From Rateless to Distanceless: Enabling Sparse Sensor Network Deployment in Large Areas

Wan Du, Zhenjiang Li, Jansen Christian Liando, Mo Li
School of Computer Engineering, Nanyang Technological University, Singapore
{duwan, lzjiang, cjansen, limo}@ntu.edu.sg

Abstract

This paper presents a distanceless networking approach for wireless sensor networks sparsely deployed in large areas. By leveraging rateless codes, we provide distanceless transmission to expand the communication range of sensor motes and fully exploit network diversity. We address a variety of practical challenges to accommodate rateless coding on resource-constrained sensor motes and devise a communication protocol to efficiently coordinate the distanceless link transmissions. We propose a new metric (expected distanceless transmission time) for routing selection and further adapt the distanceless transmissions to low duty-cycled sensor networks. We implement the proposed scheme in TinyOS on the TinyNode platform and deploy the sensor network in a real-world project, in which 12 wind measurement sensors are installed around a large urban reservoir of $2.5km * 3.0km$ to monitor the field wind distribution. Extensive experiments show that our proposed scheme significantly outperforms the state-of-the-art approaches for data collection in sparse sensor networks.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication; D.2.2 [**Network Protocols**]: Protocol architecture

General Terms

Design, Experimentation, Performance

Keywords

Wireless sensor network, Sparse deployment, Rateless codes, Environmental monitoring

1 Introduction

In many sensing applications for environmental monitoring [8, 33, 44, 9], spatially-sparse sampling suffices to gain adequate knowledge of the environmental phenomena in large areas, since spatial variation is limited and the environmental data is normally spatiotemporally correlated. In these applications, sensors are sparsely deployed, e.g., hundreds of meters away from each other. Traditional wireless sensor networks are not designed for such a sparse network setting. A dense network is assumed to be deployed with sensor motes of short communication distance, which results in a significant deployment waste, as many sensor nodes do not contribute to sensing data but just to maintaining the network connectivity.

Some low-power sensor devices have been developed for long-distance communication, like TinyNode [11] and Fleck-3 [8]. They provide long communication distance with low data rates. For instance, TinyNode adopts the Semtech XE1205 RF radio that increases the receiver sensitivity by a built-in low-noise amplifier and a baseband amplifier. TinyNode is able to achieve a theoretical communication distance up to 1.8km by lowering the bit rate to 1.2kb/s. While those long-distance devices provide the opportunity of building a sparse sensor network across large areas, we find the communication ranges may be significantly impaired in practice because the high sensitivity of receivers for decoding weak signals on the other hand makes decoding vulnerable to the multi-path effect from surrounding obstacles, e.g., buildings, vehicles, water surface, etc. Our in-field measurement demonstrates that the maximum communication distance of TinyNode ranges from 230 meters to 960 meters in different environments. Similar reduced communication ranges have also been observed by P. Corke et al. in [8].

In this paper, we design a software-based long-distance networking approach to provide DistanceLess Transmissions (DLT) with rateless erasure codes. DLT encodes data into rateless units and continuously adds redundancy by sending more encoded units. It is able to gradually lower down the effective data rate and thus significantly augment the communication distance beyond the current hardware limit. At the same time, the distanceless transmission is able to best exploit the link capacity and automatically adjust to a suitable effective bit rate for both near and far receivers. In distanceless transmission style, DLT can make efficient use of those conventionally unfavorable long-distance links. Data transmission becomes distance oblivious and can easily fit to potential receivers at different distances. As a result, the network connectivity can be enriched and the network diversity can be fully exploited.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SenSys'14, November 3–6, 2014, Memphis, TN, USA.
Copyright © 2014 ACM 978-1-4503-3143-2 ...$10.00

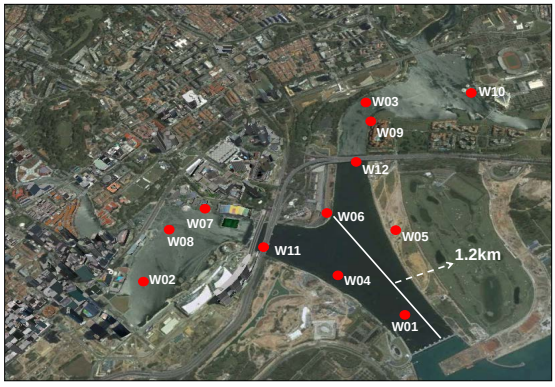
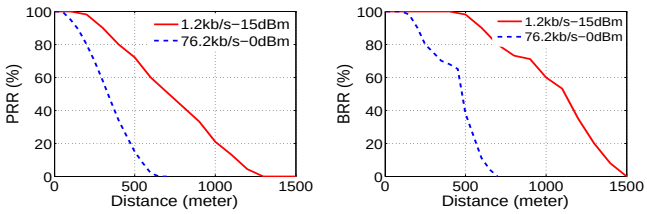


Figure 1: Locations of deployed wind sensors on and around an urban reservoir in Singapore.

Translating the idea into a practical system, however, entails a variety of challenges. Rateless codes are usually designed for high-end devices and may incur infeasible decoding overhead for resource-constrained sensor nodes. For one link transmission, the receiver should decode the packet rapidly and inform the transmitter timely to terminate the continuous transmission. We implement Luby Transform (LT) code [35] on TinyNode by carefully addressing the probelms of encoding efficiency and decoding delay. We also propose a link layer protocol to coordinate the synchronized rateless transmissions. When growing the per-link transmissions to network-wide data forwarding, we devise the Expected Distanceless Transmission Time (EDTT) metric that evaluates the link quality with rateless transmissions and best exploits the network diversity. EDTT can be easily incorporated into the Collection Tree Protocol (CTP) [17] for network data collection. We finally extend DLT to work with low duty-cycled MACs, that has been the *de facto* sensor network setting for energy conservation. Integration with duty-cycled MACs and a sequence of optimization issues were never considered in conventional rateless code design. The final design of DLT is significantly optimized in fully exploiting the network diversity. To the best of our knowledge, DLT is the first distanceless networking design that supports data collection in sparse sensor networks deployed across large areas.

We implement and test DLT in a real-world application, in which 12 wind sensors are deployed to cover a 2.5km*3.0km urban reservoir in Singapore [9]. Extensive experiments are performed and the results show that DLT improves the data delivery reliability over the state-of-the-art data collection protocols (e.g., CTP, ORW [24] and Seda [15]) by up to 26%, shortens the packet latency by 55%, and reduces the energy consumption by 41%.

The rest of this paper is organized as follows. The motivation of DLT is presented in Section 2. The DLT design and implementation are detailed in Section 3. Section 4 introduces the deployment and experiment results. Section 5 reviews related works and Section 6 concludes this paper.



(a) Packet level (b) Byte level

Figure 2: Maximum communication ranges.

2 Motivation

Need for long-distance low-power communications. In many environment monitoring applications, such as forest monitoring [33, 8], soil moisture measurement [44], ground water quality monitoring [26], etc., sensors may be sparsely deployed to cover a wide area. Long-distance communication helps to connect sensors far away from each other and reduce unnecessary deployment of relay nodes. In our recent project for wind measurement, we deploy 12 wind sensors in a 2.5km*3.0km urban water reservoir that measure the wind distribution over and around the water surface [9]. Figure 1 depicts the locations of the deployed wind sensors. The distance between two nodes ranges from 300m to 1.2km. In such a typical sparse sensor network, long-distance communication is desired, or extra sensor nodes have to be deployed to ensure network connectivity.

It is viable to apply technologies, like WiMAX and cellular communication, to achieve long-distance communications. However, the power consumption of WiMAX (about 200 mW) and cellular modules (typical 500 mW transmission power) is too high for typical sensor motes powered by batteries (about 54 mW). In addition, extra data cost may be incurred (e.g., more than 4500$ annual cost for the 12 wind sensors using a cellular data plan). In this paper, we investigate how the long-distance low-power radios could be used to form a multi-hop network to interconnect the sparsely deployed sensors. We does not consider other hardware aided solutions, e.g., using high transmission power, special hardware like high gain or directional antennas. Power consumption is a major consideration. Excessively higher power will be incurred to ensure communication quality over longer distances. In many places, such high transmission power in the ISM band is prohibited, e.g., the maximum transmission power of 868MHz that TinyNode uses is limited to 25mW (14dBm) for outdoor use in Singapore and Europe. On the other hand, those solutions add additional hardware overhead and impair the generality, e.g., most general MAC and routing approaches are based on omnidirectional antennas and cannot be applied on directional antennas.

Communication distance and network connectivity. Some low-power sensor motes have been specifically developed for long-distance communication, e.g., TinyNode [11] and Fleck-3 [8]. TinyNode offers 9 different data rates from 1.2 kb/s to 76.2 kb/s and 4 power levels from 0 dBm to 15 dBm with a step of 5 dBm. The receiving sensitivity could be as high as -121 dBm at the 1.2 kb/s bit rate, which provides

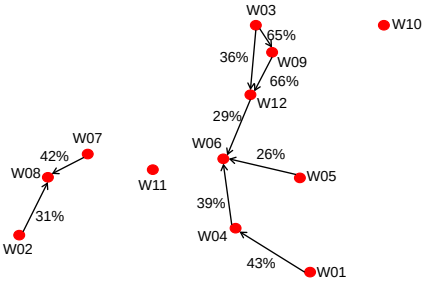


Figure 3: The network topology over packet-level links. The number on each link indicates its PRR.

