Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links

Shuo Guo, *Student Member, IEEE,* Liang He, *Member, IEEE,* Yu Gu, *Member, IEEE,* Bo Jiang, *Student Member, IEEE,* and Tian He*, *Member, IEEE*

Abstract—Flooding service has been investigated extensively in wireless networks to efficiently disseminate network-wide commands, configurations and code binaries. However, little work has been done on low-duty-cycle wireless sensor networks in which nodes stay asleep most of the time and wake up asynchronously. In this type of network, a broadcasting packet is rarely received by multiple nodes simultaneously, a unique constraining feature that makes existing solutions unsuitable. In this paper, we introduce Opportunistic Flooding, a novel design tailored for low-duty-cycle networks with unreliable wireless links and predetermined working schedules. Starting with an energy-optimal tree structure, probabilistic forwarding decisions are made at each sender based on the delay distribution of next-hop receivers. Only *opportunistically early* packets are forwarded via links outside the tree to reduce the flooding delay and redundancy in transmission. We further propose a forwarder selection method to alleviate the hidden terminal problem and a link-quality-based backoff method to resolve simultaneous forwarding operations. We show by extensive simulations and test-bed implementations that Opportunistic Flooding is close to the optimal performance achievable by oracle flooding designs. Compared with Improved Traditional Flooding, our design achieves significantly shorter flooding delay while consuming only $20\% \sim 60\%$ of the transmission energy.

Index Terms—Wireless Sensor Networks, Low-Duty-Cycle Networks, Flooding.

1 Introduction

Wireless Sensor Networks have been used for many applications such as military surveillance [12], infrastructure protection [47] and scientific exploration [36]. Miniaturized into a cubic centimeter package and deployed without wired power supply, sensor nodes have very limited amount of energy. On the other hand, there is a growing need for sustainable deployment of sensor systems [40], [47], [52], [53] to reduce operational cost and ensure service continuity. To bridge the gap between limited energy supplies and application lifetimes, a sensor network has to be operated on very low duty cycles, i.e., a sensor node is active for only a short period of time between two long dormant periods. In order to deliver a packet, a sender may have to wait for a certain period of time (termed *sleep latency* [8]) until its receiver becomes active. Sleep latency degrades the performance (e.g., delay and energy consumption) of various kinds of data forwarding designs in low-duty-cycle networks. While pioneering projects have been proposed for low-duty-cycle unicasts [8], [25], [50], research is surprisingly inadequate for low-duty-cycle flooding, an important function for disseminating network-wide commands, alerts and configurations [12], time synchronization [26], and code binaries [14]. Intended for fast network-wide data dissemination, a flooding design should be not only reliable to reach every node and keep them updated and consistent, but also time efficient with less energy cost.

There are two features that make flooding in low-duty-cycle networks challenging. First, a packet is unlikely to be received by multiple nodes simultaneously as in always-awake networks. To broadcast a packet, a sender has to transmit the same packet multiple times if its receivers do not wake up at the same time. Thus flooding in such networks is realized essentially by multiple unicasts. Second, unlike wired networks, wireless communication is notoriously unreliable [49]. A transmission is repeated if the previous transmissions are not successful due to wireless loss. The combination of low-duty-cycle operation and unreliable links necessitates the design of a different flooding mechanism than those found in wired networks and always-awake wireless networks.

One straightforward solution could be building a routing tree for flooding. Yet this type of solutions have been shown [17], [30] to be fragile, because the failure of a parent node prevents all its subtree nodes from receiving messages, even if the network is still connected. Furthermore, existing tree-based solutions could be made energy efficient only at the cost of long delays, as they only forward packets via a single route.

This work introduces Opportunistic Flooding: a flooding method specially designed for low-duty-cycle wireless sensor networks. Its main objective is to *reduce redundancy in transmission while achieving fast dissemination*. Our solution inherits the reliable nature of traditional flooding, allowing packets to travel along multiple paths. The key novelty of this work lies in the *forwarding decision making*. A node forwards a packet with a higher probability if the packet arrives *opportunistically earlier*, such that flooding packets are always delivered along fast paths. Specifically, our contributions are as follows:

- To the best of our knowledge, this is the first network design for flooding in sensor networks with extremely low duty cycles and unreliable channels.
- This work proposes *delay-driven opportunistic forwarding*.

- *Corresponding author.
- *Shuo Guo is now with Arista Networks. L. He and Y. Gu are with Singapore University of Technology and Design, Singapore. Bo Jiang is with University of Massachusetts Amherst. Tian He is with University of Minnesota-Twin City, USA.*
- *An early version of this work is published at ACM MobiCom'09.*

We propose a recursive and distributed method to compute the probability mass function (*pmf*) of forwarding delays at each node along an energy-optimal tree. The computed *pmf* is then used as the guideline in forwarding decision making to reduce the flooding delay opportunistically.

- To alleviate the hidden terminal problems without the hefty RTS/CTS overhead, we propose a forwarder selection method that allows forwarding nodes with good link quality to overhear each other. We also propose a link-quality-based backoff method to resolve simultaneous transmissions among forwarding nodes.

The rest is organized as follows: Section 2 describes the motivation behind the work. Section 3 defines the network model and assumptions. Section 4 introduces our main design, and Section 5 discusses practical issues followed by their evaluation in Sections 6 and 7. Section 8 discusses the related work and Section 9 concludes the paper.

2 Motivation

To bridge the gap between lifetime requirements of sensor applications [40], [52] and the limited availability of energy through fixed-budget batteries or energy harvesting [24], it is critical to have an energy-efficient communication architecture. This section identifies the need for low-duty-cycle communication designs in general and the flooding design in particular.

2.1 The Need for Low-Duty-Cycle Operation

Typically, the energy used in communication can be optimized through (i) physical-layer transmission rate scaling [43]; (ii) link-layer optimization for connectivity, reliability, and stability [33]; (iii) network-layer enhancement for forwarders and routes [3], [46]; and (iv) application-layer improvements for both content-agnostic and content-centric data aggregation and inference [11], [28]. Although these solutions are highly diverse, they all assume a network in which nodes are *ready to receive* packets and focus mainly on the transmission side.

In contrast, wireless networks with *intermittent receivers* have captured little attention, despite the fact that communication energy is consumed mostly by being ready for potential incoming packets, a problem commonly referred to as *idle listening*. For example, the widely adopted CC2420 radio [41] draws 19.7mA while receiving or idle listening, which is larger than the 17.4mA for transmitting. More importantly, packet transmission time is usually very brief (e.g., 1.3 ms to transmit a TinyOS packet using a CC2420 radio), while the duration of idle listening can be orders of magnitude longer. For example, most environmental applications, such as Great Duck Island [40] and Redwood Forest [42], sample the environment at relatively low rates (on the order of minutes between samples). With a comparable current draw and a 3∼4 orders of magnitude longer duration waiting for reception, idle listening is a major energy drain that accounts for the communication energy cost. To reduce the energy penalty in idle listening, a node has to run at a low-duty-cycle state and turn off its radio most of time.

