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

[1]Zhichao Cao, [1,2]Yuan He, [1,2]Yunhao Liu
[1]Department of Computer Science and Engineering, HKUST
[2]MOE Key Lab for Information System Security, School of Software, TNLIST, Tsinghua University
caozc@cse.ust.hk, {he, yunhao}@greenorbs.com

Abstract—**Energy-constrained wireless sensor networks are duty-cycled, relaying on multi-hop forwarding to collect data packets. A forwarding scheme generally involves three design elements: media access, link estimation, and routing strategy. Most existing studies, however, focus only on a subset of those three. Disregarding the low duty cycle nature of media access often leads to over-estimate of link quality. Neglecting the characteristic of bursty loss over wireless links inevitably consumes much more energy than necessary and underutilizes wireless channels. The routing strategy, if not well tailored to the above 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 using multivariate Bernoulli link model. Further incorporated with synchronized rendezvous, L^2 enables sensor nodes to work in a lazy mode, keep their radios off as long as they can, and allocates the precious energy for only a limited number of promising transmissions. We implement L^2 on real sensor network testbeds. The results show that L^2 outperforms state-of-the-art approaches in terms of energy efficiency and network yield.**

I. INTRODUCTION

Wireless sensor networks (WSNs) [1] [2] [3] are mostly battery powered and are thus energy constrained. In order to save energy, sensor nodes are made duty-cycled and rely on multi-hop forwarding to deliver data packets to the sink.

Design of data forwarding mechanism is a crucial and challenging issue in low duty cycle WSNs. A widely adopted low duty-cycle protocol is X-MAC-UPMA [4] [5] (called X-MAC for short), 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, as is called Low Power Listening (LPL) [12]. LPL has fine performance under low data rates in sparse WSN deployment. Sending long preambles, however, overly consumes energy in communications, and underutilizes wireless channels [20] [21].

The preambles could actually be shortened, if there is a coordination mechanism between sender and receiver. One way to shorten the preamble length is to exert global time synchronization at the cost of extra overhead. Under such a scenario, the packet delivery ratio (PDR) becomes highly dependent on the synchronization accuracy. It is actually very difficult to set an ideal tradeoff between PDR and synchronization overhead.

Forwarding packets in the form of short preambles brings additional challenges on link estimation. Since multiple copies are transmitted continuously over a short period, it demands precise characterization of transient link quality instead of long-term average link quality. Unfortunately, the state-of-the-art schemes of link estimation, like 4-bit [6], do not well address this issue in the context of low duty cycle WSNs.

There are in fact promising opportunities when we take into account link burstiness. Link burstiness refers to the phenomenon that the success rate of packet transmissions over a link during a very short period is highly correlated with each other. The bursty loss might cause performance degradation if it is not carefully considered in the data forwarding mechanism.

In this paper, we propose L^2, a practical design of data forwarding in 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 [7] 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 coordinate multiple forwarders with different wake-up schedules. It helps to improve 2.4% PDR and save 15.3% energy 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 with some primary experimental observations and discussions. Section III elaborates on the design of L^2. We discuss the implementation issues and show the results of performance evaluation in Section IV. We conclude in Section V.

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) [10] for data collection. 4-bit [6] is utilized for link estimation. The transceiver of every node works in the asynchronous duty-cycled mode with X-MAC, the default media access mechanism in TinyOS 2.1.1. The performance from the view of both PDR and energy are examined and discussed in the following subsections.

A. Energy and Channel Utilization

The energy consumption spent on the transceiver usually dominates the energy consumption on a sensor node, which is

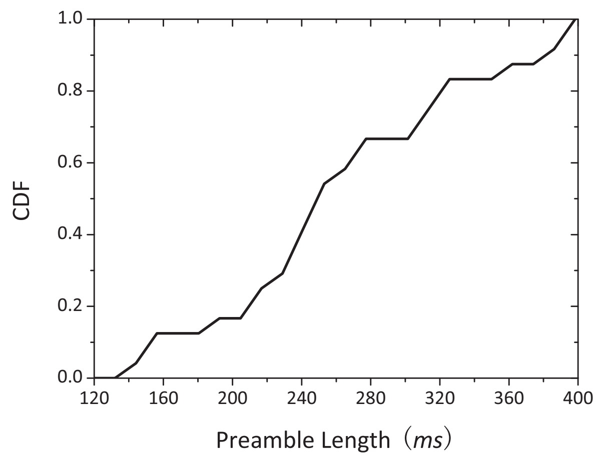


Figure 1. CDF of preamble length by hooking CTP, 4-bit, and X-MAC

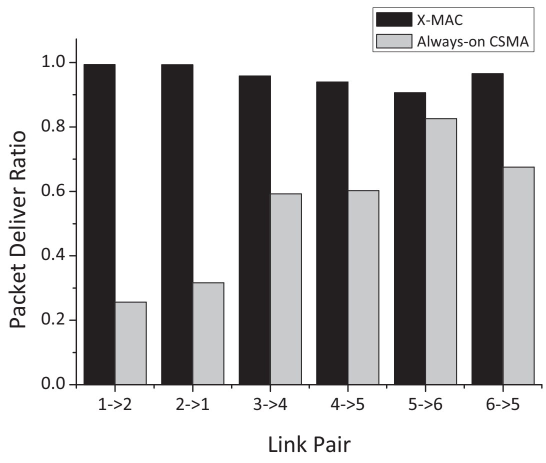


Figure 2. Comparison of the network layer PDR

measured by Quanto [11]. In the context of low power WSN, the preferable and practical media access mechanism should be flexibly adapted to different traffic pattern, energy-efficient with respect to control overhead, and easily scalable to different forms (sparse or dense) of deployments.

