www.redpel.com +917620593389

Link Quality Aware Code Dissemination in Wireless Sensor Networks

Wei Dong, *Member, IEEE*, Yunhao Liu, *Senior Member, IEEE*, Zhiwei Zhao, *Student Member, IEEE*, Xue Liu, *Member, IEEE*, Chun Chen, *Member, IEEE*, and Jiajun Bu, *Member, IEEE*

Abstract—Wireless reprogramming is a crucial technique for software deployment in wireless sensor networks (WSNs). Code dissemination is a basic building block to enable wireless reprogramming. We present ECD, an Efficient Code Dissemination protocol leveraging 1-hop link quality information based on the TinyOS platform. Compared to prior works, ECD has three salient features. First, it supports dynamically configurable packet sizes. By increasing the packet size for high PHY rate radios, it significantly improves the transmission efficiency. Second, it employs an accurate sender selection algorithm to mitigate transmission collisions and transmissions over poor links. Third, it employs a simple impact-based backoff timer design to shorten the time spent in coordinating multiple eligible senders so that the largest impact sender is most likely to transmit. We implement ECD based on TinyOS and evaluate its performance extensively via testbed experiments and simulations. Results show that ECD outperforms state-of-the-art protocols, Deluge and MNP, in terms of completion time and data traffic (e.g., about 20 percent less traffic and 20-30 percent shorter completion time compared to Deluge).

Index Terms—Wireless sensor network, dissemination, reprogramming

1 Introduction

Recent advances in microelectronic mechanical systems and wireless communication technologies have fostered the rapid development of networked embedded systems like wireless sensor networks (WSNs) [1], [23]. WSN applications often need to be changed after deployment for a variety of reasons—reconfiguring a set of parameters, modifying tasks of individual nodes, and patching security holes. Many large-scale WSNs [12], however, are deployed in environments where physically collecting previously deployed nodes is either very difficult or infeasible. Wireless reprogramming is a crucial technique to address such challenges.

In our recent efforts in deploying a large-scale sensor network system—GreenOrbs [10] during December, 2010-April, 2011, we regularly face the requirement of software upgrade. For example, our software version increases from version 158 to version 285 in the SVN repository (SVN is short for subversion which is a software versioning and revision control system distributed under an open source license). Although some modifications are intended for good readability and maintenance, we manually identify that at least 15 (i.e., 12 percent) of the updates are critical to system performance, including fixing link estimation bugs, adding additional metrics for diagnosis, etc.

Code dissemination is a basic building block for wireless reprogramming [4], [21]. Existing code dissemination protocols (represented by Deluge [6] and MNP [7]) adopt several key techniques to ensure high reliability and performance. First, they exchange control-plane messages for high reliability [6], [7], and [21]. Second, they segment a large code object into fixed-sized pages for pipelining [6], [7]. The page transmission time and inter-page negotiation time (which involves exchanges of control-plane messages) are therefore two major contributors to the overall completion time.

However, existing protocol designs exhibit their inefficiency in two main aspects. First, the data throughput efficiency—the ratio between the network throughput and PHY data rate—degrades rapidly as the PHY rate increases. For example, given the packet size of approximately 36 bytes in both Deluge and MNP (both were originally designed for the 19.2 Kbps CC1000 radio), the efficiency ratio for the current 250 Kbps CC2420 radio is only 14.3 percent. Second, the current sender selection algorithm in MNP [7] (for addressing the broadcast storm problem [14]) does not consider link quality information and needs multiple rounds of message exchanges, resulting in transmission redundancy and long completion time.

To address the first issue, we would like to increase the packet size to improve the transmission efficiency for high PHY rate radios (e.g., 250 Kbps CC2420). This approach is appropriate for code dissemination because the traffic is always saturated and there are no delay constraints on individual packets. However, it would be inflexible to fix the packet size to its maximum allowable size as a fixed packet size may not be appropriate for all platforms under all the conditions [3]. Therefore, we support dynamically

- *W. Dong, Z. Zhao, C. Chen, and J. Bu are with the Zhejiang Key Lab of Service Robot, College of Computer Science, Zhejiang University, Hangzhou 310027, China. E-mail: {dongw, chenc, bjj}@zju.edu.cn.*
- *Y. Liu is with the School of Software and TNLIST, Tsinghua University, Beijing 100084, China. E-mail: yunhao@greenorbs.com.*
- *X. Liu is with the School of Computer Science, McGill University. E-mail: xueliu@cs.mcgill.ca.*

Manuscript received 24 Mar. 2013; revised 18 June 2013; accepted 27 June 2013. Date of publication 15 July 2013; date of current version 13 June 2014. Recommended for acceptance by W. Jia.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.176

www.redpel.com +917620593389

configurable packet sizes in our protocol design (see the supplementary file which is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.176 for the design details).

To address the second issue, we leverage 1-hop neighbors' link quality information learned over the air to improve the sender selection accuracy. We dynamically estimate the impacts of senders by considering both uncovered neighbors (i.e., neighbors that do not receive an entire page) and the link qualities to those neighbors. A node's transmission is considered more effective if the node has more uncovered neighbors with good link qualities. Considering link qualities help to put less weight on potential senders with poor link qualities to their neighbors, thus mitigating transmissions for accommodating low PRR (PRR is short for Packet Reception Ratio) receivers. This is especially important when large packets are transmitted over the air. Given many candidate senders, our design needs to ensure that the best sender transmits while avoiding simultaneous retransmission attempts that can lead to duplicates or collisions. MNP [7] needs multiple rounds of message exchanges and explicit requests from receivers, resulting in high transmission overheads and long delays. We address this issue by proposing a fast sender selection mechanism that does not require explicit coordination. The basic idea is to prioritize sender transmissions so that the best sender with the largest impact is most likely to transmit.

We incorporate the above design principles into a new code dissemination protocol, ECD. We implement it based on the TinyOS operating system. We examine ECD's performance extensively through a 25-node testbed as well as TOSSIM [8] simulations. Results show that 1) by supporting large packets, ECD significantly shortens the completion time for the TelosB platform with 250 Kbps CC2420 radio; 2) by supporting accurate sender selection that leverages information learned over the air, ECD effectively reduces contentions and collisions, resulting in fewer packet transmissions; 3) by the impact-based backoff timer design, ECD reduces the inter-page negotiation time, also shortening the completion time.

In summary, the contributions of this paper are as follows.

- We identify the inefficiency of previous sender selection algorithms in code dissemination and devise a more accurate metric leveraging link quality information, which effectively reduces transmission collisions and transmissions over the poor links. We also propose practical techniques to address specific challenges such as priority inversion and concurrent transmissions.
- We propose an impact-based backoff timer design to shorten the time spent in coordinating multiple eligible senders so that the largest impact sender is most likely to transmit.
- We implement the dissemination protocol based on the TinyOS operating system. We conduct extensive evaluations via testbed experiments and simulations. Results show that ECD effectively improves the performance in terms of completion time and data transmission overhead.

