for each block, respectively, in bytes; n is the number of blocks in each frame. Each block includes block data and a 2-byte block overhead (1-byte sequence number and the 1-byte CRC). This assumption also will hold in the following simulation, since there is a high probability that the original file will be reconstructed when $e_d = 0.05$.

4.2.1 BSC Channel

Let us consider the BSC model that governs the channel. In such a model, each transmitted symbol is erroneous with a certain fixed error probability P_{BER} [25]. Therefore, it is possible to compute the frame error probability P_{FER} when the frame length is known in bits. We consider wireless sensor network radio, a known preamble pattern, which serves to achieve receiver synchronization that precedes each transmitted frame. A node in receiving mode continuously draws in bits from the radio [8]; the entire frame can be delivered to the upper layer once the previously received bits exactly match the preamble sequence; otherwise, the receiver misses the frame, since it cannot be sure of the real sender and the intended receiver of the frame. We assume L_{data}, $L_{overhead}$ and L_{block} are respectively the length of frame payload, frame overhead (including physical preamble and MAC header) and the length of the block, in bits. Some frame-level channel parameters are derived as follows:

- $L_{frame} = L_{overhead} + L_{data}$, where L_{frame} is the total frame length in bits.
- The probability of error-free frame: $P_g = (1-P_{BER})^{L_{frame}}$.
- The probability of payload error: $P_{FCR} = (1-P_{BER})^{L_{overhead}} - P_g$, therefore, the probability of frame reception is $P_g + P_{FCR}$.
- The probability of frame loss: $P_{FLR} = 1 - P_g - P_{FCR} = 1-(1-P_{BER})^{L_{overhead}}$, obviously, we have $P_{FLR} + P_g + P_{FCR} = 1$.
- The overall probability of frame error: $P_{FER} = P_{FCR} + P_{FLR} = 1-(1-P_{BER})^{L_{frame}}$.

- The probability of erroneous block: $P_{KER} = 1-(1-P_{BER})^{L_{block}}$.

We employ three typical BER observed in our channel experiments to simulate channel utilization. **Fig. 7** plots the utilization of frame ARQ for different frame sizes based on simulating over these loss channels. For our scheme, we assume 4 blocks are encapsulated in every frame (n = 4). We also plot the utilization for different block sizes in **Fig. 8**. Compared to the frame retransmission scheme, the optimal achievable utilization of our scheme is increased by more than 17% over different loss channels.

Several interesting observations can be drawn from **Fig. 7** and **Fig. 8**. First, the utilization increases by more than 17% compared to frame ARQ. The utilization of frame ARQ is dependent on the P_{FER} ; while the utilization of our scheme is dependent on P_{KER} . The variation of P_{FER} does not change P_{KER} drastically, since error-free bits in an erroneous frame happen to cluster and can be utilized efficiently in our scheme. Second, we observe that a block size of 20-30 bytes is nearly optimal for most channel conditions. Therefore, we will evaluate our scheme with a block size of 25 bytes in our real evaluation experiments.

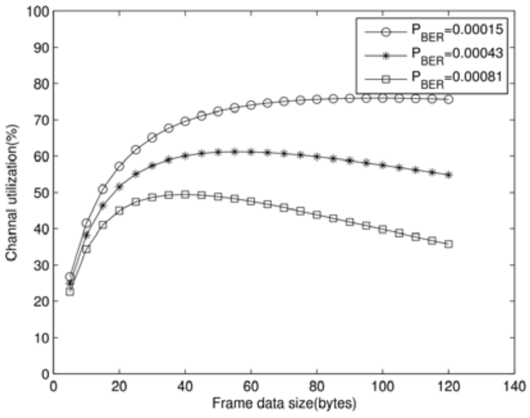

Fig. 7. Channel utilization of ARQ under BSC channel model

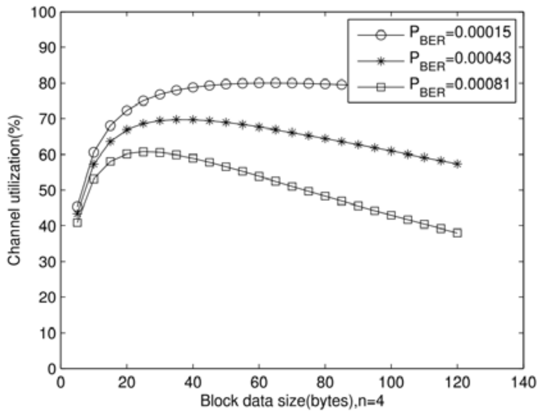

Fig. 8. Channel utilization of our scheme under BSC channel model (4 blocks per frame)