Due to the asynchronous nature, X-MAC or B-MAC [12] does not issue any control message. The coordination between the sender and receiver is thus loose. Consequently, in order to send a packet, a sender occupies the channel for half of a sleep interval in average.

We carry out a data collection experiment, in which we hook CTP, 4-bit and X-MAC directly with the default setting and use 25 TelosB nodes on a testbed. To verify the efficiency of channel utilization, each node individually calculates its preamble length in average. Figure 1 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 of a single packet transmission time ($\approx$4~10 *ms*). When the traffic load is high or the deployment becomes dense, frequent overhearing and contention probably further degrade the energy efficiency.

Some existing works, such as SCP [13], PW-MAC [14], and A-MAC, address the above problem. The usual ideas are based on synchronization and/or receiver-initiated transmission. Both time synchronization and channel probing induce extra control cost. Moreover, the efficacy of those approaches highly relies on the accuracy of time synchronization or the successful reception of channel probing. Most of those existing works, however, do not take into account the influence of link dynamics or traffic dynamics in real sensor networks. L^2 mitigates these cost and unreliability by means of piggybacking upper layer traffic, precise link estimation, and forwarding scheduling.

B. Link Estimation

4-bit link estimation treats the transmission of a network layer packet as an independent Bernoulli trial. While 4-bit considers link layer acknowledgement, it neglects the actual link layer behavior to transmit an upper layer packet. It exhibits quite different results from the comparisons between always-on CSMA and low power media access. Taking X-MAC as an example, when a network layer packet comes, the transceiver blindly sends preamble packets until the timer expires or an acknowledgement is received. The receiver might hear multiple preamble packets before it returns to sleep or issues the acknowledgement. Since the successful reception of any preamble packet implies that the packet is received, it is probable that the link quality is overestimated by X-MAC.

To verify the guess above, we carry out a controlled experiment using 3 pairs of TelosB nodes. We adjust the distance of each pair to ensure their link quality is different. Each node broadcasts 200 packets with 10 *ms* interval and calculates the PDR as the link quality. Figure 2 shows the result. The quality of all links is estimated to be better with X-MAC than that with always-on CSMA. Moreover, all PDR under X-MAC are higher than 90%. Misled by such overestimates of link quality, a node tends to make improper forwarding decisions.

Some works like β-factor [15], M&M model [7], and STLE [16] emphasize on the bursty characteristic of short-term link behavior. The observation of β-factor says that if the interval of two continuous transmissions is within 500 *ms*, the transmissions tend to be dependent. Meanwhile, note that the interval of adjacent preamble packets is at most 8 *ms*, which is the time spent to wait acknowledgement. Clearly the burstiness should be considered when characterizing the low duty-cycle link behavior. If the transmission over a bursty link experiences preamble packet loss, it is better to stop transmission immediately instead of continuing preambles.

STLE utilizes an overhearing scheme to estimate the short-term link behavior, but that scheme appears to be inappropriate when the wake-up schedule of nodes becomes diverse. Multi-level Markov model requires sufficient data for training and relatively long time for computation. In comparison, based on precise counting of the lost preamble packets, L^2 develops a practical online multivariate Bernoulli link model to depict the bursty loss on wireless links.

C. Forwarding

CTP selects a neighbor with the lowest transmission cost, measured in ETX [17]. In X-MAC, all nodes wake up periodically and their schedules are different from each other. When a transmission fails, the sender continues to occupy the channel and retransmits preambles. Neglecting the duty-cycle nature of media access, a deterministic scheme probably misses some

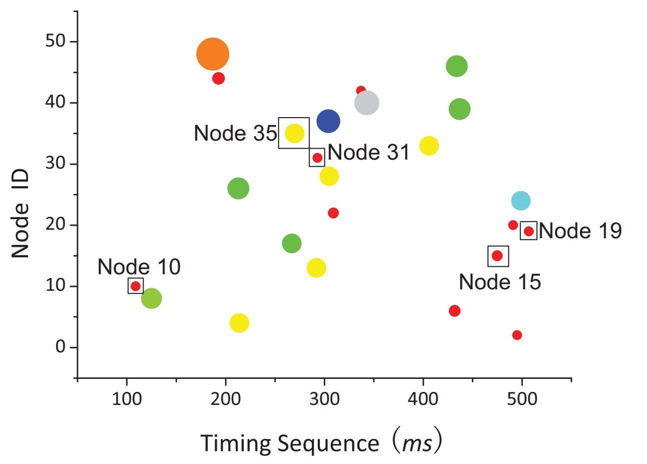


Figure 3. An illustrative example of the potential opportunity and benefit to forward a pakcet to multiple receivers

"early chances" to forward packets and thus underutilizes the channel.

Figure 3 takes a transient snapshot of the experiment in subsection 3-A and sorts all 25 nodes according to the interval between the current time and the next wake-up time of a node. A bigger circle denotes larger ETX. Nodes with the same color have the same parent. Let's take node 35 as an example. Its current parent is node 10. Node 31, 15, and 19, whose ETX is the same with node 10, 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.