The rest of this paper is structured as follows. Section 2 introduces the related work. Section 3 describes the motivation behind the work. Section 4 presents the design details. Section 5 describes the evaluation results, and finally, Section 6 concludes the paper and gives future research directions.

2 Related Work

Deluge [6] is perhaps the most popular code dissemination protocol used for reliable code updates. It uses a three-way handshake and NACK-based protocol for reliability, and employs segmentation (into pages) and pipelining for spatial multiplexing. Here we describe it in three different phases. 1) Advertisement (ADV), Request (REQ), and Page Transmission (DATA). In this phase, each node advertises about its local code objects. Note that a code object may not be complete during the process of transmission. Deluge enforces strict ordering of page transmission. When a node (receiver) learns that another node (sender) has more available pages, it will send a request and prepare to receive the data packets. When the sender receives a request, it will transition from the IDLE state to the TX state. Then it starts transmitting the requested data packets in the current page. 2) Page Retransmission. If a receiver loses some packets in a given page, it will remain in the RX state, sending requests to the most recently heard neighbor for the missing packets. On the other hand, the sender will enter the IDLE state whenever it completes transmitting a requested page. When it receives a request for retransmission purpose, it sends the packets immediately. When all missing packets in a given page are received, the receiver enters the IDLE state and prepares to receive the next page. 3) New Page Transmission. The sender and receiver will enter the IDLE state whenever a page completes. At this time, the receiver will prepare to receive the next page and increase the advertising frequency by resetting its advertisement timer. When the sender hears an inconsistent advertisement from any of the receivers, it will also reset its advertisement timer to let the receivers learn about the fact that a neighbor with more available pages is in the vicinity. The receiver will request to the sender for the next page. After the sender receives a new page request, it will start transmitting immediately.

A problem in Deluge is that when a sender receives requests from receivers, it will start transmitting data packets after a specified timeout. It is probable that multiple senders in a neighborhood start transmitting concurrently, causing serious collisions. To address this issue, MNP [7] incorporates a sender selection algorithm which attempts to guarantee that in a neighborhood there is at most one source transmitting at a time. In MNP, source nodes compete with each other based on the number of distinct requests they have received. The sender selection is greedy in that it tries to select the sender that is expected to have the most impact. MNP also reduces the active radio time of a sensor node by putting the node into ''sleep'' state when its neighbors are transmitting. This effectively reduces the idle listening problem and avoids overhearing.

Compared to the above two works, our work has two main differences. First, we enable dynamically configurable packet sizes to support large packets to improve the dissemination performance. Second, we employ an accurate and fast sender

www.redpel.com
+917620593389

selection algorithm to alleviate concurrent transmissions and transmissions over poor links.

Sprinkler [13] uses the localization service at each node to construct a connected dominating set (CDS). It uses TDMA to schedule packet transmissions among the CDS nodes to reduce energy consumption by minimizing packet transmissions. We mitigate the problem by limiting the number of concurrent transmitters in a neighborhood via dynamic sender selection. We also provide an option to mitigate the impact of hidden terminals (as discussed in the last paragraph in Section 4.2). Sprinkler, on the other hand, employs a TDMA-based approach. Sprinkler assumes a unit disk radio model. Nodes can be assigned the same time slot (and can concurrently transmit) only when the distance is sufficiently large (i.e., $5(R/\sqrt{5})$) where R is the reliable communication radius of the sensor node). However, whether there are collisions in practice depends on whether the unit disk radio model is accurate enough as well as whether the model keeps unchanged over time. We think both approaches are useful for particular application scenarios while our solution is more appropriate for randomly deployed (e.g., the density assumption of Sprinkler is not satisfied) and dynamically changing networks.

The data transmission rate in WSNs is generally considered low. However, this does not hold true for code dissemination: in the network reprogramming process, nodes need to disseminate a large code object as soon as possible. Hence, improving channel utilization is highly favored. While there are many effective approaches to improve channel utilization, including MIMO, rate adaptation [16], [20], they are not readily available for commonly used sensor nodes. Therefore, we propose adapting the packet size to improve the channel utilization, especially for high PHY rate radios (e.g., 250 Kbps CC2420). Dynamically adapting packet sizes has been investigated in [3], [15], but they have not practically applied to the broadcast/dissemination scenario. The design in our work does not aim to derive an optimal packet size. Rather, it aims to support dynamically configurable packet sizes, so that system designers can configure the packet payload size for improved performance, e.g., according to network models [2] or experimental observations.

There is also a rich literature in sender selection in general broadcast protocols. Sender selection is effective in mitigating the broadcast storm problem in wireless networks [14]. A number of approaches have been proposed to select the next forwarder [11], [18], [22]. In DCB [11], nodes maintain 2-hop neighbor information. When a sender broadcasts a packet, it selects the next forwarders in such a way that 1) 2-hop neighbors are covered and 2) 1-hop non-forwarders need to be covered by at least two forwarders. In RBP [18], every node rebroadcasts the packet for the first time. Then each node adjusts the number of retries based on the neighborhood density. Our work differs from those works in three main aspects. First, these protocols are not required to be 100 percent reliable. Second, in broadcast protocols, packet size is mainly determined by the application programs rather than underlying mechanisms which may incur delays. Third, in ECD, packets within a page are transmitted in a succession, hence we need a special design to align the estimation period, avoid priority inversions, and alleviate collisions.

CF [22] is a recent work that exploits spatial link correlation to mitigate ACK overhead. ECD differs CF in three aspects. First, CF is used for flooding a single packet while ECD is used for disseminating large code objects consisting of multiple pages and packets. CF does not guarantee 100 percent reliability while ECD employs handshake and negotiation to achieve 100 percent reliability. Second, the basic intentions for sender selection are the same for both protocols, i.e., selecting the sender which can cover the most number of uncovered (i.e., not yet received) receivers in a neighborhood. The techniques employed in those two protocols, however, are different. CF relies on ACKs and exploits link correlation to estimate the expected number of uncovered nodes while ECD relies on NAKs (i.e., REQ messages) and the link qualities to estimate the expected number of uncovered receivers. Third, both protocols rely on backoff timers to prioritize the transmissions. In CF the backoff period is simply calculated as the reciprocal of the impact. In ECD, the backoff period is more carefully optimized by minimizing the probabilities of ''priority inversion'' and transmission collisions.

Link estimation is important for sender selection. LEEP [19] is a passive link estimation protocol that can be invoked in proactive protocols to update neighbors' link qualities. For example, LEEP can be built on top of a broadcasting service. The LEEP protocol attaches a LEEP header (including a packet seqno) to the broadcast beacon. When a node receives a beacon, it can estimate the inbound link quality (i.e., the link quality from the current node to its neighbor) by inferring the number of missing packets. LEEP also attaches a LEEP footer (including inbound link qualities) to the broadcast beacon, so that each node can obtain the outbound link quality (i.e., the link quality from a neighbor to the current node) from the received beacon.

3 Motivation

This section identifies the need to incorporate a more accurate sender selection algorithm.