4.2.2 Gilbert Channel

We adopt the well-known two-state Markov chain Gilbert model to derive the frame-level parameters to simulate the performance of our scheme under burst channel. For such a channel, the channel is divided into “good” state and “bad” state: in the good state, only correct bits are generated by the model ($P_g = 0$), in the bad state, bits are incorrect with a certain probability $P_b \approx 1$. We will refer to the model with $P_g = 0$ and $P_b = 1$ as the simplified Gilbert model[2]. We further assume the state transition probabilities are $P_{gg} = \beta$, $P_{gb} = 1-\beta$, $P_{bb} = \alpha$ and $P_{bg} = 1-\alpha$, where P_{xy} denotes the probability of transition from state x to state y (g and b represent the good and bad state, respectively). Therefore, the steady state probabilities of being in states good and bad are $\pi_g = \frac{1-\alpha}{1-\alpha+1-\beta}$ and $\pi_b = \frac{1-\beta}{1-\alpha+1-\beta}$. In the simplified case, the average bit error rate is

$$P_{BER} = P_g\pi_g + P_b\pi_b = \frac{1-\beta}{1-\alpha+1-\beta} \tag{3}$$

[2] In Gilbert-Elliot model, P_g is allowed to be non-zero, and $P_g \in (0,1]$.

In [26], the two parameters (α and β) of the simplified model are expressed in terms of the more meaningful quantities P_{BER} and ρ. This yields

$$\alpha = P_{BER} + \rho(1 - P_{BER}) \tag{4}$$

$$\beta = (1 - P_{BER}) + \rho P_{BER} \tag{5}$$

Where, $\rho = \alpha + \beta - 1$ is the correlation of two consecutive error bits[3]. In a sense, it represents the probability of error burst. Ultimately, the probability that a correct frame is received is shown as follows:

$$1 - P_{FER} = \pi_g P_{gg}^{L_{frame}} = \frac{1-\alpha}{1-\alpha+1-\beta} \beta^{L_{frame}} = (1 - P_{BER})(1 - P_{BER} + \rho P_{BER})^{L_{frame}} \tag{6}$$

The right-hand term of the equation indicates the probability of starting in the good state and remaining in the good state for the entire duration of the frame. The frame loss rate is dependent on frame overhead. Hence, the overall probability of frame reception is given by:

$$1 - P_{FLR} = (1 - P_{BER})(1 - P_{BER} + \rho P_{BER})^{L_{overhead}} \tag{7}$$

Similarly, the probability of a correct block is obtained as follows:

$$1 - P_{KER} = (1 - P_{BER})(1 - P_{BER} + \rho P_{BER})^{L_{block}} \tag{8}$$

We simulate the channel utilization using six typical channel models that we observed from long term experiments. Details of these loss models are shown in **Table 1**. Loss models 1 and 2 have the same mean P_{BER} , but different ρ. Model 2 has bigger ρ, which demonstrates it is higher probability to experience longer consecutive correct or erroneous bits. That is, the channel encounters more burstiness. So do loss models 3 and 4. Loss models 5 and 6 represent scenarios of high P_{BER} .

