Link Correlation Aware Data Dissemination in Wireless Sensor Networks

Zhiwei Zhao, *Student Member, IEEE,* Wei Dong, *Member, IEEE,* Jiajun Bu, *Member, IEEE,* Yu Gu, *Member, IEEE,* Chun Chen, *Member, IEEE*

Abstract—**Reprogramming is a crucial technique for software deployment and update in wireless sensor networks (WSNs). Bulk data dissemination is an important building block for enabling wireless reprogramming. Dissemination for reprogramming is required to be fast and energy efficient. Recent studies show that wireless link correlation can have a significant impact on the performance of bulk data dissemination. In this paper, we propose Correlated Dissemination (CD), a fast and energy efficient bulk data dissemination protocol for reprogramming in WSNs. The main contribution of CD is that it disseminates data according to a novel structure called correlated tree which considers both link qualities and link correlations. The construction of correlated tree is lightweight and allows for more efficient data transmission. Besides, it selectively and dynamically employs rateless codes to improve the performance, especially in circumstances where link correlation is inherently weak. We implement CD based on TelosB testbed with both TinyOS and OpenWSN (with 802.15.4e MAC). Evaluation results show that, compared with previous works, CD greatly improves the dissemination performance in terms of completion time, transmission cost and energy efficiency.**

Index Terms—**Wireless and industrial sensor networks, data dissemination, reprogramming, link correlation**

I. Introduction

Wireless Sensor Networks (WSNs) are replacing the traditional wired industrial communication systems since the Industrial Wireless Sensor Networks (IWSNs) offer several advantages including easy and fast installation and lowcost maintenance [1], [2].

IWSN/WSN applications [3]–[7] often need to be modified for a variety of reasons – changing node functionalities, fixing bugs, patching security holes, etc. As many IWSNs/WSNs are deployed and operated in environments where physically collecting network nodes is difficult or unfeasible. Wireless reprogramming is a significant and crucial technique to address this challenge [8], [9].

Dissemination is a basic building block to enable wireless reprogramming and attracts many research attentions in recent years [10]–[16]. Dissemination for reprogramming has two requirements. Firstly, the delay for code dissemination needs to be minimized. Since the network would be temporarily down during the dissemination of the update object, it is of great importance to reduce the dissemination time. Secondly, it requires 100% reliability. Due to the data size and reliability requirements, most existing works segment a large data object into several pages for a page-by-page, pipelined transmission. A three-way handshake (ADV-REQ-DATA) protocol is typically used to ensure data consistency. Some works also adopt many optimizations to further improve the performance. For example, Rateless Deluge [17] employs rateless codes in order to reduce the transmission overhead over lossy links.

Recent studies show that link correlation can greatly affect protocol performance. Srinivasan et al. observe that either Deluge or Rateless Deluge can outperform the other with different link correlations [18]. Several works [19], [20] further exploit link correlation to improve the flooding efficiency in wireless networks. These works are designed for efficient flooding and cannot be directly applied to bulk data dissemination.

In this paper, we explore how to exploit link correlation in the context of bulk data dissemination for efficient reprogramming in WSNs. The resulting protocol is called Correlated Dissemination (CD). CD exploits link correlation in a lightweight manner to improve the dissemination performance. Consider that there are three nodes in which node A broadcasts a packet to nodes B and C. We say links A$\rightarrow$B and A$\rightarrow$C are strongly correlated if C will receive/lose a packet with high probability under the condition that B receives/loses the same packet. We observe that with similar packet error rates, when link correlation is strong, the transmission efficiency tends to be high. Based on the observation, we propose a novel dissemination tree structure called *correlated tree* which considers both link qualities and link correlations. The dissemination is performed based on the correlated tree. In the correlated tree, a node selects its parent with strong link quality as well as strong link correlation, so that the total number of transmissions can be reduced.

Another important feature of CD is that it selectively uses rateless codes [17] to further improve the performance when link correlation is inherently weak. With rateless codes, a node can decode a data page when receiving a sufficient number of distinct, encoded packets. Therefore, the number of transmissions in a neighborhood is determined by the largest number of missing packets on the neighbor with the worst link quality, instead of the total number of missing packets

Manuscript received April 20, 2014; revised September 17, 2014 and January 24, 2015; accepted March 10, 2015.

W. Dong is the corresponding author (phone&fax: (86)571-87953955; e-mail: dongw@zju.edu.cn).

Z. Zhao, J. Bu, C. Chen are with the College of Computer Science, Zhejiang University, China. E-mail: {zhaozw, bjj, chenc}@zju.edu.cn

Y. Gu is with the Future Systems Department, IBM Research-Austin, USA. E-mail: yugu@us.ibm.com

in the neighborhood. Hence, the transmission cost can be reduced. On the other hand, the use of rateless codes incurs a large decoding overhead, causing delays in the dissemination process. Previous experiments show that rateless codes can improve the performance with weak link correlation while impede the performance with high link correlation [18]. CD incorporates an accurate model to estimate the benefits of rateless codes, and thus can selectively employ rateless codes with respect to the network conditions.

The key novelties of CD are summarized as follows:

- *First and Most Important, CD Employs A Novel Structure Called Correlated Tree, Which Explicitly Considers Both Link Correlations And Link Qualities For Efficient Dissemination* (Section IV-C).
- *It Selectively Chooses Coding Strategies According To The Wireless Conditions, To Achieve The Minimum Dissemination Delay* (Section IV-C).

We implement CD based on TinyOS [21] and OpenWSN [22] (with 802.15.4e MAC for IWSNs), and conduct extensive experiments and TOSSIM simulations. The evaluation results show that compared with Deluge and Rateless Deluge, CD (1) reduces the completion time by 38.5% and 56.2% respectively; (2) reduces the number of transmissions by 59.8% and 26.9% respectively; (3) reduces the energy consumption by 69.1% and 74.2% respectively.

The rest of this paper is structured as follows. Section II describes related work, Section III describes our motivation. Section IV presents the design details of CD. Section V presents the compliance with 802.15.4e MAC protocol. Section VI shows the evaluation results. Section VII gives discussions on open issues. Finally, we conclude this paper in Section VIII. Please note that some complementary technical issues and insights can be found in our technical report [1].

II. Related works

Deluge [12] is a popular dissemination protocol based on TinyOS [21], which employs ADV-REQ-DATA three-way message exchanges to ensure eventual consistency. A Deluge node periodically broadcasts ADVs to claim its latest data version. Upon receiving ADVs, a node replies a REQ if a newer data version is found. After receiving REQs, a node begins broadcasting DATA packets. If some DATA packets are lost during the transmission, receivers will repeatedly send REQs, asking for the missing packets. Deluge segments a bulk data object into data pages to allow parallel transmissions of different pages in the network.