3.1 Sender Selection

Sender selection is a well studied technique to reduce contentions and collisions in broadcast protocols. It was also adopted in a previous code dissemination protocol—MNP [7] for improved performance. The basic principle in these algorithms is to select the best sender for forwarding the data while avoiding simultaneous transmissions from other neighboring nodes. This involves two main aspects. First, an accurate metric should be devised to estimate senders' impacts. Second, efficient mechanisms should be designed to coordinate transmissions of eligible senders so that the largest impact sender is most likely to transmit.

Sender selection in broadcast protocols is a special case of sender selection in code dissemination protocols (i.e., one page, one packet). Sender selection in code dissemination protocols is more complex. In particular, 1) code

www.redpel.com
+917620593389

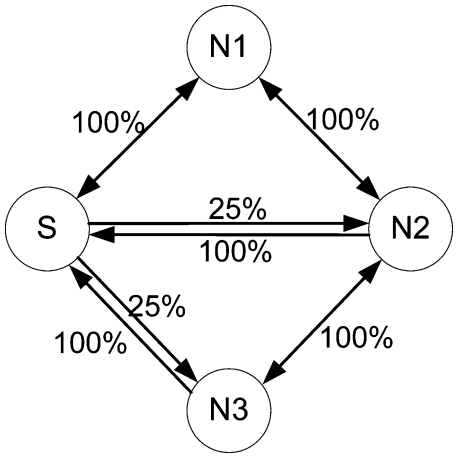


Fig. 1. Example of sender selection. The figures near the edges indicate the link qualities in terms of packet reception ratio (PRR).

dissemination should be 100 percent reliable, so senders are enforced to receive ACK/NACK after transmission; 2) neighboring nodes should determine whether a page transmission is in the vicinity because the packet transmissions within a page are not continuous.

MNP [7] is a code dissemination protocol that incorporates sender selection. The metric it uses for sender selection is the received number of distinct requests. A sender with more requests is more suitable for transmission because its transmission can cover a larger number of receivers. The best sender is selected via multiple rounds: the existence of the best node makes other neighboring nodes inactive; then the receivers explicitly request the code from the active sender. The problem therein is that it can cause biased estimation of nodes' real impacts, resulting in unnecessary packet transmissions.

Fig. 1 illustrates the benefits of our accurate sender selection algorithm. Assume S is the source node and there are 20 packets per page. In Fig. 1, we indicate the link qualities by the figures above the edges. For example, the link quality from S to N2 is 25 percent while the reverse link quality is 100 percent. Therefore, S can directly communicate with N1, N2, N3. However, the link qualities are different. Since the link quality from S to N1 is 100 percent, after S's transmission of a page, N1 can receive all the packets. On the other hand, since the link qualities from S to N2 and N3 are 25 percent, N2 and N3 can thus receive $25\% \times 20 = 5$ packets.

With MNP's approach, S will still be the next sender because it will receive the most number of requests. Hence, in MNP, $20/25\% = 80$ transmissions are needed for S to cover all N1, N2, N3.

ECD considers link qualities. By one round of page transmission, S cannot cover one receiver whereas N1 can cover N2. Therefore, N1 will be the sender (covering N2 by one page transmission), followed by N2 being the next sender (covering N3 by one page transmission). In summary, three steps are involved in our approach:

1. In the first round, S transmits a page consisting of 20 packets. N1 receives all packets while N2 and N3 receive 5 packets.
2. In the second round, N1 will be next sender, N1 transmits 15 packets requested by N2 (N2 received 5 packets in the first round).
3. In the third round, N2 will be the next sender, N2 transmits 15 packets requested by N3 (N3 received 5 packets in the first round).

Hence, our approach requires $20(\text{by S}) + 15(\text{by N1}) + 15(\text{by N2}) = 50$ packet transmissions. The key insight of our approach is to utilize 1-hop neighbors' link quality information, so that we can put less weight on nodes with poor outbound link qualities (S in this case), resulting in fewer packet transmissions.

Distinct from sender selection in broadcast protocols, the neighboring nodes that failed the competition need to estimate the length of ongoing page transmission to avoid simultaneous transmissions. In traditional broadcast protocols in which a single packet is broadcasted to all network nodes, there is no such a problem. This is because a neighboring node that can hear the transmission will not transmit as the channel is busy during the transmission. In code dissemination protocols, the unit of transmission is a page which consists of multiple packets. This is because the page size (e.g., 1104 bytes) is much larger than the maximum packet size (e.g., 128 bytes) on sensor nodes. The transmission of a page cannot fully occupy the channel because there are CSMA backoffs between two successive packets. Without a careful design, it is possible that a neighboring node will start transmitting when the channel is idle between two successive packet transmissions, causing possible collisions at the receiving node. ECD addresses this issue by attaching small meta-data to packets, so that neighboring nodes can accurately predict the ongoing activities in the neighborhood.

ECD also needs to shorten the time spent in coordinating the transmissions of multiple eligible senders so that the largest impact sender is most likely to transmit. Instead of using MNP's approach which involves multiple rounds of message exchanges, we employ a simple impact-based backoff timer design that does not require explicit coordination among neighboring nodes.

We will describe how we accurately estimate senders' impacts in Section 4.1 and how we coordinate the transmissions of eligible senders in Section 4.2.

4 Design

ECD adopts several key design principles: 1) segmentation and pipelining for scalability in large-scale networks; 2) inter-page negotiation for 100 percent reliability, and 3) adaptive advertisement through Trickle timer [9] to reduce the control-plane traffic. The Trickle timer controls the send rate so that the send rate decreases exponentially when the network is steady and increases to its maximum when inconsistency is detected.

ECD consists of five main phases. Fig. 2 shows the state transition diagram of ECD. Initially, a node (say S) stays in the IDLE state, advertising about the completed pages it has received. When another node (say R) hears the ADV message, and finds that S has more pages, it sends an REQ message to S and transitions into the RX state. R transitions

www.redpel.com
+917620593389

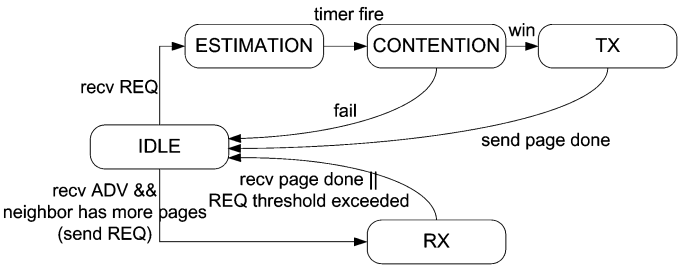


Fig. 2. State transition diagram of ECD.

back to the IDLE state under two conditions: 1) when it completely receives the current page; 2) when a specified number of REQ messages are sent and no DATA packets are received. When S overhears the REQ message (note the REQ message may not be destined to S), it enters the ESTIMATION state, waiting for more requests to estimate its impact. When the estimation timer fires, S enters the CONTENTION state. In the contention period, nodes (that overhear the REQ messages) backoff according to their estimated impacts. The largest impact node will likely have the shortest backoff time, and starts to transmit first. The winner of the competition enters the TX state while other nodes that failed the competition will enter the IDLE state. The winner will finally go back to the IDLE state when the requested packets are sent out.