Table 1. Loss channel model

Loss Model	Mean P_{BER}	ρ
1	0.00015	0.3
2	0.00015	0.9
3	0.00043	0.3
4	0.00043	0.9
5	0.00081	0.3
6	0.00081	0.9

We plot the utilizations of the ARQ scheme and our scheme in **Fig. 9** and **Fig. 10** under different channels. The conclusions from the BSC channel also hold here. In addition, in a low burst channel ($\rho = 0.3$), the utilization gains vary between 12%-15% when the frame data size (or block data size) is 30 bytes; in an extreme burst channel ($\rho = 0.9$), our scheme still achieves up to at least 8% gains, even though the ARQ scheme is considered to "favor" especially high burstiness (compared to FEC). Hence, our scheme can obtain significant performance under different channel conditions. Additionally, some simulations in [21] had shown that the utilization increases monotonically as the number of blocks in one frame increases. Our scheme also holds this result, but only $n = 4$ will be chosen in our evaluation experiments due to

[3] When ρ , the errors of consecutive bits are independent, it can be regarded as a BSC channel.

the limitation of the transceiver buffer of the radio chip.

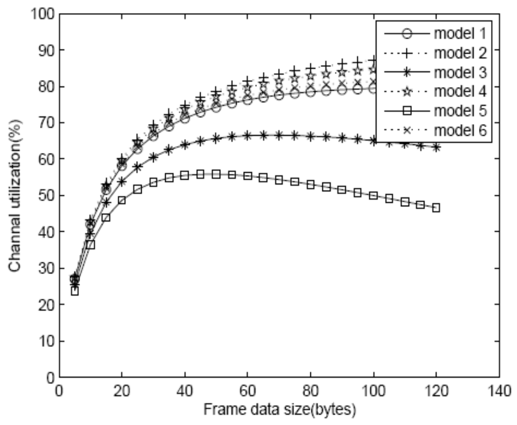

Fig. 9. Channel utilization of ARQ under Gilbert channel model

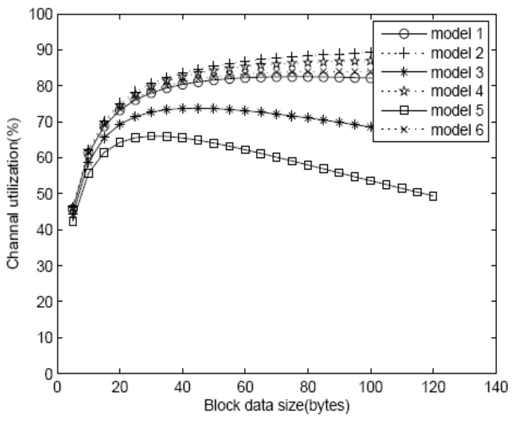

Fig. 10. Channel utilization of our scheme under Gilbert channel model (4 blocks per frame)

5. Performance Evaluation

We have evaluated the performance of our scheme by performing single-hop and multi-hop experiments on a wireless sensor network testbed. Each sensor node in our testbed is a Zigbex mote with a 7.3728MHz ATmega128L microcontroller, 4K Byte internal SRAM, 128K bytes ISP flash memory, and an IEEE 802.15.4 compatible 2.4 GHz Chipcon-CC2420 radio chip. The CC2420 radio chip has a 250 Kbps effective data rate and 128 bytes data transmission buffer. We assume that the source is a traditional mote with adequate memory for encoding. The programs in the motes provide the data link layer and physical layer interface. The sink mote is connected to a PC and records all network activity and each mote in the network logs local statistics and transmits them to the sink after all experimental data are delivered completely.

5.1 Single Hop

Three single-hop experiments are performed between a pair of motes. The first experiment shows the benefits from the optimal block size. We measure the channel utilization for four different block sizes (**Table 2**). The data in the table represents the observed channel utilization over half hour duration. We notice that the optimal utilization for our scheme is achieved with a block data size of 25 bytes. This experimental result is in accordance with our analytical model in the previous section.