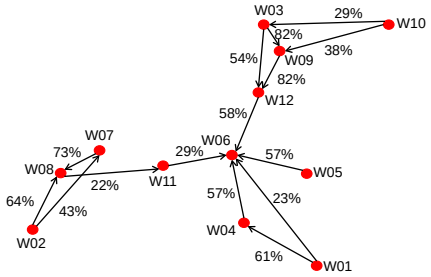


Figure 4: The network topology over byte-level links. The number on each link indicates its BRR.

the longest communication distance, a theoretical range of 1.8km. The communication range, however, may severely degrade in practice due to multi-path effect and interference.

We conduct a series of in-field measurements using TinyNode in three representative environments: an open field, an urban road and a lake. For each experiment, we configure the transmitter to continuously send packets to the receiver. We measure the packet reception of the receiver at different communication distances. The packet size is set to 76 bytes. Although we take TinyNode as a vehicle in the measurements, we believe similar results may also apply to other long-range radios, as they normally achieve long communication distances through high receiver sensitivity enabled by low bit rates.

Figure 2a depicts the measured average Packet Reception Rate (PRR) corresponding to different communication distances at the highest bit rate with the minimum power (76.2kb/s-0dBm) and the lowest bit rate with the maximum power (1.2kb/s-15dBm) respectively. The maximum communication distances of all other configurations are between these two curves. In Figure 2a, we see that the communication range achieved in practice is much smaller than its theoretical value (i.e., 1.8 km). The measurement results reveal that although a wide range of tunable configuration parameters (e.g., bit rates and transmission power) are provided, TinyNode offers inadequate channel adaptation capability in many practical situations.

Figure 3 presents the formed network topology when we directly employ TinyNode to interconnect the 12 deployed wind sensors shown in Figure 1. We measure PRR between each transceiver pair operating with the highest transmission power and the lowest bit rate, which produce the longest communication distance. Figure 3 depicts all links with a PRR higher than 20%, where W06 is the sink. Many links in the network are disconnected and most connected links suffer from high packet loss.

Although PRR decreases rapidly as the communication distance increases, we find that the erroneous bits in majority of corrupted packets are few. This observation inspires us to extend the communication distance by fully leveraging the correct bits contained in each received packet. We thus investigate the Byte Reception Rate (BRR)[1], which measures the correct bytes received by the receiver over the total bytes transmitted by the transmitter. In Figure 2b, we measure the BRR for different communication distances. The results demonstrate that significant increase of communication range can be achieved with a relatively high BRR. When we adopt the BRR metric to revisit the network connectivity, a highly connected network topology with byte-level links can be obtained, as Figure 4 depicts.

Solutions. To mitigate the distance limitation in sparse networks, we leverage rateless codes to extend communication distance. Transmitted as a stream of encoded units, rateless codes can automatically approach the data rate corresponding to the channel quality. We can thus largely release the distance constraints. The network diversity, measured by the number of potential next-hop receivers available for each node, can also be significantly enriched. To fully exploit the network diversity that can be achieved in Figure 4, we propose a DistanceLess Transmission (DLT) approach to best adapt to different communication distances.

In DLT, a transmitter sends unlimited encoded rateless blocks (each of several bytes in our implementation) and different receivers can recover the original data by accumulating sufficient correct blocks according to their own channel condition. DLT breaks the data transmissions into byte-level block transmissions and can adapt the effective data rate to the byte-level link qualities. The data transmission is made distanceless, i.e., in a same data transmission, different effective data rates can be achieved for receivers at different communication distances. The network diversity as shown in Figure 4 can thus be best exploited.

Challenges. To implement DLT in a practical system, the following two major challenges need to be addressed. (1) The current rateless codes need to be tailored to provide link communications on resource-constrained sensor motes. For instance, the decoding process needs to be accelerated to enable timely feedback from receivers to transmitters. (2) An appropriate link quality metric needs to be devised to quantify the distanceless transmission gain on different links and select routing paths for network wide data forwarding. The duty-cycled MAC should also be carefully incorporated for better energy efficiency.

3 DLT design

DLT provides reliable and efficient data collection across sparse wireless sensor networks. At the link layer, DLT

[1]We focus on the correctly received bytes rather than bits as in [30] because bytes can better reflect the information available in partial packets.

leverages rateless codes to improve the transmission quality over links of different communication distances. At the network layer, DLT incorporates its link design into the common routing stack of sensor networks in both full-active and low duty-cycled mode based on a new link metric.

3.1 Rateless codes for sensor motes

Many light-weight rateless codes, e.g., LT code, Random Linear (RL) code and Online code, encode data into rateless units and automatically achieve a proper bit rate for a given link. They are viable for low-profile wireless sensors. With those rateless codes, nodes divide one packet into k blocks, denoted as $\{B_1, B_2, \cdots B_k\}$, which are used to generate encoded rateless blocks, $\{Y_1, Y_2, \cdots\}$. For one rateless block, a certain number of randomly selected original data blocks are linearly combined. Each rateless block is attached with an one-byte Cyclic Redundancy Check (CRC) checksum. Once a node receives m ($m \geq k$) clean rateless blocks that pass the CRC checking, it can use the Gaussian Elimination (GE) algorithm or the Belief Propagation (BP) algorithm to recover the original packet.

Decoding efficiency of a block-based rateless code is calculated as k/m, which measures how many additional blocks ($m-k$) are required to recover the original packet. RL code has the optimal decoding efficiency (100%), whereas its decoding time is extraordinarily long, because it uses modular multiplication with random numbers in a finite field to linearly combine original blocks. LT and Online codes use the light-weight exclusive disjunction (XOR) operations but degrade the decoding efficiency. The performance of online code is highly determined by complex parameter tuning [43]. On the contrary, LT code is robust and well balances between decoding efficiency and computation complexity. It can recover the original packet from $k + O(\sqrt{k}\ln^2(k/\delta))$ encoded blocks with a successful probability of $1-\delta$ and an average computational overhead of $O(k\ln(k/\delta))$ [35]. We thus choose LT code in our design.

Encoded blocks in LT code are generated by the bitwise modulo-2 sum of d original blocks that are randomly and uniformly chosen from the k original blocks, where $d = 1, 2, \ldots, k$. For the encoded block $Y_i, 1 < i < \infty$, the selection of degree d is determined by a probability distribution $\rho(d) = \{p_j, 1 < j < k\}$, where p_j is the probability that $d(= j)$ original blocks are selected to encode Y_i. The decoding efficiency of LT code depends on the degree distribution. The default robust Soliton distribution in LT code is mainly optimized for long packets containing thousands of blocks in cellular or satellite communication. Its decoding efficiency is low for the small packets in wireless sensor networks. For instance, it requires 26.9 encoded blocks to recover a packet of 16 original blocks. We thus implement the degree distribution optimized in SYNAPSE [39], which reduces the requested blocks to 17.9.

3.2 DLT link

We enable the distanceless link transmissions and address the decoding issue to implement LT code on sensor motes.

3.2.1 Distanceless link coordination

With DLT, a transmitter encodes data into rateless blocks and transmits an encoded stream. At a given time point, the nodes with different distances to the transmitter may receive different number of clean blocks. As the transmitter keeps sending the encoded block stream, all receivers will succeed in decoding by accumulating sufficient clean blocks. For a single link transmission, after recovering a packet, the receiver should inform the transmitter to terminate its transmission and release the channel immediately.

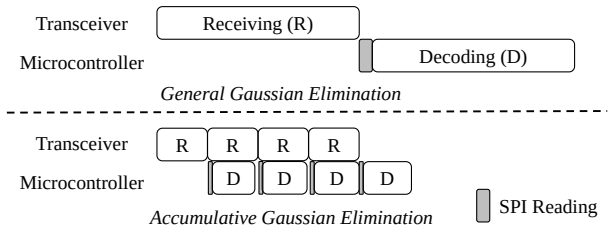


Figure 5: Parallel rateless reception and decoding.

As low-power wireless radio is half-duplex, we let the transmitter send *frames*, where one frame contains multiple blocks. Before transmitting the next frame, the transmitter waits for the feedback (e.g., ACK or NAK) from a receiver in a short time interval (e.g., 0.5ms). Upon receiving one frame, if the decoding succeeds, the receiver replies with an ACK to terminate the transmission; otherwise, it replies with a NAK containing the number of missing blocks and the transmitter sends another frame containing the requested number of rateless blocks.

To enable rateless link transmission, the receiver needs to timely feedback to the transmitter after successful decoding. The BP algorithm is computationally lightweight. It however imposes strict requirements on the degree of received clean blocks, deteriorating the decoding efficiency. We choose the GE algorithm, which can decode the packet successfully as long as k linearly independent blocks are received. The computational complexity of GE is relatively high, i.e., $O(k^3)$ for decoding k original blocks, which may not satisfy the timing requirement of link transmissions. We tackle the high computational complexity issue of GE and propose a fast decoding approach.

3.2.2 Fast decoding

To decode one packet using the GE algorithm, receivers require the encoding coefficient matrix I used by the transmitter for generating the rateless blocks. The matrix is a binary matrix. The width of the matrix is equal to the number of original blocks k and each column of the matrix corresponds to one original block. Each row indicates how a rateless block is encoded. The blocks whose corresponding column is equal to 1 are XORed to calculate the encoded block. In DLT, we let transmitters generate I using a random number generator. Receivers can reproduce an identical matrix using the same seed.