We introduce some assumptions and terminologies as follows.

- The code dissemination process is usually initiated at a sink node. But it can also be initiated at multiple nodes. This is because our protocol is epidemic and ensures eventual consistency of the code object.
- We do not require sensor nodes synchronized. However, in our sender selection process, we implicitly synchronize (and align) the estimation period during which the impact is being estimated. Otherwise, there would be hidden terminal problems. We discuss this issue in more detail in Section 4.1.
- The code object for real sensor applications varies from 10 to 48 kB. To enable parallel transmissions, we divide the entire code object to fixed-sized pages. In this paper, we adopt a page size of 1104 bytes. In the current implementation, the page size can be adjusted at the compile-time. The packet size, on the other hand, can not only be adjusted at the compile-time, but also at the run-time. For a code object of 10 kB, the number of pages is $\lceil 10\ \text{kB}/1104\ \text{B} \rceil = 10$. A page consists of several packets. Depending on the packet payload size, the number of packets in a page also varies. For example, with a payload size of 23 bytes, the number of packets in a page is $1104/23 = 48$. The packets in a page are sent in a batch to improve throughput.
  A node receives the pages in sequential order. This means that a node cannot receive a high order page earlier than a low order page. Like Deluge, ECD enforces sequential order of page reception: after page i is received, the node will only accept packets from page $i+1$. After all packets in the current page are received, the packet payloads are saved onto the external flash according to their order within the page. The node can then start transmitting or receiving the next page. When a node receives a page, it advertises about its available pages. When a node learns that another node has more available pages, it requests for the page and the page transmission starts. In this way, ECD does not require a node to have an entire code object to serve other nodes. Instead, whenever a node has more available pages, it can serve other nodes having fewer available pages. So as long as the transmissions at two links do not collide, different pages can be transmitted concurrently at these two links.
- There are two kinds of meta-data mentioned in this paper. 1) Code meta-data. The code meta-data contains information about the code object, e.g., the name of the application, code object size, the number of pages, etc. 2) Packet meta-data (e.g., `pendingPktNum`, `impact`, etc). The packet meta-data is piggybacked in three kinds of packets, i.e., ADV, REQ, DATA. The meanings are introduced where needed.

4.1 Impact Estimation

As we discussed in Section 3.1, MNP's sender selection algorithm may spend too many transmissions on poor links. This should be avoided in our protocol design.

We need to estimate the number of uncovered nodes and the outbound link qualities to them.

To estimate the number of uncovered nodes, we use the REQ messages sent by uncovered nodes when missing packets are detected. There are two differences between ECD's REQ mechanism and Deluge's REQ mechanism. First, multiple eligible senders overhear the REQ message and may be responsible for sending requested packets in the REQ message that is not destined for them. This enlarges the set of eligible senders so that we select the best one. Second, we note that in Deluge [6], REQ messages may be suppressed if another REQ message for the same page is overheard. This mechanism, however, will lead to biased estimation in our protocol design. For this reason, in ECD, the uncovered nodes send REQ messages unless there is an ongoing page transmission.

The setting of the estimation period for sending and receiving REQ messages depends on the time window during which nodes randomize the transmissions of REQs. If the time window is too small, the probability of REQ collisions will increase. On the other hand, if the time window is too large, it will cause a long dissemination delay. In this paper, we set the estimation period to be consistent with Deluge's default settings to make a fair comparison. As the REQ message is transmitted out in the time window of [16, 256) ms in Deluge, we set the estimation period to be $16 + 256 = 272$ ms. Note that these values can be configured at compile-time to further optimize the performance for a particular network.

We will analyze that the above setting will not cause serious collisions, especially for high PHY rate radios. To

www.redpel.com
+917620593389

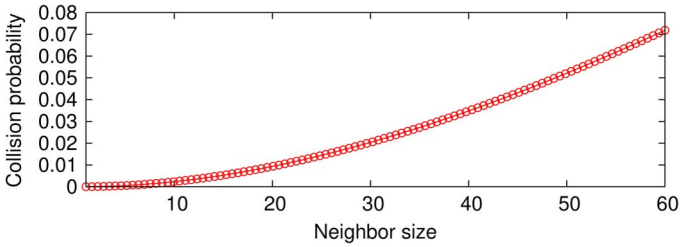


Fig. 3. Collision probability vs. number of hosts (N).

illustrate this, we here use a simple analytical model [5]. We have $CW = 272$ because we use a milli-timer for sending REQ so that the idle slot equals to 1 ms in our dissemination protocol. According to [5], the attempt probability is

$$P_e(CW) = \frac{2}{CW + 1}. \quad (1)$$

P_t, the probability of a successful transmission in a given slot, if N hosts contend for the channel is (such an event requires a transmission attempt by a single host and the absence of all the others)

$$P_t(CW, N) = NP_e(1 - P_e)^{N-1}. \quad (2)$$

The collision probability in a slot is,

$$P_c(CW, N) = 1 - P_t(CW, N) - P_i(CW, N) \quad (3)$$

where

$$P_i(CW, N) = (1 - P_e)^N \quad (4)$$

is the probability of an idle slot.

We plot the collision probability vs. number of hosts in Fig. 3. As the figure shows, for the CC2420 radio, the collision probability keeps below 10 percent even when N increases to 60.

Compared to MNP, ECD has the additional overhead of link estimation. To estimate the link qualities, we incorporate the LEEP link estimation protocol [19] into our design. LEEP is a passive link estimation protocol that can be invoked in proactive protocols to update neighbors' link qualities. In our current implementation, we attach a small LEEP header (containing a seqno) to ADV, REQ, and DATA messages. Each node uses these messages to estimate the inbound link qualities from neighboring nodes. It is worth noting that during dissemination, DATA messages are broadcasted to all neighboring nodes and can be used for inbound link estimation. This process effectively calibrates estimated link qualities via control-plane messages. Moreover, we attach the LEEP footer (containing node IDs and their inbound link qualities) to ADV messages. Therefore, the outbound link qualities can be obtained by periodically exchanging the inbound link qualities encompassed in the ADV messages. Since both the LEEP header and footer are piggybacked in three existing message types in dissemination protocols, we consider the introduced transmission overhead acceptably small.

Combining the requests sent by uncovered nodes and the link qualities to the requesters, we can estimate a sender's impact. Intuitively, if a sender has more uncovered nodes with good link qualities, this node should be the next sender.

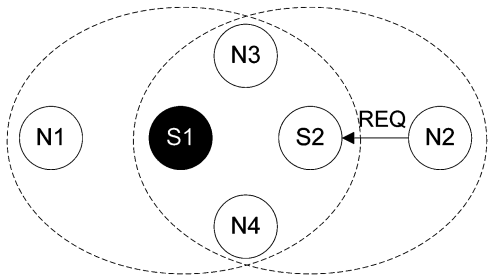


Fig. 4. Estimation period alignment in a neighborhood. The black node (S1) is currently transmitting a page.