Most traditional approaches follow the framework of Deluge and transmit native packets during dissemination [15], [16], [23]–[25]. In recent years, several coding-based dissemination protocols in IWSNs/WSNs are proposed, such as Rateless Deluge [17], Synapse [26] and ReXOR [27]. These protocols use various coding schemes to reduce the number of transmissions. In all these protocols, an encoded packet can be decoded at multiple receivers once sufficient number of encoded packets are received. Hence, the retransmissions

[1]The report is publicly available at http://www.emnets.org/pub/zzw/cd-tie-tech-rep.pdf

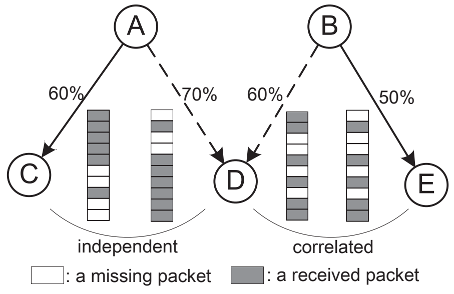


Fig. 1. An example in which A and B intend to transmit a page of 10 packets to C, D and E

are greatly reduced and the performance can be significantly improved in lossy networks. On the other hand, the decoding procedure can also bring a relatively long decoding delay, especially when the link qualities are good and the number of retransmissions is inherently small.

SYREN [28] is another recent work that employs rateless codes. It uses link correlation to infer the ACKs of correlated links, which can reduce the negotiation overhead. CD differs from SYREN in two major ways: Firstly, SYREN does not guarantee 100% reliability, which is a crucial issue in bulk data dissemination. In contrast, CD efficiently ensures 100% reliability. Secondly, SYREN always uses rateless codes regardless of the link conditions, while CD *selectively* employs rateless codes, thus achieving the minimum transmission delay in a neighborhood.

In high level, CD combines the benefits of both coding and non-coding approaches. It incorporates an accurate model to predict the performance of both strategies and can selectively utilize the strategy that achieves the best performance given the network condition. A more comprehensive review of the literature can be found in the technical report.

III. Motivation

In this section, we use some examples to motivate our work.

A. Exploiting link correlation

Fig. 1 illustrates an example in which nodes A and B intend to send the same page consisting of 10 data packets to C, D and E. The percentage value above each directed edge indicates the link quality. Link A→C and link A→D are independent while link B→D and B→E are strongly correlated, i.e., when D loses a packet, E loses the same packet with high probability. The rectangles near each node represent the packet receptions with a white rectangle indicating a missing packet and a grey rectangle indicating a correctly received packet.

If we disseminate packets according to the energy efficient tree introduced in [29] in which a node selects its parent if the parent has the best incoming link quality to the node, D should select A as its parent as the link quality of A→D is better than B→D. In this situation, A should retransmit 4+3=7 packets in the next round to cover both C and D while B should retransmit 5 packets in the next round to cover E. Totally, 7+5=12 transmissions are required in the next round.

If we take link correlation into account and let D select B as its parent, however, only 4+5=9 retransmissions are required

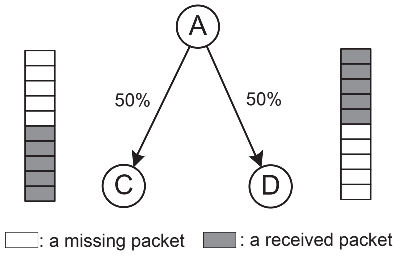


Fig. 2. An example in which A intends to transmit a page of 10 packets to C and D

in the next round, reducing 3 packet transmissions compared with the tree in [29].

More generally, with the same packet error rates, independent links tend to have a higher number of differing lost packets while correlated links tend to have a higher number of common lost packets. This implies that selecting a sender with correlated links will be more beneficial as it results in less transmission overhead to cover the intended receivers.

Section IV-C describes how we can construct a correlated tree that takes link correlation into account.

B. Selectively employing rateless codes

Fig. 2 shows an example in which the link correlation between two links with the same parent is inherently weak.

In this case, the transmission of native packets incurs 10 retransmissions in the next round, incurring 4×10=40 ms delay with the assumption that each transmission takes 4ms. On the other hand, the transmission of encoded packets incurs only 5 retransmissions in the next round, incurring 4×5+12ms=32ms delay where the 12ms is the decoding delay of 5 encoded packets (Rateless Deluge on TelosB motes). We can see that the use of rateless codes can reduce the transmission delay by 8ms (20%) in this case.

It is also worth noting that rateless codes can degrade the performance when the link correlations are strong. For example, assume the correlation of links A→C and A→D is 100%, i.e., C and D lose the same five packets. In this case, the transmission of 5 native packets incurs less delay than the transmission of 5 encoded packets, as the latter introduces decoding delay.

Section IV-D describes how we can accurately predict the performances of both strategies and dynamically select the best strategy according to the network conditions.

IV. Design

In this section, we will present CD's design details.

A. Overview

CD is a bulk data dissemination protocol that has two salient features. Firstly, it exploits link correlation to improve the dissemination performance. Secondly, it selectively employs rateless codes when the link correlation in a neighborhood is inherently weak. Overall, CD consists of four phases: **Initialization phase**. In this phase, the network nodes obtain hop counts and exchange necessary control information. **Correlated tree construction phase**. Based on the information collected during the initialization phase, a correlated tree is constructed in a distributed manner. **Coding decision phase**. Each node incorporates a model to estimate the performance of coding and non-coding strategies and selectively employs rateless codes if it is beneficial. **Dissemination phase**. Each node starts dissemination according the strategy (i.e. coding or non-coding). CD uses a prioritized REQ mechanism to ensure reliability and efficiency.

We now present the design details of CD in different phases in the following subsections.

B. Initialization phase

In this phase, the network nodes transmit and exchange necessary control information as follows.

We first give necessary notations as follows:

Notations	Descriptions
q_{ki}	The link quality from node k to node i.
B_{ki}	The bit vector indicating the packet receptions/losses of the link $k \rightarrow i$
$B_{ki}(m)$	The mth bit in B_{ki}, with "1" denoting a packet loss and "0" denoting a packet reception.
$c_k(i,j)$	The link correlation of $k \rightarrow i$ and $k \rightarrow j$
$p_k(j\|i)$	The conditional probability that $k \rightarrow j$ loses a packet given that $k \rightarrow i$ loses the same packet.
$c_k(i,S)$	The link correlation of $k \rightarrow i$ and the link set $\{k \rightarrow x \| x \in S\}$
h_k	Node k's hop count
$m_i(k)$	The integrated metric calculated by node i to estimate k's potential as a sender.

