Exploiting Concurrency for Efficient Dissemination in Wireless Sensor Networks

Yi Gao, *Student Member*, *IEEE*, Jiajun Bu, *Member*, *IEEE*, Wei Dong, *Member*, *IEEE*, Chun Chen, *Member*, *IEEE*, Lei Rao, *Member*, *IEEE*, and Xue Liu, *Member*, *IEEE*

Abstract—Wireless sensor networks (WSNs) can be successfully applied in a wide range of applications. Efficient data dissemination is a fundamental service which enables many useful high-level functions such as parameter reconfiguration, network reprogramming, etc. Many current data dissemination protocols employ network coding techniques to deal with packet losses. The coding overhead, however, becomes a bottleneck in terms of dissemination delay. We exploit the concurrency potential of sensor nodes and propose MT-Deluge, a multithreaded design of a coding-based data dissemination protocol. By separating the coding and radio operations into two threads and carefully scheduling their executions, MT-Deluge shortens the dissemination delay effectively. An incremental decoding algorithm is employed to further improve MT-Deluge's performance. Experiments with 24 TelosB motes on four representative topologies show that MT-Deluge shortens the dissemination delay by 25.5-48.6 percent compared to a typical data dissemination protocol while keeping the merits of loss resilience.

Index Terms—Wireless sensor networks, network protocols

1 Introduction

Wireless sensor networks (WSNs) can be successfully applied in a wide range of applications such as military surveillance [1], forest protection [2], volcano monitoring [3], habitat monitoring [4], etc. After the network is deployed, the network operators often need to disseminate data to a network of sensors, enabling software bug fixing, security holes patching, parameter reconfiguration [5], sensor network reprogramming [6], etc. Therefore, data dissemination is an essential building block of WSN applications.

There are many bulk data dissemination protocols [5], [7], [8], [9], [10], [11], [12], [13], [14], [15] in the literature. Existing approaches can be divided into two categories: non-coding-based approaches and coding-based approaches. The performance of non-coding-based approaches [7], [8], [14], [15] degrades in lossy networks with unreliable wireless links. It is reported in a survey [6] that the completion time of Deluge [7], a popular protocol, is almost an hour on a network with 100 nodes. The reason is that when the link qualities become poor, the number of retransmissions grows dramatically which further makes the data packets easy to be collided. Coding-based approaches [9], [10], [11] employ network coding techniques [16] to encode the native packets into encoded packets at the sender side. The salient feature of this technique is that the sender does not retransmit all lost packets of its receivers, but encodes some new encoded packets and retransmits them. When multiple receivers obtain enough encoded packets, they can decode them and obtain the native packets. The coding-based approaches can reduce the retransmissions significantly. However, the encoding and decoding introduce computation overhead and can be time consuming. For example, decoding a page consisting of 48 packets in a coding-based approach [10] costs 6.96 seconds.

We observe that when a sensor node is sending or receiving packets, the radio is busy but the CPU utilization is low; and when the sensor node is encoding or decoding packets, the CPU utilization is high but the radio is nearly idle. These two observations inspire us to exploit the concurrency potential of sensor nodes. We propose MT-Deluge, a novel multithreaded design, in this paper. MT-Deluge exploits the concurrency potential of sensor nodes by separating the coding operation and radio operation into different threads. With a multithreaded design, we are able to amortize the overheads of encoding and decoding into the idle-wait durations during the radio operations, so as to reduce the dissemination delay of coding-based approaches.

Integrating multithreading with current coding-based approaches has several practical challenges. First, when a node receives a sufficient number of packets, how to recover the native packets as soon as possible? Second, when multithreading is utilized, how to synchronize multiple threads correctly and efficiently? To address the first challenge, an incremental decoding algorithm is proposed. Instead of starting the decoding procedure after receiving all packets, the receiver starts to decode after obtaining a small number of packets. The incremental decoding

- *Y. Gao is with the Zhejiang Provincial Key Laboratory of Service Robot, College of Computer Science, Zheda Road 38, Hangzhou 310027, Zhejiang University, China, and the School of Computer Science, McGill University, Montreal, QC H3A 0E9, Canada. E-mail: gaoyi@zju.edu.cn.*
- *J. Bu, W. Dong, and C. Chen are with the Zhejiang Provincial Key Laboratory of Service Robot, College of Computer Science, Zhejiang University, Zheda Road 38, Hangzhou 310027, China. E-mail: {bjj, dongw, chenc}@zju.edu.cn.*
- *L. Rao is with the General Motors Electrical & Controls Systems Research Lab, Palo Alto, CA 94306. E-mail: lei.rao@gm.com.*
- *X. Liu is with the School of Computer Science, McGill University, 3480 University Street, Room 318, Montreal, QC H3A 0E9, Canada. E-mail: xueliu@cs.mcgill.ca.*

Manuscript received 17 Nov. 2011; revised 12 June 2012; accepted 13 June 2012; published online 22 June 2012.
Recommended for acceptance by L.E. Li.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-11-0841.
Digital Object Identifier no. 10.1109/TPDS.2012.195.

algorithm can make use of the idle durations in receiving, mitigating the impact of decoding to dissemination delay. To address the second challenge, we introduce an adaptive packet-level synchronization to synchronize multiple threads correctly. Besides, when the receiver is receiving data packets from the sender, the start time of the decoding procedure is optimized to get best performance.

In this paper, we provide a detailed description of the implementation of MT-Deluge. In particular, the incremental decoding algorithm is highlighted since it plays a key role to shorten the dissemination delay in the whole design. We evaluate the efficiency of MT-Deluge in a testbed consisting of 24 Telosb [17] motes. Experiments on four different topologies show that MT-Deluge shortens the dissemination delay by 25.5-48.6 percent compared to a typical data dissemination protocol while keeping the merits of loss resilience.

Compared with our preliminary version of MT-Deluge [18], this paper includes many important extensions as follows:

- We describe the motivation of MT-Deluge in a new section. Two motivating examples are analyzed more comprehensively.
- We introduce the related works in detail.
- We analyze the impacts of several important parameter settings.
- For single-hop networks, we improve the protocol and evaluate this improvement.
- We give the methodology of the experimental study.
- We conduct more experiments to evaluate MT-Deluge and reveal more system insights.

The rest of this paper is organized as follows: Section 2 introduces the related work. The background and two illustrative motivating examples are given in Section 3. Section 4 gives an overview of MT-Deluge. Section 5 describes the detailed design and implementation. Section 6 gives the real testbed evaluation results and demonstrates that MT-Deluge shortens the dissemination delay by 25.5-48.6 percent. Finally, Section 7 concludes the paper and presents the future work.

