L^2: Lazy Forwarding in Low-Duty-Cycle Wireless Sensor Network

Zhichao Cao, *Member, IEEE, ACM*, Yuan He, *Member, IEEE*, Qiang Ma, *Member, IEEE*, and Yunhao Liu, *Senior Member, IEEE*

***Abstract*—In order to simultaneously achieve good energy efficiency and high packet delivery performance, a multihop forwarding scheme should generally involve three design elements: media access mechanism, link estimation scheme, and routing strategy. Disregarding the low-duty-cycle nature of media access often leads to overestimation of link quality. Neglecting the bursty loss characteristic of wireless links inevitably consumes much more energy than necessary and underutilizes wireless channels. The routing strategy, if not well tailored to the above two factors, results in poor packet delivery performance. In this paper, we propose L^2, a practical design of data forwarding in low-duty-cycle wireless sensor networks. L^2 addresses link burstiness by employing multivariate Bernoulli link model. Further incorporated with synchronized rendezvous, L^2 enables sensor nodes to work in a lazy mode, keep their radios off most of the time, and realize highly reliable forwarding by scheduling very limited packet transmissions. We implement L^2 on a real sensor network testbed. The results demonstrate that L^2 outperforms state-of-the-art approaches in terms of energy efficiency and network yield.**

***Index Terms*—Energy constraint, forwarding scheme, low duty cycle, wireless sensor network.**

I. Introduction

Wireless sensor networks (WSNs) [1], [2] are mostly battery-powered and thus energy-constrained. In order to save energy, sensor nodes are duty-cycled and rely on multihop forwarding to deliver data to the sink.

Design of data forwarding mechanism, which guarantees the packet delivery ratio (PDR) and keeps the energy consumption low, is a crucial and challenging issue in low-duty-cycle WSNs. A widely adopted low-duty-cycle protocol is X-MAC [3], in which sensor nodes sleep and wake up asynchronously. To guarantee transmission of a data packet from sender to receiver, the sender has to keep sending multiple copies of the same packet (called *preamble*) for a long period that exceeds the sleeping period of the receiver, called Low Power Listening (LPL) [9]. LPL has fine performance under low data rates in sparse deployments. Sending long preambles, however, overly consumes energy in communications and underutilizes wireless channels [16], [17]. The preambles can be shortened if there is a coordination mechanism between sender and receiver. One way is to exert global time synchronization at the cost of extra overhead. Under such a scenario, the PDR becomes highly dependent on the synchronization accuracy. It is often very difficult to set an ideal tradeoff between PDR and synchronization overhead. On the other hand, forwarding packets in the form of short preambles brings additional challenges on link estimation. It demands precise characterization of transient link quality instead of long-term link quality [20]. Unfortunately, the state-of-the-art schemes of link estimation, e.g., 4-bit [4], do not well address this issue in the context of low-duty-cycle WSNs.

In this paper, we propose L^2, a data forwarding approach for low-duty-cycle WSNs. With L^2, a node is able to dynamically schedule data forwarding to multiple good parents instead of one deterministic parent. Accordingly, with low-cost synchronized rendezvous (each node synchronizes the wake-up schedule of its neighbors to its local time), a long preamble in X-MAC is shortened and divided into multiple short ones. L^2 incorporates a multivariate Bernoulli link model [5] for link estimation. Based on the precise characterization of transient link quality, L^2 maximizes delivery yield of each short preamble. The surprising result of using L^2 is that good energy efficiency and high PDR are achieved simultaneously. Major contributions of this work are as follows.

1) We introduce a new and realistic link estimation scheme based on multivariate Bernoulli link model, which characterizes bursty links in low-duty-cycle WSNs.
2) We design an efficient dynamic forwarding algorithm by coordinating multiple forwarders with different wake-up schedules. It helps to improve PDR by 2.4% and save energy by 15.3% than the deterministic way.
3) We implement L^2 and evaluate its performance on a real WSN testbed. Our experiments demonstrate that L^2 gains 20.5% improvement over DSF [8] in terms of energy efficiency and 3.3% improvement over A-MAC [9] in terms of PDR.

The rest of the paper is organized as follows. Section II summarizes the related works and discusses the research motivation. Section III elaborates on the design of L^2. We present the implementation details and show the results of performance evaluation in Section IV. We discuss some open issues in Section V. We conclude in Section VI.

Manuscript received November 16, 2012; revised August 01, 2013; accepted February 25, 2014; approved by IEEE/ACM Transactions on Networking Editor S. Chen. Date of publication April 02, 2014; date of current version June 12, 2015. This work was supported in part by the NSFC Major Program 61190110, the National Basic Research Program (973 program) under Grant 2014CB347800, the NSFC under Grant 61170213, and the National Science Fund for Distinguished Young Scholars under Grant 61125202.

The authors are with School of Software, TNLIST, Tsinghua University, Beijing 100084, China (e-mail: caozc@greenorbs.com; he@greenorbs.com; maq@greenorbs.com; yunhao@greenorbs.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2014.2310812

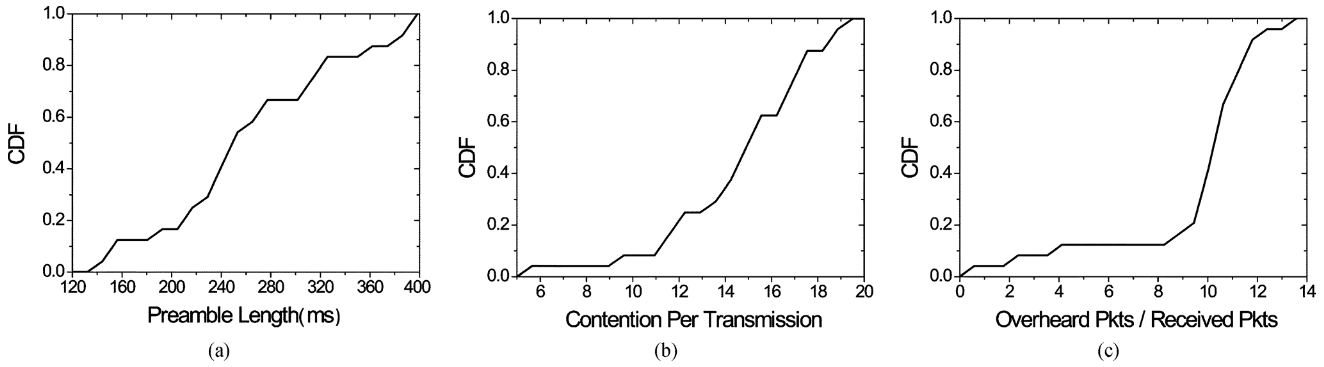


Fig. 1. CDF of the performance of channel utilization by hooking CTP, 4-bit, and X-MAC. (a) CDF of the average preamble length. (b) CDF of the contention number per transmission. (c) CDF of overhear number per received packets.