By knowing the coefficient matrix I, the GE algorithm decodes a packet in two steps: triangularization and backward substitution. They aim to obtain a triangular coefficient matrix using linear operations of rows in I. If I has full rank, the data packet can finally be recovered. However, the GE algorithm is time consuming for low-profile sensor

Algorithm 1 Accumulative Gaussian Elimination.

Input: Y_i: the new received block. R: rank of the coefficient matrix I.
Output: Decoding result (*isSolved*) and decoded original blocks.
Insert the coefficient vector of Y_i to the j_{th} of the coefficient matrix I;
$j = R+1$;
for $n = j$; $n <= i$; $n++$ **do** //Try another temporal blocks.
  for $m = 1$; $m <= j$; $m++$ **do** //Triangularization
    if $I_{jm} != 1$ **then**
    I_j=$I_m \bigoplus I_j$; Y_j=$Y_m \bigoplus Y_j$;
    end if
  end for
  if $I_{j,j} == 1$ **then** //Triangularization successes.
    for $m = j$; $m > i$; $m--$ **do** //Backward substitution
      if $I_{j,m} == 1$ && $m != j$ **then**
        I_m=$I_m \bigoplus I_j$; Y_m=$Y_m \bigoplus Y_j$;
      end if
      $R++$;
    end for
    return checkCoefficientMatrixFullRank(I);
  else//Triangularization fails. Use the previously received blocks.
    I_j=$I_j \bigoplus I_{n+1}$;Y_j=$Y_j \bigoplus Y_{n+1}$;
  end if
end for
return False;

devices, e.g., it takes 1.2ms for TI MSP430 microcontroller to decode a 64-byte packet composed of 8 blocks, which is much longer than the waiting interval of 0.5ms.

We accelerate LT decoding based on two key observations. First, the decoding time of a frame is much less than the receiving time on sensor motes, e.g., it takes 8ms to receive a 76-byte frame with the bit rate of 76.2kb/s on Semtech XE1205 radio and 1.2ms to decode the same frame on TI MSP430 microcontroller. Second, before starting the decoding process using the GE algorithm, the microcontroller has to wait until the whole frame is received. Thus, we shorten the frame processing delay by paralleling the GE decoding with the frame receiving. As illustrated in Figure 5, nodes start updating the coefficient matrix as long as the first two encoded blocks are received and perform the GE decoding during the reception of next block.

3.2.3 Accumulative Gaussian elimination

We develop an Accumulative GE (AGE) algorithm to parallelize the GE decoding with the frame receiving. Unlike the existing incremental GE algorithms [3], which only performs the triangularization incrementally, AGE strives to finish both triangularization and backward substitution before new blocks arrive. Algorithm 1 describes how a new received block is added in AGE decoding accumulatively. The key idea is to transform the top left submatrix in the coefficient matrix into an identity submatrix step by step using the GE algorithm as new blocks are accumulated gradually. If a new block cannot be used immediately to extend the submatrix, it will be stored temporarily and utilized later when more blocks are received.

An example of AGE decoding is illustrated in Figure 6, in which a packet can be decoded from 4 encoded blocks. The receiver starts decoding when two blocks are received (step a). It tries to convert the submatrix highlighted by the dashed square into an identity matrix by switching the first two rows (step b: triangularization) and replacing the first

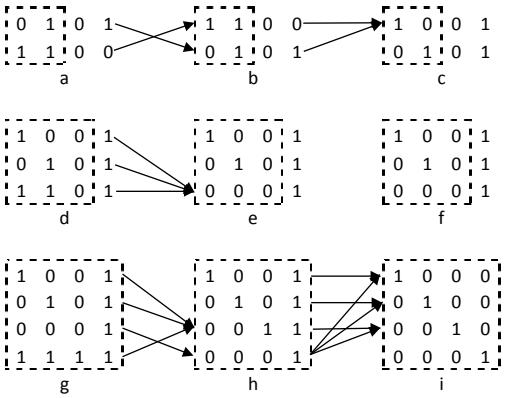


Figure 6: An example of AGE decoding.

row with the XOR of the first two rows (step c: backward substitution). When the third block is delivered from the radio to the microcontroller, the receiver inserts it to the third row (step d) and performs triangularization (step e). However, this step fails and thus the receiver stores the second block as a temporal block for future use without performing backward substitution (step f). When the last block is received, the original data can be decoded by eliminating all "1" values in the last row by triangularization (step h) and in the last column by backward substitution (step i).

With the AGE algorithm, both triangularization and backward substitution are nearly completed prior to the reception of the last block. The receiver only processes the coefficient matrix for the last block to recover the original packet. The decoding latency is thus significantly reduced from 1.2ms to 0.4ms and the receiver can promptly send a feedback to the transmitter within an ACK waiting interval.

3.3 DLT networking

Traditional link quality metrics for packet routing, e.g., ETX, are not suitable for distanceless transmissions, because they evaluate links based on the packet reception statistics. DLT transmits fine-grained rateless blocks. The number of blocks contained by each frame is dynamically adjusted and the frame length is variable for different transmissions. We therefore propose a tailored metric to evaluate the per-link transmission quality, which can be seamlessly integrated into CTP for a network-wide distanceless data collection. We further propose a routing protocol to optimize the performance in low duty-cycled sparse sensor networks with limited network diversity.

3.3.1 Expected distanceless transmission time

BLock Reception Rate (BLRR). BLRR directly describes the channel loss in block-level transmissions. It is the ratio between the clean blocks received by the receiver and the total blocks sent by the transmitter. The BLRR for a given block size (e.g., L_b), denoted as BLRR_b, can be measured directly based on data transmissions. The receiver inserts a payload of one byte in its feedback message. For an ACK, the one-byte payload presents the number of received clean blocks; otherwise, it refers to the number of missing blocks. Based on the information, the transmitter can cal-

culate its *BLRR* after each transmission. To minimize the measurement jitter, we apply a weighted moving average to obtain a relatively stable BLRR.

$$BLRR_b = \alpha * BLRR_b^{new} + (1-\alpha) * BLRR_b^{old} \quad (1)$$

where α is a weighting factor and the setting of α is experimentally determined according the variation of wireless channels. In our deployment, a weighting factor of 0.92 provides the best performance, which reveals that the channel in our deployment field is highly dynamic.

BLRR cannot differentiate two links if their block sizes are not the same. For instance, on a link of high byte error rate, if a large block size is used, the BLRR is low; otherwise, a small block size results in a higher BLRR. Simple comparison between two BLRRs of different block sizes cannot represent the actual channel condition. To best accommodate to different channel conditions, however, we must adjust block size dynamically (Section 3.4 describes the block size adaptation algorithm) used in DLT. Therefore, we propose Expected Distanceless Transmission Time (EDTT) to evaluate the distanceless link transmissions.

Expected distanceless transmission time. EDTT averages the BLRRs of different block sizes. We denote the length of an original data packet as L_{data}. With rateless transmissions, the data packet is divided into L_{data}/L_b blocks to generate unlimited rateless blocks. To decode the packet, receivers need $(L_{data}/L_b) * (m_b/k_b)/BLRR_b$ rateless blocks, where k_b/m_b is the decoding efficiency of LT code for k_b original blocks. For each rateless block, we add one-byte CRC and thus $(L_b+1)*8$ bits need to be transmitted. The time needed to complete the transmission of all those blocks, called Distanceless Transmission Time (DTT), can be calculated as:

$$DTT_b = \frac{L_{data} * m_b * (L_b+1) * 8}{L_b * k_b * BLRR_b * R}. \quad (2)$$

where R is the transmission bit rate. If we denote the set of all possible block sizes as $\mathcal{L}$, EDTT can be calculated as:

$$EDTT = \sum_{b \in \mathcal{L}} P_b * DTT_b, \quad (3)$$

where P_b is the probability to use b_{th} block size and DTT_b is its transmission time. We set P_b as the usage frequency of each block size in last M transmissions. In our deployment, M is set to 100. If one block size is not used in the past M transmissions, its usage frequency is equal to 0. Based on Equation (2) and (3), we can calculate the EDTT for each link by measuring its BLRR.

Integrating DLT with CTP. With EDTT, we integrate DLT with the *de facto* routing protocol in wireless sensor networks, Collection Tree Protocol (CTP), with the minimal modification to the existing protocol stack. We replace ETX in the CTP implementation in TinyOS by EDTT. Each node selects the path with the minimum cumulative EDTT to the sink to transmit packets. The per-link EDTT value is included in each transmitted frame. If a node receives a packet yielding a lower cumulative EDTT value to the sink, it updates its routing table. EDTT of an individual link is updated by data transmissions and proactive probes. Beacon packets are transmitted periodically with a pre-defined payload. The beacon transmission period is adjusted according to the Trickle algorithm [27]. Upon receiving a beacon, the erroneous bits are known and we thus can calculate the BLRR for each block size. By doing so, we obtain the EDTT for all block sizes using one beacon.