The concept of dynamic forwarding has been proposed by DSF [8] and anycast techniques [18] to improve the energy efficiency, PDR, or end-to-end delay in low duty cycle WSNs. If the short term bursty loss of packets is taken into account, however, the energy cost distribution on multiple forwarders is likely to be inefficient. Moreover, some auxiliary mechanisms are often demanded, like FSTP [19] protocol adopted by DSF, which consume extra communication energy and ROM space. Based on a short-term link model, L^2 efficiently schedules forwarding to multiple receivers while enhancing the PDR. L^2 is thus more energy efficient than the existing dynamic forwarding protocols. There is not any extra cost or requirement to implement such scheduling.

III. Design

In this section, we first introduce the data collection network model, and then describe the adaptive window-based transmission model. After that we present the link estimation method, and then introduce the dynamic forwarding algorithm.

A. *Network Model and Assumptions*

We assume the data collection network is a static network with a single sink. All nodes generate data periodically and forward all data to sink via multi-hop routing. Besides the data packets, routing beacon is used to coordinate the nodes' behavior in a local area. Based on the information contained in the beacons, nodes realize neighbor discovery, link quality initialization, and local time synchronization.

We assume the transceiver of a node alternatively changes its state between dormant and active to save energy. Each node periodically wakes up to sense the channel, which is contrast to the fixed sleep interval in X-MAC. The period lengths of all nodes are identical and fixed. For broadcasting, a sender keeps transmitting the preambles for a whole sleep interval. For unicasting, with the low-cost local synchronization, a sender adopts window-based transmission to shorten the preamble length. The transmission is ended either the sender receives an acknowledgement or the preamble length exceeds an adaptive period (called *transmission window*). The larger the size of transmission window is, the more the energy is wasted when the busty loss occurs. As X-MAC does, if the receiver detects the channel is busy, it keeps awake for a predefined period (called *receiving window*). When the receiver receives a new packet, the receiving window is refreshed and restarts from the beginning.

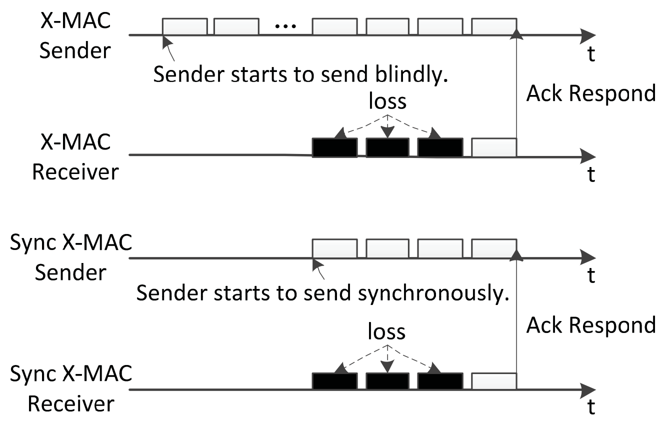


Figure 4. X-MAC vs. Synchronized X-MAC transmission and reception

B. *Window-based Unicast*

The basic design of the window-based unicast is shown in Figure 4. In contrast to the blind sending of X-MAC, the sender knows the time, at which the receiver wakes up. Thus it may start to send the preamble just a short while before the wake-up time of the receiver. 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 packets 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. In this way, one can avoid wasting energy on idle listening and useless sending when bursty loss occurs.

For locally synchronized rendezvous, a node aligns the wake-up schedules of other nodes to its local time. Specifically, as illustrated in Figure 5, for node *i*, $t_i(\bullet)$ indicates the most recent wake-up time of node *i*. For each neighbor *j* of node *i*, we keep a constant interval $offset_i(j)$ between $t_i(\bullet)$ and $t_j(\bullet)$, when *i* receives a routing beacon from *j*. According to $offset_i(j)$ and local wake-up schedule of node *i*, the wake-up schedule of any neighbor node *j* can be known by node *i*.

$offset_i(j)$ is calculated according to MAC layer timestamp [19], which is piggybacked on the routing beacon. As Figure 5 shows, the SFD (Start of Frame Delimiter) interrupt appears on both sender and receiver sides at the same time before the real

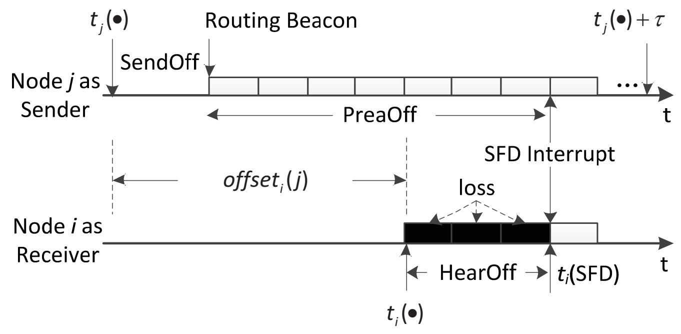


Figure 5. The way of local synchronization by MAC layer timestamp piggybacked on routing beacon