2.2 The Need for a New Flooding Design

The traditional flooding method and many of its advanced versions [14], [21], [23], [29], [38] have shown their good performance in terms of delivery ratio, delay and energy cost in many always-awake networks. These solutions, however, suffer severe performance degradation if directly applied to low-duty-cycle networks. In those designs, a node starts broadcasting a packet as soon as it receives it from its previous-hop node. In a low-duty-cycle network where two neighbors seldom wake up at the same time, a broadcasting packet cannot be received by many nodes simultaneously. The delivery ratio becomes even worse when unreliable links and collisions are taken into account. We conducted a series of simulations by decreasing the duty-cycle from 100% to 1% in a randomly generated network with 200 nodes. The working schedules are randomly generated, and we count the percentage of nodes that can receive a broadcasting packet using a pure flooding scheme, i.e., a node broadcasts upon first receiving a packet. Fig.1 shows how the performance degrades as the duty-cycle decreases. Even under ideal conditions (i.e., no collisions and perfect links), only 5% of the packets are successfully delivered in a 2% duty cycle network, a clear indication that traditional methods are not suitable for low-duty-cycle networks, if used directly.

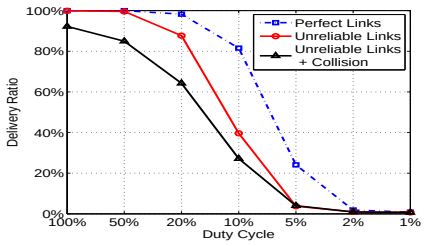


Fig. 1. Traditional Flooding in Low-Duty-Cycle Networks

One could argue that traditional flooding methods can be adapted to low-duty-cycle networks by permitting (i) multiple transmissions of the same packet based on the neighbor schedules, and (ii) ARQ-based mechanism to deal with unreliable links, but this still has problems. First, it suffers from a high energy cost due to collisions. When a node wakes up, many of its neighbors attempt to start transmissions simultaneously. With unreliable links [33], [49], it is difficult to resolve collisions, since the nodes cannot sense each other's transmissions. Second, even without collisions, the number of redundant transmissions is large especially when the network density is high. Due to these limitations in traditional flooding methods, it is necessary to have a tailored design for low-duty-cycle networks, which motivated our work in the present paper.

3 Preliminaries

This section defines the network model and assumptions related to our opportunistic flooding design.

3.1 Network Model

Consider a sensor network where each node has two possible states: an active state and a dormant state. An active node is able to sense an event, transmit a packet or receive a packet. A dormant node turns off all its function modules except a

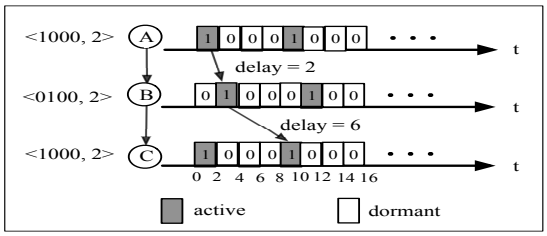


Fig. 2. A Low-Duty-Cycle Network Model

timer to wake itself up. All nodes have their own working schedules. These schedules are shared with neighboring nodes and are normally asynchronous in order to reduce information redundancy among temporally correlated sensing data [10], [45]. A dormant node wakes up when (i) it is scheduled to switch to the receiving state, or (ii) it has packets to send to an active neighbor. *In short, a node can transmit a packet at any time, but can only receive a packet when it is active.*

For sensing purposes, the working schedules of sensor nodes are normally periodic [12], [40], [42]. Without loss of generality, suppose T is the working period of the whole network (e.g, T can be any common multiple of the periods of all nodes). T can be further divided into a number of time units of length τ where τ is appropriate for a round-trip time. Then each node picks one or more time units as its active state, in compliance with its duty cycle. Denote the active and dormant states by '1' and '0', respectively. The ith node's working schedule can then be represented as $< w_i, \tau >$, where w_i is a string of '1's and '0's denoting the schedule and τ is the length of each time slot. In a low-duty-cycle network, we can compress the representation of w_i by keeping only the offset values of active states.

Fig.2 shows an example of our network model. In this example, T is 8 time units and is divided into 4 time slots, each of which is 2 time units long ($i.e., \tau = 2$). Nodes A and C with schedule $< 1000, 2 >$ are active during the first 2 time units and dormant during the next 6 time units; node B with schedule $< 0100, 2 >$ is active during the second 2 time units and dormant for the rest of the period. All nodes periodically change their states based on their predetermined working schedules. Suppose node A has a packet to send to B at time 0. Since a node can only receive a packet when it is active, node A has to wait until time 2 to start the transmission. Similarly, to transmit the packet to C, node B has to wait until time 8. If both links are perfect, the total delay of this packet from A to C is 4 time slots, and thus the delay is $4\tau = 8$ time units.

3.2 Assumptions

Suppose the source nodes have flooding packets to be sent throughout the whole network. We make several assumptions as follows:

1) As the traffic intensity is typically light in low-duty-cycle networks, we simply our consideration by focusing on the scenario where only one flooding will be in process at any time [6].
2) A node sets up its working schedule $< w_i, \tau >$ and shares it with all its neighbors when joining the network, a process normally referred as low-duty-cycle rendezvous [5]. After rendezvous, a node knows the schedules of all its one-hop neighbors. A node changes its schedule only after the new schedule is shared to all its neighbors.
3) We assume the existence of unreliable links and collision. We define the *link quality* of a wireless link as the probability that the transmission through this link is successful. Link quality can be measured using probe-based methods in [3], [46] or through low-cost piggybacking on regular data traffic. We assume a relatively stable environment in which link qualities can be updated infrequently (e.g., every ten minutes). We will show in Section 6 that our design also works in the presence of mild fluctuations in link qualities. If two or more ongoing transmissions are within the communication range, a collision occurs and none of them will succeed. Note that collisions can be sometimes resolved by the *capture effect* [20]. However, even with the capture effect, simultaneous transmissions still consume extra energy since only one useful packet out of multiple transmissions is delivered. We try to avoid simultaneous transmissions for the sake of energy efficiency and do not consider capture effect in our design.
4) The network is locally synchronized so that each node knows when to send packets to its neighbors. Local synchronization can be achieved by using a MAC-layer time stamping technique, as described in FTSP [26], which achieves an accuracy of 2.24μs with the cost of exchanging a few bytes of packets among neighboring nodes every 15 minutes. Since τ is typically between $2,000\mu$s to $20,000\mu$s, the accuracy of 2.24μs is sufficient.
5) An hop count is used to denote the minimum number of hops from a node to the source. Hop count can be easily obtained by letting each node broadcast its own hop count as soon as it is calculated or updated. Initially, the source node has a hop count of 0. It broadcasts this information and all its neighbors have a hop count of 1. Similarly, the neighbors of hop-1 nodes that have not yet been assigned a hop count have a hop count of 2, and so on. To ensure the logical network topology is loop free, we simplify our consideration by focussing on the scenario that a node only forwards a flooding packet to nodes with larger hop counts. However, the proposed opportunistic flooding is applicable to any logical network topology. We also compare the opportunistic flooding with the energy-optimal and delay-optimal schemes in Section 6, and the results show that the proposed opportunistic flooding achieves a well-balanced and near-optimal flooding performance in terms of both the flooding delay and energy cost.

3.3 Performance Metrics

In this paper, we define the following two performance metrics to evaluate the performance of a flooding design.

1) **Flooding delay**: defined as the time elapsed from a message being sent out by the source until it reaches 99% of the nodes in the network. Due to the imperfection of the links, the flooding delay exhibits inherent randomness.

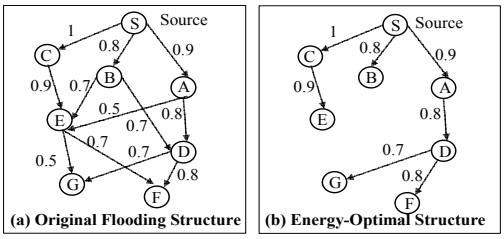


Fig. 3. DAG-Based Flooding Structure

Here we propose to use the average flooding delay as a measure of network performance.