3.3.2 Low duty-cycled networks

In wireless sensor networks, nodes are usually duty-cycled to prolong the network lifetime. To provide a general and comprehensive design for data collection in environmental monitoring applications, we extend the DLT design to low duty-cycled mode. Low-power listening (LPL) has been widely adopted to schedule two asynchronous transceivers in low duty-cycled sensor networks. With the default implementation of LPL in TinyOS, BoX-MAC [36], the transmitter sends a long preamble of data packets. When a node wakes up, it first checks the channel for a short duration. It attempts to receive the packet if the channel is sensed to be busy; otherwise, it goes back to sleep again. We introduce the DLT design based on LPL for duty-cycled sensor networks. As a matter of fact, other types of duty cycling schemes, e.g., receiver-initiated A-MAC [13], can also be similarly integrated into DLT.

LPL has been integrated into many existing routing protocols, like CTP and ORW. In CTP with LPL enabled, nodes transmit a long preamble until their target receiver wakes up. ORW reduces the latency and energy consumption by enabling opportunistic routing of the first waken forwarder. Nodes check whether they can make progress for a packet delivery by considering both their cumulative ETX distance to the sink and the number of their potential forwarders. More potential forwarders imply that the per-hop transmission latency will be low.

The existing protocols, however, mainly focus on dense sensor networks with rich network diversity. By taking into account the unique features of sparse sensor networks (e.g., extremely lossy links and low network diversity), we devise several optimization schemes to better incorporate the distanceless transmissions in low duty-cycled mode. Due to the low network diversity in sparse sensor networks, we need to make full usage of each potential transmission opportunity. Nodes with DLT maintain an EDTT parameter for each potential receiver and choose the minimum cumulative EDTT as their EDTT. When a node wakes up and hears a preamble, it decodes the header of a frame and verifies whether it should forward the packet. For verification, the node compares its EDTT with the transmitter's, which is contained in the frame header of each transmission. If its EDTT is smaller than the transmitter's, it becomes a forwarder for that transmitter.

In DLT, instead of repeating the same data packet in the preamble, nodes transmit a stream of rateless frames as preamble. Each frame contains different rateless blocks such that diverse frames are continuously pumped out. Potential forwarders can recover the data packet by receiving sufficient rateless blocks. For multiple receivers, the optimal frame length is different. We configure the length of preamble frames according to the channel condition to the nearest forwarder since it normally requires the least amount of rateless

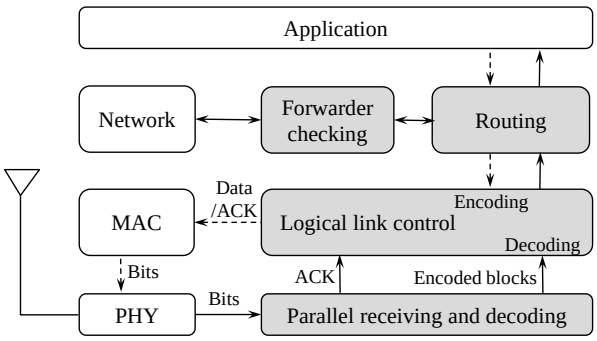


Figure 7: Architecture of DLT.

blocks. When a receiver far away from the transmitter wakes up first and verifies that it is eligible to relay this packet, it sends a NAK with the number of missing blocks. Upon receiving the NAK, the transmitter adjusts the frame length and transmits proper number of rateless blocks to adapt to the wireless channel condition of the respective forwarder. The number of blocks contained in next frame is adjusted based on both the number of clean blocks already received by the forwarder (N_{rec}) and the channel quality, as expressed in the following:

$$N_b = \frac{(N_{data} - N_{rec}) * m_b}{BLRR_b * k_b} \quad (4)$$

In sparse sensor networks, each node only possesses a few of potential forwarders. It is rare that multiple forwarders simultaneously succeed in decoding and their feedbacks collide. To handle this problem, the transmitter transmits frames with a default frame length after the ACK waiting timer expires. When a forwarder receives a frame for the decoded data packet, it transmits an ACK with $1/2$ probability to migrate potential collisions.

3.4 Implementation details

We implement DLT in TinyOS. We introduce the architecture of DLT and the techniques which enable the DLT implementation.

DLT architecture. Figure 7 depicts the architecture of DLT. We implement four major modules compatible to the existing 802.15.4 networking stack with the minimal modifications to current protocol components in TinyOS.

To transmit a data packet, the *routing* module adds a header before the application payload, including EDTT, source address and sequence number. The processed data packet is then delivered to the *logical link control* module to generate rateless blocks and assemble frames. The optimization of transmission parameters, e.g., block size and frame length, are also performed in the logical link control module. Frames are finally passed to the existing MAC layer for transmissions using LPL and CSMA/CA.

For receiving, the PHY layer loads the received bytes in a buffer after detecting a preamble. The *fast decoding* module retrieves blocks from the buffer and passes them to the logical link control to start decoding. Nodes maintain a forwarding cost for each neighbor in the routing module.

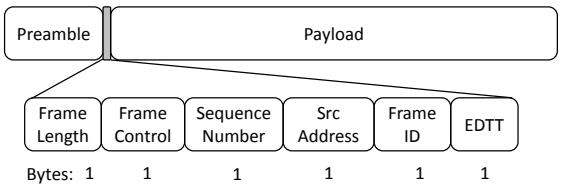


Figure 8: Frame format in DLT.

When a decoded packet is passed to the routing module, the node either relays the packet to the CTP parent or the first waken neighbor with a smaller forwarding cost. The link quality metric is updated periodically with the default mechanism in the network layer.

Frame format. The frame format in DLT is depicted in Figure 8. "Frame Length", i.e., the number of bytes contained in the frame. "Frame Control" contains control information, e.g., two bits in this field indicate the frame type; one bit describes whether an ACK is required; and the rest are reserved for future extension. "Sequence Number" is the original data packet index and "Src Address" is the ID of the node which generates this data packet. Different frames encoded from the same data packet are identified by their "Frame ID". The ID is set to 1 for the first frame and is increased gradually for the following frames. The EDTT of the transmitter is used to verify forwarding before the decoding of MAC payload. If the node is not a forwarder to the current transmission, it will discard the received frame without decoding.

Block size adaptation. Given the channel condition, different block sizes L_b may result in different BLRRs. A small block size can preserve correct bytes with higher granularity, whereas it requires more CRC overhead. In DLT, block size in each frame is adapted dynamically according to the current channel condition. We propose a simple heuristic algorithm to dynamically adapt the block size. We adapt the block size according to the variation of BLRR. When the BLRR reaches an upper bound τ_h, it indicates that the number of error bytes in the received frame is low and we can increase the block size to reduce the CRC overhead. When the BLRR decreases to a lower bound τ_l, there are too many erroneous blocks and the block size needs to be reduced. In our implementation, τ_h and τ_l are set to 91% and 72% respectively, and three levels of block size, i.e., 4, 8 and 16 bytes, are used. The block size of a frame is indicated by 2 bits in its "Frame Control" field of the MAC header.

Predictable encoding coefficient matrix. To reduce the transmission overhead, we do not transmit the random number generator seed used for encoding and decoding along with the data packet. Instead, a fix seed is used. To generate the coefficient matrix I for decoding, the corresponding row of one received block can be identified by "Frame ID" and the offset of the block in that frame. If the CRC checking of a block fails, it will be discarded and its corresponding row will be deleted from I. If the transmitter does not receive a feedback from any receivers in an ACK waiting duration, it transmits another frame with the same number of rateless blocks in the previous frame. The new frame contains the

Table 1: The performance of implemented LT code on TinyNode using the AGE decoding algorithm.

Number of blocks		4	8	16
Decoding time (ms)	GE	0.9	2.4	10.1
	IGE	0.5	0.8	3.4
	AGE	0.4	0.4	1.4
Overhead (blocks)	Robust Soliton	2.99	6.03	10.87
	SYNAPSE+GE	1.81	1.96	1.83
	SYNAPSE+AGE	1.7	1.91	1.9
RAM (kB)		4.4	4.5	5.0

next piece of encoded blocks and an increased frame ID. It can thus bring novel information to the receiver in case that the feedback of the previous frame was lost. When a node receives a frame ID increased by j, where $j \geq 2$, it identifies the row of the i-th block in this frame as $N_{blk} + b \times j + i$, where N_{blk} is the number of blocks in the last correctly received frame and b is the number of blocks contained in the current frame.

LT code on sensor motes. Table 1 tabulates the decoding efficiency, decoding time, and memory occupancy measured on TinyNode (TI MSP430 microcontroller) for different k values. The results reveal that the proposed AGE algorithm can significantly reduce the decoding time without impacting the decoding efficiency. The decoding time is measured from the last block received until the decoding completed under parallel decoding and receiving.

To fully pipeline the frame receiving and decoding, the total decoding time should be less than the receiving time. However, the decoding time is 30.5ms, which is much larger than the time (i.e., 8 ms) to transmit a packet of 76 bytes at 76.2kb/s bit rate on TinyNode. To accelerate the decoding, we find that the random number generation used to reproduce the encoding coefficient matrix is time consuming in TinyOS. We trade RAM memory for encoding and decoding speed. Instead of generating the coefficient matrix every time a frame is received, we fix the random number generator seed and store the coefficient matrix in RAM. For a 64-byte frame of 8 original blocks, we save a matrix of 160 rows for 160 encoded blocks (20 times larger than the number of original blocks). It is sufficient for some extremely lossy links with a block error rate around 94% (150/160), but only occupies 160 bytes of RAM. By doing so, we can reduce the decoding time from 31 ms to 2.4 ms.