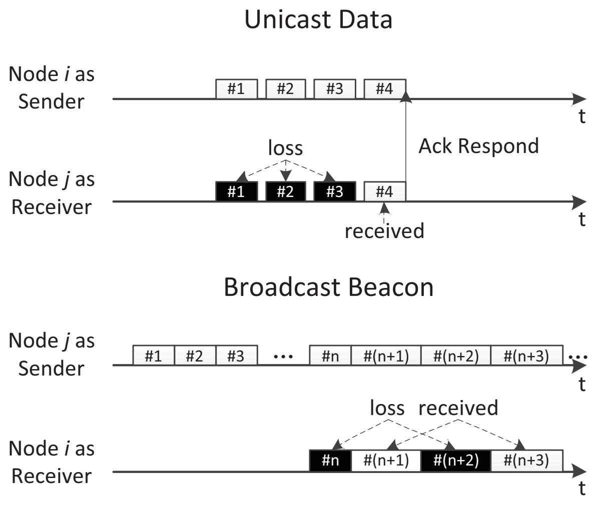


Figure 6. Link estimation based on window transmission

payload is transmitted. In the response of SFD interrupt, the sender adds *SendOff* and *PreaOff* into the payload and then begins to send. Assume the sleep interval of all nodes is τ. Then $offset_i(j)$ is calculated as follows:

$$offset_i(j) = (SendOff + PreaOff - HearOff) \bmod (\tau) \qquad (1)$$

We do not need extra communication cost for network-wide synchronization. Each node just makes use of the routing beacon to keep aware of its neighbors' wake-up schedules. In practice, however, the expected synchronized rendezvous often deviates from the actual one. One cause is the different clock drifts between a pair of nodes. Another cause is that the different processing speed or task queue length incurs the software skew, which makes the wake-up time fluctuate. Moreover, the difference in transceiver initiation processes introduces certain mismatch. We take the following methods to cope with those uncertainties.

By storing the 4 recent historical offset records, *Linear Regression* is used to calibrate the skew incurred by clock drift and software delay. Besides the transceiver initiation process, the sender needs extra time to load the payload into TX buffer of the transceiver, which usually takes several milliseconds. To ensure that the sender begins to send 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.

Besides the accuracy problem, 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 jammed timings. Thus we set 5 *ms* interval between any two consecutive preamble packets of routing beacons that contains the synchronization information.

For better scalability, 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 the correct time prediction of the events such as reboot, configuration update, and newly connected node.

Actually, there still exist a little bias in timing control of synchronized rendezvous, due to possible beacon loss and other unexpected factors. The subsequent schemes with L^2 design will resolve such limitations.

C. Link Estimation

L^2 counts both data traffic and beacons for link estimation. For accurate 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 *macdsn* (MAC layer data sequence number) as shown in Figure 6.

For unicast data, as mentioned in the previous subsection, the scope of *macdsn* is from 1 to TW_LENGTH. According to the *macdsn* of the acknowledged packet, the sender gets to know how many preamble packets are lost. For example in Figure 6, the first 3 preamble packets are lost.

For broadcast beacon, however, to ensure every neighbor can receive a beacon, the sender does not synchronize with the schedule of any neighbor. Thus 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 Figure 6, the n^{th} and the $(n+2)^{th}$ packets are lost, while the $(n+1)^{th}$ and the $(n+3)^{th}$ packets are received. The loss of the n^{th} packet (the actual first preamble packet that is intended to arrive at node *i*) might not be known by node *i*, because node *i* does not gets aware of the broadcast until the $(n+1)^{th}$ packet is successfully received. Thus the case of broadcast could not be counted perfectly with respect to link estimation.

Based on the above introduction, 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 preamble packet as an independent event as 4-bit does. 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 Figure 6, P=2/3. Due to the imperfect counting, P is larger than the actual probability 1/2. Nevertheless, that result is much more accurate than 4-bit, which yields an estimate of 1. Thus P is mainly utilized to initialize and update the link quality when there is little data traffic. Moving average is further introduced in to make the estimation adaptive to link dynamics.

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. The basic idea is transmissions of preamble packets respectively hold their own successful delivery probability. That model differs from the original single component multivariate Bernoulli model in that the delivery probability of packets is dependent with each other.

More specifically, a unicast transmission is terminated when the sender receives an acknowledgement or the timer exceeds the TW_LENGTH bound. We define the successful delivery of the k^{th} packet in the transmission window as: ***The previous k-1 preamble packets all fail to arrive at the receiver. The receiver receives the k^{th} preamble packet successfully and the sender also receives the acknowledgement of the k^{th} 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 data traffic.

Combining the above two link models together, we define P^k as the probability that at least one preamble packet could be successfully received, when the transmission window is k:

$$\begin{aligned} P^1 &= p_1 \\ P^k &= P^{k-1} + (1-P^{k-1})\bullet p^k \end{aligned} \tag{2}$$

When the p^k is not initialized (e.g. all preambles are acknowledged before the k^{th} packet), we set p^k as P. Then P^k reflects the PDR of network layer packet. Different transmission window size determines different network PDR and energy consumption. Accordingly, if a link suffers heavy bursty loss ($p^k \rightarrow 0$, while $k > 1$), the transmission window should be kept small to avoid useless retransmissions.

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 from the k^{th} to the TW_LENGTHth is successfully received while the previous (k-1) preamble packets fail to arrive at the receiver. We have

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

When RP^k is smaller than a preconfigured threshold (denoted by BURSTY_LOSS), it implies that the chance for the left (TW_LENGTH-k+1) preamble packets to arrive at the receiver is very low, indicating a potential bursty loss. In order to save energy while preserving PDR, it is wise to avoid transmitting those packets. In this way, a sender is able to determine an appropriate window size for preamble transmission.

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.

D. Dynamic Forwarding Scheme

Based on the discussion above, different sizes of the transmission window lead to different levels of energy consumption and PDR. The goal of L^2's forwarding scheme is to maximize PDR while keep the corresponding energy cost low. The basic idea is to distribute the total energy budget to multiple potential forwarders, which wake up before a bounded delay, so that the expected PDR is maximized.