In ECD, we use $\varepsilon(u)$ as the metric to estimate the sender's impact. $\varepsilon(u)$ considers both the number of uncovered nodes and the link qualities to them

$$\varepsilon(u) = \sum_{k \in U(u)} p(u, k) \quad (5)$$

where $U(u)$ contains u's uncovered nodes, $p(u, k)$ is the link quality from u to k.

Let's look at the example shown in Fig. 1. After S finishes sending one complete page, we use the metric defined above to estimate the impacts of S and N1. $\varepsilon(S) = 1 \cdot (0.25) + 1 \cdot (0.25) = 0.5$, $\varepsilon(N1) = 1$. Therefore, we will select N1 as the next forwarder. This metric will put less weight on nodes with poor outbound link qualities, thus avoiding transmissions over these poor links.

Another challenge for impact estimation is how to align the estimation periods of different nodes within a neighborhood. To illustrate the importance, let's look at the example shown in Fig. 4. In this example, S1 is currently transmitting a page consisting of, say 20 packets. As N2 is 2 hops away, it cannot hear S1's transmission. It thus sends an REQ message to S2, requesting the missing packets. When S2 receives the REQ message, S2 cannot start the estimation period because a page transmission is in the vicinity. If S2 starts the estimation period (because of received REQ) and then starts data transmission shortly. It would cause collisions at N3 and N4. This is different from sender selection in traditional broadcast protocols. In broadcast protocols, a single packet fully occupies the channel, so S2 will not transmit when S1 is currently transmitting (because the channel is busy). However, in code dissemination protocols, there is a large time gap between successive packets within the same page (each single packet performs backoff). Hence there is probability that S2 will transmit the page concurrently with S1. This will introduce contention delays in the MAC layer, or even collisions at some nodes (e.g., N3 and N4).

To address this problem, we attach a `pendingPktNum` field to each data packet, indicating the number of remaining packets that the sender intends to transmit. With this information and the expected transmission time for a single packet, we can estimate the end time of the ongoing page transmission

$$t_{\text{page}} = t_{\text{pkt}} \cdot \texttt{pendingPktNum} \quad (6)$$

www.redpel.com
+917620593389

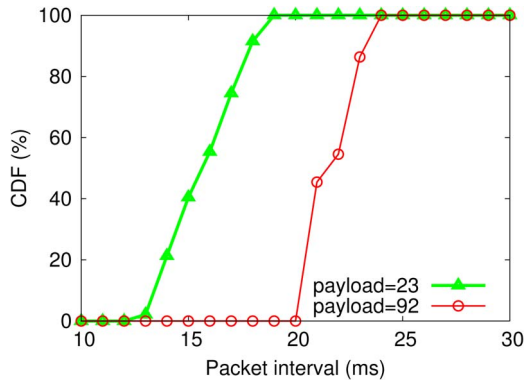


Fig. 5. CDF of packet transmission time within a page.

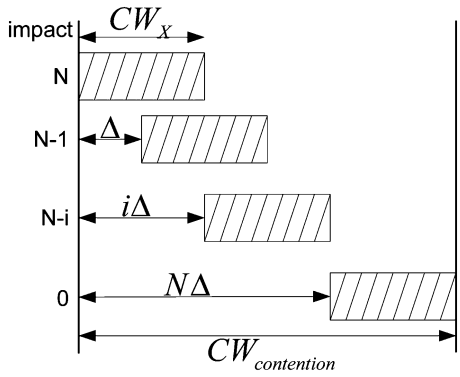


Fig. 6. Backoff timer design.

where t_{pkt} is the packet transmission time. Fig. 5 shows the CDF of the packet transmission time under different payload length of the dissemination protocol.

4.2 Transmission Prioritization

All nodes that overhear the REQ messages and have the requested page are eligible senders. From these eligible senders, we want to select the best sender.

Both ECD and MNP, compared to Deluge, have the additional overhead of transmission prioritization during which different senders contend so that the largest impact sender is most likely to transmit. MNP selects the best sender via explicit coordination: eligible senders broadcast their impacts in the ADV messages, then the receivers request to the sender that has received the maximum number of distinct requests. This method introduces high overheads. The receivers must send multiple requests for receiving the missing packets. Some requests are merely for impact estimation while some other requests will be followed by useful data transmissions. ECD uses an implicit backoff mechanism which can significantly reduce the time spent in coordinating multiple eligible senders so that the largest impact sender is most likely to transmit (see the simulation study in the supplementary file available online).

Assume N is the maximum number of eligible senders. The maximum contention period is $CW_{\text{contention}}$. Generally, the larger the impact of node u, the shorter u's backoff time. We need to compute the backoff time for a given node u.

First, node u's backoff time is randomly selected in a range $[a, b]$. We use CW_X to denote the range width, i.e., $CW_X = b - a$. Therefore, the backoff time of u is

$$t_{\text{backoff}}(u) = a + X \tag{7}$$

where X is a random period of time generated from $[0, CW_X)$.

Second, we determine the start of the range, a. It is determined according to node u's impact, $\varepsilon(u)$ (in the range of $0 \sim \text{N}$). The larger the impact, the shorter the backoff time. Hence,

$$a = (N - \varepsilon(u))\Delta \tag{8}$$

where Δ is a constant to prioritize the transmissions of nodes with different impacts. Combining Eq. (7) and (8), we obtain

$$t_{\text{backoff}}(u) = (N - \varepsilon(u))\Delta + X. \tag{9}$$

Finally, we usually need to have a bound on the largest backoff time to guarantee the worst-case performance. As depicted in Fig. 6, we bound the largest backoff time to $CW_{\text{contention}}$, i.e. $N\Delta + CW_x = CW_{\text{contention}}$. Therefore,

$$\Delta = \frac{CW_{\text{contention}} - CW_X}{N}. \tag{10}$$

With this backoff time design, the largest impact node is most likely to transmit first. Other nodes will cancel the data transmission and enter the IDLE state.

Different choices of $CW_{\text{contention}}$ and CW_X have different impacts. Usually, we need a relatively short contention period for fast propagation. However, if the contention period is too small, we may not effectively differentiate transmissions of different priorities. There is also a tradeoff in determining CW_X. If CW_X is large, there is possibility that the best sender loses the competition. If CW_X is small, we may not effectively randomize the transmissions from senders that have the same impact.

We set $CW_{\text{contention}} = 256$ ms so that it is comparable to the setting of the estimation period. We determine CW_X according to two constraints:

1. The probability that a low impact sender transmits before a high impact sender should be as small as possible. We denote this probability as P_1.
2. The probability of collisions among senders with the same impact should be as small as possible. We denote this probability as P_2.

We set CW_X such that $P_1 + P_2$ can be minimized.

P_1 is the probability that both the high impact sender and low impact sender choose the overlapping window. Moreover, the low impact sender has an earlier transmit time, i.e.,

$$P_1 = \frac{CW_X}{CW_{\text{contention}}} \cdot \frac{CW_X}{CW_{\text{contention}}} \cdot \frac{1}{2}, CW_X > \Delta. \tag{11}$$

We also note that $P_1 = 0$ when $CW_X \leq \Delta$.