2 Related Work

Data dissemination is a fundamental building block in WSNs. Many approaches have been proposed during the past decade. Existing approaches can be divided into two categories, i.e., non-coding-based and coding-based approaches. Deluge [7] is the *de facto* standard non-coding-based dissemination protocol in TinyOS. In Deluge, every sensor node advertises the version number of its current program, then nodes with older version programs request program updates. Then, these receivers get the data packets from the sender and uses an NACK-based mechanism to improve the reliability. The program image is divided into fixed size pages which are further divided into native packets to send. When a sensor node has received an entire page, it forwards this page to other nodes before receiving the next page. This mechanism enables spatial multiplexing which reduces the total delay significantly. MNP [8] is another non-coding-based approach which employs distributed sender selection to improve the dissemination efficiency. In addition, MNP employs sleep scheduling to reduce the energy cost. In order to speed up the dissemination procedure, Stream [15] reduces the code size by preinstalling the dissemination protocol on the external flash beforehand. Elon [19] introduces replaceable component to further reduce the transferred code size.

Because packets can be lost in wireless networks with unreliable links, retransmissions are necessary to provide high reliability. The performance of non-coding-based approaches degrades rapidly in lossy networks. In order to reduce the number of retransmitted packets, network coding techniques have been widely used in WSNs. Rateless Deluge [10] is a typical coding-based data dissemination protocol. In Rateless Deluge, random linear coding is used to reduce the retransmission overhead and two additional optimizations (i.e., precoding and ACKless transmission) further improve the performance. It is reported in [10] that Rateless Deluge can get 15-30 percent savings in the data plane and 50-80 percent in the control plane than original Deluge. Adapcode [12] also employs random linear codes. Unlike Rateless Deluge, the receivers in Adapcode collect encoded packets from multiple senders instead of a single sender. Synapse [9] is another coding-based approach which uses Fountain Codes. All these approaches need to perform Gaussian elimination in the decoding phase which is time consuming on current resource-constrained sensor nodes. Our previous work ReXOR [11] uses XOR-based coding technique to improve the decoding efficiency.

Different from these previous works, MT-Deluge exploits the concurrency potential of sensor nodes and is able to shorten the dissemination delay effectively.

3 Background and Motivation of MT-Deluge

In this section, we first briefly describe a coding-based approach, Rateless Deluge [10], in order to give a background of our approach. Then, we demonstrate the motivation of MT-Deluge by several illustrative examples.

3.1 Background on Random Linear Code

Without loss of generality, we take Rateless Deluge [10] as an example. Rateless Deluge employs random linear code to encode native packets before transmission. Receivers can recover the native packets when a sufficient number of encoded packets have been received. Rateless Deluge exploits spatial diversity so that a single encoded packet can be recovered at multiple receivers, reducing the total number of transmissions. The main workflow of Rateless Deluge is described as follows [10].

At the sender side, a long page X is divided into k native packets $\{X_1, X_2 \cdots X_k\}$. These k native packets are further encoded into $m > k$ encoded packets $Y_1, Y_2 \cdots Y_m$ which are linear combinations of the native packets, i.e., $Y_i = \Sigma_{j=1}^{k} \beta_{i,j} X_j$, where $\beta_{i,j}$ are randomly chosen numbers. At the receiver side, the encoded packets can be decoded by solving the set of linear equations by Gaussian elimination with back substitution after k linearly independent packets have been received. The benefit of network coding is shown in the following example. Suppose three receivers only get

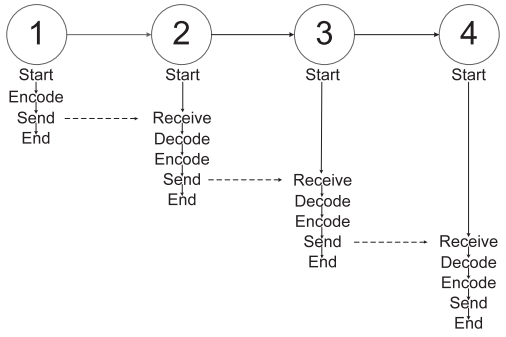


Fig. 1. Example 1, sequential workflow of Rateless Deluge.

$k-1$ packets (the lost packets are distinct). In non-coding-based approaches, the sender needs to retransmit all three packets. In Rateless Deluge, the sender only needs one additional encoded packet as long as it is linearly independent with the previous $k-1$ packets. After decoding, the receivers obtain the native packets and become new senders for other nodes.

By applying the random linear code, Rateless Deluge is able to reduce the retransmissions significantly. However, the encoding and decoding introduce computation overhead which will prolong the dissemination delay directly.

3.2 Motivating Examples

The first example demonstrates how the computation overheads affect the dissemination delay and the potential of using multithreading design in coding-based data dissemination. Network coding reduces the number of retransmitted packets, but it also introduces computation overheads. The time complexity of Gaussian elimination involved in the decoding procedure is $O(k^3)$. As reported in [10], on Tmote-sky motes, decoding a page consisting of 48 packets costs 6.96 seconds. Long decoding delay is a bottleneck in terms of the overall dissemination delay. In Example 1, we set up a small network consisting of four sensor nodes and these nodes sequentially obtain a series of packets using random linear codes. Fig. 1 shows the behaviors of the sensor nodes during the experiment. We can see that every encode/decode operation prolongs the dissemination delay directly. We also measure the average CPU utilization of these four sensor nodes. We modify the basic scheduler implementation (i.e., SchedulerBasicP) of TinyOS 2.1 to make the scheduler track the task running time. Then, the task running time is accessed by a new interface in the scheduler implementation. The CPU utilization can be easily obtained by dividing the task running time by the total running time. The CPU utilization is below 14 percent when nodes are performing radio operations. If we could make good use of the computation ability during the radio operations for coding purpose, we can mitigate the impact of coding overhead to the dissemination delay. In MT-Deluge, the computation overheads are amortized into the packet sending/receiving procedure and hence the dissemination delay is shortened significantly.

One could argue that the receivers can simply forward the received encoded packets before decoding them. Then, the decoding overhead does not contribute to the dissemination delay. The second example shows this straightforward method has severe problems. In Example 2, we consider another small network as shown in Fig. 2, node s transmits four packets (i.e., p_1 to p_4) to nodes r_1, r_2, and r_3 using random linear codes. When node r_1 receives four encoded packets (i.e., p_1^* to p_4^*) from node s, it immediately forwards these four encoded packets to nodes r_2 and r_3 without decoding them. However, this solution can hardly obtain the performance gains of network coding in practice. For example, when node r_2 and node r_3 lose one distinct packet (e.g., node r_2 loses p_3^* while node r_3 loses p_2^*), node r_1 has several choices here.