II. Motivation and Related Work

This work is motivated by our real-world experience from GreenOrbs [1], a large-scale WSN to collect forestry environmental data. GreenOrbs adopts the Collection Tree Protocol (CTP) [8] for data collection. 4-bit [4] is utilized for link estimation. The transceiver of every node works with X-MAC, the default low-duty-cycle media access mechanism in TinyOS 2.1.1.

A. Energy and Channel Utilization

We carry out an experiment, in which we hook CTP, 4-bit, and X-MAC directly with the default setting and use 25 TelosB nodes on a testbed. The network diameter is 4 hops, when the transmission power is set at $\mathrm{Tx\_Power} = 1$ and the wireless channel of CC2420 is set at 26 in TinyOS. Every node generates data packets periodically. The interval between two consecutive packets is a random value in [2 s, 4 s]. The packet length is 62 B. The node density, defined as the average number of neighbors of a node, is 8.7.

To verify the energy efficiency, a node individually calculates its preamble length in average. Fig. 1(a) shows that the expected preamble length is 263.5 ms, which is about half of the default sleep interval (512 ms). It is much longer than what is necessary for a single preamble packet transmission ($\approx 4 \sim 10$ ms).

Due to the relatively long preamble, the probability of the channel contention increases dramatically. Fig. 1(b) shows the cumulative distribution function (CDF) of the average times of contention per transmission. It shows that a sender contends the channel about 14.5 times in average during a preamble transmission. The severe contention leads to inefficient channel utilization and reduces the throughput [24].

A node may overhear the preamble packets sent to other nodes. Overhearing wastes energy and does not enhance the forwarding quality. Fig. 1(c) shows the CDF of the ratio of the overheard packets to the total received packets. In average, a node receives only one packet while overhearing 9.5 packets.

Some existing works, such as SCP [10], PW-MAC [11], and A-MAC [7], address the above problem. The general idea is based on synchronization and/or receiver-initiated transmission. The efficacy of those approaches highly relies on the accuracy of time synchronization or the successful reception of channel probing. L^2 proposes a low-cost local synchronization scheme and develops a window-based transmission to cope with synchronization variance and link burstiness.

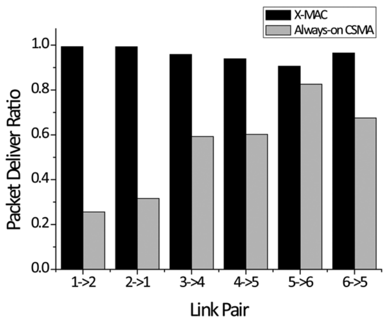


Fig. 2. Comparison of the network-layer PDR.

B. Link Estimation

We carry out an experiment using three pairs of TelosB nodes with diverse link quality. Each node broadcasts 200 packets at 10-ms intervals and calculates the PDR. Fig. 2 shows that the quality of all links is overestimated with X-MAC than with always-on CSMA. In the experiment, a node keeps awake for 20 ms when it detects the busy channel. The packet length is 43 B. The transmission time of each preamble packet is about 1.4 ms. Thus, a receiver may hear over 10 copies of the same packet. As a result, when the link quality is very poor, the PDR without X-MAC is as low as lower than 0.3, while the PDR with X-MAC can be close to 1 in X-MAC. Misled by such overestimates of link quality, a node tends to make improper forwarding decisions.

The observation of β-factor [12] says that if the interval of two continuous transmissions is within 500 ms, the transmissions tend to be dependent, called bursty. Meanwhile, the interval of adjacent preamble packets is at most 8 ms, which is the time spent to wait acknowledgment. Thus, the insistent preamble transmission over bursty links is inefficient.

STLE [13] employs an overhearing scheme to estimate the short-term link behavior. Based on the history data, M&M [5] trains a multilevel Markov model to depict the short-term dependency among transmissions. 4C [22] predicts the successful packet delivery probability based on the combination of information from different aspects, namely PRR, physical-layer information, and the prediction model trained with history data. TALENT [23] applies machine learning methods to develop an online prediction model to predict the link quality in the near future. L^2 develops a practical online multivariate Bernoulli model, which characterizes both short-term link behavior and

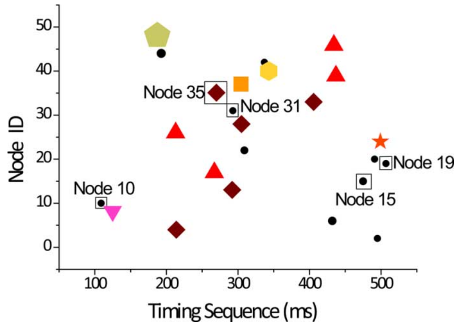


Fig. 3. Illustrative example of the potential opportunity and benefit to forward a packet to multiple receivers.

long-term delivery quality, to depict the burstiness on wireless links.

C. Forwarding

CTP selects a neighbor with the lowest transmission cost, measured in ETX [14], as the next-hop forwarder. The fail-and-retry mechanisms are adopted to ensure successful packet delivery. When a transmission fails, the sender continues to retransmits preambles. Neglecting the duty cycle nature, the deterministic scheme probably misses some ”early chances” to forward packets and thus underutilizes the channel.

Fig. 3 takes a snapshot of the experiment in Section II-A and sorts all 25 nodes according to the interval between the current time and the next wake-up time of a node. Different nodes may have a common routing parent. We plot such nodes with the same legend in the figure. We use node 35 as an example. According to the collected trace, node 10 is node 35's parent. Nodes 31, 15, and 19 have same ETX with node 10. They are also the neighbors of node 35. Suppose a packet at node 35 needs to be sent at about 300 ms. It is clearly a beneficial opportunity to involve 31, 15, and 19 as the potential forwarder, such that the preamble length is shorter than singly relying on node 10 for forwarding.

In dynamic forwarding [6], [15] of low-duty-cycle WSNs, the sender selects multiple candidate forwarders. When any candidate forwarder receives the packet, the sender will be acknowledged and stop the transmission. The subsequent forwarding responsibility is taken over by the next-hop forwarder that receives the packet. ORW [21] exploits opportunistic routing in low-duty-cycle WSNs and addresses a packet to multiple potential forwarding nodes. Based on our proposed adaptive link model, L^2 efficiently schedules forwarding to multiple receivers while enhancing the PDR with the energy and delay constraints.

III. Design

Fig. 4 illustrates the design of L^2. The link estimation layer maintains the neighbors' wake-up schedule. Based on the schedule, the preamble is shortened as a small window. Meanwhile, we develop a conditional probabilistic model to depict the successful delivery probability of the window-based transmission. L^2 selects a set of forwarders with dynamic lengths of transmission windows to forward the data packets.