P_2 can be calculated according to Eq. (3):

$$P_2 = P_c(CW_X, M) \tag{12}$$

where M denotes the maximum number of neighbors with same impact.

www.redpel.com
+917620593389

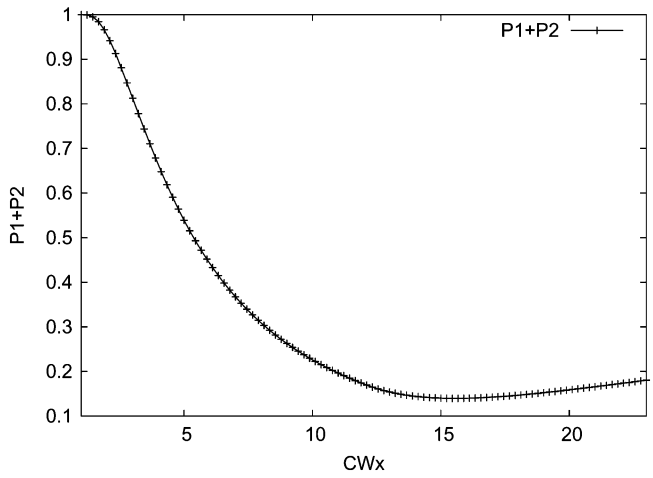


Fig. 7. $P_1 + P_2$ ($CW_{\text{contention}} = 256$ ms, $N = 20$, and $M = 5$).

Fig. 7 shows how the sum of P_1 and P_2 varies with CW_X given $CW_{\text{contention}} = 256$ ms, $N = 20$, and $M = 5$. We see that $CW_X \approx 15$ can minimize the sum of probabilities. Hence we set CW_X to be 15 ms.

Additionally, we devise specific mechanisms to avoid priority inversion and concurrent transmissions from senders that have the same impact.

To avoid priority inversion (i.e., a low impact node is transmitting instead of the largest impact one), we attach the `impact` field into each data packet. Suppose a low impact node is currently transmitting, the large impact node will start contending with the node. When the low impact node hears that a large impact node is transmitting, it will finally enter the IDLE state. If both impacts are the same, we use node IDs to break the tie.

To alleviate concurrent transmissions and collisions from senders of the same impact, we should restrict the number of eligible senders when there is little diversity in their impacts. When there are very few REQ messages in the neighborhood, it is likely that most of the nodes have already received the current page. In this case, we only allow the senders that the REQ message is destined to be eligible senders.

The sender selection scheme, while reduces contention and collisions, does not prevent them in the case of hidden terminals. Mechanisms to mitigate hidden terminals may introduce additional overhead that outweigh the achieved benefits, as demonstrated in [7]. We provide an option to mitigate the impact of hidden terminals: nodes that experience collisions can send a ''shut down'' message to prohibit interfering nodes from transmitting simultaneously. For example, consider a scenario where nodes A and C are out of communication range of each other, but both can send messages to and receive messages from node B. A is currently transmitting a page to B. It is possible that C will transmit another page to B since C cannot hear the transmission from A. In this case, B will experience collisions. To deal with this, B can send a ''shut down'' message including the node ID which has the largest impact B knows, say, node A. When C receives the message, it knows that a larger impact node, A, is currently transmitting to C. Therefore, the transmission will be postponed and the collision will be resolved.

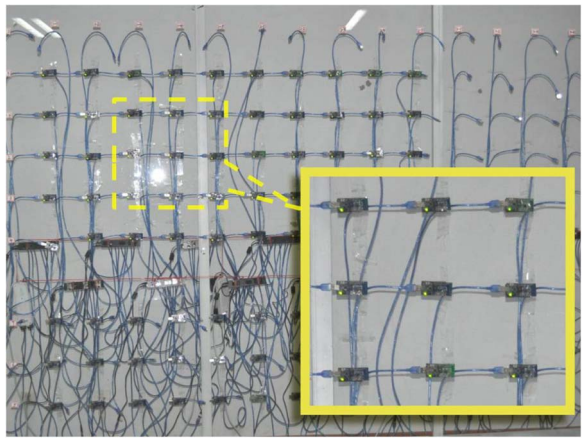


Fig. 8. Testbed.

5 Evaluation

We use Deluge [6] and MNP [7] for performance comparisons. We use two primary metrics—completion time and data traffic to evaluate the protocols.

5.1 Methodology

We evaluate the performance of ECD and Deluge on a 25-node TelosB testbed as shown in Fig. 8. In the testbed, the wires (USB lines) are used to burn programs to the sensor nodes. The wires can also be used to collect detailed logging information so that the collection process does not affect the performance of wireless protocols. We want to emphasize that our dissemination process is performed via the wireless channel.

To simulate the multihop behavior, we set the CC2420 power level to 1 and 2. The topology used for evaluation is a 5×5 grid where the base node is located at the bottom left corner. We use Deluge's default configurations (e.g., 23 bytes protocol payload, 48 packets per page). The page size of ECD is 1104 bytes (the same as in Deluge). We inject a program consisting of 10 pages (i.e., approximately 10 kB). Each experiment is conducted five times.

To get the performance metrics, we have written a statistic reporting component and a sniffer component. The statistic reporting component is wired to each of the 25 nodes. With this component, the base node broadcasts time synchronization beacons at the maximum power to synchronize all the other nodes. All the other nodes locally record their transmitted data packets and completion time (synchronized to the base node). After the experiments, all the nodes (including the base node) broadcast their statistics at the maximum power. The sniffer component is installed to a separate node called the sniffer node. It is able to hear all transmissions within the radio range. After the dissemination process, all other nodes report their statistics with the maximum transmission power so that the sniffer node can hear and transfer the results to PC via serial communications for later analysis.

5.2 Testbed Experiments

5.2.1 *Experiment with Power Level 1*

Fig. 9a gives the network topology we have conducted the experiment. The red star denotes the location of the base