Firstly, the sink node initiates a flooding for each network node to obtain its hop count. The hop count is used to avoid loops in the correlated tree construction phase. Secondly, each node k (with hop count h_k) periodically broadcasts HELLO messages. A HELLO message contains the node ID k and a sequence number. Node i is a 1-hop downstream node of node k ($h_i = h_k + 1$). It keeps HELLO message receptions for sender k in a bit vector B_{ki} with each bit representing whether a HELLO message is received (where 0 denotes a successful reception and 1 denotes a failed reception). We use $B_{ki}(m)$ to denote the m-th bit in the bit vector. Due to the memory limits, node i only keeps bit vectors for the latest HELLO messages of each neighboring node (e.g., the latest 30 messages) and uses moving average [30] for link estimation. It is worth noting that many applications already use periodic HELLO/beacon messages for maintaining network connectivity [31], thus we can reuse these HELLO/beacon messages to reduce the measurement overhead.

Thirdly, with the bit vectors, node i can calculate its incoming link quality as:

$$q_{ki} = 1 - \frac{\sum_{m=1}^{|B_{ki}|} B_{ki}(m)}{|B_{ki}|} \tag{1}$$

where $|B_{ki}|$ denotes the length of the bit vector B_{ki}.

We define the link correlation $c_k(i,j)$ as the probability that j loses a packet provided i loses the same packet (i.e., the percentage that packet losses on $k \rightarrow i$ are covered by packet losses on $k \rightarrow j$). We calculate $c_k(i,j)$ as follows:

$$c_k(i,j) = p_k(j|i) = \frac{\sum_{m=1}^{|B_{ki}|} B_{ki}(m) \& B_{kj}(m)}{\sum_{m=1}^{|B_{ki}|} B_{ki}(m)} \tag{2}$$

For the example shown in Fig. 1, the correlation of $B \to D$ and $B \to E$ is $\frac{4}{4} = 100\%$, which means the packet losses on $B \to D$ are 100% covered by the packet losses on $B \to E$. The correlation of $B \to E$ and $B \to D$ is $\frac{4}{5} = 80\%$, which means the packet losses on $B \to E$ are 80% covered by the packet losses $B \to D$.

We further extend the correlation concept to describe how the packet losses on a particular link are covered by the union of packet losses on a set of links with a common sender. We define

$$c_k(i,S) = \frac{\sum_{m=1}^{|B_{ki}|} B_{ki}(m) \& B_{kS}(m)}{\sum_{m=1}^{|B_{ki}|} B_{ki}(m)} \tag{3}$$

where S denotes the set of links and $B_{iS} = \bigcup_{j \in S} B_{ij}$. Both link quality (Eq.(1)) and link correlation (Eq.(2) and Eq.(3)) are calculated at the receiver side.

C. Correlated tree construction phase

In this phase, nodes construct a correlated tree in a distributed manner. The correlated tree considers both link qualities and link correlations in a neighborhood in order to improve the dissemination performance.

Firstly, potential sender node k broadcasts a CLAIM message to initiate the parent selection. The CLAIM message includes all its 1-hop downstream nodes (including i). A CLAIM message from node k can be described as $< k, \{j, B_{kj}\}_{h_j = h_k + 1} >$ where k denotes the source node ID from which the CLAIM message originates, j denotes a downstream node of k, and B_{kj} is the reception vector. We call $\{j\}_{h_j = h_k + 1}$ a group of k, indicating that any node j can be a child of node k. Upon receiving the bit vectors in the CLAIM message, node i updates q_{ki} and $c_k(i,S)$.

Secondly, node i selects its parent. Node i can select k as parent only when i is included in k's CLAIM. When multiple CLAIMs include i, i will favor the group in which it has higher correlation with the remaining nodes in that group. Strictly speaking, i favors a group if the packet losses on i have a high chance to be covered by packet losses in that group. In this case, the addition of i to that group is supposed not to introduce additional overhead because the lost packets will be retransmitted anyway. We denote the set of nodes in k's group except i as S (i.e., $\forall j \in S$, $h_j = h_k + 1$ & $j \neq i$). This correlation can be denoted as $c_k(i,S)$ and can be calculated according to Eq. (3).

Node i should also favor the group in which it has better link quality q_{ki} from the parent k. Therefore, we propose a new metric $m_i(k)$ that combines the effects of link correlation and link quality within the group: $m_i(k) = \alpha c_k(i,S) + (1-\alpha) q_{ki}$ with $\alpha = 0.5$. Node k with the largest metric value will be selected as parent by i. Note that the metric is general: with $\alpha = 0$, the resulting tree will be the energy efficient tree described in [29], which considers only link quality. With $\alpha = 1$, the metric takes only link correlation. With $0 < \alpha < 1$, both link quality and correlation are considered. We will study the impact of different α values in section VI-B.

Thirdly, after parent selection, node i broadcasts PSLT (Parent Selection) messages to make notifications in the neighborhood so that the network can keep a consistent view. With the above three steps, the correlated tree is constructed.

We revisit the example in Fig. 1. After receiving CLAIM messages from both A and B, D calculates the new metric $m_D(A)$ and $m_D(B)$. $m_D(A) = \alpha \times c(D,C) + (1-\alpha) \times q_{AD} = 0.5 \times 0 + 0.5 \times 0.7 = 0.35$ while $m_D(B) = \alpha \times c(D,E) + (1-\alpha) \times q_{BD} = 0.5 \times 1 + 0.5 \times 0.6 = 0.8 > 0.35$. Hence, node D will select node B as its parent, resulting improved dissemination performance.

D. Coding decision phase

In this phase, each node builds a model to estimate the transmission delay in its group in order to decide whether to employ rateless codes.

κ factor [18] experimentally observes that link correlation significantly affects the protocol performance: Deluge performs better when the link correlation is strong while Rateless Deluge performs better when the link correlation is weak. We would like to build a mathematical model so that each node can incorporate this model to make coding decisions. We consider the transmission delay of a page within a group with parent k.

We introduce the following notations before the modeling.

Notations	Descriptions
N	The page size in the number of packets
q_{kw}	The worst link quality of k's outbound links.
τ	The transmission delay of a single packet (including the MAC backoff)
D	The decoding time of rateless codes
S	The set of downstream nodes of i: $\forall j \in S$, $h_j = h_i + 1$
n	The number of downstream nodes, i.e., $n = \|S\|$
w	The child node of k that has the worst link quality: $q_{kw} = \min_{i \in S}(q_{ki})$
$P_n^k\{X = m\}$	The probability that k needs to transmit m packets to cover n receivers
$P_{n-1/n}^k$	The probability that at least one node in the remaining $n-1$ nodes loses a packet from k, given that the n-th node loses the same packet, i.e., $P_{n-1/n}^k = c_k(n, S_{n-1}) = p_k(S_{n-1}\|n)$ where S_{n-1} denotes the remaining $n-1$ nodes
t_{REQ}	The transmission delay of REQ message (including the MAC backoff)
t_{coding}	Transmission delay of a page when using rateless codes
t_{native}	Transmission delay of a page when using native packets

(1) **Transmission delay with rateless codes**. We use the default TinyOS CSMA/CA MAC layer protocol. The 802.15.4 MAC operates as follows. When a packet is delivered to the MAC layer, the MAC layer performs an initial backoff without sensing the channel. The backoff timer is randomly set in [0,CW_{init}) where CW_{init}=9.8ms. When the initial backoff timer fires, the MAC layer performs clear channel assessment (CCA) to check whether the channel is busy. If clear, the data is sent immediately. Otherwise, the MAC layer performs a congestion backoff. The congestion backoff timer is randomly set in [0,$CW_{congestion}$) where $CW_{congestion}$ ≈2.5ms. The MAC layer repeats the congestion backoff until the channel is clear.