A. Window-Based Unicast

The design of the window-based unicast is shown in Fig. 5. The transmission is ended when either the sender receives an

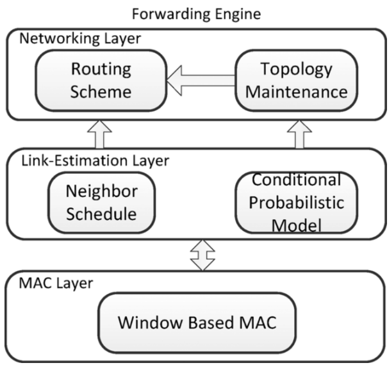


Fig. 4. Conceptual architecture of the L^2 design.

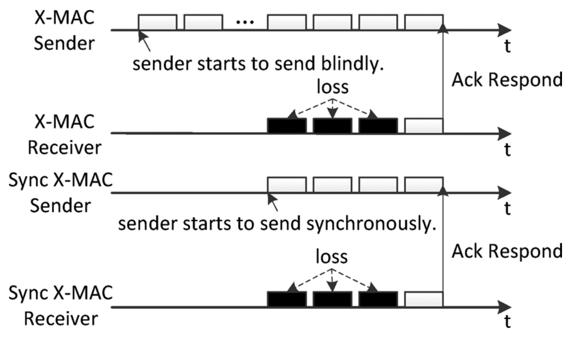


Fig. 5. X-MAC versus synchronized X-MAC transmission and reception.

acknowledgment or the preamble length exceeds an adaptive threshold (called *transmission window*). If the receiver detects the channel is busy, it keeps awake for a predefined period (called *receiving window*. In contrast to the blind sending of X-MAC, a sender knows the time at which the receiver wakes up. Thus, it may start to send the preamble just a short while before the receiver wakes up. The prefix sending is utilized to compete for the channel with other potential senders and to keep the receiver awake. If the receiver does not receive any packet in the receiving window, it returns to sleep. Thus, the maximum preamble length should not exceed the size of the receiving window. We use TW_LENGTH to denote the predefined size of receiving window.

For local synchronized rendezvous, a node aligns the wake-up schedules of other nodes to its local time. Specifically, as illustrated in Fig. 6, for node i, $t_i(\cdot)$ indicates the latest wake-up time of node i. For each neighbor j of node i, we keep a constant interval $offset_i(j)$ between $t_i(\cdot)$ and $t_j(\cdot)$. $offset_i(j)$ is calculated according to MAC-layer timestamp, which is piggybacked on the routing beacon to avoid extra energy cost. As Fig. 6 shows, the Start of Frame Delimiter (SFD) interrupt appears on both sender and receiver sides at the same time before the real payload is transmitted. Assume the sleep interval of all nodes is τ. Then, $offset_i(j)$ is calculated as follows:

$$\begin{aligned} &offset_i(j) \\ &\quad = (k\tau + SendOff + PreaOff - HearOff) \bmod \tau \end{aligned} \quad (1)$$

where k is an integer and $k\tau$ is used to keep $offset_i(j)$ as a positive value.

In practice, the expected synchronized rendezvous often deviates from the actual one. We propose the following method

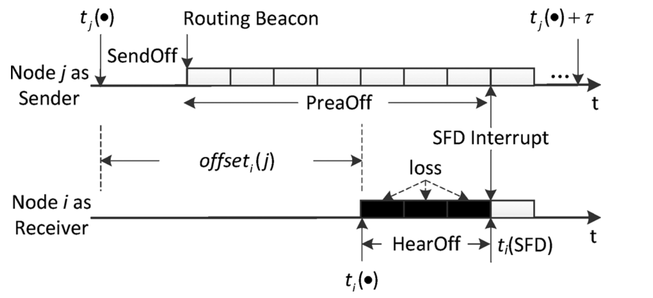


Fig. 6. Way of local synchronization by MAC-layer timestamp. $t_i(\cdot)$ is the latest wake-up time of node i. Routing Beacon is the beginning of the beacon preamble. SFD interrupt happens simultaneously on both sender and receiver to send/receive the packet. $offset_i(j)$, SendOff, PreaOff, and HearOff indicate the time offsets among these events.

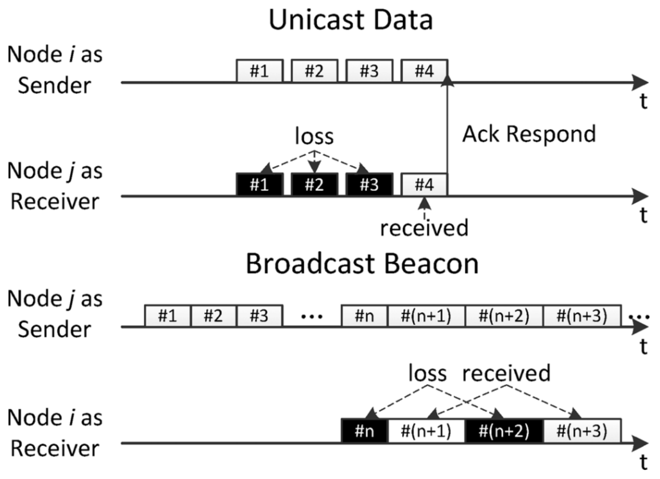


Fig. 7. Link estimation based on window transmission.

to cope with those uncertainties. By storing the four recent historical offset records, Linear Regression is used to calibrate the skews incurred by clock drifts and software delays. Besides the transceiver initiation process, the sender needs extra time to load the payload into memory buffer of the transceiver, which usually takes several milliseconds. To ensure that the sender begins transmission exactly before the transceiver initiation of the receiver is done, a conservative and empirical guard time, 10 ms, is set. The sender initiates the transmission 10 ms earlier than the expected wake-up time of the receiver.

The reliability of MAC-layer timestamp on each packet is not ideal in the current TinyOS 2.1.1 radio stack. The consecutive SFD interrupts are so close to each other. Sometimes it becomes impossible to deal with those interrupts in such jamming timings. Thus, we set 5-ms interval between two consecutive preamble packets of routing beacons.

It is necessary to add another bit, R (*refresh* bit) in a routing beacon to inform the change of local wake-up schedule. After receiving the routing beacon with R bit set, each node refreshes all records of the source node and initializes it again. This ensures correct time prediction of the events such as reboot, configuration update, and newly connected nodes.

B. Link Estimation

L^2 counts both data traffic and beacons for link estimation. The task of L^2 is to precisely identify how many preamble packets have been successfully received by the receiver during the preamble transmission and reception process. L^2 assigns each preamble packet with a MAC-layer data sequence number ($macdsn$) as shown in Fig. 7.