www.redpel.com
+917620593389

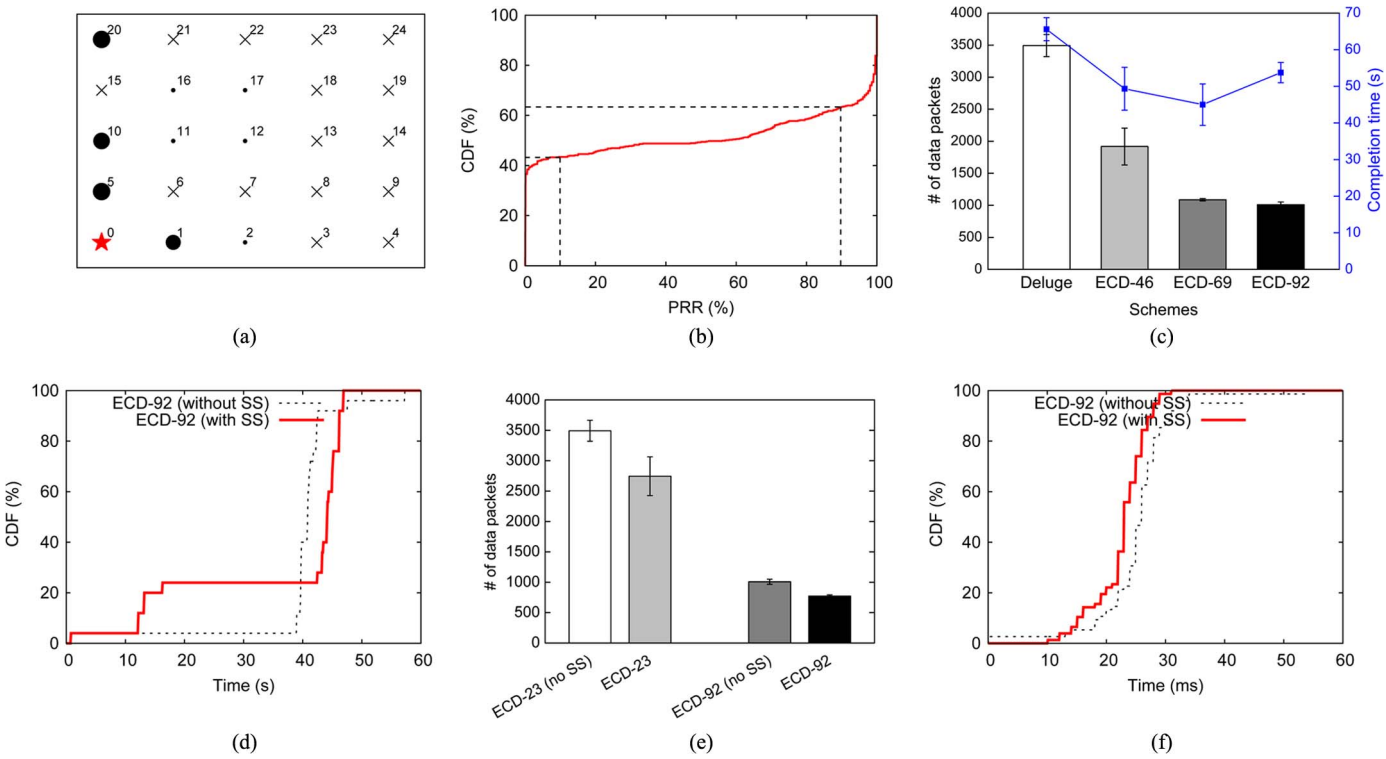


Fig. 9. Testbed results with power level $= 1$. (a) Testbed topology. (b) CDF of link qualities. (c) Impact of packet sizes. (d) CDF of node completion times. (e) Impact of sender selection. (f) CDF of inter-packet arrival times.

node. All other nodes are denoted by circles (there is a link from the base node to this node) or crosses (there is no link from the base node to this node). The radius of the circle represents the quality of link. For example, link quality of $0 \rightarrow 5$ is 100 percent while link $0 \rightarrow 2$ is only 25 percent. We place the sniffer node near node 12 to overhear radio transmission activities in the vicinity. We notice that nodes 11, 12, 16, 17, 20 can communicate directly with node 0, but nodes 6, 7, 15 cannot. This is because the quality of reception is not only related to the distance but also the noise floor of the radio hardware [17].

Before we conduct the experiment, we first measure the link quality of each node pair. Fig. 9b gives the CDF of link qualities. We can see that 36 percent links are good with link quality > 90 percent, 20 percent links are intermediate with link quality varied between 10 percent and 90 percent. Additionally, 44 percent of the links are poor (with link quality < 10 percent) or disconnected, which represents a proper multihop setting.

Fig. 9c shows the impacts of different packet payload sizes without sender selection. For data packets, we can see that large packets effectively reduces the transmitted packet number. It is obvious as for large packets, fewer number of packets constitute the entire code object. For the completion time, however, using the largest packets (92 bytes payload) does not lead to the best performance. This is because large packets are more susceptible to wireless losses. With ECD, we can dynamically configure the packet payload length. For example, using packet payload length of 69 bytes can shorten the completion time by 29.2 percent, compared to Deluge. The performance degradation at 92 payload length illustrates that we should incorporate sender selection into our protocol design to alleviate transmission collisions and transmissions over poor links, which causes larger negative impacts to large packets.

Fig. 9d shows impacts of sender selection on completion time for large packets (i.e., 92 bytes payload length). With sender selection, ECD can shorten the completion time from 58 s to 47 s, a 19 percent reduction. We also observe that, with sender selection, more number of nodes complete receiving the code object earlier. This is because our impact metric favors senders that cover more number of uncovered node with good link qualities. High PRR receivers can complete the code object earlier in time. For example, at time 20 s, there are 6 completed nodes with sender selection while there is only 1 completed node without sender selection.

Fig. 9e shows the impacts of sender selection on the number of data packets. At 23 bytes packet payload, ECD with sender selection can reduce the number of data packets by 22.8 percent. At 92 bytes packet payload, ECD with sender selection can reduce the number of data packets by 20 percent.

To get further insights why sender selection improved the performance, Fig. 9f compares the inter-packet arrival time for packets within a page observed on the sniffer node. We can see that without sender selection, the inter-packet arrival time is about 25 ms. With sender selection, the inter-packet arrival time can be reduced to about 21 ms. This is because there will be more contentions and collisions in protocols without sender selection. Considering that the expected congestion backoff time is 1.25 ms under TinyOS CSMA MAC, it means that sender selection reduces 3-4 times collisions for each packet transmission.

www.redpel.com
+917620593389

5.3 More Evaluation Results

Please refer to the supplementary file available online for more testbed results with power level 2 and simulation studies compared to MNP [7].

6 Conclusion

In this paper, we present ECD, an Efficient Code Dissemination protocol for wireless sensor networks. Compared to prior works, ECD has three salient features. First, it supports dynamically configurable packet sizes. By increasing the packet size for high PHY rate radios, it significantly improves the transmission efficiency. Second, it employs an accurate sender selection algorithm to mitigate transmission collisions and transmissions over poor links. Third, it employs a simple impact-based backoff timer design to shorten the time spent in coordinating multiple eligible senders so that the largest impact sender is most likely to transmit. We implement ECD based on TinyOS and evaluate its performance extensively. Results show that ECD outperforms state-of-the-art protocols, Deluge and MNP, in terms of completion time and data traffic.

Future work leads to two directions. First, we would like to incorporate effective sleep scheduling algorithm into our design. Second, we would like to examine effectiveness of our design in large-scale WSN systems.

Acknowledgment

This work is supported by the National Science Foundation of China Grants 61202402, 61070155, the National Key Technology R&D Program (2012BAI34B01), the Fundamental Research Funds for the Central Universities, the Research Fund for the Doctoral Program of Higher Education of China (20120101120179), NSFC Distinguished Young Scholars Program under Grant 61125202, the National Basic Research Program of China (973 Program) under Grant 2006CB303000, and the Demonstration of Digital Medical Service and Technology in Destined Region.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, ''Wireless Sensor Networks: A Survey,'' *Comput. Netw.*, vol. 38, no. 4, pp. 393-422, Mar. 2002.