We first calculate the transmission delay for single packet (including MAC backoffs). Assume there are no transmission collisions, the expected backoff time t_b is 9.8/2=4.9ms. The packet transmission time is $\frac{S_p}{R}$, where S_p is the packet size (including the packet header) and R is the radio transmission rate. Then the expected data packet transmission time $\tau = t_b + \frac{S_p}{R}$(ms). Similarly, $t_{REQ} = t_b + \frac{S_r}{R}$(ms) where S_r is the REQ packet size.

Next we calculate the expected number of transmissions for N packets. We note that the number of transmissions depends on the worst link quality in the group, q_{kw}. The expected number of data packet transmissions is $\frac{N}{q_{kw}}$ and the expected number of REQ messages is $\frac{1}{q_{kw}}$. The transmission delay is then calculated as:

$$t_{coding} = \frac{N}{q_{kw}}\tau + \frac{1}{q_{kw}}t_{REQ} + D \tag{4}$$

(2) **Transmission delay without rateless codes**. We would like to calculate the expected number of transmissions (of a single data packet) at node k (to cover all n downstream nodes), E_{pkt}. We first calculate the probability that the number of transmissions are more than m times, $P_n^k(X > m)$, which equals to the probability that m transmissions could not cover all the n nodes:

$$P_n^k(X > m) = (1-q_{kn})^m + P_{n-1}^k(X > m) - ((1-q_{kn}) \times P_{n-1/n}^k)^m \tag{5}$$

where $(1-q_{kn})^m$ denotes the probability that m transmissions cannot cover the n-th node, $P_{n-1}^k(X > m)$ denotes the probability that m transmissions cannot cover the remaining $n-1$ nodes, i.e., there is at least one node which cannot be covered by m transmissions, and $((1-q_{kn}) \times P_{n-1/n}^k)^m$ denotes the probability that m transmissions cannot cover the n-th node and at least one node in the remaining $n-1$ nodes.

We can calculate $P_n^k(X > m)$ recursively starting from $P_1^k(X > m) = (1-q_{k1})^m$ as follows:

$$P_n^k(X > m) = \sum_{r=1}^{n}((1-q_{kr})^m - ((1-q_{kr}) \times P_{r-1/r}^k)^m) \tag{6}$$

We note that $P_{0/1}^k = 0$ according to the definition.

Therefore, the probability that the expected number of transmissions is k can be calculated as:

$$P_n^k(X = m) = P_n^k(X > m-1) - P_n^k(X > m) \tag{7}$$

To cover all n nodes, the expected number of transmissions of a single packet can then be calculated as follows:

$$E_{pkt} = \sum_{m=1}^{+\infty} m P_n^k(X = m) \tag{8}$$

Hence, the time for a page transmission is:

$$t_{native} = N E_{pkt}\tau + \frac{1}{q_{kw}}t_{REQ} \tag{9}$$

CD uses Eq. (9) and Eq. (4) to estimate the delay performance of the two strategies and selectively uses rateless codes when $t_{coding} < t_{native}$. Note that we assume that each node sends REQs to k, if it has packet losses after a round of page transmission is finished. Node k collects and combines all REQs before the next round of transmission.

E. Data dissemination phase

In this phase, CD disseminates the bulk data according to the established correlated tree and transmission strategy (i.e., coding or non-coding).

The sink initiates the data dissemination. After each node receives an entire page, it starts transmitting the current page. We use the mechanism in MNP [23] to resolve possible contentions. Firstly, a node will postpone its transmission when it overhears that a page transmission with lower page number. Secondly, with the same page numbers, a node will postpone its transmissions when it has a larger node ID. As each node has a unique ID, such mechanism is effective.

CD employs a prioritized REQ mechanism to reduce the transmission cost of REQs and data packets. The intuition is that in a group of nodes, we should let the node with the worst link quality request first so that the following data transmissions benefit all remaining nodes with a high probability, especially when the link correlation is strong.

The prioritized REQ mechanism works as follows. Firstly, each node estimates the end time of a page transmission by observing the packet index within a page. Secondly, starting from the end time, each node starts a prioritized timer to control the transmission of the REQ. The priority is determined by the link qualities. Basically speaking, the node with the worst link quality will most likely to transmit the REQ first.

CD incorporates the coordinated schedules proposed in [32]. C, P and Q slots are used for transmitting, receiving and sleeping respectively. This is for two purposes. Firstly, this allows effective pipelining. Secondly, this reduces the energy consumption on sensor nodes. Neighboring nodes are locally synchronized such that when a node is in P slot, its parent is in Q slot and its children are in C slot.

Validations of the model and system optimizations with prioritized REQs can be found in the technical report.

V. Compliance with IEEE 802.15.4e

IEEE 802.15.4e [33] is chartered to define a MAC amendment to the existing standard 802.15.4-2006, to better support industrial markets. Compared with the typical TinyOS MAC [21], it has the following distinct features: (1) Time is slotted and globally synchronized in 802.15.4e. (2) 802.15.4e incorporates time synchronized channel hopping (TSCH) to improve the link reliability. (3) Instead of a constant back-off timer window (0∼BW-1, where BW≈10), 802.15.4e uses an exponential initial back-off timer window (0∼ $2^{BE}-1$, where BE can vary from 3 to 5, the default values macMinBE and macMaxBE). The expected back-off time duration is longer than that of TinyOS MAC.

In general, 802.15.4e MAC improves the dissemination performance. The reason is that (1) TSCH improves the link reliability, thus reducing number of retransmissions. (2) The multi-channel schedules allow a more efficient pipeline manner: under TinyOS MAC, only nodes from three hops away in the same path can transmit packets simultaneously in order to avoid intra-path collisions. While under 802.15.4e MAC, nodes from only two hops away are able to transmit at the same time in different channels.

Link correlation also have a strong impact on dissemination performance under 802.15.4e. The problem is similar: (1) When a node has multiple potential senders, which should it select? (2) For a given channel, or a set of channels, is it better to use rateless codes or not? The key problem to migrate CD's design to 802.15.4e is that it should establish the correlated tree structure as well as slot schedules. To support both peer to multiple peers (P2MP) and multiple peers to peer (MP2P) traffics, we assume the flexible slot schedules, which support bidirectional communications: when the data flows are from leaf nodes to sink node, the default routing topology and schedules are used. When the data flows are from sink node to leaf nodes, the established correlated tree structure topology and corresponding schedules are used. We note that the two directions rarely happen at the same time since the latter is often used for network update or reconfiguration, during which nodes are in different states and the network is temporarily unavailable for data collections.