Table 2. Percentage utilization of different block sizes

Block Size (Bytes)	Percentage Utilization
15	60.7%
20	67.4%
25	69.2%
30	63.1%

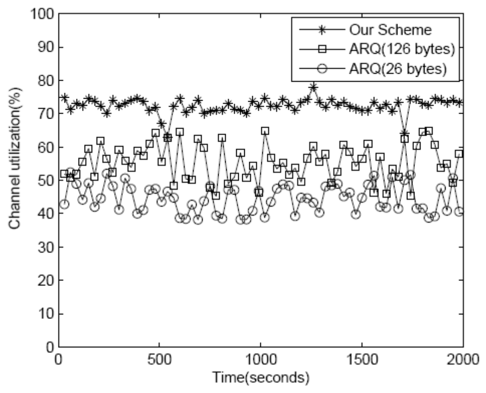

Fig. 11. Channel Utilization Comparison

Our scheme shows optimal and stable channel utilization with respect to frame ARQ in the second experiment. Some remarkable performances are measured in the following experiment on the real channel with a natural $P_{BER} = 0.00052$. We consider frame sizes of 126 bytes and 26 bytes for frame ARQ. A frame size of 126 bytes has a low packet overhead, but high FER. Conversely, a frame size of 26 bytes has a high frame overhead, but low FER. **Fig. 11** plots the utilization of our scheme and frame ARQ over a half hour duration. Each point in the graph is obtained by observations of 30 seconds. Our scheme achieves nearly consistent utilization, while two kinds of frame ARQ schemes have high fluctuations. Meanwhile, by decoupling these two factors of frame size and FER, our scheme is always able to achieve better channel utilization.

The third experiment compares our scheme and frame ARQ in terms of the time to deliver a file; this is related to system utilization efficiency. Our implementation of the encoding and decoding algorithm is based on a PC with 1.7GHz P4 CPU and 1GB RAM. The time for encoding and decoding a reasonable range of file sizes, which might be used for a sensor network, is very small; this is negligible in regard to greater transmission time. Other experiments show that the encoding time for a 1MB size file is less than 0.23 seconds and decoding time is less than 0.29 when the reconstruction factor e_d is 0.05. We measure the file transmission time by our scheme and frame ARQ at the rates of 2, 5 and 10 frames per second. In our scheme, each frame consists of four blocks and the block data size is 25 bytes. Whilst, in the frame ARQ scheme, the entire frame data size is 100 bytes, excluding the physical layer and MAC layer headers. **Table 3** compares the average time taken by our scheme and frame ARQ to transmit a file reliably for different size source files and frame sending rates.

Table 3. Comparison of transmission time for different file sizes and frame sending rates

File Size (K Bytes)	Frame send rate (frames/second)	Our Scheme (seconds)	Frame ARQ (seconds)	Performance Gain
10	2	54.27	69.63	30.0%
	5	23.11	30.43	35.7%
	10	12.54	16.28	36.5%
50	2	258.32	342.25	32.8%
	5	105.47	143.73	37.4%
	10	56.60	76.30	38.5%

We define performance gain as the ratio of the file transmission time decrease between our scheme and the frame ARQ scheme to the file transmission time without any error and delivery overhead, expressed as a percentage. The performance gain is shown in **Table 3**. We notice that, the delivery efficiency of our scheme is consistently better than that of the frame ARQ scheme. The performance gains for delivery of different file sizes were always greater than 30%.

5.2 Multi-Hop

Some evaluation experiments are conducted on 6-hops network consisting of seven motes, where topology is shown in **Fig. 12** and the averaged packet error rate over six runs is shown in **Fig. 13 (a)** and **Fig. 13 (b)**.

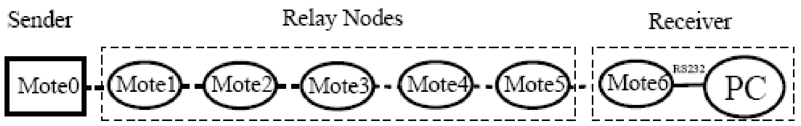

Fig. 12. Multi-hop experimental testbed

In our experiments, different link qualities are obtained by changing the distance between two neighbor nodes, which is often referred to as *Space* . An interesting phenomenon is observed in the multi-hop channel condition experiments. The transmission between any two neighbor nodes with the same space can experience widely varying packet loss rate, shown in **Fig. 13 (a)**. This means there is no deterministic relationship between distance and link quality for only one hop. In the extreme case, neighbors with smaller space experience very high packet loss due to packet collision and the hidden terminals problem. However, accumulative packet loss rate beyond four hops is related to the distance, which is shown in **Fig. 13 (b)**. In ARQ, for a significant packet loss hop, hop-by-hop negative acknowledgement packets will increasingly aggravate link quality of the current hop. **Fig. 14** plots the time to deliver 20K, 50K and 100K byte files using our scheme over the 6-hop network. The delivery time is almost proportional to the file size over several channels. The performance is hardly affected by the significant short-term fluctuating hop-by-hop link quality.