[2] W. Dong, C. Chen, X. Liu, J. Bu, and Y. Liu, ''Performance of Bulk Data Dissemination in Wireless Sensor Networks,'' in *Proc. IEEE/ACM Int'l. Conf. DCOSS*, 2009, pp. 356-369.

[3] W. Dong, X. Liu, C. Chen, Y. He, G. Chen, Y. Liu, and J. Bu, ''DPLC: Dynamic Packet Length Control in Wireless Sensor Networks,'' in *Proc. IEEE INFOCOM*, 2010, pp. 1-9.

[4] W. Dong, Y. Liu, X. Wu, L. Gu, and C. Chen, ''Elon: Enabling Efficient and Long-Term Reprogramming for Wireless Sensor Networks,'' in *Proc. ACM SIGMETRICS*, 2010, pp. 49-60.

[5] M. Heusse, F. Rousseau, R. Cuillier, and A. Duda, ''Idle Sense: An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs,'' in *Proc. ACM SIGCOMM*, 2005, pp. 121-132.

[6] J.W. Hui and D. Culler, ''The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale,'' in *Proc. ACM SenSys*, 2004, pp. 81-94.

[7] S. Kulkarni and L. Wang, ''Energy-Efficient Multihop Reprogramming for Sensor Networks,'' *ACM Trans. Sens. Netw.*, vol. 5, no. 2, p. 16, Mar. 2009.

[8] P. Levis, N. Lee, M. Welsh, and D. Culler, ''TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications,'' in *Proc. ACM SenSys*, 2003, pp. 126-137.

[9] P. Levis, N. Patel, D. Culler, and S. Shenker, ''Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks,'' in *Proc. USENIX Symp. NSDI*, 2004, p. 2.

[10] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, and J. Zhao, ''Does Wireless Sensor Network Scale? A Measurement Study on GreenOrbs,'' in *Proc. IEEE INFOCOM*, 2011, pp. 873-881.

[11] W. Lou and J. Wu, ''Towards Broadcast Reliability in Mobile Ad Hoc Networks with Double Coverage,'' *IEEE Trans. Mobile Comput.*, vol. 6, no. 2, pp. 148-163, Feb. 2007.

[12] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X.-Y. Li, and G. Dai, ''Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest,'' in *Proc. ACM SenSys*, 2009, pp. 99-112.

[13] V. Naik, A. Arora, P. Sinha, and H. Zhang, ''Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Wireless Embedded Devices,'' in *Proc. IEEE RTSS*, 2005, p. 286.

[14] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, ''The Broadcast Storm Problem in a Mobile Ad Hoc Networks,'' in *Proc. ACM MobiCom*, 1999, pp. 151-162.

[15] Y. Sankarasubramaniam, I.F. Akyildiz, and S.W. Mclaughlin, ''Energy Efficiency Based Packet Size Optimization in Wireless Sensor Networks,'' in *Proc. IEEE Int'l. Workshop Sens. Netw. Protocols Appl.*, 2003, pp. 1-8.

[16] S. Sen, N. Santhapuri, R.R. Choudhury, and S. Nelakuditi, ''AccuRate: Constellation Based Rate Estimation in Wireless Networks,'' in *Proc. USENIX Symp. NSDI*, 2010, p. 12.

[17] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, ''Understanding the Causes of Packet Delivery Success and Failure in Dense Wireless Sensor Networks,'' Stanford Univ., Stanford, CA, USA, 2006, Tech. Rep. SING-06-00.

[18] F. Stann, J. Heidemann, R. Shroff, and M.Z. Murtaza, ''RBP: Robust Broadcast Propagation in Wireless Networks,'' in *Proc. ACM SenSys*, 2006, pp. 85-98.

[19] TinyOS TEP 124: The Link Estimation Exchange Protocol (LEEP). [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/html/tep124.html

[20] M. Vutukuru, H. Balakrishnan, and K. Jamieson, ''Cross-Layer Wireless Bit Rate Adaptation,'' in *Proc. ACM SIGCOMM*, 2009, pp. 3-14.

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

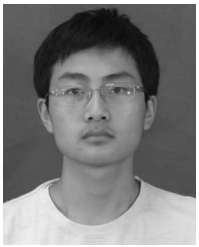
[22] T. Zhu, Z. Zhong, T. He, and Z.-L. Zhang, ''Exploiting Link Correlation for Efficient Flooding in Wireless Sensor Networks,'' in *Proc. USENIX Symp. NSDI*, 2010, p. 4.

[23] X. Liu, Q. Wang, L. Sha, and W. He, ''Optimal QoS sampling frequency assignment for real-time wireless sensor networks,'' in *Proc. IEEE RTSS*, 2003, pp. 308-319.

Wei Dong received the BS and PhD degrees in computer science from Zhejiang University, China, in 2005 and 2010, respectively. He was a postdoc fellow in the Department of Computer Science and Engineering, Hong Kong, University of Science and Technology, Hong Kong 2011. He is currently an Associate Professor at the College of Computer Science, Zhejiang University. His research interests include embedded systems and wireless sensor networks. He is a Member of the IEEE.

Yunhao Liu received the BS degree from the Automation Department, Tsinghua University, China, in 1995, the MA degree from Beijing Foreign Studies University, China, in 1997, and the MS and PhD degrees in computer science and engineering from Michigan State University, East Lansing, MI, USA in 2003 and 2004, respectively. He is currently a Professor in the School of Software and TNLIST, Tsinghua University. He is serving as the Associate Editors-in-Chief for *IEEE Transactions on Parallel and Distributed Systems*, Associate Editor for *IEEE/ACM Transactions on Networking* and *ACM Transactions on Sensor Networks*. His research interests include distributed systems, wireless sensor networks, privacy and security. He is a Senior Member of IEEE, and a distinguished speaker of the ACM.

www.redpel.com
+917620593389



Zhiwei Zhao received the BS degree from the College of Electronic and Information, Xi'an Jiaotong University, in 2010. He is currently a PhD student at the College of Computer Science in Zhejiang University. His research interests mainly focus on wireless sensor networks. He is a Student Member of the IEEE.

Xue Liu is an associate professor in the School of Computer Science at McGill University. He received his PhD in Computer Science from the University of Illinois at Urbana-Champaign. His research interests are in computer and communication networks, real-time and embedded systems, distributed systems, cyber-physical systems, green computing, and smart grid. He serves on the editorial board of IEEE Transactions of Parallel and Distributed Systems, IEEE Transactions on Vehicular Technology, and IEEE Communications Surveys and Tutorials.

Chun Chen received BS degree in mathematics 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 at Zhejiang University. His research interests include embedded systems, image processing, computer vision, and CAD/CAM. He is a Member of the IEEE.

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 College of Computer Science and the Deputy Dean of the Department of Digital Media and Network Technology at Zhejiang University. His research interests include embedded system, mobile multimedia, and data mining. He is a Member of the IEEE and ACM.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

www.redpel.com
+917620593389