2) **Energy consumption**: measured by the total number of transmissions initiated by a flooding packet from the source. The receiver-side energy is determined by their predefined working schedules, which are not changed by flooding designs. Therefore, we use only the sender-side energy as the performance metric when comparing different flooding designs with the same duty-cycled schedules.

4 Main Design

We present the design of the Opportunistic Flooding in this section.

4.1 Design Overview

Based on the network model, a flooding packet can only be forwarded from nodes with smaller hop counts to those with larger ones. As a result, the flooding structure of the network is a directed acyclic graph (DAG) as shown in Fig.3(a). The weights of the directed edges are the corresponding link qualities. A tree structure can be obtained from the DAG by retaining for each node only the incoming edge with the highest weight, as shown in Fig.3(b). As we will show later, this tree is the energy optimal for flooding among all trees compatible with the DAG. In other words, if a flooding packet is forwarded according to this tree, (i.e., a node only receives packets from its parent), the expected total number of transmissions is minimized.

We observe, however, that flooding via the energy-optimal tree may result in a long flooding delay, since the parent of a node in the energy-optimal tree may not receive a specific packet as early as its other parents in the DAG, due to the opportunistic nature of wireless communication. Based on this observation, the key idea of opportunistic flooding is to *utilize links outside an energy-optimal tree if transmissions via these links have a high chance of making the receiving node receive the packet "statistically earlier" than its parent on the energy-optimal tree*. As a summary, the proposed opportunistic flooding is applied on top of the energy-optimal tree and can further improve the flooding performance in terms of the flooding delay.

The flooding structure of Opportunistic Flooding is dynamically changing. A node forwards a flooding packet to a next-hop node if and only if this transmission is expected to deliver a new packet to that node, instead of an old one. The packet to be forwarded *opportunistically* should be *statistically earlier* than the packet that would otherwise be delivered via the energy-optimal tree. In order to forward opportunistically early packets while avoiding late ones, opportunistic flooding consists of three major steps, as illustrated in Fig.4:

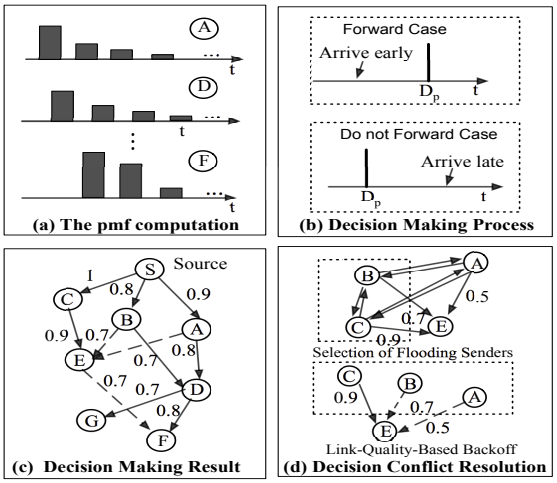


Fig. 4. Design Overview of Opportunistic Flooding

1) **The *pmf* Computation**: Due to unreliable links, the delay of a flooding packet arriving at each node from its parent in the energy-optimal tree is a random variable. As shown in Fig.4(a), the probability mass function (*pmf*) of this delay for each node is first derived to guide the decision making process. From the *pmf*, each node computes its p-quantile delay as the statistically significant threshold and shares it with all its previous-hop nodes.
2) **Decision Making Process**: As shown in Fig.4(b), a packet is forwarded opportunistically via the links outside the energy-optimal tree only if this forwarding can significantly reduce the delay (Note that the p-quantile delay is used to control the statistical significance). Specifically, a node makes its forwarding decision locally based on three inputs: (i) the receiving time of the flooding packet, (ii) the link quality between itself and the next-hop node, and (iii) the p-quantile. Fig.4(c) shows one example of the final structure of decision making. This structure is dynamically changed for different flooding packets.
3) **Decision Conflict Resolution**: With the distributed decision process, multiple nodes may decide to forward the same packet to a common neighbor, which is called *decision conflict*. Conflict resolution techniques are designed to avoid collisions and save energy further (Fig. 4(d)).

With dynamic decisions per packet, Opportunistic Flooding permits a packet to travel along an opportunistically-faster route instead of a fixed one via the energy-optimal tree. Also, late packets are not forwarded to reduce redundancy and save energy. The design details are given in the following subsections.

4.2 Flooding Energy Cost and Delay

The flooding process can be evaluated from two aspects: the energy consumption and the delay.

4.2.1 About Energy Optimality

In a low-duty-cycle network, the probability that a node has two neighbors with identical working schedules is very low. For example, in a network with a 5% duty-cycle, the working period

is divided into 20 time units and each node randomly chooses one of them as an active state. The probability that two nodes will choose the same active time slot is only 5%. As a result, flooding in low-duty-cycle networks is essentially realized by a number of unicasts. Normally a node needs to forward a packet to its neighbors (with a larger hop count) one-by-one due to their different working schedules.

As discussed in 4.1, an energy-optimal tree is constructed based on the DAG by retaining for each node only the incoming edge with the best link quality. When the flooding process is strictly realized by unicasts, the energy-optimality of this tree among all trees compatible with the DAG can be easily proved by contradiction: in an energy-optimal tree, if there exists a node whose incoming link does not have the best link quality among all incoming links in the DAG, then given that the network has a low duty cycle and flooding is realized by a number of unicasts, the new tree obtained by replacing this link with the best one is more energy efficient, which contradicts with the assumption on its energy optimality. Note that if multiple nodes wake up simultaneously, the energy-optimal tree obtained will deviate from the actual flooding structure with the highest energy efficiency. In this case, identifying such an optimal flooding structure is equivalent to finding a Minimum Connected Dominating Set given the nodes deployment, which is proven to be NP-hard [4]. However, since the multiple-receiver scenario is rare in low-duty-cycle networks, the aforementioned flooding tree still achieves a good approximation of energy optimality.

4.2.2 About Delay Optimality

Flooding delay is another critical metric to evaluate the flooding performance. In the ideal case where all the links are perfect, a delay-optimal tree in the DAG can be identified based on solely the scheduled active slots of sensor nodes. However, with the unreliable links in practice, the delay-optimal flooding structure cannot be a tree anymore. Figure 5 (which is part of the DAG in Fig. 3) shows a simple demonstration on the non-existence of the delay-optimal tree. Let us consider the case that for a flooding process starts at time 0, E and D receive the packet at time t, and F will wake up at time instances $t+4, t+8, \cdots$. Either link $D \rightarrow F$ or $E \rightarrow F$ will be adopted if the delay-optimal tree exists. When the former link is adopted, the expected delay for F to receive the packet is

$$t + 4 \times 0.8 + 8 \times (1 - 0.8) \times 0.8 + \cdots = t + 4.99,$$

and if the latter is adopted, the expected delay is

$$t + 4 \times 0.7 + 8 \times (1 - 0.7) \times 0.7 + \cdots = t + 5.71.$$

However, if both of the links are adopted, the expected delay would be

$$\begin{aligned} & t + 4 \times (1 - (1 - 0.8)(1 - 0.7)) \\ & + 8 \times (1 - 0.8)(1 - 0.7) \times (1 - (1 - 0.8)(1 - 0.7)) \\ & + \cdots = t + 4.26, \end{aligned}$$

which is smaller than both the above cases. This means to achieve the minimal delay, more links should be adopted in the flooding process, and thus more transmissions are needed, which on the other hand increases the energy consumption of sensor nodes.