A. Constructing correlated tree and determining channels for active slots

We slightly modify the design of our design as follows: (1) Each node measures the link quality as well as link correlation using periodic beacons (as described in Section IV-B) for all 16 channels. (2) Each node choose the best l channels as candidate channels used in active slots as senders, and estimates the average ETX_{pkt} to its downstream neighboring nodes. Note that l is the number of available channels in the schedules. (3) Each node transmits a CLAIM message to its downstream nodes in default channel, containing the ETX_{pkt} and l channels. (4) When receiving the CLAIM messages, a node i calculates the integrated metric $m_i(k)$ for each potential sender k as described in Section IV-C, and selects the node with maximum $m_i(k)$. At the same time, it obtains its channel offsets maintained by the selected node k. After the above steps, a correlated tree structure and channels used for active slots are done for each node. Next, we schedule the selected channels and slots for TSCH.

B. Slot scheduling

For each node in slot *ASN* (Absolute Slot Number), the channel to be used is calculated as follows: `channel = (ASN+channelOffset)/l` where `channelOffset` is the default channel used at the beginning slot, l is the number of available channels. Since *ASN* is globally shared in a network, once we determine the `channelOffset` and available channels for each node, the slot scheduling is done. After the tree construction, each node has obtained its own available channels for active slots. The only thing we should do is to determine the `channelOffset`. We divide all slots into odd number slots and even number slots. For a specific node, it either transmits in odd slots and receives in even slots, or transmits in even slots and receives in odd slots. For odd and even number of slots, we assign different `channelOffset` for a node. The assignment of `channelOffset` should obey the following rules:

Rule 1: Neighboring nodes with the same hop count should have different offset for sending slots.

Rule 2: Nodes in the same path should be in the same sending slots with different channel offset when they are two hops away.

Rule 3: When a sender node is in its sending slot (even/odd), its receivers should be in receiving slot (odd/even). The offsets of the sender and the receivers should be the same.

An example schedule can be found in the technical report. A limitation of such strategy is that it may suffer starvation of channels when employed in dense networks. When there are not enough available channels, we simply use CSMA/CA to resolve possible interference and collisions. For the scheduling algorithms, we can simply follow the heuristics of existing approaches [34].

VI. Evaluation

In this section, we first study the effect of CD's selective coding decision. Then we conduct experiments on a 5×10 TelosB nodes testbed and compare the performance of CD with Deluge and Rateless Deluge in terms of completion time, transmission cost, and energy efficiency with different experimental configurations. We also evaluate the performance of CD, Deluge and Rateless Deluge in the IEEE 802.15.4e standard with OpenWSN[2] and TelosB motes.

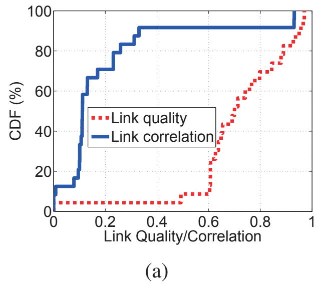
A. Methodology

We set the CC2420 power to -32.5 dBm to form a multi-hop network in our testbed. In the network, 50 nodes are used, and each relay node has 2-4 downstream links. Fig. 3(a) shows the cumulative distribution function (CDF) of pair-wise link qualities and link correlations. With the power setting, link qualities are different with different link pairs. We can see that 17.4% links are good links with packet reception ratio (PRR) $> 90\%$, 13.1% links are intermediate links with PRR in the range of 10%–90%, and 69.5% links are poor links with PRR $< 10\%$. The solid line shows the CDF of average outbound link correlation for each node. We can see that there are good link correlations as well as poor link correlations, i.e., with correlation coefficients close to 0 and 1 respectively.

In CD, we set the page size and packet payload size as Deluge's default settings in order for a fair comparison. The page size is 48 packets per page and packet payload size is 23 bytes. For the rateless codes, we use fountain code in SYNAPSE++ [26].

We use three key metrics for comparison: completion time, number of transmissions and energy consumption. Energy consumption of each node can be obtained by summing up the energy consumptions of all operations involved in the dissemination process.

[2]The OpenWSN project serves as a repository for open-source implementations of protocol stacks based on Internet of Things standards, using a variety of hardware and software platforms. https://openwsn.atlassian.net/



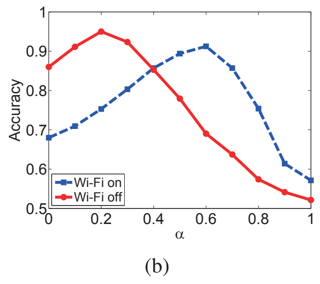


Fig. 3. Link behaviors and parent selection accuracy. (a) Link quality and correlation of the testbed; (b) Parent selection accuracy with varying α.

B. Impact of α

In the tree construction phase, node i selects node k with the largest $m_i(k)$ as parent. Intuitively, a node with larger metric value is expected to have less transmission cost. Apparently, the weighting factor of link correlation in the metric, α, is the impacting key parameter.

In this subsection, we tune the α value and run CD in a simple network under different link conditions. We would like to see whether a node can accurately select the node that achieves better performance using different α values, and find the α value that achieves the best performance.

We consider the case for parent selection with a three hop network: one sink node S, two relay nodes A and B and three leaf nodes C,D and E. We let S disseminate 100 packets to the network. Node D is the common potential receiver of both A and B. After receiving CLAIM messages from A and B, node D calculates the metrics m_A and m_B, and selects the node (A or B) with larger metric value as parent.

We tune the α value from 0 to 1 (with step 0.1). For each α value, we repeat node D's parent selection for 100 times. After D selects one node as parent and the dissemination is finished, we manually set the other node as parent node and repeat the experiment to compare the dissemination transmissions. If node D's selection achieves fewer transmissions than the other choice, the selection is right. Otherwise, the selection is wrong. Then we can get the parent selection accuracy of D by calculating the ratio of right selections among all parent selections.

We conduct two separate experiments under different environments: one with Wi-Fi interference (more correlated links [18]) and the other without Wi-Fi interference (more independent links).

Fig. 3(b) shows the parent selection accuracies with different α values and different conditions. First of all, we can see that neither α=0 nor α=1 achieves the best performance, which means both link quality and link correlation are important impacting factors to be considered. Next, the best α is different under different conditions. Under Wi-Fi interference, $\alpha = 0.6$ achieves the best performance. Without Wi-Fi interference, $\alpha = 0.2$ achieves the best performance. The reason is that when there is Wi-Fi interference, there are more correlated links to exploit. Thus we should weight link correlation more (i.e., using a large α) to exploit the correlated links. Otherwise, when link correlation is inherently weak in the network, link correlation will not help improving dissemination performance. Then we should use a small α.