For one hop delivery of a packet, we assume the energy budget and the delay constraint are predefined application-specific constant. Thus the hop count from a node to the sink denotes the maximum energy cost and end-to-end delay to deliver a packet to the sink. We define EDQ (*expected delivery quality*) as a routing metric. It is equal to the quotient of path EPDR (*expected packet delivery ratio*) and HOP (*hop count of the path*).

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

We use EDQ$_i$, EPDR$_i$ and HOP$_i$ to denote EDQ, EPDR and HOP of node i, respectively.

It is easy to see that a greater value of EDQ implies either a higher PDR or a lower energy cost and delay. The EDQ of a node can be calculated recursively. Specifically, for the sink, its EPDR is 1 and HOP is 0.

$$\text{EPDR}_{\text{sink}} = 1;\ \text{HOP}_{\text{sink}} = 0 \tag{5}$$

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 as k, these metrics are calculated as:

$$\begin{aligned} &\text{EPDR}_i(j) = \text{EPDR}_j \bullet \text{P}_i^k(j);\ \ \text{HOP}_i(j) = \text{HOP}_j + 1 \\ &\text{EDQ}_i(j) = \frac{\text{EPDR}_i(j)}{\text{HOP}_i(j)} = \frac{\text{EDQ}_j \bullet \text{P}_i^k(j)}{1 + 1/\text{HOP}_j} \end{aligned} \tag{6}$$

Let f_{best}^i denote the by-default selected best forwarder for node i. Based on the observation in Section II, a sender should involves all possible neighbors for forwarding a packet, as long as the candidate forwarder's EDQ is not less than the best forwarder. In this way, one can enhance the PDR while do not build up the energy cost or delay of packet forwarding.

Now it comes to introducing the algorithm of scheduling packet transmissions to multiple candidate forwarders. For node i, we assume a sending task happens at t_i. Node i first finds a set of candidate forwarders $FC_i = \{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}$, sorted in the order of the time offset (i.e. how soon the forwarder will wake up). The sizes of corresponding transmission windows of the forwarders are $\{l_1, l_2, \ldots, l_m\}$. We assume the energy budget and the single hop delay constraints are Ψ and Ω, respectively.

Based on the above discussion and the link model presented in the previous subsection, the expected PDR of dynamic forwarding via node i can be calculated as follows:

$$\text{EDQ}_i(\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}) = \frac{p_i^{l_1}(f_1^{t_i})\bullet \text{EDQ}_i(f_1^{t_i})}{\text{HOP}_{f_1^{t_i}} + 1} + (1 - p_i^{l_1}(f_1^{t_i}))\bullet \text{EDQ}_i(\{f_2^{t_i}, \ldots, f_m^{t_i}\}) \tag{7}$$

Then the optimization problem of dynamic forwarding is:

Maximum: EDQ$_i$($\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}$)

Subject to:

$$0 \le OFFSET(f_1^{t_i});$$

$$OFFSET(f_m^{t_i}) \le \Omega;$$
$$\textstyle\sum_1^m l_k \le \Psi;$$
$$\forall k = 1,2,...,m. \quad l_k \le \text{TW\_LENGTH};$$
$$\forall k = 1,2,...,m. \quad RP_i^{l_k}(f_k^{t_i}) \ge \text{BURSTY\_LOSS};$$
$$\forall k = 1,2,...,m. \quad \text{EDQ}_{f_{best}^i} \le \text{EDQ}_{f_k^{t_i}};$$
$$\forall k = 1,2,...,m. \quad p_i^0(f_k^{t_i}) = 0;$$

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

It can be proved that the above optimization problem is NP-hard. Given the limited computing resources in a WSN, we propose a greedy algorithm to achieve sub-optimal performance. Instead of optimizing the path PDR, the greedy algorithm optimizes the one hop PDR. In other words, we ignore the specific difference in the path delivery qualities among different candidate forwarders and focus on the one hop delivery quality. Given the selection criterion of candidate forwarders, such approximation does not degrade the overall PDR. We define the expected PDR of dynamic forwarding via node *i* using the greedy algorithm as follows:

$$\begin{aligned} &\text{EDQ}_i^*(\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\}) \\ &= p_i^{l_1}(f_1^{t_i}) + (1 - p_i^{l_1}(f_1^{t_i})) \bullet \text{EDQ}_i^*(\{f_2^{t_i}, \ldots, f_m^{t_i}\}) \end{aligned} \tag{8}$$

The greedy forwarder scheduling problem is formalized as:

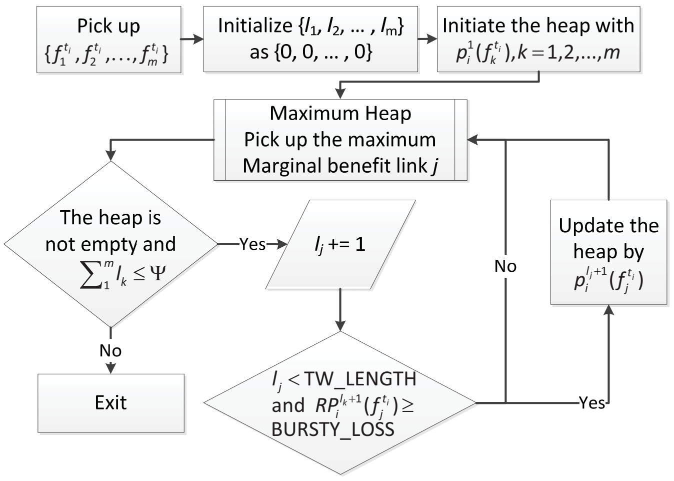
Maximum: $\text{EDQ}_i^*(\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\})$