With our proposed AGE algorithm, the decoding time can be finally reduced to 0.4ms. From Table 1, we also see that the decoding overhead of our AGE algorithm is the same with the traditional GE algorithms. Furthermore, the RAM cost shown in Table 1 is the footprint of our implementation of DLT including specifications of all protocol layers but not just AGE decoding. The memory cost is well controlled and can be supported by current sensor motes, e.g., TinyNode and TelosB, which are composed of TI MSP430 microcontroller possessing a RAM memory of 10KB.

4 Evaluation

We evaluate DLT and compare it with other data collection protocols on our wind measurement sensor network.

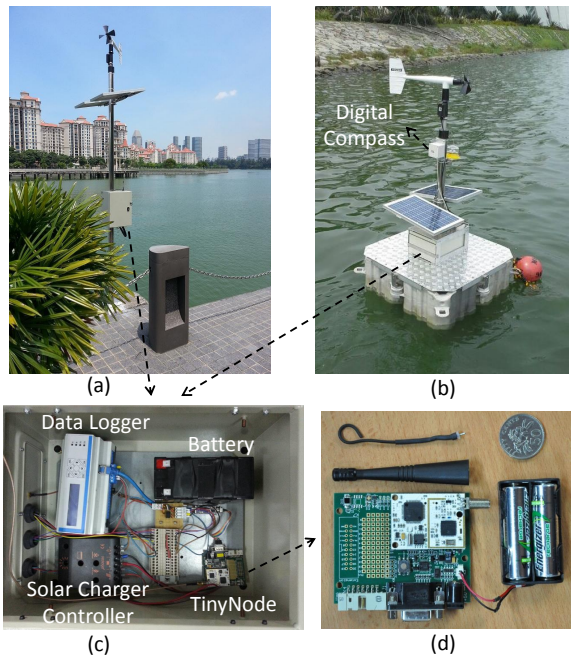


Figure 9: Wind measurement sensors installed on land (a) and floating on the water surface (b); Electronic devices in a weatherproof box (c); TinyNode with an omnidirectional antenna extended outside the box (d).

4.1 Deployment and experimental setting

In our application, 12 wind sensors are installed in Marina Reservoir of Singapore (a typical urban water field of $2.5km * 3.0km$), as depicted in Figure 1, to measure the wind distribution on the water surface. The sensor locations have already been optimized by a sensor placement approach [9]. The average line of sight distance between two sensors in the network is 720m. The maximum distance is 1000m and the minimum distance is 300m.

Figure 9 presents the wind measurement sensors, including sensors installed on the land and floating on the water surface. The wind monitor model 05305L of R.M. YOUNG is used to measure the wind direction and speed. The OS5000 3-axis digital compass from OceanServer provides the direction offset of the floating platform. With an early version of the data collection system, each sensor is equipped with a data logger to record the sensor data and send the data back to our server via cellular network. At present, TinyNode retrieves the sensor readings from the anemometer via its analog-to-digital converter and a multi-hop network is built using 12 TinyNode sensor motes to collect sensor data. The data logger is used to record the system debugging information, including data generation, packet transmission and receiving. Solar panels are used to harvest energy, which is stored in a rechargeable battery and further used to power all electronic devices. The energy harvested by the solar panel provides a power budget of $\sim$55.2 Wh/day, where the wind sensor and data logger consume $\sim$51.9 Wh/day, leaving $\sim$3Wh/day to the communication module. We employ duty-cycled DLT with such limited power budget.

Besides energy efficiency, the data collection is required to be reliable and fast. The sensor readings are processed to generate the distribution of wind stress on the water surface. Data loss from any sensor nodes will impair the accuracy of the derived wind distribution. Furthermore, the wind distribution is used as input to study the water hydrodynamics and water quality in the entire reservoir with a 3-D limnological model [9]. If problems arise, special physical or chemical treatments will be taken, e.g., draining the water through a barrage, starting the bubble-plume system to improve water mixing, or adding algaecide to control algal blooms. As the calculation of ecological model is time consuming (about 2-3 min), to enable timely treatment, the data collection system is required to provide real-time data monitoring, at least faster than the ecological calculation.

In our experiment, one wind data sample is described by 4 bytes (2 bytes for wind direction and 2 bytes for wind speed). Each sample is associated with a time stamp of 4 bytes. Wind sensors make a measurement every 10s and send 6 samples together to the sink every minute. With a 8-byte network layer header (same as CTP) and a 8-byte field describing the status of physical devices, a data link layer payload is 64 bytes. The total packet length is 76 bytes, including 6-byte PHY layer header and 6-byte frame header. In our implementation, the data link layer payload of 64 bytes is encoded into rateless blocks by LT code. The block size could be 4, 8 and 16 bytes, corresponding to 16, 8 and 4 original blocks in one data packet.

4.2 Methodology

We compare the performance of DLT with the following benchmark protocols.

CTP [17] is the *de facto* routing protocol for sensor networks. We run CTP with BoX-MAC [36], the default low-power listening MAC protocol in TinyOS. To transmit a data packet, the transmitter sends a long preamble until the target receiver wakes up. The preamble is a series of data packets separated by an ACK waiting interval.

ORW [24] is the most recent routing protocol designed for low duty-cycled sensor networks following the opportunistic principle. It uses Expected Duty Cycled wakeups (EDC) to control the size of forwarder set and adds a weight in EDC to reflect forwarding cost. The weight parameter is set to 0.1, the best setting reported in [24].

Seda [15] is a block-level link transmission method. It divides one packet into blocks, each of which is associated with a CRC and a sequence number. When a node receives a corrupted packet, it replies the transmitter with the sequence number of erroneous blocks and the transmitter retransmits those blocks. As Seda outperforms other Forward Error Correction (FEC) and Automatic Repeat-reQuest (ARQ) methods in sensor networks [15], we do not compare DLT with them individually. In this experiment, we integrate Seda with ORW (as ORW generally outperforms CTP) for the performance comparison.

Metrics. The main task of DLT is to collect data in sparse sensor networks reliably and efficiently. We concern the following 3 metrics for the performance evaluation.

Data yield is the ratio between the amount of data packets received at the sink and the total amount of data packets generated by all sensors in the network. In the experiments, packets may be lost when 1) buffer overflows due to network congestion, or 2) continuous failures after the maximum number of channel access (*macMaxCSMABackoffs*) or transmission attempts (*macMaxFrameRetries*). As the default setting in IEEE 802.15.4 standard, *macMaxCSMABackoffs* and *macMaxFrameRetries* are set to 4 and 3 respectively.

End-to-end latency is measured from the time when the original source node generates a data packet to the time when the packet is received by the sink. In our wind measurement sensor network, sensor nodes maintain a 2-packet buffer for each neighbor. A node must drop the oldest packets from one neighbor if more than 2 packets are in the respective buffer.

Energy efficiency is measured by duty cycle, i.e., the portion of time when the radio is on. Duty cycle is a good proxy of energy consumption for wireless sensors, since the two main energy-consuming components on sensor motes (i.e., microcontroller and radio) have similar work schedule and the radio consumes similar levels of energy while transmitting and receiving.

Table 2: Performance (PRR, BRR and BER) of two links with different communication distances and data rates.

Rate (kb/s)	W01-W04 (550m)			W01-W06 (1000m)		
	PRR	BRR	BER	PRR	BRR	BER
76.2	0.25	0.46	0.07	0.04	0.15	0.11
1.2	0.43	0.61	0.05	0.09	0.23	0.07

4.3 Results

We show the experiment results at both link level and network level. We first focus on the periodical data collection and consider another flexible traffic pattern later.

4.3.1 Single link

Using the in-field measurements on several single links, we study the performance of different link-transmission approaches and the proposed block size adaptation algorithm.

In-field measurements. Table 2 presents three link-level performance metrics (PRR, BRR and BER) measured at W04 and W06 when W01 is transmitting at different data rates. The pairs of W01-W04 and W01-W06 represent links with short (around 550m) and long (around 1000m) communication ranges, respectively. PRR and BRR are calculated based on all transmitted packets. BER is the byte error rate of all received packets, not including the lost packets.

In Table 2, the BRRs of all links are much higher than the relative PRRs and the BER in the corrupted packets is low. The results confirm to our observation in Section 2 that the bandwidth utilization in sparse sensor networks can be significantly improved and the long communication distance can also be achieved if we can efficiently enable byte-level transmissions. Although both PRR and BRR increase for the long-distance link to W06 when the bit rate is reduced, the highest bit rate (76.2kb/s) still offers the largest throughput (PRR*Rate), which is probably due to the combined effect of interference and signal attenuation. We set the bit rate of all approaches to 76.2kb/s during the experiments.

Block size adaptation. Table 3 presents the goodput achieved by different block sizes for the packet traces col-

Table 3: Goodput (kb/s) achieved by different block sizes.