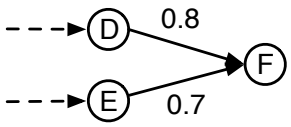


Fig. 5. Delay-optimal flooding structure cannot be a tree.

Thus with unreliable links, the energy-optimal tree may result in long flooding delay, while the delay-optimal flooding structure will introduce more energy cost. The proposed opportunistic flooding *improves the flooding process based on the energy-optimal tree, and establishes a balance between the flooding delay and the energy cost.*

4.3 The Delay *pmf* of the Energy-Optimal Tree

This section computes the packet delay distribution (*pmf*) when packets are forwarded according to the energy-optimal tree.

4.3.1 The Computation of pmf

Given an energy-optimal tree, the flooding packet delay of each node is a random variable due to unreliable links. In order to guide the decision making process of neighboring nodes, it is important to calculate the distribution of the delay. We call a node with hop count l a level-l node. Suppose the ith active time unit of a level-l node is $t_l(i)$, the packet delay *pmf* is denoted by a set of tuples $\{(t_l(i),\ p_l(i))\}$, where $p_l(i)$ is the probability of receiving the packet at time $t_l(i)$.

The *pmf* computation process starts from the level-0 node (the source) and propagates through the network level by level. Initially, the source is always awake and the probability that it receives the packet with delay 0 is 100%, i.e., the *pmf* of the source is $\{(0,100\%)\}$. Then a level-1 node calculates its *pmf* based on the *pmf* of the level-0 parent according to the energy-optimal tree. Similarly, a level-$(l+1)$ node calculates its *pmf* based on that of its level-l parent. Given the *pmf* of this level-l node ($t_l(i)$ and $p_l(i)$ for any i), and its level-(l+1) child with active time units $t_{l+1}(j)$ for any j, we calculate the probability that it receives the flooding packet at its jth active time slot as

$$p_{l+1}(j) = \sum_{i:t_l(i)<t_{l+1}(j)} p_l(i) q (1-q)^{n_{ij}}, \qquad (1)$$

where q is the link quality satisfying $q \in (0, 1]$, n_{ij} is the number of the level-$(l+1)$ node's active time units between $t_l(i)$ and $t_{l+1}(j)$. $p_l(i)q(1-q)^{n_{ij}}$ is the probability that the packet which arrives at the level-l node at its ith active time unit is first delivered to the level-$(l+1)$ node at its jth time unit. Clearly, a node's *pmf* can be derived from its parent's *pmf* recursively, with initial *pmf* (0,100%) at the source.

Fig.6 shows an example of the *pmf* computation process, where node A computes its *pmf* first based on the link quality 0.9 and its own working schedule. The probability that node A receives the packet for the first time at time 10 is 0.9. At time 20, the probability becomes $(1-0.9) \times 0.9 = 0.09$. Then node D computes its *pmf* based on A's *pmf*. For node D at time 15, the probability is the multiplication of the link quality and the

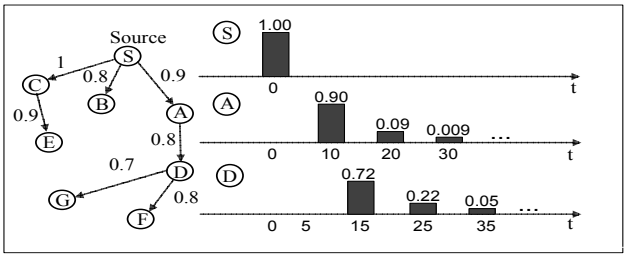


Fig. 6. The *pmf* Computation

probability that node A receives the packet at time 10, which is $0.9 \times 0.8 = 0.72$. For node D, at time 25, the probability is the sum of the probability that (i) node A receives the packet at time 10 and succeeds at the second transmission, and (ii) node A receives the packet at time 20 and succeeds in the first transmission, which is $0.9 \times (1-0.8) \times 0.8 + 0.09 \times 0.8 = 0.22$. Similarly, all the nodes within the network compute their *pmf*s once the *pmf*s of their parents become available.

4.3.2 Complexity Analysis

At each node, the number of possible delay values equals the number of entries to be calculated in the *pmf* computation. Theoretically, the delay *pmf* may have infinitely many entries. However, we can accurately approximate the *pmf* by including first M entries, so that the cumulative probability of the rest entries is less than a small value (i.e., 1%). For example, in Fig.6, node A's *pmf* contains only two entries: (10, 0.9) and (20,0.09). In this case, $M = 2$.

Eq. (1) takes quadratic time $O(M^2)$. However, linear time is achievable with the following recursive formulation

$$\begin{aligned} p_{l+1}(j) &= \sum_{i:t_l(i)<t_{l+1}(j)} p_l(i)q(1-q)^{n_{ij}} \\ &= \sum_{i:t_l(i)<t_{l+1}(j-1)} p_l(i)q(1-q)^{n_{i,j-1}}(1-q) \\ &\quad + \sum_{i:t_{l+1}(j-1)\le t_l(i)<t_{l+1}(j)} p_l(i)q \\ &= p_{l+1}(j-1)(1-q) + \sum_{i:t_{l+1}(j-1)\le t_l(i)<t_{l+1}(j)} p_l(i)q. \end{aligned} \tag{2}$$

For example, the probability for D to receive the packet at time 35 (Fig. 6) is calculated with (2) as $0.22 \times (1-0.8) + 0.009 \times 0.8 = 0.0512 \approx 0.05$.

Each node needs only its parent's *pmf* to complete the computation, which amounts to just a few packets. The calculation is repeated when the link qualities are updated. We discuss how frequently the link quality needs to be updated in Section 5.2.

4.4 Decision Making Process

As discussed in Section 4.1, only opportunistically early packets are forwarded to reduce flooding delay. Upon receiving a flooding packet, a node judges if its transmission to a next-hop node could make the node receive the packet for the first time with a high probability. If so, such a transmission helps reduce the flooding delay and is considered *Needed*. Otherwise it only consumes more energy and is considered *Redundant*.

A node finds its p-quantile delay based on its *pmf*, which is referred as D_p, and shares it with its parents. By definition, if a flooding packet arrives at this node later than D_p, the probability that this packet has been already received by this node from its parent is greater than p.

Suppose a level-l node A receives a packet at its ith active time unit with delay $t_l(i)$ and intends to make a forwarding decision toward one of its level-$(l+1)$ neighbors B with active units $t_{l+1}(j)$s (A is not the parent of B on the energy-optimal tree). A computes the expected packet delay (*EPD*) at B if A forwards the packet to B. Specifically, if A transmits to B, the *EPD* from A to B can be computed using the following equation

$$EPD = \sum_{j:t_{l+1}(j)>t_l(i)} t_{l+1}(j)q(1-q)^{n_{ij}}, \tag{3}$$

where q is the link quality, n_{ij} is the number of B's active time units between $t_l(i)$ and $t_{l+1}(j)$. Eq. (3) is essentially the sum of a geometric series, which can be calculated with a closed form. To reduce the amount of floating-point computation, an alternative way is to make use of the expected number of transmissions. For a link with link quality q, $\lceil \frac{1}{q} \rceil$ transmissions are expected for a successful packet delivery. Thus A finds in B's working schedule the $\lceil \frac{1}{q} \rceil$th active time slot after $t_l(i)$ (the time that this packet arrives at A) and takes it as B's *EPD*.

After the *EPD* is computed, A compares this value with B's delay threshold D_p to decide if this transmission (from A to B) is opportunistically needed. If *EPD*$\le D_p$, the probability that B has already received this flooding packet via the energy-optimal tree is no greater than p. So, this packet is considered *Needed*. If *EPD*$> D_p$, the chance that B has already received this packet is more than p, and this packet is considered *Redundant*. Thus A will not forwarded it to B.