Subject to:

$$0 \le OFFSET(f_1^{t_i});$$
$$OFFSET(f_m^{t_i}) \le \Omega;$$
$$\textstyle\sum_1^m l_k \le \Psi;$$
$$\forall k = 1,2,...,m. \quad l_k \le \text{TW\_LENGTH};$$
$$\forall k = 1,2,...,m. \quad RP_i^{l_k}(f_k^{t_i}) \ge \text{BURSTY\_LOSS};$$
$$\forall k = 1,2,...,m. \quad \text{EDQ}_{f_{best}^i} \le \text{EDQ}_{f_k^{t_i}};$$
$$\forall k = 1,2,...,m. \quad p_i^0(f_k^{t_i}) = 0;$$

In this scenario, $\text{EDQ}_i^*(\{f_1^{t_i}, f_2^{t_i}, \ldots, f_m^{t_i}\})$ 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 the more energy to the forwarder with the maximum marginal effect to increase the overall PDR.

Figure 7 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. At the beginning, the sizes of all transmission windows are initialized as 0 and the heap is initialized as the delivery probability of the first preamble packet of all links. Then, for link *j* picked up into the heap, the size of the transmission window l_j is added by 1. If the size of the transmission window over link *j* is smaller than TW_LENGTH and the delivery of sending more preamble packets is not bursty, we add the delivery probability of the next packet delivery into the heap. The process will not stop until there is no extra energy budget to allocate or the heap is empty. According to the determined schedule of forwarders, the link layer transmits the preamble packets. The temporal complexity of heap sorting is $O(\log m)$ and the size of heap is $O(m)$. Therefore the temporal and storage complexity of the greedy algorithm is $O(\Psi \log m)$ and $O(m)$.



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, … , 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 $RP_i^{l_k+1}(f_j^{t_i}) \ge$ BURSTY_LOSS
No
Yes
Update the heap by $p_i^{l_j+1}(f_j^{t_i})$

Figure 7. The process of greedy forwarder scheduling.

In practice, the wake-up times of different forwarders might be close to each other. Such forwarders should not be selected simultaneously into the candidate set. If one of them is the best forwarder, L^2 keeps it and ignores the others. Otherwise, L^2 just keeps the one with the maximum EDQ.

IV. Implementation and Evaluation

A. Implementation

We implement L^2 on TinyOS 2.1.1 system. The three main components of L^2 implementation are window-based transmission, multivariate Bernoulli link estimation, and dynamic forwarding scheme. To verify the efficacy and efficiency of each component, we compile 6 different programs with different combinations of functional components, as shown in Table I.

Deterministic L^2 implements the window-based transmission and Bernoulli link estimation model, but only the best forwarder is used to relay data packets. Dynamic L^2 further contains the dynamical forwarding scheme. For comparison, we also implement DSF and A-MAC. We carry out indoor (25 TelosB motes) and outdoor (40 TelosB motes) experiments for performance evaluation.

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 for different purposes, such as neighbor discovery, link estimation, network

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

state update, and local time synchronization. Multiplexing the broadcast beacon reduces the control overhead of L^2. The frequency of routing beacon is controlled by Trickle Timer [23]. Smart unicast is initiated by the dynamic forwarding scheme.

While the separation of broadcast and unicast brings much benefit to the forwarding performance, it also requires more memory storage. Table I indicates that the window-based transmission, which differs Sync X-MAC from X-MAC, costs 9292 bytes ROM and 937 bytes RAM space, respectively.

For the implementation of multivariate Bernoulli link model, besides except the original statistics for long term estimation, a node also maintains 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. As 4-bit does, when a node accumulates sufficient observations of a link, it updates the link estimation by keeping an exponential moving average.

For the implementation of dynamic forwarding, all nodes update their current best forwarder and path EDQ periodically. A node obtains its forwarding schedule using the greedy scheduling algorithm. As we mentioned in the previous section, temporal time and spatial complexity of the greedy algorithm is $O(\Psi\log m)$ 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 wakes up in the future Ω period.

B. Performance Evaluation

We set the sleep interval at 512ms as the same with X-MAC. All nodes (include the sink) execute such a sleep wake-up schedule. In the application layer, the period 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 as 1 to limit the transmission range and ensure multi-hop communication. We run each program for half hour. For each node, about 500 packets can be collected by the sink in that period.

1) Performance Overview

In order to evaluate the network yield and energy efficiency of different approaches, we compare their performance with respect to average PDR and radio-on duty cycle 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 deals with the bursty loss problem. DSF increases PDR, but due to bursty loss, the energy efficiency might be unsatisfactory. Based on such observations, dynamic L^2 achieves improved PDR and decreased energy consumption simultaneously.

TABLE II. PERFORMANCE COMPARISON OF DIFFERENT APPROACHES

Version	Average Performance				
	PDR	*Duty Cycle*	*Preamble Length*	*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

Table II summarizes the PDR and the radio-on duty cycle of different approaches. We can see that compared with X-MAC, the improvement of energy efficiency by dynamic L^2 is (13.4–7.2)/13.4=46.2%. Corresponding to our expectation, the PDR of deterministic L^2 is the same with that of Sync X-MAC, but the energy efficiency has been improved 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. These results demonstrate that the multivariate Bernoulli link model enables better energy efficiency than 4-bit does.