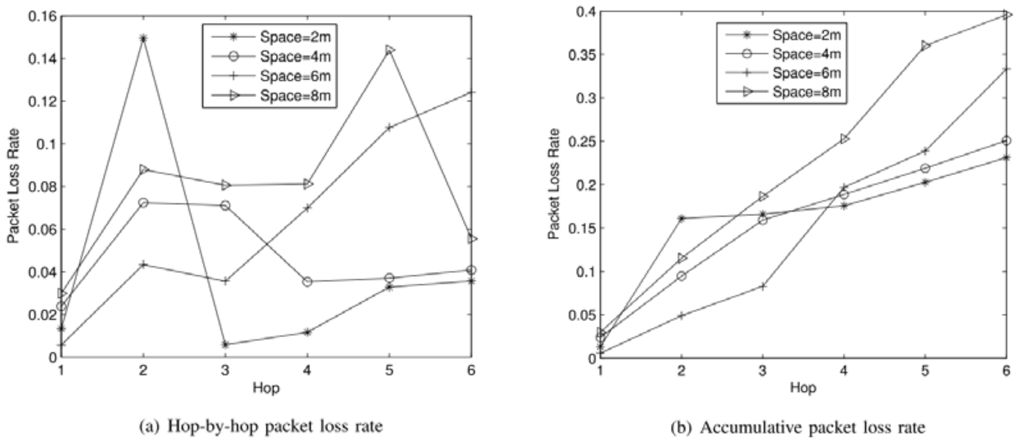

Fig. 13. Experimental channel characteristics

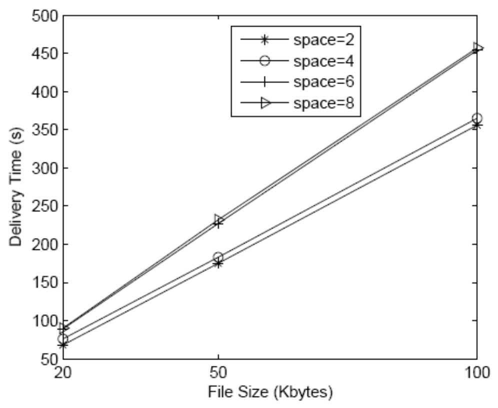

Fig. 14. File delivery time on different spaces over a 6-hop network

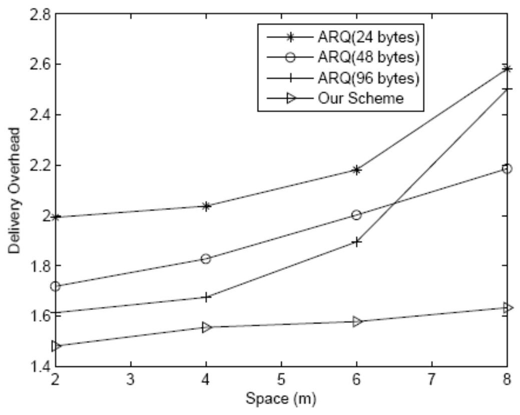

Fig. 15. Delivery overhead comparison between our scheme and different frame size ARQ schemes

Finally, we compare our scheme to ARQ in terms of *delivery overhead* that is relative to channel utilization. Our scheme is expected to have less overhead to transmit the file to the receiver reliably, since our scheme does not require acknowledgement and retransmission whilst recovering correctly received blocks from the error frame. *Delivery overhead* is defined as follows:

$$Delivery\ \ overhead = \frac{total\ number\ of\ bytes\ sent\ by\ all\ nodes}{the\ size\ of\ \ file\ sent \times number\ of\ hops} \tag{9}$$