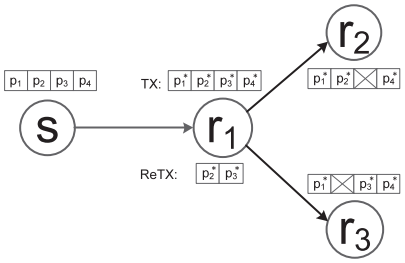


Fig. 2. Example 2, a network consisting of four nodes, p_i are native packets, p_i^* are encoded packets.

1. Retransmits two encoded packets, p_3^* and p_2^*. This is similar as Deluge, in this case there are no benefits of network coding.
2. Decodes the four encoded packets and generates a newly encoded packet p_5^* and then transmits it to the two receivers. This is similar as Rateless Deluge, the decoding delay directly contributes to the dissemination delay.
3. Encodes the lost encoded packets p_2^* and p_3^* into two *reencoded* packets p_5^{**} and p_6^{**} and transmits them. Then, node r_2 has $p_1^*,\ p_2^*,\ p_4^*\ p_5^{**}$, and p_6^{**}, node r_3 has $p_1^*,\ p_3^*,\ p_4^*\ p_5^{**}$, and p_6^{**}. Both node r_2 and r_3 have five unknown variables (e.g., r_2 does not know $p_1,\ p_2,\ p_3,\ p_4$, and p_3^*) and five equations, so the native packets can be obtained. However, as we can see in this example, node r_1 still needs to retransmit two packets, no benefits of network coding. When there are more than two receivers and the lost packets satisfy some conditions, the number of retransmissions can de reduced. Scalability is another major problem of this *reencode* mechanism. Consider a network with more than two hops, the sensor nodes will encode packets more than twice (e.g., $p_7^{***},\ p_8^{****}$, etc.). The decoding overhead will be huge for nodes far from the sink node. Therefore, this mechanism can hard perform efficiently in multihop networks.

In general, simple patching to the original mechanism does not fundamentally address the irreconcilability of coding complexity and dissemination efficiency.

4 MT-Deluge Overview

MT-Deluge exploits the concurrency ability of the sensor nodes and speeds up the dissemination of coding-based protocols. In this section, we give an overview of MT-Deluge. In MT-Deluge, when a code image needs to be disseminated,

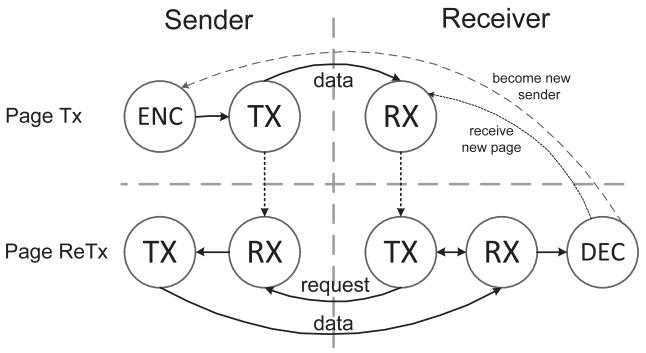


Fig. 3. State transition diagram of a coding-based protocol.

the image is divided into many pages. Each page is further split into a series of packets.

4.1 Background of Single-threaded Design

Before describing MT-Deluge, we introduce the workflow of a coding-based protocol of a single thread. Fig. 3 depicts the main state diagram transition of a coding-based protocol. Every sensor node advertises about its local images periodically. When a node learns that one of its neighbor nodes has more available pages than itself, the node will start sending request to its neighbor. This advertise-request mechanism allows every node in the sensor network to know the current information about its neighbors. As shown in Fig. 3, the workflow of a typical coding-based data dissemination protocol is divided into two phases: page transmission and page retransmission.

Page transmission: Fig. 3 depicts three different states of the sender and receiver. There is one more state, $MAINTAIN$ which is not shown. The sender and receiver finish the advertise-request operations in $MAINTAIN$ state and we do not include this $MAINTAIN$ state to avoid complexity. After the advertise-request procedure, the sender enters the first state, Encoding (ENC). The k native packets belonging to one page are encoded using different algorithms in different protocols. For example, Rateless Deluge [10] and Adapcode [12] use random linear codes, Synapse [9] uses Fountain codes, ReXOR [11] uses XOR-based codes to encode the native packets into encoded packets. In Rateless Deluge, after the k native packets are encoded, the sender enters into Transmission (TX) state and transmits the encoded packets to the receiver. At the same time, the receiver is in Receiving (RX) state and receives the encoded packets.

Page retransmission: Due to unreliability of wireless links, the receiver may lose several encoded packets. Therefore, the page retransmission phase is needed for error recovery. Unlike the first phase, this phase is initiated by the receiver. The receiver enters into the TX state and transmits a request including the information of the lost packets. Since the page number and packet number are included in every packet, the receivers can obtain the information of the lost packets easily. After the sender receives the request in the RX state, the sender enters into the TX state again and transmits the requested packets to the receiver. At the same time, the receiver switches to the RX state to receive the requested packets. If the receiver has obtained enough packets for decoding, it enters into the Decoding (DEC) state and starts to decode. Otherwise, the receiver enters into the TX state and starts to send the request again.

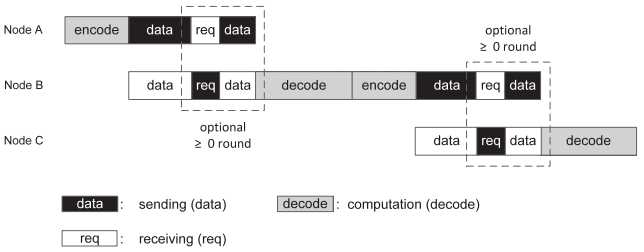


Fig. 4. Workflow of a single-threaded design.

These two phases constitute the workflow of one page transmission. A large code image usually has many pages. To disseminate an entire image, all the pages need to be disseminated. After the decoding procedure is finished, there are two possibilities. 1) the sender starts to transmit the next page, and 2) the receiver becomes a new sender and start to transmit pages to other sensor nodes. In this paper, we focus on accelerating the one page transmission. The multipage dissemination is accelerated as well and its evaluation results will be given in the evaluation section.