For unicast data, the scope of $macdsn$ is from 1 to TW_LENGTH. According to the $macdsn$ of the acknowledged preamble packet, the sender knows the number of lost preamble packets. For example in Fig. 7, the first three preamble packets are lost.

For broadcast beacon, the $macdsn$ just begins at 1 and increases by 1 each time, until the timer of preamble transmission expires. For node i in the lower subfigure of Fig. 7, the nth and the $(n+2)$th packets are lost, while the $(n+1)$th and the $(n+3)$th packets are received.

Based on the above points, L^2 establishes two different probability models to estimate long-term and short-term link behavior, respectively.

In long-term view, L^2 treats each transmission of a preamble packet as an independent event. Define P as the probability of a successful preamble packet transmission. It is quantified as the ratio of successful packet reception of broadcast beacons. For example in Fig. 7, $P = 2/3$. P is mainly utilized to initialize and update the links with low data traffic.

In short-term view, L^2 treats the continuous transmission of preamble packets as dependent events. L^2 employs a single component TW_LENGTH-dimensional multivariate Bernoulli model to depict the behavior of unicast data. We define the successful delivery of the kth preamble packet in the transmission window as follows: *The previous $k-1$ preamble packets all fail to arrive at the receiver. The receiver receives the kth preamble packet successfully, and the sender also receives the acknowledgment of the kth preamble packet.* The probability of this event is denoted by p^k, which actually reflects the marginal effect to transmit one more preamble packet in the transmission window. This model is mainly instantiated based on unicast data traffic.

By combining the above two models, we define P^k as the probability that a preamble packet is successfully received when the transmission window is k

$$\begin{aligned} P^1 &= p^1 \\ P^k &= P^{k-1} + (1 - P^{k-1}) \cdot p^k. \end{aligned} \tag{2}$$

When p^k is not initialized (e.g., all preambles are acknowledged before the kth packet), we set p^k as P. P^k reflects the PDR of network-layer packets. Different transmission window sizes determine different network PDR and energy consumption.

Suppose the sender sends preambles using the maximum window size TW_LENGTH, we define RP^k as the probability that at least one preamble packet since the kth to the (TW_LENGTH)th is successfully received, while the previous $(k-1)$ preamble packets fail to arrive at the receiver. We have

$$\begin{aligned} RP^{\text{TW\_LENGTH}} &= p^{\text{TW\_LENGTH}} \\ RP^k &= p^k + (1 - p^k) \cdot RP^{k+1}. \end{aligned} \tag{3}$$

When RP^k is smaller than a predefined threshold (denoted by BURSTY_LOSS), it implies the existence of bursty. The chance for the left (TW_LENGTH $-\ k + 1$) preamble packets to arrive at the receiver is very low. In order to save energy while preserving PDR, it is wise to avoid transmitting those packets.

In the following sections, for the link from node i to j, we use $p_i^k(j)$, $P_i^k(j)$, and $RP_i^k(j)$ to denote the above three perspectives of link quality, respectively.

C. Dynamic Forwarding Scheme

L^2 incorporates a dynamic forwarding scheme, which coordinates multiple forwarders to help forwarding. The goal of L^2 is to maximize the network yield with specified energy budget and bounded per-hop delay. We define a routing metric *Expected Delivery Quality* (EDQ), which equals to the quotient of path *Expected Packet Delivery Ratio* (EPDR) and *path Hop count* (HOP).

$$EDQ = \frac{EPDR}{HOP}. \tag{4}$$

We use EDQ_i, $EPDR_i$ and HOP_i to denote EDQ, EPDR, and HOP of node i, respectively.

We fix the times of preamble packets that every packet forwarder is allowed to try. Then, the maximum amount of energy that can be spent for forwarding a packet from source node to the sink is proportional to the hop count of the routing path. The actual energy consumption is often much less than the energy budget. The path EPDR denotes the expected yield. The ratio of HOP to EPDR, namely the reciprocal of EDQ, quantifies the aggregated and conservative power consumption to successfully deliver a packet along a multihop path. Therefore, given a fixed total energy budget (the sum of maximum energy spent on all the hops) to forward a packet, EDQ quantifies the throughput of multihop forwarding with L^2.

The EDQ of a node can be calculated recursively. Specifically, for the sink, its EPDR is 1 and HOP is 0

$$\begin{aligned} EPDR_{\text{sink}} &= 1 \\ HOP_{\text{sink}} &= 0. \end{aligned} \tag{5}$$

Suppose node j is one of node i's neighbors, we use $EPDR_i(j)$, $HOP_i(j)$, and $EDQ_i(j)$ to respectively denote EPDR, HOP, and EDQ obtained when forwarding the data to j. When the size of transmission window is set at k, these metrics are calculated by

$$\begin{aligned} EPDR_i(j) &= EPDR_j \cdot P_i^k(j) \\ HOP_i(j) &= HOP_j + 1 \\ EDQ_i(j) &= \frac{EPDR_i(j)}{HOP_i(j)} = \frac{EDQ_j \cdot P_i^k(j)}{1 + 1/HOP_j}. \end{aligned} \tag{6}$$

The forwarder selection process includes two phases. First, the node selects one of its parents as the best forwarder f_{best}, via which the routing path from the current node to the sink has the highest EDQ. Second, the current node selects a set of potential candidate forwarders. Those candidate forwarders should have equal or higher EDQ than f_{best} in terms of the EDQ of the path from such a forwarder to the sink. In this way, L^2 ensures that no matter which candidate forwarder finally receives and forwards the packet, the resulting EDQ is equal or higher than the path EDQ offered by the previously selected best forwarder.

The sizes of the transmission window for different forwarders are determined by their diverse link qualities and the energy budgets. For node i, we assume a sending task is initiated at t_i. According to the delay bound Ω, node i finds a set of candidate forwarders $FC_i = \{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}$, sorted in the temporal order (i.e., how soon the forwarder will wake up). According to the link quality and the energy budget Ψ, node i determines the sizes of corresponding transmission windows of the forwarders as $\{l_1, l_2, \ldots, l_m\}$ to maximize the expected EDQ.