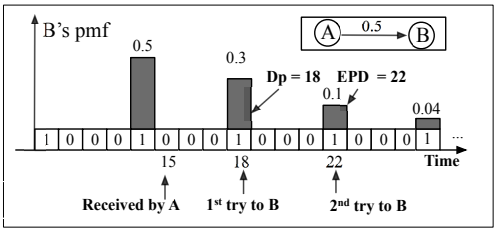


Fig. 7. An example of Decision Making

Fig. 7 shows an example of the decision process with $p = 0.8$. Denoting the *pmf* of node B at its k'th active slot as $p_l^B(k)$, the p-quantile delay D_p of B is calculated by first identifying its earliest active slot till which the packet will be received by B with probability no smaller than p

$$i = \min\{j : \sum_{k=1}^{j} p_l^B(k) \ge p\}, \tag{4}$$

then D_p is assigned as the time of B's ith active slot.

In the example shown in Fig. 7, because the first two active slots of B achieves a receiving probability of $0.5 + 0.3 = 0.8$, and the second active slot is at time 18, thus $D_p = 18$. During the initialization, B shares the D_p value with A. Suppose a packet arrives at node A at time 15. Since the link quality is 0.5, A is expected to transmit twice to forward the packet to

B successfully. The first try is at time 18 (which is the first active time unit of B after time 15) and the second try is at time 22. Therefore, the *EPD* from A to B is 22. Since $EPD = 22 > D_p = 18$, this packet is considered *Redundant* and is not forwarded to B.

Note that p is a control parameter for balancing the delay and energy cost. With a larger value of p, more packets are likely to be considered as *Needed* and hence forwarded opportunistically. This increases the chance of fast delivery, provided that collisions are handled appropriately. It also increases the number of transmissions, leading to higher energy cost. On the other hand, as p becomes smaller, fewer packets are forwarded opportunistically via energy-suboptimal links, which improves the energy efficiency but increases the delay. Clearly, the value of p strikes a balance between delay and energy, which we will evaluate in Section 6.

4.5 Decision Conflict Resolution

This subsection deals with the conflicts when two or more nodes transmit to the same node simultaneously.

4.5.1 The Selection of Flooding Senders

In wireless communication, a certain percentage of collisions are caused by the Hidden Terminal Problem, which is more likely to occur in a low-duty-cycle network, because all transmissions to a receiver are limited to the small time window when it is active. If the hidden terminal problem occurs, both senders will keep sending but neither of them will succeed.

One possible solution is to use TDMA-based approaches to schedule the transmissions of different senders at different time slots to avoid collisions. However, due to the unpredictable wireless loss along the multi-hop links from the source to each node, the time each sender receives the flooding packet is highly dynamic, making TDMA–based approaches less efficient. Another possible solution is to use the RTS/CTS control packets as they are in CSMA/CA. However, adding control packets into every transmission is very costly, especially when the hidden terminal problem occurs infrequently under low traffic loads.

The key idea of our solution is to select a reduced sender set for each node, so that all sending nodes can hear each other to avoid the hidden terminal problem. We use a link quality threshold l_{th} to determine whether a link is good or not. All links between the selected senders should have a link quality better than l_{th}. The selection process goes as follows: First, a node only receives flooding packets from nodes with smaller hop counts. These nodes are the candidates for the flooding senders, and are sorted in descending order of their link qualities. The candidate with the best link quality is always included in the sender set, and the rest are selected inductively. Consider the best candidate that has not yet been tested. If the link qualities between this candidate and all the currently selected senders are better than l_{th}, this candidate is added to the sender set; otherwise, this link is disabled. All the candidates are tested one-by-one in descending order of their link qualities.

Denote H as the total number of sender candidates. In the worst case, a time of $O(H)$ is needed to check whether each candidate should be added to the sender set. For each checking, we need to examine the link qualities between all the nodes that have been already added to the sender set and the candidate node under consideration, which implies another $O(H)$ time in the worst case. Thus the computation complexity to construct the sender set is $O(H^2)$. The impacts of l_{th} on both the flooding delay and energy cost will be evaluated in Section 6.

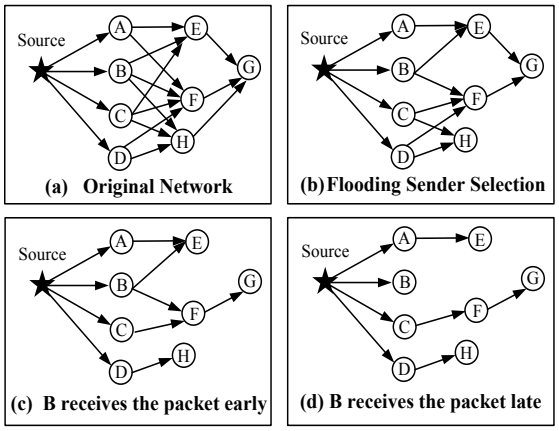


Fig. 8. Different Flooding Structures

4.5.2 Link-Quality-Based Backoff

Once a sender set is formed, we need to resolve the conflicts within the set. We propose a link-quality-based backoff scheme that not only resolves collisions but also reduces redundant transmissions to save energy.

Ideally, a node with better link quality should have a higher priority to grab the channel and start a transmission earlier. Suppose the backoff time bound is $T_{backoff}$ and the maximum size of the sender set is W. $T_{backoff}$ is divided into W slots for different backoff durations. A sender computes its backoff duration $t_{backoff}$ as

$$t_{backoff} = (\lfloor W(1-q) \rfloor)\frac{T_{backoff}}{W} + X, \tag{5}$$

where q is the link quality and X is a random period of time generated from $[-\frac{T_{backoff}}{W}, +\frac{T_{backoff}}{W}]$ if $1 \leq \lfloor W(1-q) \rfloor \leq W-1$ and from $[0, +\frac{T_{backoff}}{W}]$ if $\lfloor W(1-q) \rfloor = 0$. This ensures $t_{backoff}$ is non-negative and within the backoff bound. Introducing such randomness into this equation reduces the chance of collision when two or more nodes have the same link quality.

When multiple nodes within the communication range have the same packet to send to the same node, they back off first before transmission and the one with the best link quality starts first. If this transmission is detected by other nodes, they will abort their own transmissions and mark them as *Redundant*. To ensure the transmission from the best-link node is detectable by other nodes, the best-link node can keep occupying the channel after its sending of packet is finished until the current time slot is passed. This will not increase the energy consumption of the best-link sender significantly because the energy consumption rates when it is in active and transmitting states are comparable to each other, e.g., the current draw of a MicaZ mote at active and transmitting states are 8 mA and 11–17.4 mA, respectively [41].

Fig. 8 shows an example of Opportunistic Flooding. The original DAG of the network is shown in (a). After sender selection, all nodes in the same sender set should have good enough links between them, and the result is shown in (b), with some links deleted from (a). For the flooding of a specific packet, (c) and (d) present the two cases on the forwarding decision of node B. If the B receives the packet early enough such that its forwarding to E and F is *Needed*, B will forward the packet to E and F, as shown in (c). On the other hand, if B receives the packet too late, it will not forward the packet to neither E or F, because these forwarding is *Redundant*, as shown in (d).

5 Practical Issues

In this section we discuss practical issues that could affect the performance of Opportunistic Flooding.