4.2 Multithreaded Design Overview

We propose MT-Deluge to improve the efficiency of coding-based dissemination protocols. The intuition of multi-threaded design is to allow the coding operations and the radio operations to work concurrently. Note that in the multithreaded design, a sensor node can decode and send packets simultaneously, so there is no separate TX state or DEC state. Therefore, we use the time line figure to describe the workflow. Figs. 4 and 5 use time lines to describe the one page transmission of single-threaded protocol and multithreaded protocol. In Figs. 4 and 5, the black box represents the sending operation, the white box represents the receiving operation, and the gray box represents coding operations (i.e., encoding and decoding). This convention is also used in several following figures. There are three nodes. Node A sends one page to node B and node B sends the same page to node C. In the single-threaded design as shown in Fig. 4, node B only starts to send the page to node C after decoding and encoding. In the multithreaded design as shown in Fig. 5, a radio thread and a computation thread are executed concurrently. Node B

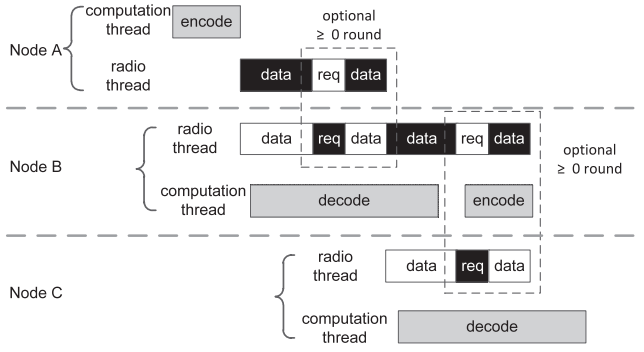


Fig. 5. Workflow of a multithreaded design.

can forward the page to node C once it has collected a small number of encoded packets sent by node A. The computation thread can start to decode even earlier than the sending thread. Once node B has obtained several packets, the computation thread can start to decode. Once node B has obtained enough encoded packets, the decoding procedure ends after a short period of time. Therefore, the page dissemination time can be shortened.

It is worth noting that Fig. 5 only illustrates an ideal situation. In MT-Deluge, we need to consider several practical design issues. For example, when node C loses some packets and sends a request to node B, node B may have not finished the decoding and encoding procedures. Then, node C must wait for node B. We will elaborate on this issue in more details in Section 5.

4.3 Improvement for Single-Hop Networks

We design MT-deluge mainly for multihop networks which are widely used in all kinds of applications. For single-hop networks, we make some modifications to make MT-Deluge work more efficiently. The detailed description of this improvement can be found in Section 1 of the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.195.

5 Multithreaded Design and Implementation

The implementation of multithreaded design has two main challenges. One is to shorten the total dissemination delay by minimizing the waiting time throughout the dissemination procedure. The other is to synchronize different threads running on each sensor node correctly and efficiently. To address the first challenge, we propose an incremental decoding algorithm to start the decoding procedure earlier. To synchronize multiple threads correctly, a packet-level synchronization mechanism is employed. In addition, an adaptive page size adjustment scheme is utilized to optimize the page size given the link quality. We implement MT-Deluge based on Rateless Deluge [10], a typical coding-based data dissemination protocol which has both encoding and decoding procedures. MT-Deluge changes the page transfer mechanism from single-threaded design to multithreaded design. In this section, we describe the solutions and the implementation details to address the challenges.

5.1 Incremental Decoding Algorithm

In Rateless Deluge, a page is decoded after the receiver has received all encoded packets. After decoding, the native packets are obtained. Then, the receiver encodes the native packets and sends them to other nodes. In MT-Deluge, the multithreaded design allows the computation thread and radio thread to run concurrently. Therefore, an obvious benefit is that the receiver can just forward the received encoded packet to other nodes and decode the encoded packets at the same time. Fig. 6 shows node B's workflow using this design.

This workflow is straightforward when multithreaded design is employed. However, it causes the radio thread to wait a relatively long time period which is shown as a long

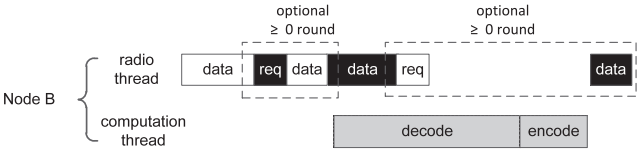


Fig. 6. Workflow of node B without incremental decoding.

blank area in Fig. 6. We can see that there are two *black boxes* which mean two different procedures of "sending data." The first "sending data" procedure is to forward the encoded data packets to other nodes. These encoded packets are encoded by the sender (node A), thus node B can forward them without waiting for the decoding and encoding procedures. Unlike the first one, the second *black box* procedure is a data retransmission procedure. As mentioned in the Section 1, a node cannot forward the encoded packets from the sender in data retransmission procedure because the encoded packets have already been received by the receivers with high probability. Thus, the node has to encode native packets to retransmit. Therefore, as shown in Fig. 6, the second "sending data" has to wait for the decoding and encoding procedures.

Comparing Fig. 6 with Fig. 5, it is easy to see that in Fig. 5, the decoding procedure starts earlier and the total time cost is lower. The reason is that an incremental decoding algorithm is employed in MT-Deluge. The basic idea is to start the decoding procedure after a small number of encoded packets have been obtained instead of starting it after *all* encoded packets have been obtained. Next, we describe the decoding procedure in Rateless Deluge generally and give the detailed description of the incremental decoding algorithm in MT-Deluge.

In Rateless Deluge, the decoding procedure consists of three parts: generating the coefficients matrix, getting the echelon matrix by Gaussian elimination, and performing back substitution. The receiver needs to recover the coefficients matrix that the sender uses by the initial seeds attached in the encoded data packets. Then, Gaussian elimination is used to get the echelon matrix, followed by the back substitution which recovers the native packets. In MT-Deluge, the whole incremental decoding is divided into four procedures. The description and the analysis of the Algorithm are given in Section 2 of the supplementary file, available online.

5.2 Multithread Synchronization

We use an adaptive packet-level synchronization mechanism to provide correct and efficient synchronization between multiple threads. The mechanism has three key features. First, packet-level synchronization between the radio thread and the computation thread provides high precision. This feature ensures the correctness. Second, the actual workflow of each sensor node is adaptively chosen at the runtime. Third, different page sizes are chosen according to different link qualities in order to shorten the total delay. These two features improve the efficiency.

Compared with packet-level synchronization, state-level synchronization is another choice in the implementation. In fact, we use state-level synchronization to synchronize radio thread and computation thread outside the incremental

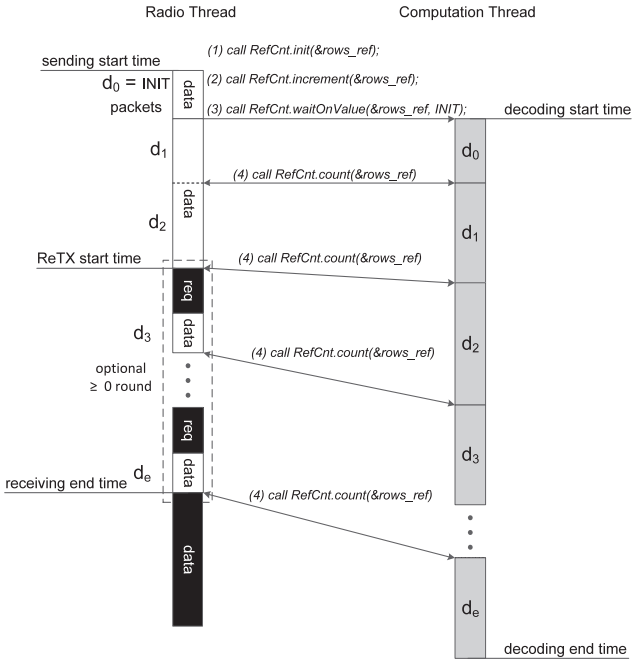


Fig. 7. Packet-level synchronization details.