According to the link model presented in Section III-B, the expected EDQ of certain energy distribution can be calculated by

$$\begin{aligned} EDQ_i(\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}) &= \frac{P_i^{l_1}(f_1^{t_i}) \cdot EDQ_{f_1^{t_i}}}{1 + 1/HOP_{f_1^{t_i}}} \\ &\quad + (1 - P_i^{l_1}(f_1^{t_i})) \cdot EDQ_i(\{f_2^{t_i}, \ldots, f_m^{t_i}\}). \end{aligned} \tag{7}$$

In the right side of (7), the first item depicts the EDQ of the path through f_1, and the second item depicts the EDQ provided by the rest of forwarders if f_1 fails to receive the packet. Then, the optimization problem of energy distribution is

$$\begin{aligned} &\textbf{Maximize}: \quad EDQ_i(\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}) \\ &\textbf{Subject to}: \\ &\quad OFFSET(f_m^{t_i}) \le \Omega \\ &\quad \sum_{k=1}^{m} l_k \le \Psi \\ &\quad \forall k = 1, 2, \ldots, m. l_k \le TW\_LENGTH \\ &\quad \forall k = 1, 2, \ldots, m. RP_i^{l_k}(f_k^{t_i}) \ge BURSTY\_LOSS \\ &\quad \forall k = 1, 2, \ldots, m. EDQ_{f_{\text{best}}^i} \le EDQ_{f_k^{t_i}} \\ &\quad \forall k = 1, 2, \ldots, m. p_i^0(f_k^{t_i}) = 0. \end{aligned} \tag{8}$$

The first constraint means any acceptable candidate forwarder $f_k^{t_i}$ must wake up during $[t_i, t_i + \Omega]$. The second constraint means the total energy spent on preamble packets cannot exceed the energy budget. The third constraint means the size of the transmission window of a single forwarder should not exceed TW_LENGTH. The fourth constraint means every transmission window should be reasonably allocated so as to avoid bursty loss. The fifth constraint denotes the basic selection criterion of a candidate forwarder. The sixth constraint defines the boundary conditions.

The above optimization problem is NP-hard. We skip the proof here. Considering the resource constraints in WSNs, we propose a greedy algorithm to achieve suboptimal performance. Instead of optimizing the path EDQ, the greedy algorithm seeks to maximize one hop PDR. Given the selection criterion of candidate forwarders, such approximation does not degrade the overall EDQ. We define the expected EDQ of dynamic forwarding via node i using the greedy algorithm as follows:

$$\begin{aligned} &EDQ_i^*(\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}) \\ &\quad = P_i^{l_1}(f_1^{t_i}) + (1 - P_i^{l_1}(f_1^{t_i})) \cdot EDQ_i^*(\{f_2^{t_i}, \ldots, f_m^{t_i}\}). \end{aligned} \tag{9}$$

The greedy forwarder scheduling problem is formalized as

$$\begin{aligned} &\textbf{Maximize}: \quad EDQ_i^*(\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}) \\ &\textbf{Subject to}: \\ &\quad OFFSET(f_m^{t_i}) \le \Omega \\ &\quad \sum_{k=1}^{m} l_k \le \Psi \\ &\quad \forall k = 1, 2, \ldots, m. l_k \le TW\_LENGTH \end{aligned}$$

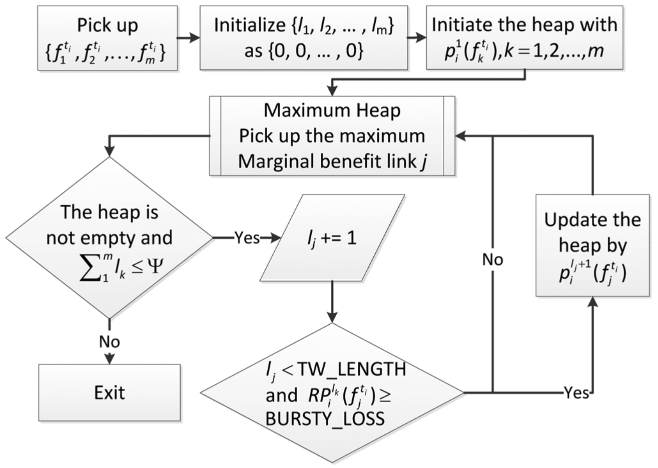

Pick up $\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}$
Initialize $\{l_1, l_2, \ldots, l_m\}$ as $\{0, 0, \ldots, 0\}$
Initiate the heap with $p_i^1(f_k^{t_i}), k=1,2,\ldots,m$
Maximum Heap Pick up the maximum Marginal benefit link j
The heap is not empty and $\sum_1^m l_k \le \Psi$
Yes
l_j += 1
No
Exit
$l_j <$ TW_LENGTH and $RP_i^{l_k}(f_j^{t_i}) \ge$ BURSTY_LOSS
No
Yes
Update the heap by $p_i^{l_j+1}(f_j^{t_i})$

Fig. 8. Process of greedy forwarder scheduling.

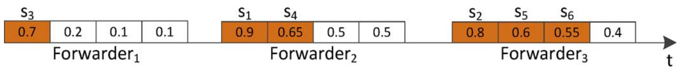


Fig. 9. Example of greedy forwarder scheduling.

$$
\begin{aligned}
&\forall k = 1,2,\ldots,m. RP_i^{l_k}(f_k^{t_i}) \ge BURSTY\_LOSS \\
&\forall k = 1,2,\ldots,m. EDQ_{f_{\text{best}}^i} \le EDQ_{f_k^{t_i}} \\
&\forall k = 1,2,\ldots,m. p_i^0(f_k^{t_i}) = 0. \qquad (10)
\end{aligned}
$$

In this scenario, $EDQ_i^*(\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\})$ is independent from the temporal order of $\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}$. Given the set of candidate forwarders and the corresponding link quality, a node only needs to allocate more energy to the forwarder with the maximum marginal effect to increase the overall PDR.

Fig. 8 illustrates the algorithm of greedy forwarder scheduling for node i. The maximum heap is the data structure to filter the link j with the maximum marginal effect when transmitting one more preamble packet. The time complexity of heap sorting is $O(\log m)$, and the size of heap is $O(\mathrm{m})$. Therefore, the time and space complexity of the greedy algorithm is $O(\Psi \log m)$ and $O(\mathrm{m})$. In practice, the wake-up time of different forwarders might be close to each other. Such forwarders should not be simultaneously selected into the candidate set. Only the one with the maximum EDQ will be selected. Take Fig. 9 as an example, we assume the TW_LENGTH and energy budget of each forwarder are four and six preamble packets. The value in each cell indicates the conditional delivery probability in the corresponding preamble slot. According to the greedy forwarding algorithm, six slots are selected in the descending order of the conditional delivery probability, namely $\{s_1, s_2, s_3, s_4, s_5, s_6\}$. The actual order to send preambles is $\{s_3 \rightarrow s_1 \rightarrow s_4 \rightarrow s_2 \rightarrow s_5 \rightarrow s_6\}$.

IV. Implementation and Evaluation

A. Implementation

We implement L^2 on TinyOS 2.1.1. The three main components of L^2 are window-based transmission, multivariate Bernoulli link estimation, and dynamic forwarding scheme. To verify the efficacy and efficiency of each component, we compile six different programs with different combinations of those functional components, as shown in Table I. Dynamic L^2 contains all three components. For comparison, we also implement DSF, A-MAC, and deterministic L^2. Deterministic L^2 consists of the window-based transmission and Bernoulli link estimation model, but it only uses the best forwarder to relay data packets. We carry out indoor (25 TelosB motes) and outdoor (40 TelosB motes) experiments for performance evaluation. The network diameters of the two testbeds are 4 and 9 hops, respectively.