Links \ Block size (byte)	4	8	16	Optimal	Adapt
W01->W04	52.5	56.7	57.9	59.4	58.8
W01->W06	41.8	40.8	36.5	44.9	43.4

lected on two links. The goodput of a received frame with block size L_b can be calculated as:

$$S_i = \frac{N_{cleanblk} * L_b}{N_{totalblk} * (L_b + 1)} * R \quad (5)$$

where R is the bit rate. $N_{cleanblk}$ and $N_{totalblk}$ represent the number of correctly received blocks and the total number of blocks in one packet, respectively. We calculate the goodput achieved by several fixed block sizes for each packet in the traces. The goodput of optimal adaptation is the average goodput calculated by the best block size of each packet. From Table 3, we see that one fixed block size is not sufficient for all links. For the short link from W01 to W04, a large block size is preferred; the other link of long communication distance, however, works best with a small block size due to more erroneous bytes in the traces. In addition, even for one single link, the block size should be adapted according to the channel dynamics. The goodput achieved by the proposed adaptation algorithm in Table 3 demonstrates that our heuristic algorithm captures the channel variation and approaches the optimal solution. We will show next that the 1-byte CRC overhead of each block is much smaller than the substantial gain derived from rateless transmissions and block size adaptation.

Goodput on single link. Figure 10 depicts the CDF of goodput achieved by ARQ, Seda and DLT on the long-distance link from W01 to W06. We measure the goodput that all approaches can achieve to transmit 100 packet traces, considering CRC overhead, packet retransmission, CSMA-based multiple access overhead and ACK loss. Seda and DLT-8 use a fixed block size of 8 bytes, and DLT enables the proposed block size adaptation algorithm. The results in Figure 10 show that Seda improves the average goodput of ARQ by 1.4X via block-level retransmissions and DLT achieves a goodput improvement of 2.1X over ARQ through distanceless transmissions. If the proposed block size adaptation algorithm is enabled, the goodput gain could be further increased to 2.3X. Although Seda provides block level transmissions, DLT possesses two unique advantages. First, DLT proactively adapts to the wireless channels by transmitting proper number of encoded blocks before each transmission; however, Seda can only recover the corrupted packet by passively retransmitting the erroneous blocks. Second, the performance of Seda highly relies on the correct reception of feedback packets. In case of ACK loss, Seda has to retransmit the data packet, whereas DLT only needs to transmit more rateless blocks. New blocks can be combined with the previous blocks to recover the original data. In our deployment, 10% ACK loss is observed. The link asymmetry confirms to the experiments on IEEE 802.15.4 links [41].

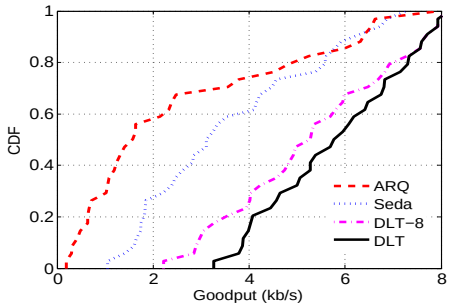


Figure 10: Link-level goodput of different approaches.

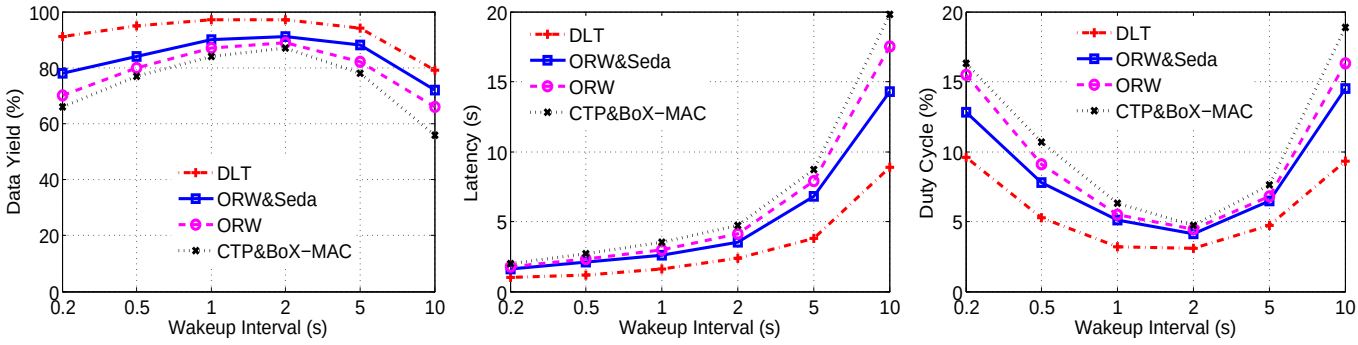
4.3.2 *Network Performance*

In this section, we run the benchmark approaches one by one on the deployed sensor network. Each experiment lasts for 2 hours. In our application, each node sends its data to the sink (W06) every minute. The sink is always in active mode and is connected to internet directly. All the results presented in Figure 11 are based on the packet generation rate of 1 min. We evaluate the performance of all approaches with different wakeup intervals. In low duty-cycled sensor networks, wakeup interval is a crucial parameter to achieve the best network performance given a fixed traffic load.

The results reveal that DLT outperforms the other approaches for all wakeup intervals and it can provide high performance for a large range of wakeup intervals. On average, DLT achieves substantial performance improvement over CTP, ORW, and ORW-Seda. In particular, DLT increases the data yield of CTP, ORW and ORW-Seda by 26%, 15% and 10%, respectively. It reduces the packet latency of CTP, ORW and ORW-Seda by 55%, 49% and 44%. DLT also improves the energy efficiency of CTP, ORW and ORW-Seda by 41%, 31% and 27%.

Data yield. Figure 11a shows the data yield of different protocols under various wakeup intervals. Compared with benchmark protocols, DLT produces less traffic loads in the network, since it shortens the preamble transmission by utilizing the opportunistic forwarding from distant neighbors and accelerates the data transmissions by better adapting to the dynamic wireless channels. It encounters less collision and congestion, and thus provides higher data yield.

When the wakeup interval is small, it is highly possible that multiple nodes are awake at the same time. Data yields are low due to the high probability of collisions. Many packets are dropped after the maximum number of transmission attempts. Especially for sparse sensor networks, traditional communication schemes have to transmit a packet many times when the channel is lossy. The transmission of one data packet may be longer than one wakeup interval. As a result, it will likely collide with the transmissions from other neighbors in the next wakeup interval. DLT mitigates such problems since it shortens the link transmissions and reduces the probability of lengthy packet transmissions. As the wakeup interval increases, the data yield of DLT becomes stable. Compared with the other approaches, DLT provides high performance for a wider range of wakeup intervals. For



(a) Data Yield (b) End to End Latency (c) Energy Consumption

Figure 11: Overall performance of different wakeup intervals.

large wakeup intervals, data yields decrease due to traffic congestion. Long preambles occupy the channel for a long duration, which reduces the transmission chance of other nodes. Moreover, they are susceptible to collisions.

Data latency. Figure 11b presents the average end-to-end latency of data packets. The latency augments as the wakeup interval increases, as long preamble needs to be transmit before the forwarder wakes up. However, compared with the benchmark protocols, DLT has a much slower increasing speed, since it achieves short data frame transmissions and benefits more from the distant forwarders enabled by distanceless transmissions.

When the wakeup interval is large, the latency of DLT is even less than one wakeup interval. In the multi-hop network, latency is reduced by opportunistic forwarding. The nodes close to the sink forward the packets from other nodes if they wake up earlier than the default forwarder, e.g., the node selected by CTP. Moreover, even without opportunistic forwarding, it is possible that the forwarders along a packet delivery path wake up sequentially. Since the distanceless link transmission of DLT is short and optimized, the packet has a high probability to be relayed sequentially without missing the wakeup of any forwarders. The latency difference between CTP and ORW is small, because opportunistic forwarding is rare if long-distance links are not utilized.

Energy consumption. Figure 11c depicts the duty cycles achieved by different protocols. Lower duty cycle indicates higher energy efficiency. DLT can achieve the best energy efficiency for most wakeup intervals. When a small wakeup interval is used, the energy consumption is high due to more collisions and more CSMA-based multiple access overhead. DLT transmits the packet with much less attempts attributed to its optimal utilization of channel bandwidth. For a large wakeup interval, more energy is consumed by the transmission of long preamble packets. Since DLT leverages better the distant forwarders which may wake up earlier than the default forwarder, it enables shorter preamble transmissions and thus smaller probability of collisions.

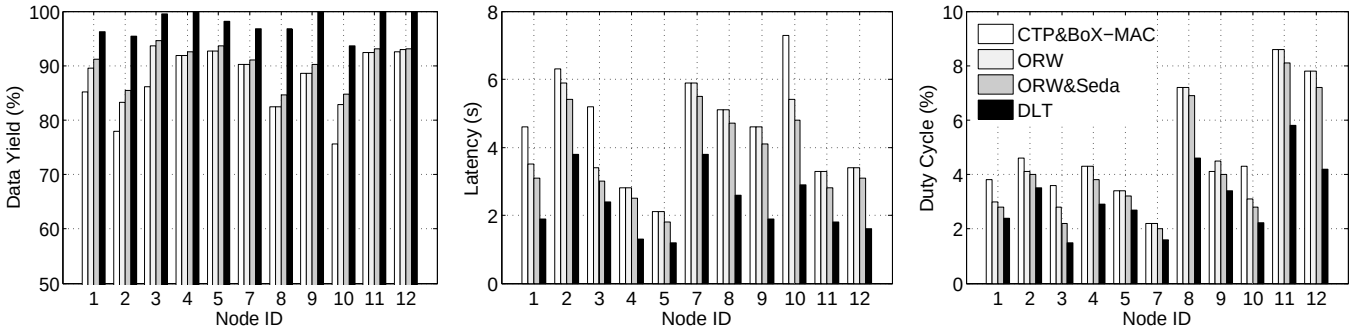
4.3.3 Performance per Node

The experiments in this section are conducted with a wakeup interval of 2s which enables the best performance of CTP and ORW. Figure 12 demonstrates the performance of each node in the network except the sink, which has direct internet access. From the results, we see that DLT can improve the reliability and efficiency of all the nodes regardless of their location in the network. Compared with the other approaches, the gain of DLT mainly comes from two parts: better utilizing wireless channel bandwidth and fully exploiting the enriched network diversity enabled by distanceless transmissions.