TABLE I
DELAY COMPARISON WITH CORRELATED FLOODING

Protocol \ Network size	10x10	20x20	30x30
CD	0.448	1.208	2.368
Correlated Flooding	2.506	5.185	8.287

C. Comparison with Correlated Flooding

In this section, we compare the tree construction delay overheads of Correlated Flooding and CD.

Correlated Flooding [20] exploits link correlation for efficient flooding in low duty cycle sensor networks. There is a tree construction phase in both Correlated Flooding and CD. As discussed in Section II, we do not directly compare Correlated Flooding and CD in terms of the dissemination performance. The reason is that Correlated Flooding only deals with flooding one single packet at a time and do not ensure 100% reliability. As a result, data object segmentation, multi-hop pipelining and the three-way handshake mechanism are not incorporated in Correlated Flooding. Hence, it is unfair to directly compare the dissemination performance metrics.

Instead, the construction of the structure is with the same context and also considers link correlation. Then we compare the structure construction delay of CD and CF. Since TOSSIM executes the node program in PC's clock, we compensate for the computation delay in order to model the actual computation delay on sensor nodes.

Table I compares the tree construction delays for Correlated Flooding and CD in three grid topologies where adjacent nodes have perfect link qualities and other nodes have no direct communication links. The message delivery delays of both Correlated Flooding and CD are similar. The main difference is that Correlated Flooding involves a computation intensive k-means algorithm to group the downstream nodes in each tree level. Hence, the computation delay of k-means (which is approximately 225ms) is accumulated at each tree level. We see that CD's tree construction delay is much shorter than Correlated Flooding in different network topologies since CD incorporates a lightweight tree construction.

D. Comparison with Deluge and Rateless Deluge

We tune the data object size, communication channels and nodes distances, to study the performance of CD under different scenarios. We first give the evaluation results in two typical channel 26 and 16, of which 26 is non-overlapped with WiFi (strong link quality but weak link correlation) while 16 overlaps with WiFi (weak link quality but strong link correlation).

1) Channel 26: Fig. 4(a) compares the performances of CD, Deluge and Rateless Deluge in terms of the number of transmissions. The x-axis indicates the data object size in pages. We can see that (1) CD reduces the data transmissions by 44.3% compared with Deluge. There are two reasons. Firstly, the construction of correlated tree helps to select senders with good link qualities and link correlations. Secondly, CD selectively uses rateless codes, resulting fewer packet transmissions compared with using native packets. (2)

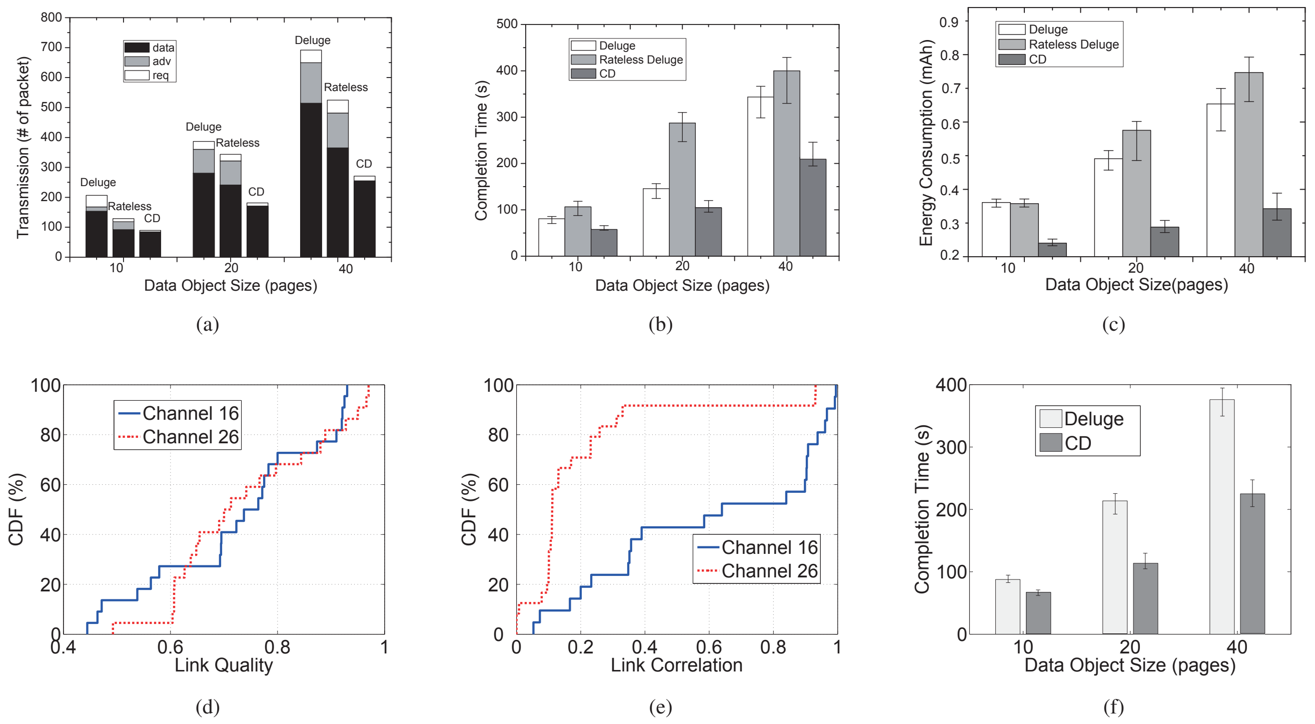


Fig. 4. Testbed results. (a) Average number of transmissions. (b) Completion Time. (c) Average energy consumption (d) CDF of link quality. (e) CDF of link correlation. (f) Completion time in Channel 16.

CD also reduces the data transmissions by 27.5% compared with Rateless Deluge. This is interesting as one would think that in a given neighborhood, using rateless codes achieves the smallest number of transmissions. Selectively using rateless codes is supposed to yield more transmissions than always using rateless codes. The reason why CD can reduce the transmissions resides in the fact that the senders are different. The correlated tree employed by CD helps to select better senders with better outbound link qualities and link correlations than Rateless Deluge. (3) CD greatly reduces the number of control packet transmissions compared with both Deluge and Rateless Deluge. CD adopts a "push"-based policy when all other nodes do not possess the data object, resulting much fewer ADV traffic. Moreover, CD employs prioritized REQ scheme with which nodes with good link qualities can effectively suppress their REQs.

Fig. 4(b) compares the performance of CD with Deluge and Rateless Deluge in terms of completion time. (1) Compared with Deluge and Rateless Deluge, CD reduces the completion time by 55.2% and 37.8%. There are two reasons. Firstly, the construction of correlated tree in CD helps to select good senders, resulting in fewer packet transmissions. Secondly, CD intelligently uses rateless codes only when its delay performance is better than using native packets. From the figure, we also see that as the data object size increases, the reduction becomes larger.