TABLE I
Programs With Different Components

Version	Memory Utilization	
	RAM(bytes)	***ROM(bytes)***
CTP + 4-bit + X-MAC (X-MAC)	5490	27666
CTP + 4-bit + Sync X-MAC (Sync X-MAC)	6427	36958
CTP + 4-bit + A-MAC (A-MAC)	5410	29794
DSF + 4-bit + Sync X-MAC (DSF)	6507	38070
Deterministic L^2	6933	39386
Dynamic L^2	6933	39572

1) *Window-Based Transmission:* In the implementation of window-based transmission, we divide the original LPL component of X-MAC into two separate parts, namely *unified broadcast* and *smart unicast*. The unified broadcast transmits the routing beacon, which synthetically carries all information from different layers. The frequency of routing beacon is controlled by TrickleTimer [19]. Smart unicast is initiated by the dynamic forwarding scheme. However, Table I indicates that the window-based transmission, which differs Sync X-MAC from X-MAC, costs extra 9292 B ROM and 937 B RAM, respectively.

2) *Link Estimation:* For the implementation of multivariate Bernoulli link model, a node maintains both long-term statistical estimation and the estimation based on every preamble packet in the transmission window. According to the *macdsn* of the acknowledged preamble packet, a node updates the local estimation records. When a node accumulates sufficient observations of a link, it updates the link estimation in an exponentially weighted moving average method.

3) *Dynamic Forwarding:* A node updates its current best forwarder and path EDQ periodically. It obtains its forwarding schedule using the greedy scheduling algorithm. As we mention in Section III-C, time and spatial complexity of the greedy algorithm are $O(\Psi \log m)$ and $O(m)$. m is determined by the wake-up schedule of forwarders and the delay bound Ω. In our default setting, Ψ is 30 and Ω is the length of a sleep interval. In other words, the link layer transmits at most 30 preamble packets to all forwarder candidates, which wake up in the future Ω period.

We measure the time of task executions through experiments. We set the Ω (the delay bound) as 512 ms, which equals to the node's sleep interval. In our experiments, m (the number of next-hop forwarders) is between 1 and 6. The minimum and maximum task execution times are 3.8 and 13.6 ms, respectively. We execute these tasks every 4 s to keep the routing table up to date. The computation complexity is affordable for typical sensor motes, e.g., TelosB.

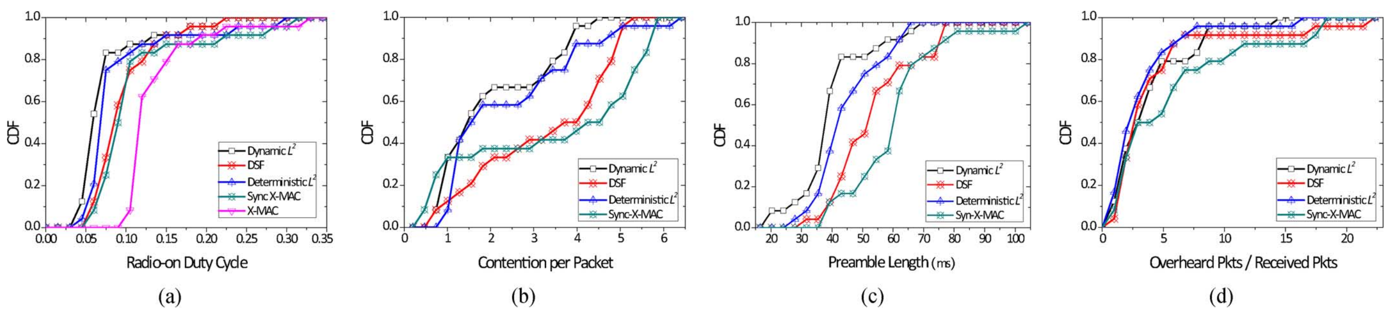


Fig. 10. (a) CDF of radio duty cycle. (b) CDF of the contention per packet. (c) CDF of the average preamble length. (d) CDF of the ratio of the overheard packets to the received packets.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT APPROACHES

Version	Average Performance				
	***PDR* (%)**	***Duty Cycle* (%)**	***Preamble Length (ms)***	***CPP*[1]**	***ROR*[2]**
X-MAC	97.1	13.4	263.5	14.5	9.5
Sync X-MAC	97.7	10.7	59.6	3.4	5.59
DSF	99.1	9.4	53.3	3.22	4.2
Deterministic L^2	97.1	8.5	44.3	2.4	3.1
Dynamic L^2	99.5	7.2	38.6	2.0	3.6

[1] CPP is the abbreviation of the contention per packet, which indicates the number of the contention to transmit one network packet.
[2] ROR is the abbreviation of the ratio of the overheard packets to the received packets.

B. Performance Evaluation

We set the same sleep interval as X-MAC does, namely 512 ms. All nodes (including the sink) follow such a wake-up schedule. At the application layer, the time interval to generate a packet is a random value between [2–4] s. The experiments with the setting are run on 25 TelosB nodes in an indoor testbed. We set the radio power at 1 to limit the transmission range and get a 4-hop network. Each program is run for half an hour. For each node, about 500 packets can be collected by the sink by the end of that period.

1) Performance Overview: We compare the performance of different approaches with respect to the average PDR and radio duty cycle (which indicates the aggregated energy consumption on a node) of all nodes. What we expect is that, with window-based transmission, the PDR increases and the radio duty cycle decreases. Deterministic L^2 improves the energy efficiency further because it addresses the bursty loss problem. DSF increases PDR, but due to bursty loss, the energy efficiency might be unsatisfactory. Dynamic L^2 achieves improved PDR and decreased energy consumption simultaneously.

From Table II, we can see that compared to X-MAC, the improvement in energy efficiency by dynamic L^2 is $(13.4 - 7.2)/13.4 = 46.2\%$. With the multivariate Bernoulli link model, the PDR of deterministic L^2 is comparable to that of Sync X-MAC, but deterministic L^2 improves the energy efficiency by $(10.7 - 8.5)/10.7 = 20.6\%$. The PDR of DSF and dynamic L^2 are both very close to 100%, while the energy efficiency of dynamic L^2 is $(9.4 - 7.2)/9.4 = 23.4\%$ better than that of DSF. Those results demonstrate that the multivariate Bernoulli link model enables better energy efficiency than 4-bit does.

To further investigate the proportion of the bursty links in our experiments, we calculate the β-factor of the links on the 25-nodes testbed, when the output power and the wireless channel are set at 1 and 26, respectively. A link tends to be bursty when its β-factor is larger than 0. The result shows that about 12.5% of all the 225 links ($A \rightarrow B$ and $B \rightarrow A$ are counted as two links) appear to be bursty.