In our scheme, each frame consisted of four blocks and the block size is 26 bytes (including 2 bytes block overhead). Several ARQ schemes have been conducted with different block sizes. We denote such schemes as ARQ(S), where S is the frame size. The intermediate nodes relay the received frame immediately after detecting and removing the erroneous blocks. Almost the same traffic load is employed in all experiments for a fair comparison. For example, in ARQ (96), four frames are transmitted per second, whilst eight frames are transmitted in ARQ (48). A 100K bytes file is delivered over our 6-hop network by each scheme. For our scheme, the sender transmits the encoding packets until the file can be decoded in the receiver. In the ARQ schemes, the entire file also is ensured to be recovered in the receiver by hop-by-hop reliable packet delivery. The experimental results are shown in **Fig. 15**. In multi-hop experiments, our scheme always has lower delivery overhead than the three kinds of ARQ schemes.

When space is more than 6m with higher packet loss rate, the delivery overhead of ARQ (96) increases rapidly, while our scheme shows a slight increase in delivery overhead. The performance gains of our scheme are more significant than obtained in the previous single hop simulation. This can be explained as follows. For the ARQ scheme, in a high loss scenario, more and more acknowledgements aggravate the network due to more packet collision. This collision leads that the ARQ scheme experiencing high FER, while FER in our scheme is less affected in a similar network experiment environment. Therefore, our scheme is expected to achieve better performance than the ARQ scheme in a higher packet loss rate network.

6. Conclusions

In this paper, we introduced a novel scheme for bulk data transmission in sensor networks. We describe the system design and implementation. We construct a new data link layer architecture and data frame format based on the erasure-resilient codes feature. This is able to simplify transmission protocol and does not require the retransmission traffic. As a result, the frame can be sufficiently large to ensure high utilization of the wireless channel, without sacrificing error recovery effectiveness. The contribution of our work is to introduce a new level of FEC, while previous approaches are classified as a bit level FEC and a packet level FEC. Our scheme uses a block level FEC, where a frame consists of some number of blocks. We can present throughput improvement provided by our scheme by analyzing performance based on models derived from real measurements of wireless channel errors. Our evaluation confirmed that the scheme improves throughput by at least 20% over a typical wireless sensor network channel. In multiple hop experiments, our scheme reduces the total number of bytes to deliver in the sensor network for reliable transmission by more than 60% compared to a frame-based retransmission scheme.

References

[1] R. E. Azouzi, T. Peyre, and A. Benslimane, "Optimal design of hybrid fec/arq schemes for real-time applications in wireless networks," in *Proc. of the 2nd ACM international workshop on Wireless multimedia networking and performance modeling*, pp.11-18, Oct. 2006.

[2] J. Paek and R. Govindan, "Rcrt: rate-controlled reliable transport for wireless sensor networks," in *Proc. of the 5th international conference on Embedded networked sensor systems*, pp.305-319, 2007.

[3] D. A. Eckhardt and P. Steenkiste, "A trace-based evaluation of adaptive error correction for a wireless local area network," *Mobile Networks and Applications*, vol.4, no.4, pp.273-287, Dec. 1999.

[4] M. Kobayashi and G. Caire, "A low-complexity approach to space-time coding for multipath fading channels," *EURASIP Journal on Wireless Communications and Networking*, vol.5, no.3, pp.437-446, Aug. 2005.

[5] G. Ferrari, P. Medagliani, S. D. Piazza, and M. Martal, "Wireless sensor networks: performance analysis in indoor scenarios," *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, no.1, Jan. 2007.

[6] P. Lettieri, C. Schurgers, and M. Srivastava, "Adaptive link layer strategies for energy efficient wireless networking," *Wireless Networks*, vol.5, no.5, pp.339-355, Oct. 1999.

[7] G. Bianchi, F. Formisano, and D. Giustiniano, "802.11b/g link level measurements for an outdoor wireless campus network," in *Proc. of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks*, pp.525-530, 2006.

[8] H. Dubois-Ferriere, D. Estrin, and M. Vetterli, "Packet combining in sensor networks," in *Proc. of ACM Sen-Sys 2005*, Nov. 2005.

[9] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer, "Understanding link-layer behavior in highly congested ieee 802.11b wireless networks," in *Proc. of SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND)*, pp.11-16, Aug. 2005.