decoding phase. If incremental decoding is not performed, state-level synchronization will be sufficient. For example, as shown in Fig. 3, when the sender enters state TX from state ENC, it can use synchronization primitives (e.g., a semaphore). However, when incremental decoding is used, state-level synchronization is not sufficient because it does not have any separate states anymore. For example, when the node is receiving packets, it is probably doing decoding at the same time. It is hard to say whether the node is in RX state or DEC state. The workflow and some related code of incremental decoding using packet-level synchronization are depicted in Fig. 7. As same as previous figures, the black box represents the sending operation, the white box represents the receiving operation and the gray box represents coding operations (i.e., encoding and decoding). In the current implementation, we use a synchronization primitive in TOSThreads [20] (an multithread extension of TinyOS) named $ReferenceCounter$ ($RefCnt$ in short) to record the received number of packets. There are four operations regarding $RefCnt$. Operation (1) initializes the value of $RefCnt$ to zero. Operation (2) increases the value of $RefCnt$ when a new encoded packet is received. Operation (3) waits till $INIT$ packets have been received. Operation (4) is used to get the current number of received packets. d_1 to d_e represent all the encoded packets received and each of them represents a segment of the page.

The $INIT$ value indicates the number of received packets before the decoding procedure starts ($INIT \in [1, k]$, k is the page size). Smaller $INIT$ can make the start time of decoding earlier. However, setting the $INIT$ too small will introduce too much synchronization overhead. As described in Fig. 7, the $waitOnValue$ method will be called many times if the $INIT$ is set to be too small. In Section 6.2, a series of experiments are conducted to evaluate this impact of $INIT$ value to the decoding speed. Therefore, there is a tradeoff between the start time and the overhead. In our implementation, we find out that setting $INIT$ to $k/3$ can get good performance.

d_0 (equals to $INIT$) is set before the program runs, but d_2 to d_e are chosen at runtime. Operation (4) is used to get the current number of received packets and these newly received packets are incrementally decoded subsequently. This adaptive mechanism to determine the number of the next packet segment can avoid wait time of both the radio and computation threads. In addition, link quality can affect the packet receiving progress. Retransmission is needed when link quality is poor, which makes the packet receiving progress much slower. Link quality can only be obtained after the program runs. Therefore, the adaptive mechanism can also work well under different link qualities.

5.3 Page Size Adjustment

The last feature is that we set different k (i.e., page size) under different link qualities. Detailed description of this adjustment can be found in Section 3 of the supplementary file, available online.

6 Experimental Study

In this section, we conduct experiments in a testbed consisting of 24 TelosB [17] motes. The performance of MT-Deluge is evaluated in four different network topologies.

6.1 Methodology

We implement MT-Deluge on TinyOS/TelosB platform and evaluate the page transmission efficiency. The experiments are conducted based on a multihop line topology, a multihop grid topology, a single-hop clique topology, and a random distributed topology. Multihop line topologies are widely used in wireless sensor network applications due to the limited transmission ranges of sensor nodes. Multihop line topology is usually used when the deployment field is a long strip (e.g., a bridge [21]). Multihop grid topology is a more representative topology in real applications. Single-hop clique topology is usually used in lab testing and some small scale applications. Random distributed topology is a common topology used in real applications. We evaluate one page transmission for the first three topologies to show the efficiency of MT-Deluge. In the random distributed topology, we evaluate the multipage transmission efficiency.

We compare Deluge, Rateless Deluge, and MT-Deluge in these four network topologies using two kinds of packet loss models: 1) forced packet loss, the nodes transmit packets at high power level over short distances, which ensure the link qualities are good, then the nodes drop packets uniformly at a certain rate; and 2) natural packet loss, the nodes transmit packets at low power level over relatively long distances, which induces natural packet loss.

According to the datasheet of TelosB, radio operations (e.g., send, receive, listen) and external flash operations (e.g., read, write, erase) cost most of the energy. More precisely, Table 1 gives the energy consumption of Telosb mote under different operations. It is easy to see that the radio operations consume more than 10 times of energy than CPU operations. Current reprogramming protocols, like Rateless Deluge and Synapse, keep the radio always on. Some other dissemination protocols [8], [22] allow the

TABLE 1
Energy Consumption of Different Operation Combinations [17]

Oprations	Current
CPU on, Radio RX/TX	20.65 mA
CPU on only	1.8 mA
CPU on, Flash Read/Write	12 mA

sensor nodes to sleep for saving energy. The sleep scheduling, however, has extra overhead (e.g., additional transmissions or computations). In MT-Deluge, we do not incorporate sleep scheduling and consider it as a direction of future work. Some effects have been devoted to reduce the flash operations during the reprogramming to save energy and prolong the network lifetime [19]. Our work can be considered as complementary to these efforts. We mainly focus on dissemination delay, which affects the energy consumption directly, in this section.

6.2 Multihop Line Topology

We put 16 TelosB motes in a straight line with short internode distance to get near 100 percent link quality. Forced packet losses are used to evaluate the performance of MT-Deluge under different loss rates. In order to avoid the nodes disseminating packets to far away nodes, we set the nodes only communicate with their adjacent neighbors. One page consisting of 24 packets is disseminated from one side of the line to another using different page transfer techniques. Experiments under this topology are conducted by forced packet losses. Fig. 8 shows the total delay using three different page transfer techniques. When the packet loss rate is zero, no retransmission is needed and network coding is not needed. Therefore, non-coding-based approach Deluge achieves the best performance. When the packet loss rate increases, network coding used in Rateless Deluge and MT-Deluge significantly improve the network performance. In our Multithreaded design, nodes forward the encoded packets and decode them concurrently. Therefore, the dissemination delay is shortened. On average, MT-Deluge consumes 48.6 percent less time than Rateless Deluge.

In order to demonstrate how MT-Deluge shortens the dissemination delay, we also make the nodes perform local logging to revealing some system insights. Every node records its current operation (e.g., radio operation, encoding/decoding) to its local memory so we can analyze it.