We notice that PDR of deterministic L^2, Sync X-MAC, and X-MAC are almost the same. The window-based transmission does not apparently improve the PDR because the synchronization bias potentially causes packet loss. Compared to deterministic L^2, dynamic L^2 improves PDR and energy efficiency by 2.4% and 15.3%, respectively. It indicates that dynamic forwarding is more adaptive to the asynchronous wake-up characteristic in low-duty-cycle WSNs. Fig. 10(a) shows the radio duty-cycle distribution of all nodes, which again demonstrates the performance advantage of dynamics L^2.

In Table II, the preamble length of Sync X-MAC is 4.4 times shorter than that of X-MAC. By using the multivariate Bernoulli link model in deterministic and dynamic L^2, the preamble length is further shortened by 25.6% and 27.6%, respectively. Fig. 10(c) also demonstrates that dynamic L^2 holds the shortest preamble. Because transmissions with long preambles easily conflict with each other, we can see that Fig. 10(b) has the similar trend with Fig. 10(c). The contention is relatively reduced in L^2.

Fig. 10(d) plots the CDF of the ratio of the overheard packets to the received packets. Lower ratio indicates lower chance of overhearing to happen. Fig. 10(d) reveals that the actual overheard ratio of most nodes is still larger than 1, i.e., a lot of overhearing occur during the experiments. Note that overhearing is related to the diversity of wake-up schedules among the nodes in a local area. The result in Fig. 10(d) implies that if the nodes in a local area are well scheduled [18], the performance of forwarding can be further improved.

2) Performance Comparison to ORW: We compare the average radio duty cycle, end-to-end delay, and path hop count of L^2, ORW, and CTP to X-MAC through the indoor testbed experiments. The time interval between packet generations is a random value between [1–2] min. All the nodes are duty-cycled. In the default ORW implementation, the sink is always on. This will significantly reduce the communication overhead at the last mile. For fair comparison in this experiment, the sink is made

TABLE III
COMPARISON OF THE PERFORMANCE WITH ORW

Sleep	Duty Cycle				Delay				Average Path Hop Count			
Interval	Trace [%]			Improve [%](vs. ORW)	Trace [ms]			Improve [%](vs. ORW)	Trace [#]			Improve [%](vs. ORW)
[ms]	L^2	ORW	CTP	Average	L^2	ORW	CTP	Average	L^2	ORW	CTP	Average
512	2.8	3.2	3	12.5	348.5	488.4	367.7	28.6	1.47	2.13	1.6	31
2048	2.2	2.8	3.4	21.4	1520.5	1635	1335.8	7	1.65	2.2	1.85	25

TABLE IV
COMPARISON OF THE FORWARDING DIVERSITY

Version	# of Forwarder Candidates	
	Average	*Standard Deviation*
DSF	1.61	0.55
Dynamic L^2	2.38	1.29

TABLE V
COMPARISON OF THE PERFORMANCE TO A-MAC

Version	Average Performance			
	PDR	*Duty-Cycle*	*Ave-Hop*	*Max-Hop*
A-MAC	94.7%	6.2%	4.0	9
Dynamic L^2	98.0%	3.3%	4.2	9

to be duty-cycled in ORW. We do the experiments with 512-ms and 2-s sleep intervals, respectively.

The results are shown in Table III. The PRR of all the three protocols is over 99%. Compared to ORW, L^2 has lower radio duty cycle, lower delay, and less average path hop count. Compared with CTP, which takes ETX as the routing metric, the average path hop count of L^2 is less. Due to the randomness of sleep schedule, the per-hop delay of each transmission is randomly distributed between 0 and the maximum sleep interval. Thus, when the sleep interval is large (2 s), the average delay of L^2 might be a bit larger than CTP.

3) Dynamic Forwarding: Table IV compares DSF and dynamic L^2 and indicates that L^2 makes more active use of potential forwarders than DSF. DSF selects the forwarder in a backtrack manner. When the best forwarder is selected, the other candidates that wake up before the best one are seldom utilized. It means the potential utility of candidate forwarders in DSF is affected by the occurrence time of sending event and the wake-up schedules of nodes. On the other hand, dynamic L^2 takes the delivery quality of each neighbor into account. It involves the neighbors, who have comparable delivery quality with the best forwarder.

4) Performance Comparison to A-MAC: We deploy 40 nodes in the form of a relatively sparse grid on a playground. A node generates a packet at a random interval between [30–60] s. The sleep interval of A-MAC is by default set at 128 ms. Based on the result in [7], 128 ms is the most adaptive sleep interval for A-MAC. The sleep interval of dynamic L^2 is set at 512 ms. We run 1-h experiments for both programs. The results are shown in Table V.

Compared to A-MAC, dynamic L^2 improves PDR by 3.3% and reduces energy consumption by about 50%. A-MAC suffers higher probe sending cost than dynamic L^2, which is due to the shorter sleep interval. Reducing the sleep interval of A-MAC, however, will probably degrade its PDR.

V. DISCUSSION

This section discusses several open issues with regard to the design and evaluation of a multihop forwarding protocol.

A. Routing Metric EDQ

No routing metric perfectly fits all kinds of applications. It is possible that with EDQ, a node might choose the forwarder with relatively lower EPDR, but small HOP. The fundamental difference between EDQ and EPDR is whether there is a precondition (or constraint) of energy budget. Specifically, when there is no energy budget on any node, the network throughput is exactly denoted by EPDR. Considering certain energy budget, however, EPDR does not always reflect the exact network throughput. When there is energy budget on every node to forward every packet, the energy budget on a certain routing path is limited as well. Under such circumstance, a metric that makes more sense is how many packets can be successfully received at the sink, given a fixed amount of energy to forward the packets from the source to the sink. Thus, EDQ actually denotes the efficiency of packet forwarding in terms of energy consumption. In the cases of this paper, EDQ is more suitable than the existing metrics. In some other applications scenarios, EPDR might be a better metric than EDQ .

B. Forwarder Set Selection

In Section III-C, we select the forwarder set according to the constraint that the EDQ of the candidate should be greater than the EDQ of the best forwarder f_{best}. The idea behind that is when we fix the topology in a deterministic way, we can make use of the potential opportunity given by routing redundancy to enable dynamic forwarding. In this way, it is guaranteed that the resulting EDQ is not worse than that in the deterministic forwarding, no matter which forwarder receives the data. It is possible that a node only chooses f_{best} and excludes all other neighbors. This only happens when the number of potential forwarders is very small. In that case, routing goes back to the deterministic way.