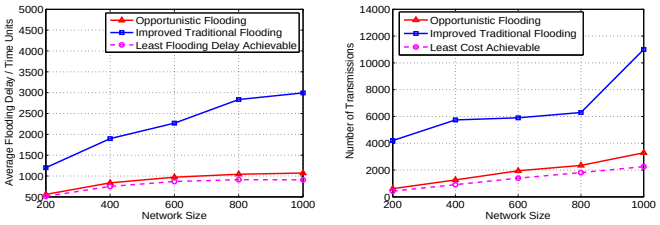
5.1 On Node Failures

In real world deployments, a sensor node can fail due to many factors such as physical damage or energy depletion. A robust flooding design should be insensitive to node failures and minor topological changes. In Opportunistic Flooding, flooding packets are forwarded through a dynamically changing structure with redundant links where the corresponding senders make the same decisions to send. The failure of an opportunistic flooding sender only results in a larger delay due to lower chances for the receivers to get "early packets". Even if its parent in the energy optimal tree fails, a node still has a high chance to receive an opportunistically early packet from other senders, thus avoiding cascading failures as in tree-based designs. We evaluate the impact of node failures in Section 6.6.

5.2 On Link Quality Change

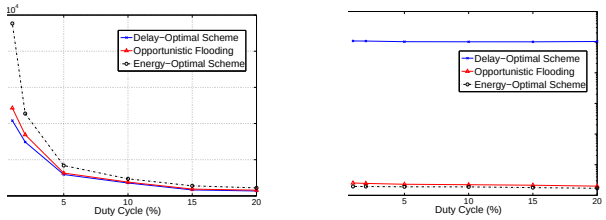
Link quality plays an important role in Opportunistic Flooding as it is a required input in almost every step of the design. It is thus preferable that the qualities of all the links do not change once they are measured. In practice, however, link quality is affected by many environmental factors and changes over time even during the interval between two measurements. Thus it is important to discuss if Opportunistic Flooding is still suitable for networks with out-of-date link quality information.

Due to the periodic measurements, the link quality may deviate slightly from the latest measured value. This deviation will possibly lead to two consequences: the loss of optimality of the energy-optimal tree (which further affects the accuracy of D_p), and an *EPD* deviating from its accurate value. However, the impact of both on Opportunistic Flooding is limited, because D_p and *EPD* only affect the decision making process. Given that the link quality has only a limited deviation, the changes of these two values are small. Thus, only a limited number of nodes will make wrong decisions, and thus either reducing the chance of receiving "early packets" or increasing the chance of sending redundant packets. We evaluate the impact of link quality change in Section 6.6.



(a) Flooding Delay vs. Network Size (b) Energy Cost vs. Network Size

Fig. 9. Network Size.



(a) Flooding Delay (b) Energy Cost

Fig. 11. Comparison with optimal schemes.

6 Evaluation

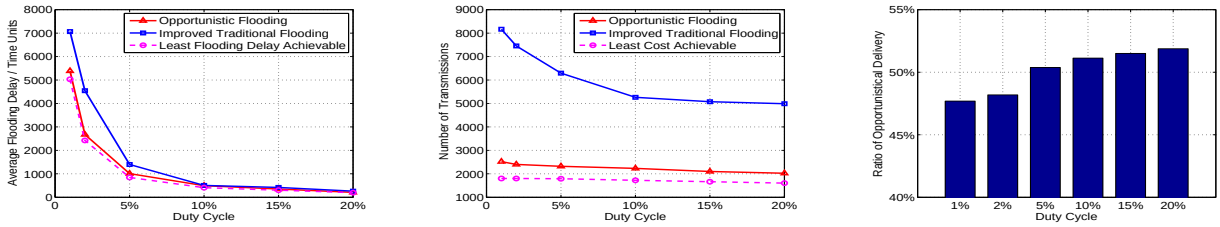
This section evaluates the performance of Opportunistic Flooding. Specifically, we compare the flooding delay and the energy cost in Opportunistic Flooding with those of traditional flooding. We also show that the performance of Opportunistic Flooding is very close to the optimal that is achievable by any flooding design. In Section 7, we provide further evaluation through a physical testbed experiment.

6.1 Simulation Setup

The networks used for the simulation are randomly generated with the size varying from 200 nodes to 1000 nodes. The links between these nodes are wireless path loss channels with shadowing effects, and the link qualities are calculated as in [54]. Unless explicitly specified otherwise, the parameters are $l_{th} = 0.7$ for sender selection and $p = 0.9$ for decision making. The flooding delay is based on a 99% delivery ratio instead of 100% to eliminate the effect of extremely low-degree nodes in a randomly generated network. All the nodes pick their active time units randomly. (Note that random schedules are for evaluation purpose only. Opportunistic Flooding works regardless of how working schedules are set up.) Since the performance is not affected by the actual length of a time unit, we measure the flooding delay by the total number of time units in all simulations. Simulation results are the averaged out of 10 network topologies, each with 1000 flooding packets.

6.2 Baseline I: Optimal Performance Bounds

We compare Opportunistic Flooding with the best performance achievable by any possible design. The optimal energy costs (the one with the least number of transmissions) is achieved when flooding packets are forwarded via the energy optimal tree. The optimal flooding delay (the one with the least flooding delay) is achieved by pure flooding with an oracle collision-free media access control. Note that the optimal energy and delay are achieved by two different methods, neither of which can achieve both the optimal delay and the optimal energy simultaneously.



(a) Flooding Delay vs. Duty Cycle (b) Energy Cost vs. Duty Cycle (c) Opportunistic Delivery Ratio

Fig. 10. Flooding Performance in Networks with Different Duty Cycles

6.3 Baseline II: Improved Traditional Flooding

To our knowledge, there is no distributed flooding algorithm specially designed for low-duty-cycle networks. Besides the optimal performance bounds described above, we also compare the performance of Opportunistic Flooding with a variation of traditional flooding. Recall from Section 2 that the performance of traditional flooding deteriorates dramatically when the duty cycle of a network is significantly reduced.

To make the comparison fair, we modify the traditional flooding method to improve its efficiency for low-duty-cycle networks and refer to it as Improved Traditional Flooding. First, it uses the same link-quality-based backoff method as Opportunistic Flooding to avoid collisions among multiple senders. Second, a node stops sending to a certain neighbor after hearing the transmission of another node. This greatly reduces the number of redundant transmissions. Third, the hidden terminal problem is alleviated by using a p-persistent backoff scheme after a fixed number of trials. These three techniques enable Improved Traditional Flooding to resolve a greater percentage of collisions, reduce energy cost, and recover from the hidden terminal problem quickly.

6.4 Performance Comparison

This section compares Opportunistic Flooding with optimal performance bounds and Improved Traditional Flooding. Due to the aggregated link dynamics over multiple hops, the flooding delays of these methods have a noticeable variance. For the sake of clarity, simulation figures only plot the average performance over multiple runs. In the test-bed evaluation, we experience less dynamics (i.e., less fluctuation on link quality over multiple hops), which allows us to plot both average and variance without reducing the figure clarity.

6.4.1 Different Network Sizes

We also evaluate the performance of Opportunistic Flooding in different network sizes as shown in Fig.9. For different network sizes from 200 to 1000, the side length of the area changes from $200m$ to $400m$ to keep a similar density.

In Fig.9, the average flooding delay and energy cost increase as the network size increases, as expected. Again, Opportunistic Flooding outperforms Improved Traditional Flooding and saves about 40% of flooding delay and 50% of energy cost. It is very close to the optimal performance, with around 10% more delay and energy cost.