Fig. 4(c) compares the performance of CD with Deluge and Rateless Deluge in terms of the average energy consumption. CD significantly reduces the energy consumption in two ways. Firstly, CD achieves a shorter completion time than Deluge and Rateless Deluge. Secondly, CD employs sleep control. In CD, non-leaf nodes can go to sleep for approximately 1-page transmission time after transmitting a page, while leaf nodes can go to sleep for 2-page transmission time after receiving a page. As radio operations consume much more energy than CPU operations [35], the coordinated schedules contribute to most of the energy reduction, thus the reduction of leaf nodes and non-leaf nodes approximates 2/3 and 1/3 respectively. An interesting phenomenon is that Rateless Deluge has a larger energy consumption than Deluge while it has fewer transmissions than Deluge. We explain this as follows: Although Rateless Deluge has fewer transmissions, its decoding procedure incurs considerably long delay (3s for decoding 32 encoded packets). As it does not incorporate any sleep mechanism, this delay greatly increases the idle listening time, resulting in that its energy consumption is the highest.

2) Channel 16: Fig. 4(d) shows the CDF of average pair-wise link quality in channel 16 and channel 26, respectively. The average link qualities for channel 16 and channel 26 are similar (0.765 and 0.81, respectively). Fig. 4(e) shows the CDF of link correlation for all link pairs of a common sender. As expected, channel 16 experiences a higher link correlation than channel 26.

Fig. 4(f) shows the completion time of CD and Deluge in channel 16. CD outperforms Deluge in both channels. However, CD's performance improvement to Deluge at channel 16 is larger than that at channel 26. This can be explained as follows. As analyzed in Section IV-C, when link quality is similar for the two channels, link correlation essentially reshapes the missing packets of each node and affects the number of common missing packets. At channel 16, there are more cases that receivers have the similar missing packets. As a result, CD has more opportunities to select senders with both good outbound link qualities as well as strong link correlations in the neighborhood. While in channel 26, the link correlation is inherently weak, and the receivers tend to have random or different missing packets. There are fewer opportunities for CD to find neighborhoods with strong link correlation. Hence, CD's performance in channel 16 is better than channel 26. In contrast, Deluge has similar performances in channel 16 and channel 26. Consequently, the performance improvement of CD to Deluge in channel 16 is larger than that in channel 26. Complementary results with 802.15.4e and system insights can be found in the technical report.

VII. Discussion

A. Wireless dynamics

When links are dynamic during the dissemination phase, the parent nodes selected in the tree construction phase in CD may be out of date and the structure may be detrimental. Under such a situation, CD is likely to become less efficient. On the other hand, Deluge-like approaches are based on an on-the-fly dissemination strategy, and are likely to dynamically select the senders with good link qualities. As a result, CD's performance will degrade and Deluge-like approaches will perform better in such situation.

A possible way for CD to deal with wireless dynamics is to re-construct the correlated tree structure every certain number (denoted as n_s) of pages transmission, such that the structure can be up to date. When deciding n_s, there is a tradeoff between the reconstruction overhead and the adaptation of wireless dynamics. When n_s is small, CD is more adaptive to the wireless dynamics, but more reconstruction overhead will be incurred. When n_s is large, CD is less adaptive to the wireless dynamics, but less reconstruction overhead will be incurred.

B. Slot scheduling

Currently in CD's design, the lengths of the three slots are the same. The reason is that (1) the receiving slot length is the same with sending slot, and (2) the sleep slot length is the same with the sending slot of next hop senders. Overall, all three slots are with the same length.

The main drawback of current design is that not all sending slots are fully occupied by packet transmissions. In some slots, there are only a few packet transmissions. Apparently, the wasted time in these slots adds to the eventual dissemination delay.

Intuitively, we would like to adaptively tune the slot length according to the ongoing transmissions for specific slots. When there is much transmission load, the slot should be long; while when there is less transmission load, the slot should be short. However, the main challenge is the consistency problem as follows: If a node changes its sending slot length according to its own transmission load, its parent and child nodes will also have to change the sleep and receiving slot lengths, which further affect the slot schedules of the whole network. More importantly, because of different transmission load, other network nodes may have conflict slot length settings.

We would like to explore the above two directions in our future work.

VIII. Conclusion

Many real world projects [36] [37] have employed bulk data dissemination protocol for network reprogramming. The dissemination completion time is a key metric for reprogramming in IWSNs/WSNs. At the same time, considering the constrained resources on sensor nodes, the dissemination protocols should also be transmission and energy efficient.

In this paper, we investigate the relationship between link correlation and the dissemination performance. Then we propose a novel dissemination protocol call CD, considering link correlation and rateless codes. The proposed correlated tree structure can effectively exploit the correlated links in a network. Evaluation results show that CD improves the dissemination performance in terms of completion time, transmissions and energy consumption. However, as discussed in VII-A, when the wireless is highly dynamic during the dissemination, CD's performance will likely degrade and the Deluge-like approaches tend to perform better.

Directions of future works: Firstly, optimizing CD to be more adaptive to wireless dynamics. Secondly, adaptive slot scheduling for further reducing the dissemination delay.

Acknowledgements

This work is supported by the National Science Foundation of China under Grant No. 61472360 and 61202402, the Fundamental Research Funds for the Central Universities, and the Research Fund for the Doctoral Program of Higher Education of China (20120101120179). Many thanks to Yonggang Wang for his efforts with this work.

References

[1] V. C. Gungor and G. P. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4258–4265, 2009.

[2] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the internet of (important) things," *IEEE Commun. Surv. Tut.*, vol. 15, no. 3, pp. 1389–1406, 2013.

[3] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, and X. Li, "Does wireless sensor network scale? a measurement study on greenorbs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 10, pp. 1983–1993, 2013.

[4] M. Li and H.-J. Lin, "Design and implementation of smart home control systems based on wireless sensor networks and power line communications," *IEEE Trans. Ind. Electron.*, vol. PP, no. 99, pp. 1–1, 2014.

[5] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun, "Distributed collaborative control for industrial automation with wireless sensor and actuator networks," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4219–4230, 2010.

[6] J. P. Carmo, P. M. Mendes, C. Couto, and J. H. Correia, "A 2.4-ghz cmos short-range wireless-sensor-network interface for automotive applications," *IEEE Trans. Ind. Electron.*, vol. 57, no. 5, pp. 1764–1771, 2010.

[7] P. Park, C. Fischione, A. Bonivento, K. H. Johansson, and A. Sangiovanni-Vincent, "Breath: An adaptive protocol for industrial control applications using wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 6, pp. 821–838, 2011.

[8] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE Trans. Ind. Electron.*, vol. 57, no. 10, pp. 3557–3564, 2010.