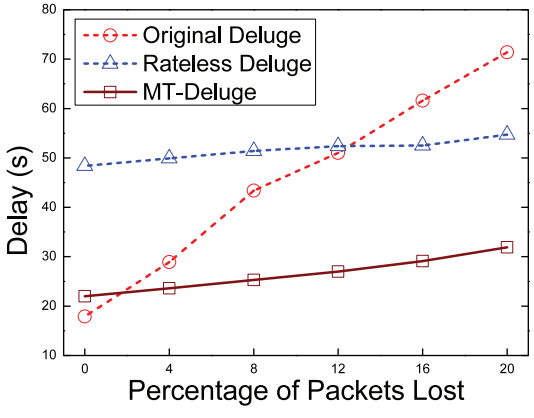


Fig. 8. dissemination delay of the line topology network consisting of 16 TelosB motes.

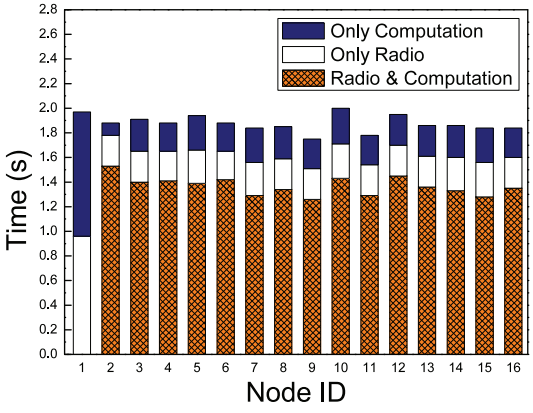


Fig. 9. Temporal statistics of different operations.

Fig. 9 gives the 16 nodes' detailed temporal statistics of different operations in the experiment. Node 1 is the initial sender, it needs to encode the whole page and send the encoded packets to node 2. Therefore, node 1 does not perform radio operations and decoding concurrently. All other nodes receive the encoded packets and forward them to the next node when it is decoding these packets at the same time. On average, more than 73 percent of the time, the radio operations, and the decoding run concurrently. Note that in Rateless Deluge, these operations are sequentially executed which is time consuming. This is the main reason that MT-Deluge can shorten the dissemination delay.

Consider node 4 in this experiment, its radio thread, and computation thread run concurrently for 1.41 seconds; only radio thread and only computation thread spends 0.24 second and 0.23 second, respectively. This means these two threads run concurrently in 75 percent of the time. We use Concurrency Degree (CD_i) to denote this ratio of node i. In theory, the most efficient way is to keep the two threads running concurrently all the time ($CD_i = 100\%$) which makes the radio and CPU resources be fully utilized. Based on this idea, we conduct a series of experiments with different parameter settings. To make the two threads run concurrently longer, we tune the $INIT$ (i.e., the number of received packets before the decoding starts) to change the incremental decoding starting time as well as the CD_i. The page size is fixed to be 24 in these experiments. Table 2 shows the dissemination delay and the average CD under different $INIT$. The minimum $INIT$ is 2 because the incremental decoding needs at least two rows to start. Therefore 100 percent CD can not be reached. When $INIT$ equals to 2, the average CD reaches the maximum which is 82.3 percent. However, the dissemination delay does not reach the minimum when $INIT$ is 2. Instead, it reaches the minimum when $INIT$ is 8. The reason is that the total decoding time is longer when $INIT$ is smaller. Fig. 10 shows the incremental decoding speed under different $INIT$. It shows that the decoding speed is faster when $INIT$ is larger. That means there is a tradeoff of the setting

TABLE 2
Dissemination Delay and Average CD under Different $INIT$

$INIT$	2	4	6	8	10	12
Delay (s)	30.5	30.2	29.2	28.1	31.2	39.1
Average CD (%)	82.3	79.9	75.6	73.2	70.1	67.7

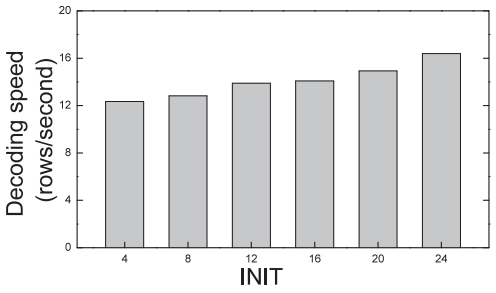


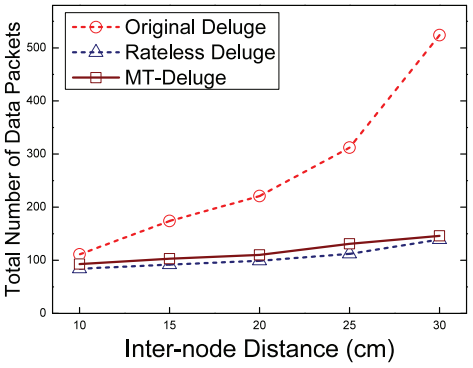
Fig. 10. Incremental decoding speed.

of $INIT$. In the current implementation, we set $INIT$ be $k/3$ where k is the page size.

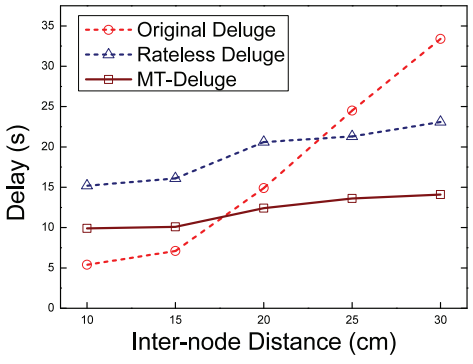
6.3 Multihop Grid Topology

We put 20 TelosB motes as a 4×5 grid and evaluate the performance of one page transmission. One corner node is chosen as the initial sender to disseminate one page consisting of 24 packets. Two metrics are used: total number of data packets and dissemination delay. Three sniffer nodes are placed in the grid to passively listen packets in the network. The results are obtained by calculating the average value of the data from the three sniffer nodes. In grid experiments, we do not set packet loss rate but use natural packet loss. The transmission power of TelosB mote can be configured at different levels (i.e., 2 to 31 [17]). We set the transmission power level be 2 and the internode distance between 10 and 30 cm. Under this setting, there are both good and poor links, more similar to the real condition. Figs. 11a and 11b show the total number of data packets and dissemination delay of three techniques, respectively. The number of data packets of MT-Deluge is similar to Rateless Deluge in grid topology but small than that of Deluge especially when the link qualities are poor. For example, when the internode distance is configured to be 30cm, the total number of data packets of Deluge is 3.5 times larger than that of MT-Deluge. This performance gains come from the network coding technique which reduce the data retransmissions. Fig. 11b shows that MT-Deluge can shorten the dissemination delay significantly in grid topology. Compared to the line topology, the relative delay reduction of MT-Deluge is smaller (i.e., 48.6 percent in line topology and 37.6 percent in grid topology). The reason is that grid topology has less hops than line topology.