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. Deterministic forwarding schemes might not change a node's parent until it loses a number of packets. Even if packet loss is identified, there is not a mechanism to provide immediate back-up forwarder. The dynamic forwarding scheme resolves this problem as it exploits much more chances to find a synchronized forwarder and is adaptive to various uncertainties by flexibly rescheduling the energy allocation from the sender to multiple forwarders.

Compared with 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 sleep wake-up characteristic in low duty cycle WSNs than deterministic forwarding. Figure 8 shows the radio duty-cycle distribution of all nodes, which again demonstrates the performance advantage of dynamics L^2.

2) Channel Utilization

TABLE II also lists the preamble length, the contention per packet, and the ratio of the overheard packets to the

[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.

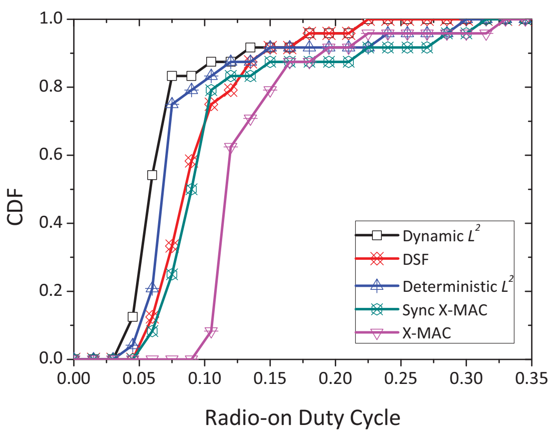


Figure 8. CDF of Duty Cycle

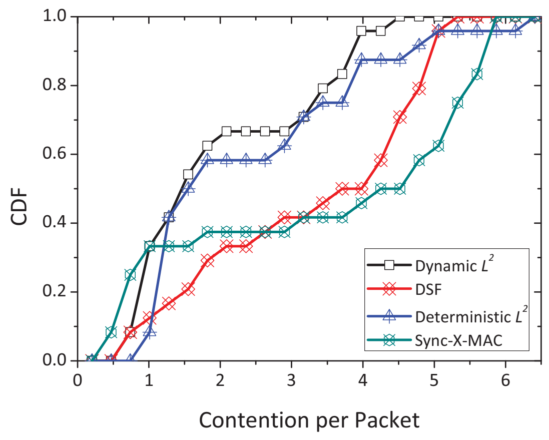


Figure 10. CDF of the contention per packet

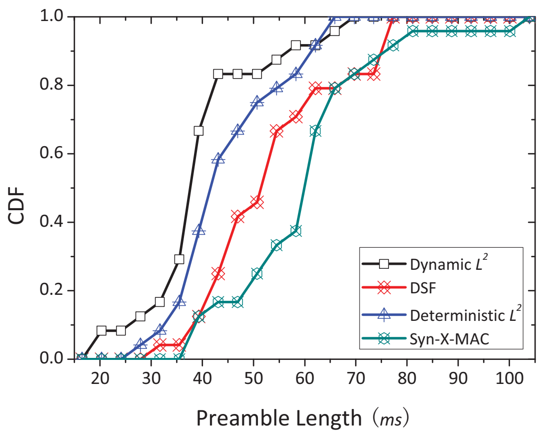


Figure 9. CDF of the average preamble length

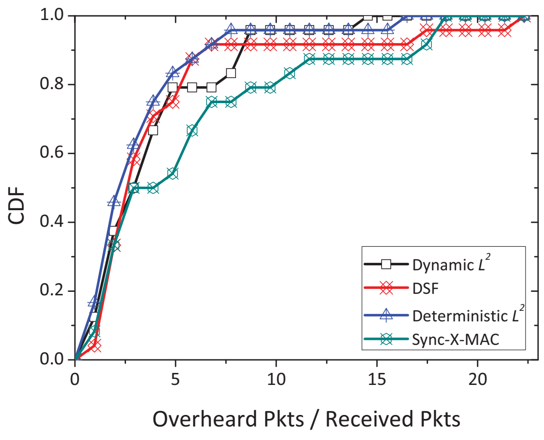


Figure 11. CDF of the ratio of the overheard packets to the received packets

TABLE III. 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 IV. COMPARISON OF THE PERFORMANCE WITH 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

received packets, which reflect the ability of different approaches with respect channel utilization.

The preamble length indicates the time of channel occupation for a single network packet transmission. 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 be further shortened by 25.6% and 27.6%, respectively. Figure 9 shows the CDF of preamble length of all 25 nodes. The results demonstrate that both the window-based transmission and the multivariate Bernoulli link model improve channel utilization.

Measures of contention indicate how crowded the channel is. Figure 10 shows the CDF of the contention per packet among all 25 nodes. Because transmissions with long preambles easily conflict with each other, we can see from Figures 9 and 10 that the contention is highly correlated with the preamble length. Moreover, the heavy contention often incurs unexpected bias of synchronization and collision so that the delivery reliability is decreased.

The dynamic forwarding scheme smartly allocates the energy to multiple forwarders. It optimizes the preamble length further and avoids the possible case that multiple nodes compete for the same forwarder in deterministic forwarding. These are validated by the results in Figures 9 and 10 that dynamic L^2 outperforms deterministic L^2 with respect to the preamble length and the chance of contention.