Data yield. The data yield of a node is the ratio between the amount of data packets received by the sink from that node and the total amount of data packets generated by that node. Relaying packets are not considered in the per-node data yield. As shown in Figure 12a, the data yield of CTP for some distant nodes, e.g., W02 and W10, is quite low, because they only possess one forwarder and their data packets have to pass through a long path composed of lossy links. ORW and ORW-Seda improve the data yield by employing multiple forwarders and DLT can achieve further improvement by proactive adaptation to the wireless channels of all potential forwarders including the distant ones.

Data latency. Figure 12b examines the average latency of packets transmitted from different nodes. Similar to data yield, packet latency of the nodes far away from the sink is large since the packets need to pass through a long path to reach the sink. DLT can accelerate this process by best leveraging distant receivers over extremely lossy links. For the one-hop neighbors of the sink (i.e., the nodes possessing a direct connection with W06 in Figure 3), DLT reduces their packet delivery latency by the efficient distanceless transmissions. The latency of W11 and W12 is slightly higher than that of W05, as their packets may be delayed when they are relaying the traffic from other nodes.

Energy consumption. Figure 12c shows the duty cycle of each node with different protocols. The energy consumption of some relaying nodes, like W08, W11 and W12, is high since they need to transmit both their own packets and the relayed packets for other nodes. DLT can improve the energy efficiency of these nodes by its elaborate link layer design to achieve reliable transmission of long communication distance. For instance, when W08 is transmitting to W11, if W06 is receiving data from W12 at the same time, the data transmission between W08 and W11 will be im-



(a) Data Yield (b) End to End Latency (c) Energy Consumption

Figure 12: Overall performance per node.

paired by the ACK packet from W06 to W12. Attributable to its block-level distanceless transmissions, DLT can tolerate such interference by further transmitting a small number of encoded blocks.

4.3.4 Overhead

Figure 13 presents the overhead of DLT introduced to each node. We separate the decoding overhead and communication overhead from the data transmissions. The communication overhead includes the time spent on ACK transmissions and CSMA channel access. The results show that DLT spends most of its active time for data transmissions. The decoding overhead is negligible compared with the data transmission or communication overhead. The decoding time of DLT is about 0.4ms (for 8 original blocks) which is much smaller than the duration of data transmission (8 ms for a data packet of 8 original blocks). The small MAC header in DLT imposes negligible overhead. However, in sparse sensor networks, due to the impact of surrounding buildings, the hidden terminal problem is more severe than that of dense sensor networks, which increases the communication overhead. DLT minimizes the communication overhead by increasing the probability of successful transmission using distanceless transmissions.

4.3.5 Robustness

We examine the robustness of each approach by inserting outages in the network. Every 30 min in a 120-min experiment, we disable a randomly chosen node for 10 min. To compare the performance of all approaches, the sequence of the selected nodes is the same for the experiments of all approaches. Figure 14 demonstrates the capacity of each approach adapting to the outages. The results of each time point in Figure 14 is the smoothed data with a 15-min moving average window. During the first outage from 30min to 40min, W11 is disabled. The data yield of all approaches decreases, since W11 connects the subnetwork, consisting of W02, W07 and W08, to the sink W06. The energy consumption of all approaches is increased because W08 spends much energy to send data to W11. In the last two outages, W09 and W04 are disabled respectively. We can see from Figure 14 that the data yield of DLT is reduced slightly whereas the performance of the other approaches degrade sharply. In those two cases, DLT can fully leverage the long-

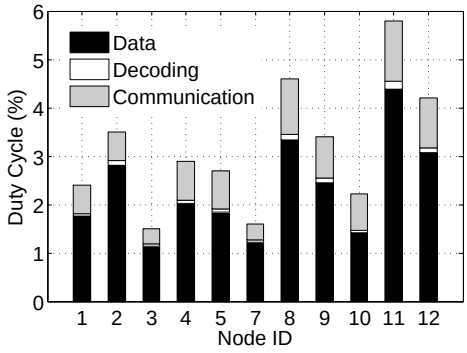


Figure 13: Overhead breakdown of DLT.

distance links to bypass the disabled nodes; however, the other approaches react slowly and cannot fully utilize the wireless channels based on packet-level retransmissions or simple block-level retransmissions.

4.3.6 Traffic patterns

In the above experiments, all nodes in the network send their data to the sink periodically. Besides such a pre-fixed traffic pattern, we also conducted some experiments to evaluate the performance of DLT for event-driven monitoring. Sensor nodes only send their data back to the sink when an interesting event occurs. We assume that the event effect is limited (e.g., a sudden change of wind direction) and can only be detected by one or two sensors. To evaluate the performance of DLT in such a flexible traffic pattern, each node in the deployed sensor network generates a packet randomly and independently in a given period. The wakeup interval of each node is set to 2s.

Table 4 presents the performance of DLT for different Event Generation Rates (EGR), which is the average number of events generated by each node every minute. For each EGR, we measure the performance of DLT in an experiment of 2 hours. From the experiment results in Table 4, we see that DLT can reliably send the event information to the sink in short time with small energy consumption for most EGRs. The reliability of data delivery is high for all EGRs lower than 2/min. When the EGR is 1/min, the performance of

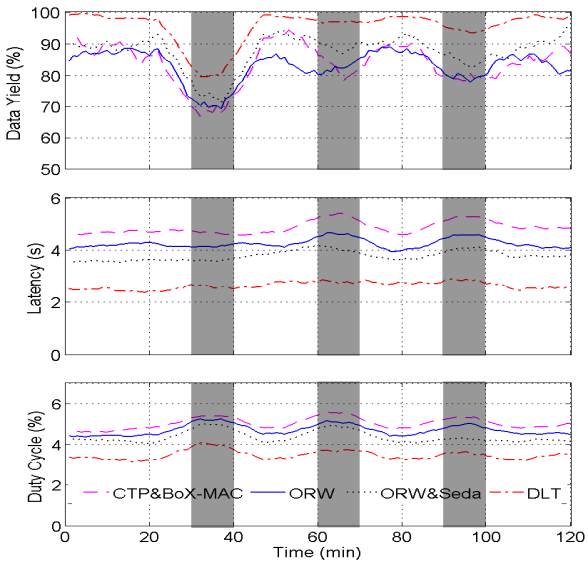


Figure 14: Performance under outage. The gray shadow indicates the duration when a node is disabled. The sequence of the disabled nodes is W11, W09 and W04.

DLT is even better than the periodical data collection pattern. As the events generated by all nodes are independent in the event-driven traffic pattern, fewer nodes transmit at the same time, and the probability of collision and congestion is smaller than the periodical data collection pattern. When the EGR is 5/min, more collisions and congestions are caused by the heavy traffic. As a result, the reliability becomes lower; besides, the latency and energy consumption increase.

5 Related works

Applications. In the last decade, a large number of sensor networks [40, 25, 22, 21, 7, 23, 1, 28] have been deployed for various applications, such as shooter detection, agriculture, healthcare and building automation. Besides, many large scale systems with hundreds of nodes [14, 20, 29, 33] have been developed, like multi-target tracking, military surveillance, temperature measurement and forest monitoring. TinyNode has been used in many projects for environmental monitoring, such as SensorScope [2] and PermaSense [42]. All the above systems are, however, densely deployed at scale, which requires large number of sensors and heavy maintenance due to network failures or environment dynamics [32]. The only deployment of sparse sensor networks, to the best of our knowledge, is a system of 9 Fleck-3 monitoring the salinity level of underground water with a mean communication distance of 800m [26]. While it is a practical deployment, its delivery rate is low, about 64%.

Rateless codes. Strider [19] and Spinal code [38] are the most recent rateless codes designed for Gaussian channels; they nevertheless cannot be implemented on low-power wireless devices due to the high computational complexity. The digital fountain approach conception is first introduced in [5]. LT code [35] enables rateless transmission of encoded blocks by XOR operations and an elegant design of the

Table 4: Performance of the event-driven traffic pattern with different event generation rates.

Performance / EGR (per min)	Yield	Latency	Energy consumption
0.1	99.4%	1.7s	0.65%
0.5	99.4%	1.7s	1.2%
1	99.3%	1.9s	2.7%
2	96.6%	2.8s	6.5%
5	75.3%	7.9s	26.7%

coding scheme. It is used for remote reprogramming in sensor networks [39] at packet-level. LT-W [34] improves the decoding efficiency by Wiedemann Solver, whereas it is difficult to be parallelized. MT-Deluge [16] employs multiple threads in TinyOS to provide concurrent operations of coding and reception. RTOC [43] adopts Online code to improve transmission reliability. The performance of online code is highly determined by parameter tuning and the evaluation in [43] is only based on simulations.