[9] D. He, C. Chen, S. Chan, and J. Bu, "Sdrp: A secure and distributed reprogramming protocol for wireless sensor networks," *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4155–4163, 2012.

[10] D. He, C. Chen, S. Chan, J. Bu, and L. T. Yang, “Security analysis and improvement of a secure and distributed reprogramming protocol for wireless sensor networks,” *IEEE Trans. Ind. Electron.*, vol. 60, no. 11, pp. 5348–5354, 2013.

[11] X. Cao, J. Chen, Y. Xiao, and Y. Sun, “Building-environment control with wireless sensor and actuator networks: centralized versus distributed,” *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3596–3605, 2010.

[12] 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.

[13] Y. W. Law, Y. Zhang, J. Jin, M. Palaniswami, and P. Havinga, “Secure rateless deluge: Pollution-resistant reprogramming and data dissemination for wireless sensor networks,” *EURASIP J. Wireless Commun. Netw.*, vol. 2011, p. 5, 2011.

[14] R. Sivakumar and I. F. Akyildiz, “Garuda: Achieving effective reliability for downstream communication in wireless sensor networks,” *IEEE Trans. Mobile Comput.*, vol. 7, no. 2, 2008.

[15] Y. Gao, J. Bu, W. Dong, C. Chen, L. Rao, and X. Liu, “Exploiting concurrency for efficient dissemination in wireless sensor networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 4, pp. 691–700, 2013.

[16] W. Dong, Y. Liu, C. Wang, X. Liu, C. Chen, and J. Bu, “Link quality aware code dissemination in wireless sensor networks,” in *Proc. IEEE ICNP*, 2011, pp. 89–98.

[17] A. Hagedorn, D. Starobinski, and A. Trachtenberg, “Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes,” in *Proc. ACM/IEEE IPSN*, 2008, pp. 457–466.

[18] K. Srinivasan, M. Jain, J. Choi, T. Azim, E. Kim, P. Levis, and B. Krishnamachari, “The κ factor: Inferring protocol performance using inter-link reception correlation,” in *Proc. ACM MobiCom*, 2010, pp. 317–328.

[19] T. Zhu, Z. Zhong, T. He, and Z. Zhang, “Exploring link correlation for efficient flooding in wireless sensor networks,” in *Proc. USENIX NSDI*, 2010, pp. 49–64.

[20] S. Guo, S. Kim, T. Zhu, Y. Gu, and T. He, “Correlated flooding in low-duty-cycle wireless sensor networks,” in *Proc. IEEE ICNP*, 2011, pp. 383–392.

[21] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, “Tinyos: An operating system for sensor networks,” *Ambient Intelligence*, vol. 35, pp. 115–148, 2005.

[22] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, “Openwsn: a standards-based low-power wireless development environment,” *Trans. Telecommun. Technol.*, vol. 23, no. 5, pp. 480–493, 2012.

[23] S. S. Kulkarni and L. Wang, “Mnp: Multihop network reprogramming service for sensor networks,” in *Proc. IEEE ICDCS*, 2005, pp. 7–16.

[24] S. Kulkarni and L. Wang, “Energy-efficient multihop reprogramming for sensor networks,” *ACM Trans. Sens. Netw.*, vol. 5, no. 2, p. 16, 2009.

[25] W. Dong, C. Chen, X. Liu, Y. He, Y. Liu, J. Bu, and X. Xu, “Dynamic packet length control in wireless sensor networks,” *IEEE Trans. Wireless Commun.*, vol. 13, no. 3, pp. 1172–1181, 2014.

[26] M. Rossi, N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, and M. Zorzi, “Synapse++: Code dissemination in wireless sensor networks using fountain codes,” *IEEE Trans. Mobile Comput.*, vol. 9, no. 12, pp. 1749–1765, 2010.

[27] W. Dong, C. Chen, X. Liu, J. Bu, and Y. Gao, “A lightweight and density-aware reprogramming protocol for wireless sensor networks,” *IEEE Trans. Mobile Comput.*, vol. 10, no. 10, pp. 1403–1415, 2011.

[28] S. I. Alam, S. Sultana, Y. C. Hu, and S. Fahmy, “Syren: Synergistic link correlation-aware and network coding-based dissemination in wireless sensor networks,” in *Proc. IEEE MASCOTS*, 2013, pp. 485–494.

[29] S. Guo, L. He, Y. Gu, B. Jiang, and T. He, “Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links,” *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2787–2802, 2014.

[30] A. Woo, T. Tong, and D. Culler, “Taming the underlying challenges of reliable multihop routing in sensor networks,” in *Proc. ACM SenSys*, 2003, pp. 14–27.

[31] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, “Protocols for self-organization of a wireless sensor network,” *IEEE Pers. Commun.*, vol. 7, no. 5, pp. 16–27, 2000.

[32] L. Huang and S. Setia, “Cord: Energy-efficient reliable bulk data dissemination in sensor networks,” in *Proc. IEEE INFOCOM*, 2008, pp. 1247–1255.

[33] IEEE Standard 802.15.4e, “Part. 15.4: Low-rate wireless personal area networks (LR-WPANs) amendament 1: MAC sublayer,” 2012.

[34] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco, and M. Dohler, “Decentralized traffic aware scheduling for multi-hop low power lossy networks in the internet of things,” in *Proc. IEEE WoWMoM*, 2013, pp. 1–6.

[35] “TelosB Datasheet.” [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf

[36] 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,” in *Proc. ACM SenSys*, 2009, pp. 99–112.

[37] X. Mao, X. Miao, Y. He, X.-Y. Li, and Y. Liu, “Citysee: urban co 2 monitoring with sensors,” in *Proc. IEEE INFOCOM*, 2012, pp. 1611–1619.

Zhiwei Zhao (S’11) received his BS degree at the College of Electronic and Information in Xi’an Jiaotong University in 2010. He is currently a PhD student at the College of Computer Science in Zhejiang University. His research interests include sensor networks and wireless computing. He is a student member of IEEE and IEEE ComSoc.

Wei Dong (S’08-M’12) received his BS and Ph.D. degrees from the College of Computer Science at Zhejiang University in 2005 and 2011, respectively. He is currently an associate professor in the College of Computer Science in Zhejiang University. His research interests include networked embedded systems and wireless sensor networks. He is a member of IEEE.

Jiajun Bu (M’06) received the B.S. and Ph.D. 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 IEEE and ACM.

Yu Gu (M’10) is currently a research scientist at Mobile Enterprise Computing Group, Future Systems Department, IBM Research-Austin. Currently his research spans mobile computing, cyber-physical systems, wireless networks, etc. Dr. Gu is the author of over 90 papers in premier journals and conferences, with the best paper/demo awards from 7 conferences. Dr. Gu is a member of IEEE, IEEE ComSoc and ACM.

Chun Chen (M’06) received his Bachelor of Mathematics degree from Xiamen University, China, in 1981, and his M.S. and Ph.D. 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.