6.4.2 Different Duty Cycles

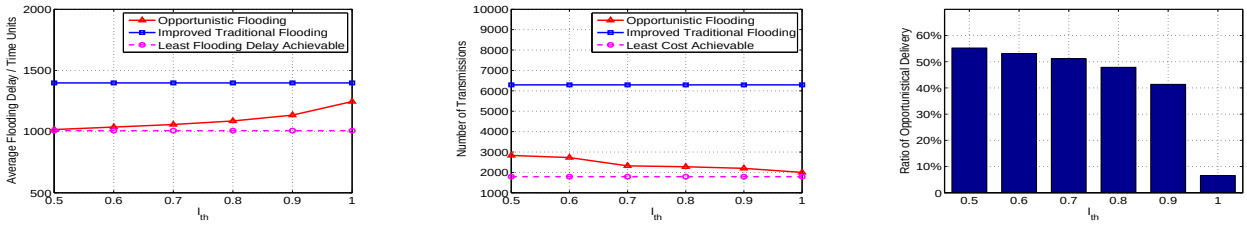
We first evaluate the performance in networks with different duty cycles, where 800 nodes are randomly deployed in a $300m \times 300m$ field. Fig.10(a) and (b) plot the flooding delay and energy cost of Opportunistic Flooding, Improved Traditional Flooding and optimal solutions. In Fig.10(a), the average flooding delay of opportunistic flooding is only around 80% of that of Improved Traditional Flooding and is very close to the optimal solution. In Fig.10(b), Opportunistic Flooding costs less than 50% of Improved Traditional Flooding while providing a shorter flooding delay. Compared with the optimal-energy solution, the number of redundant transmissions is around 400, which means that in the network consisting of 800 nodes, a node receives on average only 0.5 redundant packets for every broadcasting packet. Fig.10(c) shows the percentage of nodes whose first flooding packets are opportunistically early packets (i.e., not from its parent in the energy optimal tree), from which we can see that around 50% of packets are delivered opportunistically, significantly reducing the delay compared to Improved Traditional Flooding. We observe that this ratio increases as the duty cycle increases. This is because the probability that a node has more than one active neighbor is higher in a network with a higher duty cycle, and an opportunistically early packet is more likely to be received by more nodes.

6.4.3 Comparison with Optimal Schemes

Next we compare the performance of the Opportunistic Flooding with the energy-optimal and delay-optimal schemes (Fig. 11). We can see that the performance of Opportunistic Flooding is quite close to the respective optimal scheme, e.g., about $1.07\times$ and $1.2\times$ of the minimal achievable flooding delay and energy cost, respectively, when the duty cycle is 5%. Also note that both flooding delays under the energy-optimal scheme and energy costs under the delay-optimal scheme are significantly larger than that achieved by Opportunistic Flooding. Thus we can see that Opportunistic Flooding achieves a well-balanced and near-optimal flooding performance in terms of both the flooding delay and energy cost.

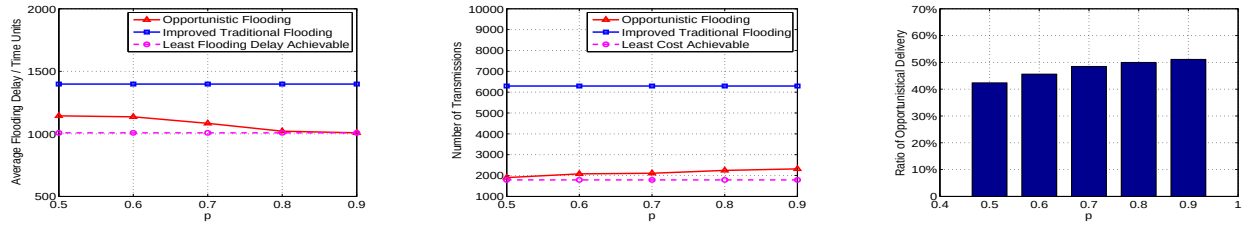
6.5 Investigation on System Parameters

In Opportunistic Flooding, we use two parameters (i.e., l_{th} and p) to control the tradeoff between delay and energy. This section addresses how to choose appropriate values for these parameters under different user requirements.



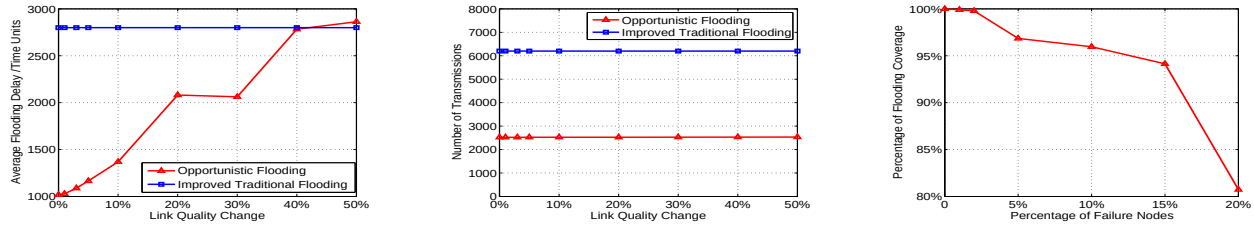
(a) Flooding Delay vs. l_{th} (b) Energy Cost vs. l_{th} (c) Opportunistic Delivery Ratio

Fig. 12. Flooding Performance in Networks with Different l_{th}



(a) Flooding Delay vs. p (b) Energy Cost vs. p (c) Opportunistic Delivery Ratio

Fig. 13. Flooding Performance in Networks with Different p



(a) Flooding Delay vs. Link Quality (b) Energy Cost vs. Link Quality (c) Coverage Ratio vs. Node Failures

Fig. 14. Evaluation of Practical Issues

6.5.1 *The Sender Set Link Quality Threshold* l_{th}

We study the impact of link quality threshold l_{th} used to build a sender set. Again we use the 800-node network which is randomly generated on a $300m \times 300m$ field with a 5% duty-cycle. p is still fixed to 0.9 while l_{th} is changed from 0.5 to 1.0. (To guarantee delivery, a node's best incoming link is always selected as flooding sender, even if it is no greater than l_{th}.)

Fig.12 plots the performance comparisons under different l_{th}. As l_{th} increases, the requirement for flooding sender selection becomes higher and fewer nodes are included in the sender set, leading to less opportunistic forwarding. This is validated by Fig.12 where we observe an increasing flooding delay, decreasing energy cost and decreasing opportunistical delivery ratio as l_{th} becomes larger. There is a significant change when l_{th} goes from 0.9 to 1 compared to the slow change when l_{th} goes from 0.5 to 0.9. This is because $l_{th} = 1$ requires that all senders have a 100% link quality, which is too strict to allow enough opportunistic packets and provide better performance.

Generally, l_{th} can be set to any value from 0.5 to 0.9, a trade-off between the flooding delay and the energy cost. If a shorter flooding delay is preferred, l_{th} can be set to 0.5 and 0.6. If energy is the most important issue, a greater l_{th} in the range of 0.8 to 0.9 is the best choice.

6.5.2 *The Quantile Probability* p

We study the impact of p, the threshold to decide whether a packet is an "early packet". Again, 800-node randomly generated networks are used. This time, l_{th} is fixed to 0.7 while p is changed from 0.5 to 0.9. Fig.13 (a), (b) and (c) plot the average flooding delay, the energy cost and the opportunistic delivery ratio, respectively. As p increases, more nodes make the decision to start transmissions so that a shorter delay and larger number of transmissions can be expected. Similarly, the choice of p is a trade-off between delay and cost. If a shorter delay is more important, a larger p of 0.8 or 0.9 is needed. (Recall that in Fig.10, the flooding delay when $p = 0.9$ is very close to the optimal delay, which means $p = 0.9$ can almost satisfy all delay requirements.) On the other hand, if a lower energy cost is more important, p can be as low as 0.5 or 0.6.