C. End-to-End Delay

The packet delivery latency is an important issue, especially considering the sleep latency in low-duty-cycle WSNs. It is possible to minimize the delay by choosing the neighbors with small sleep latency, but the path length might get longer. Though L^2 does not guarantee to minimize the end-to-end delay, the first forwarder that receives the packet will acknowledge the current transmission and be responsible for further forwarding. Thus, we do not overlook the forwarding chances to reduce the delay. On the other hand, if the sleep interval is too long, we can accordingly lower the EDQ condition of candidate forwarder selection, so as to utilize more chance to reduce the delay.

VI. Conclusion

In this paper, we propose L^2, a data forwarding technique tailored to energy-constrained low-duty-cycle WSNs. By using the window-based transmission with synchronized rendezvous, L^2 tames the bursty characteristic of wireless links and realizes precise link estimation under the context of duty-cycled communications. Based on such mechanisms, the dynamic forwarding of L^2 smartly schedules transmission of a packet to multiple receivers, thus achieving high network yield and low energy consumption simultaneously. In the future, we plan to carry out large-scale field test of L^2 and port it to different radio platforms.

References

[1] L. Mo, Y. He, Y. Liu, J. Zhao, S.-J. Tang, X.-Y. Li, and G. Dai, "Canopy closure estimates with GreenOrbs: Sustainable sensing in the forest," in *Proc. ACM SenSys*, Berkeley, CA, USA, 2009, pp. 99–112.

[2] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proc. USENIX OSDI*, Seattle, WA, USA, 2006, pp. 381–396.

[3] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proc. ACM SenSys*, Boulder, CO, USA, 2006, pp. 307–320.

[4] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, "Four-bit wireless link estimation," in *Proc. 6th Workshop Hot Topics Netw.*, Atlanta, GA, USA, 2007.

[5] A. Kamthe, M. A. Carreira-Perpiñán, and A. E. Cerpa, "M&M: Multi-level Markov model for wireless link simulations," in *Proc. ACM SenSys*, Berkeley, CA, USA, 2009, pp. 57–70.

[6] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *Proc. ACM SenSys*, Sydney, Australia, 2007, pp. 321–334.

[7] P. Dutta, S. Dawson-Haggerty, Y. Chen, C. J. . Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *Proc. ACM SenSys*, Zurich, Switzerland, 2010, pp. 1–14.

[8] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. ACM SenSys*, Berkeley, CA, USA, 2009, pp. 1–14.

[9] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. ACM SenSys*, Baltimore, MD, USA, 2004, pp. 95–107.

[10] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," in *Proc. ACM SenSys*, Boulder, CO, USA, 2006, pp. 321–334.

[11] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, "PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks," in *Proc. IEEE INFOCOM*, Shanghai, China, 2011, pp. 1305–1313.

[12] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The β-factor: Measuring wireless link burstiness," in *Proc. ACM SenSys*, Raleigh, NC, USA, 2008, pp. 29–42.

[13] M. H. Alizai, O. Landsiedel, J. Á. B. Link, S. Götz, and K. Wehrle, "Bursty traffic over bursty links," in *Proc. ACM SenSys*, Berkeley, CA, USA, 2009, pp. 71–84.

[14] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. ACM MobiCom*, San Diego, CA, USA, 2003, pp. 134–146.

[15] J. Kim, X. Lin, and N. B. Shroff, "Optimal anycast technique for delay-sensitive energy-constrained asynchronous sensor networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 2, pp. 484–497, Apr. 2011.

[16] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-MAC: A hybrid MAC for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 511–524, Jun. 2008.

[17] X. Wang, W. Huang, S. Wang, J. Zhang, and C. Hu, "Delay and capacity tradeoff analysis for motioncast," *IEEE/ACM Trans. Netw.*, vol. 19, no. 5, pp. 1354–1367, Oct. 2011.

[18] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "DESYNC: Self-organizing desynchronization and TDMA on wireless sensor networks," in *Proc. IEEE/ACM IPSN*, Cambridge, MA, USA, 2007, pp. 11–20.

[19] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proc. USENIX NSDI*, San Francisco, CA, USA, 2004, vol. 1, p. 2.

[20] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "Understanding the causes of packet delivery success and failure in dense wireless sensor networks," Stanford University, Stanford, CA, USA, Tech. Rep. SING-06-00, 2006.

[21] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," in *Proc. IEEE/ACM IPSN*, Beijing, China, 2012, pp. 185–196.

[22] T. Liu and A. E. Cerpa, "Foresee (4C): Wireless link prediction using link features," in *Proc. IEEE/ACM IPSN*, Chicago, IL, USA, 2011, pp. 294–305.

[23] T. Liu and A. E. Cerpa, "TALENT: Temporal adaptive link estimator with no training," in *Proc. ACM SenSys*, Toronto, ON, Canada, 2012, pp. 253–266.

[24] S. Chen, Y. Fang, and Y. Xia, "Lexicographic maxmin fairness for data collection in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 7, pp. 762–776, Jul. 2007.

Zhichao Cao (M'13) received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 2009, and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2013.

He is currently with the School of Software, TNLIST, Tsinghua University. His research interests include sensor networks, mobile networks, and pervasive computing.

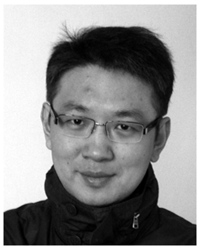
Dr. Cao is a member of the Association for Computing Machinery (ACM).

Yuan He (S'09–M'12) received the B.E. degree in computer science and technology from the University of Science and Technology of China, Hefei, China, in 2003, the M.E. degree in computer software and theory from the Chinese Academy of Sciences, Beijing, China, in 2006, and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2010.

He is now an Assistant Professor with the School of Software, Tsinghua University, Beijing, China, and the Vice Director of the IOT-Tech Center, Tsinghua National Lab for Information Science and Technology. His research interests include sensor networks, wireless networks, cloud computing, and pervasive computing.

Qiang Ma (M'13) received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 2009, and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2013.

He is currently with the School of Software, TNLIST, Tsinghua University. His research interests include sensor networking, network diagnosis and management, and mobile computing.



Yunhao Liu (M'02–SM'06) received the B.S. degree in automation from Tsinghua University, Beijing, China, in 1995, and the M.S. and Ph.D. degrees in computer science and engineering from Michigan State University, East Lansing, MI, USA, in 2003 and 2004, respectively.

He holds the Changjiang Chair Professorship with Tsinghua University.

Prof. Liu is a Senior Member of the IEEE Computer Society. He is currently an ACM Distinguished Speaker and Vice Chair for the ACM China Council. He is serving as the Associate Editor-in-Chief for the IEEE Transactions on Parallel and Distributed Systems and an Associate Editor for the IEEE/ACM Transactions on Networking and *ACM Transactions on Sensor Networks*.