In Section 5.3, we have analyzed the relationship between page size and link qualities. We got a conclusion that it is better to choose a *larger* page size in a *lossy* environment. A series of experiments on grid topology are conducted to evaluate the performance. Again, we set different internode distances (i.e., dense and sparse) to get different link qualities (i.e., large internode distance means poor link qualities). The page size is set between 18 and 48. Because we only transmit one page, the delay is directly related to the page size. For comparison, we use average delay per packet to evaluate the efficiency of different page sizes. The result is shown in Fig. 12. The optimal page size when the internode distance is 15 cm is smaller. In our experiments, setting the page size near 30 can get the shortest delay in dense network. And setting the page size near 36 is optimal in sparse network.



(a) data traffic



(b) delay

Fig. 11. Data traffic and delay of the grid topology network consisting of 20 TelosB motes.

6.4 Single-Hop Clique Topology

We also evaluate the performance of MT-Deluge in a single-hop network. The description of the experiments and the results can be found in Section 4 of the supplementary file, available online.

6.5 Multihop Random Distribution Topology

In the previous three topologies, the nodes are forced to communicate with several adjacent nodes or drop some packets. These experiment settings can reveal many system insights and evaluate the protocol precisely. In this section, we conduct a series of experiments under a random distribution topology network without controlling the neighbors or packet drop rate. Twenty four nodes are deployed randomly in three adjacent rooms. One sink node

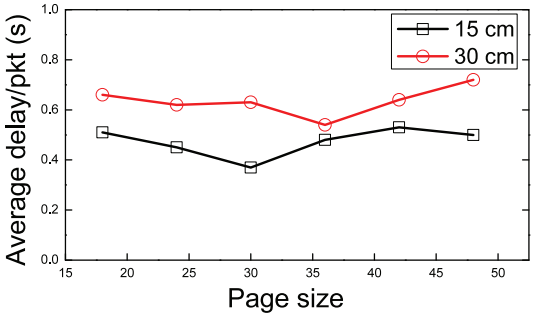
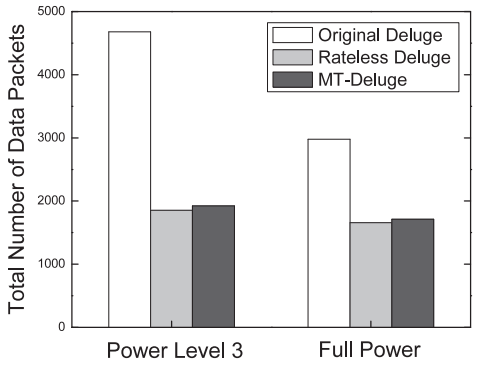
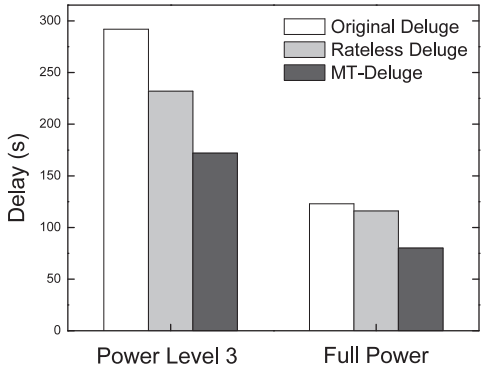


Fig. 12. Average delay per packet when the internode distance is set to be 15 or 30 cm.



(a) data traffic



(b) delay

Fig. 13. Data traffic and delay of the random distributed topology network consisting of 24 TelosB motes.

distributes a 7-page program image (*CountToLeds* benchmark in TinyOS) to all the other nodes. Every page consists of 24 packets. We set the power level to three to get high packet drop rate and use the full power level to get low packet drop rate. Note that when the power level is set to 3, a TelosB node can communicate with nodes about 5 meters away without any obstacles. In these experiments, there are many obstacles such as monitors, tables, wall, etc. Therefore, the packet drop rate is high when the power level is set to 3.

Fig. 13a shows the total number of data packets during the dissemination procedure. Both Rateless Deluge and MT-Deluge reduce the number of data packets significantly whether the power level is set to 3 or full power level. This benefit comes from the network coding used the these two approaches. Fig. 13b depicts the delays the three approaches. When the power level is set to 3, MT-Deluge costs 25.8 percent less time to complete the dissemination. When the power level is set to full power level, MT-Deluge costs 31 percent less time than Rateless Deluge. Note that the delay of original Deluge is very close to that of Rateless Deluge when full power level is set. The reason is that the packet drop rate is relatively low under this setting, the performance gains of network coding used in Rateless Deluge are neutralized by the computational cost. MT-Deluge still performs better than the original Deluge due to the efficient page transfer method.

7 Conclusion and Future Work

In this paper, we provide a multithreaded design for data dissemination in WSNs. Current coding-based data dissemination protocols introduce encoding or decoding overhead and this problem limits the scalability of these protocols. We propose MT-Deluge, a multithreaded design which allows the radio operations and the computations run concurrently. We implement it based on TinyOS, and the experiments on four topologies show that MT-Deluge is able to keep the merits of network coding techniques and has much shorter dissemination delay. For future work, we will consider designing multithreaded versions of other protocols and leveraging power management to further improve the network performance for WSNs.

Acknowledgments

This work is supported in part by the National Science Foundation of China Grants No. 61202402, 61070155, the National High Technology Research and Development Program of China (863 Program) (No. SS2012AA022814), the Fundamental Research Funds for the Central Universities (2012QNA5007, 2012FZA5017), the Research Fund for the Doctoral Program of Higher Education of China (20120101120179), the Zhejiang Province Key S&T Innovation Group Project (Grant No. 2009R50009), and the Program for New Century Excellent Talents in University (NCET-09-0685). The authors would like to thank the support from China Scholarship Council. Please contact with the corresponding author, Dr. Wei Dong (dongw@zju.edu.cn) for any further questions and comments.

References

[1] S.H. Lee, S. Lee, H. Song, and H.S. Lee, "Wireless Sensor Network Design for Tactical Military Applications: Remote Large-Scale Environments," *Proc. IEEE Military Comm. Conf. (MILCOM)*, 2009.

[2] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, and G. Dai, "Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest," *Proc. ACM Seventh Conf. Embedded Networked Sensor Systems (SenSys)*, 2009.

[3] R. Huang, W.-Z. Song, M. Xu, N. Peterson, B. Shirazi, and R. LaHusen, "Real-World Sensor Network for Long-Term Volcano Monitoring: Design and Findings," *IEEE Trans. Parallel and Distributed Systems*, vol. 23, no 2, pp. 321-329, Feb. 2011.

[4] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA)*, 2002.

[5] K. Lin and P. Levis, "Data Discovery and Dissemination with DIP," *Proc. ACM/IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN)*, 2008.