Figure 11 illustrates the CDF of the ratio of all overheard packets to the received packets. Ideally, the ratio should be smaller than 1, which means a node usually receives several packets after wake-up. However, we find the reality is that the overheard ratio of most nodes is greater than 1. Because overhearing is related to the diversity of wake-up schedules among the nodes in a local area, the result in Figure 11 suggests that if the nodes in a local area are well scheduled [22], the performance of forwarding can be further improved.

3) *Dynamic Forwarding*

As we discussed before, involving more forwarders is likely to increase the delivery reliability. Table III compares DSF and dynamic L^2 and indicates that L^2 makes more active use of potential forwarders than DSF in our relatively dense deployment. DSF selects the forwarder in a backtrack manner. When the best forwarder is selected, the other candidates waking up before the best can hardly be 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 considers the delivery quality of each neighbor. It involves the neighbors, who have comparable delivery quality with the best forwarder. Thus in the case of a dense deployment, there are usually many usable candidate forwarder. In a sparse deployment, however, dynamic L^2 is possibly degraded to deterministic L^2 as there are not many usable forwarders.

4) Performance Comparison with A-MAC

We deploy 40 nodes in the form of a relatively sparse 5×8 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*. Base on the result in [9], 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 one-hour experiments for both programs. The results are shown in Table IV.

Compared with A-MAC, dynamic L^2 improves PDR by 3.3% and reduces energy consumption by about 50%. A-MAC suffers higher probe sending frequency than dynamic L^2, which is due to the shorter sleep interval and leads to higher energy consumption. Reducing the sleep interval of A-MAC, however, probably degrades the delivery reliability.

V. Conclution

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.

Acknowledgement

This study is supported in part by NSFC Major Program 61190110, the NSFC program under Grant 61170213, National High-Tech R&D Program of China (863) under grant No. 2011AA010100, and the China Postdoctoral Science Foundation Grant No. 2011M500019.

Reference

[1] L. Mo et al., “Canopy closure estimates with greenorbs: Sustainable sensing in the forest”, in *Proceedings of ACM SenSys,* Berkeley, California, USA, 2009.

[2] W. Z. Song, R. Huang, M. Xu, B. A. Shirazi, and R. LaHusen, “Design and deployment of sensor network for real-time high-fidelity volcano monitoring”, *IEEE TPDS*, vol. 21, no. 11, pp. 1658–1674, 2010.

[3] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, “Fidelity and yield in a volcano monitoring sensor network”, in *Proceedings of USENIX OSDI,* Seattle, WA, USA, 2006.

[4] M. Buettner, G. V. Yee, E. Anderson, and R. Han, “X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks”, in *Proceedings of ACM SenSys,* Boulder, Colorado, USA, 2006.

[5] K. Klues, G. Xing, and C. Lu, “Towards a unified radio power management architecture for wireless sensor networks”, in *In Proceedings of the 1st International Workshop on Wireless Sensor Network Architecture,* Cambridge, Massachusetts, USA, 2007.

[6] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, “Four-bit wireless link estimation”, in *Proceedings of the 6th Workshop on Hot Topics in Networks,* Atlanta, Georgia, USA, 2007.

[7] A. Kamthe, M. A. Carreira-Perpinán, and A. E. Cerpa, “M&M: multi-level Markov model for wireless link simulations”, in *Proceedings of ACM SenSys,* Berkeley, California, USA, 2009.

[8] Y. Gu and T. He, “Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links”, in *Proceedings of ACM SenSys,* Sydney, Australia, 2007.

[9] 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 *Proceedings of ACM SenSys,* Zurich, Switzerland, 2010.

[10] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, “Collection tree protocol”, in *Proceedings of ACM SenSys,* Berkeley, California, USA, 2009.

[11] R. Fonseca, P. Dutta, P. Levis, and I. Stoica, “Quanto: Tracking energy in networked embedded systems”, in *Proceedings of USENIX OSDI,* San Diego, CA, USA, 2008.

[12] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks”, in *Proceedings of ACM SenSys,* Baltimore, Maryland, *USA*, 2004.

[13] W. Ye, F. Silva, and J. Heidemann, “Ultra-low duty cycle MAC with scheduled channel polling”, in *Proceedings of ACM SenSys,* Boulder, Colorado, USA, 2006.

[14] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, “PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks”, in *Proceedings of IEEE INFOCOM,* Shanghai, China, 2011.

[15] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, “The β-factor: measuring wireless link burstiness”, in *Proceedings of ACM SenSys,* Raleigh, North Carolina, USA, 2008.

[16] M. H. Alizai, O. Landsiedel, J. Á. B. Link, S. Götz, and K. Wehrle, “Bursty traffic over bursty links”, in *Proceedings of ACM SenSys,* Berkeley, California, USA, 2009.

[17] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing”, in *Proceedings of ACM MobiCom,* San Diego, CA, USA, 2003.

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

[19] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, “The flooding time synchronization protocol”, in *Proceedings of ACM SenSys*, Baltimore, Maryland, *USA*, 2004.

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

[21] X. Wang, W. Huang, S. Wang, J. Zhang, and C. Hu, “Delay and Capacity Tradeoff Analysis for MotionCast”, *IEEE/ACM TON*, vol. 19, no. 99, pp. 1354–1367, 2011.

[22] J. Degesys, I. Rose, A. Patel, and R. Nagpal, “DESYNC: self-organizing desynchronization and TDMA on wireless sensor networks”, in *Proceedings of IEEE/ACM IPSN*, Cambridge, Massachusetts, USA, 2007.

[23] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks”, in *Proceedings of USENIX NSDI*, San Francisco, California, USA, 2004.