Partial packet recovery. FEC approaches and hybrid ARQ can harness the correct bits in corrupted packets, e.g., ReedSolomon code used in ZipTx [30] to recover partial packets in WLANs. However, the existing approaches mainly focus on the error correction over well-established links and require accurate channel estimation to gauge proper redundancy to compensate for the bit error, which is difficult in sparse sensor networks. Even worse, if sensor nodes are duty-cycled, it is more challenging to achieve accurate channel estimation. DLT automatically approaches the capacity of different links. The block-level data link protocol, Seda [15], is not efficient because it needs to retransmit the exact erroneous blocks and cannot add protection before transmissions. SpaC [10] combines multiple corrupted packets to recover the original data, whereas it does not proactively adapt to channel loss.

Routing in sensor networks. Dozer [4] and Koala [37] collect sensor data through TDMA-based scheduling on a tree topology for delay-insensitive applications. They, however, are not suitable for sparse sensor networks with dynamic transmission times. DSF [18] improves the reliability and latency of data forwarding by transmitting to multiple forwarding nodes. CBF [6] builds a forwarder cluster to enable opportunistic routing in sensor networks. ORW [24] incorporates opportunistic routing in low duty-cycle sensor networks to reduce latency and energy consumption. DOF [31] finds the duplicate problem is severe in ORW when the traffic load is high. ORLP [12] extends ORW to low-power IPv6 networks.

6 Conclusions

This paper presents DLT, a low-power networking approach for sparse wireless sensor networks at large-scale. DLT expands the communication range of sensor motes and fully explores link capability by continuously transmitting rateless blocks. The network diversity can thus be enriched. We propose a link layer protocol to support distanceless link transmission, and tackle many technical challenges during the implementation of rateless codes on sensor motes. We

further propose a tailored metric EDTT for efficient data collection. EDTT can be directly integrated with CTP for network-wide data collection and further extended to the data collection in duty-cycled sensor networks. We evaluate the performance of DLT in a deployed wireless sensor network. The results show that DLT outperforms existing protocols in terms of data yield, latency, and energy consumption.

7 Acknowledgments

We would like to thank our shepherd Koen Langendoen and the anonymous reviewers for their valuable comments and suggestions that improve the quality of this paper. We also thank Zikun Xing, Cheng Liu and Wei Huang for their help on the in-field experiments. This work is supported by the Singapore National Research Foundation under its Environment and Water Technologies Strategic Research Programme and administered by the Environment and Water Industry Programme Office (EWI) of the PUB on project 1002-IRIS-09. This work is also supported in part by Singapore MOE AcRF Tier 2 MOE2012-T2-1-070 and NTU Nanyang Assistant Professorship (NAP) grant M4080738.020.

8 References

[1] Y. Agarwal, B. Balaji, S. Dutta, R. K. Gupta, and T. Weng. Duty-cycling buildings aggressively: The next frontier in hvac control. In *ACM/IEE IPSN*, 2011.

[2] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli. The hitchhiker's guide to successful wireless sensor network deployments. In *ACM SenSys*, 2008.

[3] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno. On the fly gaussian elimination for lt codes. *IEEE Commun. Lett.*, 2009.

[4] N. Burri, P. Von Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *ACM/IEEE IPSN*, 2007.

[5] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM*, 1998.

[6] Q. Cao, T. Abdelzaher, T. He, and R. Kravets. Cluster-based forwarding for reliable end-to-end delivery in wireless sensor networks. In *IEEE INFOCOM*, 2007.

[7] O. Chipara, C. Lu, T. C. Bailey, and G.-C. Roman. Reliable clinical monitoring using wireless sensor networks: experiences in a step-down hospital unit. In *ACM SenSys*, 2010.

[8] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore. Environmental wireless sensor networks. *Proceedings of the IEEE*, 2010.

[9] W. Du, Z. Xing, M. Li, B. He, L. H. C. Chua, and H. Miao. Optimal sensor placement and measurement of wind for urban ecological studies. In *ACM/IEEE IPSN*, 2014.

[10] H. Dubois-Ferrière, D. Estrin, and M. Vetterli. Packet combining in sensor networks. In *ACM SenSys*, 2005.

[11] H. Dubois-Ferrire, R. Meier, L. Fabre, and P. Metrailler. Tinynode: A comprehensive platform for wsn applications. In *ACM/IEEE IPSN*, 2006.

[12] S. Duquennoy, O. Landsiedel, and T. Voigt. Let the tree bloom: scalable opportunistic routing with ORPL. In *ACM SenSys*, 2013.

[13] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *ACM SenSys*, 2010.

[14] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler. Trio: enabling sustainable and scalable outdoor wireless sensor network deployments. In *ACM/IEEE IPSN*, 2006.

[15] R. K. Ganti, P. Jayachandran, H. Luo, and T. F. Abdelzaher. Datalink streaming in wireless sensor networks. In *ACM SenSys*, 2006.

[16] Y. Gao, J. Bu, W. Dong, C. Chen, L. Rao, and X. Liu. Exploiting concurrency for efficient dissemination in wireless sensor networks. *IEEE Trans. on Parallel and Distributed Systems*, 2013.

[17] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *ACM SenSys*, 2009.

[18] Y. Gu and T. He. Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links. In *ACM SenSys*, 2007.

[19] A. Gudipati and S. Katti. Strider: automatic rate adaptation and collision handling. In *ACM SIGCOMM*.

[20] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. Stankovic, and T. Abdelzaher. Achieving long-term surveillance in vigilnet. In *IEEE INFOCOM*, 2006.

[21] X. Jiang, M. Van Ly, J. Taneja, P. Dutta, and D. Culler. Experiences with a high-fidelity wireless building energy auditing network. In *ACM SenSys*, 2009.

[22] Y. Kim, T. Schmid, Z. M. Charbiwala, J. Friedman, and M. B. Srivastava. NAWMS: nonintrusive autonomous water monitoring system. In *ACM SenSys*, 2008.

[23] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh. Wireless sensor networks for healthcare. *Proceedings of the IEEE*, 2010.

[24] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low power, low delay: opportunistic routing meets duty cycling. In *ACM/IEEE IPSN*, 2012.

[25] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In *IEEE IPDPS*, 2006.

[26] T. Le Dinh, W. Hu, P. Sikka, P. Corke, L. Overs, and S. Brosnan. Design and deployment of a remote robust sensor network: Experiences from an outdoor water quality monitoring network. In *IEEE LCN*, 2007.

[27] P. A. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self regulating algorithm for code propagation and maintenance in wireless sensor networks. In *USENIX NSDI*, 2004.

[28] F. Li, T. Xiang, Z. Chi, J. Luo, L. Tang, L. Zhao, and Y. Yang. Powering indoor sensing with airflows: a trinity of energy harvesting, synchronous duty-cycling, and sensing. In *ACM SenSys*, 2013.

[29] C.-J. M. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao. RACNet: a high-fidelity data center sensing network. In *ACM SenSys*, 2009.

[30] K. C.-J. Lin, N. Kushman, and D. Katabi. ZipTx: Harnessing partial packets in 802.11 networks. In *ACM MobiCom*, 2008.

[31] D. Liu, Z. Cao, J. Wang, Y. He, M. Hou, and Y. Liu. DOF: Duplicate detectable opportunistic forwarding in duty-cycled wireless sensor networks. In *IEEE ICNP*, 2013.

[32] K. Liu, M. Li, Y. Liu, M. Li, Z. Guo, and F. Hong. Passive diagnosis for wireless sensor networks. In *ACM SenSys*, 2008.

[33] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, J. Zhao, et al. Does wireless sensor network scale? a measurement study on greenorbs. In *IEEE INFOCOM*, 2011.

[34] H. Lu, F. Lu, C. FOH, and J. Cai. LT-W: Improving LT decoding with Wiedemann solver. *IEEE Trans. on Information Theory*, 2013.

[35] M. Luby. LT codes. In *IEEE FOCS*, 2002.

[36] D. Moss and P. Levis. Box-macs: Exploiting physical and link layer boundaries in low-power networking. Technical Report Technical Report SING-08-00, Stanford University, 2008.

[37] R. Musaloiu-E, C.-J. M. Liang, and A. Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *ACM/IEEE IPSN*, 2008.

[38] J. Perry, P. A. Iannucci, K. E. Fleming, H. Balakrishnan, and D. Shah. Spinal codes. In *ACM SIGCOMM*, 2012.

[39] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. Harris, and M. Zorzi. SYNAPSE: A network reprogramming protocol for wireless sensor networks using fountain codes. In *IEEE SECON*, 2008.

[40] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *ACM SenSys*, 2004.

[41] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An empirical study of low-power wireless. *ACM Trans. on Sensor Networks*, 2010.

[42] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin. PermaSense: investigating permafrost with a WSN in the Swiss Alps. In *ACM EmNets*, 2007.

[43] A. D. Wood and J. A. Stankovic. Online coding for reliable data transfer in lossy wireless sensor networks. In *IEEE DCOSS*, 2009.

[44] X. Wu, M. Liu, and Y. Wu. In-situ soil moisture sensing: Optimal sensor placement and field estimation. *ACM Trans. Senor Netw.*, 2012.