[6] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming Wireless Sensor Networks: Challenges and Approaches," *IEEE Network Magazine*, vol. 20, no. 3, pp. 48-55, May/June 2006.

[7] J.W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," *Proc. ACM Second Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2004.

[8] S.S. Kulkarni and L. Wang, "MNP: Multihop Network Reprogramming Service for Sensor Networks," *Proc. IEEE 25th Int'l Conf. Distributed Computing Systems (ICDCS)*, 2005.

[9] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A.F. Harris III, and M. Zorzi, "SYNAPSE: A Network Reprogramming Protocol for Wireless Sensor Networks using Fountain Codes," *Proc. IEEE CS Fifth Ann. Conf. Sensor, Mesh, and Ad Hoc Comm. and Networks (SECON)*, 2008.

[10] A. Hagedorn, D. Starobinski, and A. Trachtenberg, "Rateless Deluge: Over-the-Air Programming of Wireless Sensor Networks using Random Linear Codes," *Proc. ACM/IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN)*, 2008.

[11] W. Dong, C. Chen, X. Liu, J. Bu, and Y. Gao, "A Light-Weight and Density-Aware Reprogramming Protocol for Wireless Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 10, no. 10, pp. 1403-1415, Oct. 2011.

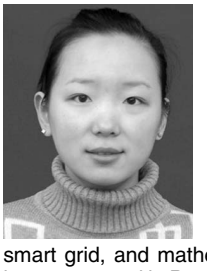
[12] I.-H. Hou, Y.-E. Tsai, T.F. Abdelzaher, and I. Gupta, "AdapCode: Adaptive Network Coding for Code Updates in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2008.
[13] W. Li, Y. Zhang, and B. Childers, "MCP: An Energy-Efficient Code Distribution Protocol for Multi-Application WSNs," *Proc. IEEE Int'l Conf. Distributed Computing Systems (DCOSS),* 2009.
[14] M.D. Krasniewski, R.K. Panta, S. Bagchi, C.-L. Yang, and W.J. Chappell, "Energy-Efficient on-Demand Reprogramming of Large-Scale Sensor Networks," *ACM Trans. Sensors Networks,* vol. 4, article 2, 2008.
[15] R.K. Panta, I. Khalil, and S. Bagchi, "Stream: Low Overhead Wireless Reprogramming for Sensor Networks," *Proc. IEEE INFOCOM,* 2007.
[16] S. Katti, D. Katabi, H. Balakrishman, and M. Medard, "Symbol-Level Network Coding for Wireless Mesh Networks," *Proc. ACM SIGCOMM,* 2008.
[17] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research," *Proc. ACM/IEEE Fourth Int'l Symp. Information Processing in Sensor Networks (IPSN/SPOTS),* 2005.
[18] Y. Gao, J. Bu, W. Dong, C. Chen, L. Rao, and X. Liu, "Exploiting Concurrency for Efficient Dissemination in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems (DCOSS),* 2011.
[19] W. Dong, Y. Liu, X. Wu, L. Gu, and C. Chen, "Elon: Enabling Efficient and Long-Term Reprogramming for Wireless Sensor Networks," *Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems,* 2010.
[20] K. Klues, C.-J. M. Liang, J.-y. Paek, M.-E.P. Levis, A. Terzis, and R. Govindan, "TOSThreads: Thread-Safe and Non-Invasive Preemption in TinyOS," *Proc. ACM Seventh Conf. Embedded Networked Sensor Systems (SenSys),* 2009.
[21] T. Zhu, Z. Zhong, T. He, and Z.l. Zhang, "Exploring Link Correlation for Efficient Flooding in Wireless Sensor Networks," *Proc. Seventh USENIX Conf. Networked Systems Design and Implementation (NSDI),* 2010.
[22] L. Huang and S. Setia, "CORD: Energy-Efficient Reliable Bulk Data Dissemination in Sensor Networks," *Proc. IEEE INFOCOM,* 2008.

Yi Gao received the BEng degree at Zhejiang University in 2009. He is currently a third year PhD student at Zhejiang University, China. From December 2008 to April 2009, he worked in the Information System College of Singapore Management University as an exchange student. From October 2011 to October 2012, he worked at McGill University as a joint training research student. His research interests include data reliability and network protocols in wireless sensor networks. He is a student member of the IEEE and the China Computer Federation (CCF).

Jiajun Bu received the BS and PhD degrees in computer science from Zhejiang University, China, in 1995 and 2000, respectively. He is a professor in the College of Computer Science and the deputy director of the Institute of Computer Software at Zhejiang University. His research interests include embedded system, mobile multimedia, and data mining. He is a member of the IEEE and the ACM.

Wei Dong received the BS and PhD degrees in computer science from Zhejiang University in 2005 and 2010, respectively. He was a Postdoc Fellow in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology from December 2010 to December 2011. He is currently an associate professor at the College of Computer Science in Zhejiang University. His research interests include networked embedded systems and wireless sensor networks. He is a member of the IEEE and the China Computer Federation (CCF).

Chun Chen received the bachelor's of mathematics degree from Xiamen University, China, in 1981, and the MS and PhD degrees in computer science from Zhejiang University, China, in 1984 and 1990, respectively. He is a professor in College of Computer Science, and the director of Institute of Computer Software at Zhejiang University. His research interests include embedded system, image processing, computer vision, and CAD/CAM. He is a member of the IEEE.



Lei Rao received the PhD degree from the Department of Electronics and Information Engineering from Huazhong University of Science and Technology, China, in 2010. In 2011, she was a postdoctoral fellow in the School of Computer Science, McGill University, Montreal, QC, Canada. She is a researcher at General Motors Electrical & Controls Systems research lab in the Silicon Valley. Her research interests include vehicular technology, greencomputing, smart grid, and mathematical modeling and optimizations. Her papers have appeared in Proceedings of the IEEE, *IEEE Transactions on Smart Grid*, IEEE INFOCOM, ACM/IEEE ICCPS, IEEE ICC, QShine, etc. She is a member of the IEEE.

Xue Liu received the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 2006. He received the BS degree in mathematics and the MS degree in automatic control, both from Tsinghua University, China. He is an associate professor in the School of Computer Science at McGill University. He has also worked as the Samuel R. Thompson Associate Professor at the University of Nebraska-Lincoln and HP Labs in Palo Alto, California. His research interests are in computer networks and communications, smart grid, real-time and embedded systems, cyberphysical systems, data centers, and software reliability. Dr. Liu has been granted three US patents and filed two other US patents, and published more than 150 research papers in major peer-reviewed international journals and conference proceedings, including the Year 2008 Best Paper Award from *IEEE Transactions on Industrial Informatics*, and the First Place Best Paper Award of the ACM Conference on Wireless Network Security (WiSec 2011). He is a member of the IEEE and the ACM.