[10] A. K. Miu, H. Balakrishnan, and C.E. Koksal, "Improving loss resilience with multi-radio diversity in wireless networks," in *Proc. of ACM MobiCom 2005*, pp.16-30, Sep. 2005.

[11] M. Chen, G. Wei, "Multi-Stages Hybrid ARQ with Conditional Frame Skipping and Reference Frame Selecting Scheme for Real-Time Video Transport Over Wireless LAN," *IEEE Transaction on Consumer Electronics*, vol. 50, no.1, pp.158-167, Feb. 2004.

[12] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp.569-584, Feb. 2001.

[13] M. Luby, "LT codes," in *Proc. of The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[14] P. Maymounkov and D. Mazi`eres, "Rateless Codes and Big Downloads," in *Proc. of Int'l Workshop on Peer-to-peer Systems (IPTPS)*, 2003.

[15] T. T. E. Hyyti and J. Virtamo, "Optimizing the degree distribution of lt codes with an importance

sampling approach," in *Proc. of RESIM*, Oct. 2006.
[16] J. S. Ahn, S. W. Hong, and J. Heidemann, "An adaptive fec code control algorithm for mobile wireless sensor networks," *Journal of Communications and Networks*, vol. 7, no. 4, pp. 489-499, 2005.
[17] K. Jamieson and H. Balakrishnan, "Ppr: partial packet recovery for wireless networks," in *Proc. of ACM SIGCOMM*, vol. 37, no .4, pp. 409-420, Oct' 2007.
[18] D. Chase, "Code combining: A maximum-likelihood decoding approach for combining an arbitrary number of noisy packets," *IEEE Trans. on Comm.*, vol.33, no.5, pp. 385-393, May 1985.
[19] J. Metzner, "Improvements in block-retransmission schemes," *IEEE Trans. on Comm.*, vol. 27, no. 2, pp. 524-532, Feb. 1979.
[20] S. Lin and P.S. Yu, "A hybrid arq scheme with parity retransmission for error control of satellite channels," *IEEE Trans. on Comm*., vol. 30, no. 7, pp. 1701-1719, July 1982.
[21] R. Ganti, P. Jayachandran, H. Luo, and T. Abdelzaher, "Datalink streaming in wireless sensor networks," in *Proc. of ACM SenSys*, Nov. 2006.
[22] D. Skordoulis, Q. Ni, H. H. Chen, A. P. Stephens, C. Liu, and A. Jamalipour, "IEEE 802.11n MAC frame aggregation mechanisms for next-generation high-throughput WLANs," *IEEE Wireless Comm.*, vol. 15, no. 1, pp. 40-47, 2008.
[23] K. l. Blackard, T. S. Rappaport, and C. W. Bostian, "Measurements and models of radio frequency impulsive noise for indoor wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 7, pp. 991-1001, Sep.1993.
[24] D. Echhard and P. Steenkiste, "Measurement and analysis of the error characteristic of in-building wireless network," in *Proc. of ACM SIGCOMM*, pp. 243-254, Aug. 1996.
[25] H. Karl and A. Willig, "Protocols and architectures for wireless sensor networks," John Wiley and Sons Ltd, 2005.
[26] J. R. Yee and E. J. Weldon, Jr., "Evaluation of the performance of error-correcting codes on a gilbert channel," *IEEE Tran. on Comm.*, vol. 43, no. 8, Aug. 1995.

Jian-Jun Lei received a B.S. from Chongqing Institute of Technology, China, in 2000, and a M.S. from the Chongqing University of Posts and Telecommunications, in 2006. He is currently a Ph.D. candidate in the Department of Computer and Information Engineering, INHA University, Incheon, Korea. His research interests include wireless communication and wireless sensor networks.

Gu-In Kwon received a M.A. degree in computer science from City University of New York (Queens College) in 1998, and a Ph.D. in computer science from Boston University, Boston, MA, in January 2005. He is Assistant Professor in the Department of Computer and Information Engineering, INHA University, Incheon, South Korea. His research is in wireless sensor networks, Internet content delivery, multicast congestion control, and overlay networking.