Based on all the comparisons, we conclude that Opportunistic Flooding approaches the optimal bounds. It outperforms Improved Traditional Flooding and saves flooding delay significantly while consuming only 20% to 60% transmission energy in almost all network settings. It could also fit different design requirements by choosing different values for its parameters.

6.6 Evaluation of Practical Issues

We inject noise into link qualities to simulate the scenario in which a flooding packet experiences different link qualities from the ones that are measured and shared in the latest update. We want to see how much the performance is affected when the link quality information known by each node is out of date. Fig.14(a) and (b) show the flooding performance with different link quality

variation ranges. As link qualities deviate from measurements, the flooding delay increases. This is because there are more nodes making wrong forwarding decisions and fewer nodes can receive "early packets". The energy cost slightly increases. We next explain why the energy cost has only a limited change in contrast with the flooding delay. Recall that a node compares EPD with D_p to make forwarding decisions, and EPD is affected by (i) the time that a flooding packet reaches this node and (ii) the link quality to the next-hop node. Since link qualities are periodically updated, both Dp and the link quality to the next-hop node remain the same in this node's computation. As a result, only the time that the packet reaches this node affects the decisions made by this node, and this is further affected by the link dynamics from the source to this node. With a random variation on link qualities, this node has roughly equal probability to receive the packet early or late, making the total number of decisions on "early packets" and the total energy cost almost unchanged. Compare Fig.14(a) and (b) we find that with inaccurate link quality information, more packets are sent as "redundant" ones as they consume energy and increase the chance of collisions without helping reduce the flooding delay at all. However, with a reasonable changing range of 30%, Opportunistic Flooding still outperforms Improved Traditional Flooding in terms of both flooding delay and energy cost.

Fig.14(c) shows the flooding coverage for different percentage of node failures. To better understand the number of nodes that fail to receive the broadcasting packet caused by the flooding design, the percentage of coverage is calculated as the number of nodes that receive the flooding packets divided by the total number of nodes subtracting the number of failure nodes. As seen in the figure, as the number of failure nodes increases, the flooding coverage decreases as expected. However, for a failure rate of no more than 10%, the flooding coverage is more than 96%, which indicates that Opportunistic Flooding is very robust to node failures and minor topology changes.

6.7 Overhead Analysis

We evaluate the overhead of Opportunistic Flooding in this subsection. In the simulation, link qualities are measured by exchanging 10 hello messages among neighbors. Since *pmf*s are computed based only on the *pmf* of the parent, only one message needs to be sent to each child. Fig.15 plots the control overhead for different "data packet size/control packet size" ratios, in which, the x-axis is the number of flooding packets sent per link quality update period and the y-axis is the ratio of the total bits sent for control packets over data packets. We see from the figure that the control overhead is reduced as the flooding packets sent per link quality update increases, as expected. When the data packet size is as small as the control packet, the overhead of Opportunistic Flooding is large especially when less than 5 flooding packets are sent during one link quality update period. When the data packet is greater than 5 times of the control packet, the control overhead becomes negligible when more than 5 flooding packets are sent per link quality update period.

In practice, a sensor network is deployed to have a number of tasks, and the cost of measuring link qualities is shared by all of them. For example, the hello messages are periodically sent to maintain the network connectivity for almost all tasks and the link qualities can be measured and updated at the same time with no extra cost. Even when flooding is the only operation of the network, we show in Fig.15 that Opportunistic Flooding still saves energy when a reasonable amount of flooding bits is sent per link quality update period.

7 Implementation and Evaluation

In addition to large-scale simulations, we implemented Opportunistic Flooding and Improved Traditional Flooding on the TinyOS/Mote platform in nesC with 30 MicaZ motes to further validate Opportunistic Flooding in practice.

7.1 Experiment Setup

We randomly deployed 30 MicaZ nodes on an in-door testbed, whose transmission power is tuned down so that the nodes form a 4-hop network. After deployment, all nodes are in the initialization phase with a 100% duty cycle. Each mote randomly generates a specified working schedule controllable by a standalone base station node. Then, starting from the source, each node broadcasts its existence and its working schedule. Upon receiving a broadcast message from a neighbor with a smaller hop count, a node updates its hop count and starts to announce its working schedule to neighboring nodes. When this process ends, all nodes have their hop counts ready and a neighbor table built with working schedules from all neighboring nodes. Followed by neighbor discovery, a node begins to measure the pairwise link quality between itself and each neighboring node in its neighbor table by counting the reception ratio of 20 packets. This information is exchanged with neighboring nodes. Consequently, the neighbor table of each node would contain both incoming and outgoing link qualities for all neighboring nodes. The link quality threshold l_{th} in the sender selection is 0.6. With such information, the *pmf* is computed and the $p-$quantile with $p = 0.9$ is shared with neighbors. After the initialization phase, all nodes switch to low-duty-cycle mode. They turn on or turn off their radios based on their working schedules. Specifically, in this experiment we set the unit time as $50ms$.

7.2 Performance Comparison

In this section, we compare the empirical flooding delay and energy consumption for both designs. For each specified duty cycle, the source sends 100 packets using either Opportunistic Flooding or Improved Traditional Flooding. In order to minimize the impact of link quality fluctuation on the performance comparison, opportunistic flooding packets and improved traditional flooding packets are sent alternatively.

7.2.1 Delay Performance

We investigated the impact of duty-cycle on delay as shown in Fig.16. At duty-cycles 2% and above, both schemes experience a comparable delay in flooding a packet to every node in the network. At the duty cycle of 1%, the delay in opportunistic flooding is about 25% shorter. Notice that Fig.16 does not show the similar significant delay reduction that we observed in the

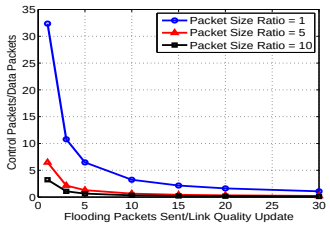


Fig. 15. Control Overhead

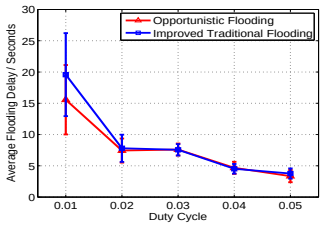


Fig. 16. Flooding Delay

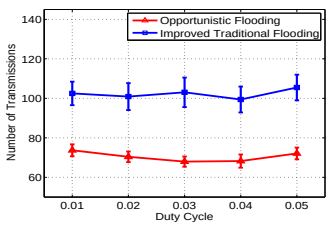


Fig. 17. Energy Cost

simulation. This is because our experiments are subject to the physical limitations of the testbed. First, we have to form a four-hop network with only 30 MicaZ nodes. Consequently, the number of flooding senders for each node is small, which reduces the chance of sending "opportunistically early" packets. Second, a pure-flooding algorithm is considered delay-optimal when a network is not congested. Due to the low connectivity in four-hop network of 30 nodes, the congestion level is not as significant as that observed in the simulation.

7.2.2 Energy Performance

Fig.17 compares average energy consumption of the two flooding designs. Due to the small network size and limited number of opportunistic links, the percentage of energy saved is not as significant as that in simulation, but is still noticeable. For example, at a duty cycle of 3%, the average flooding delay for Opportunistic Flooding and Improved Traditional Flooding are 7603ms and 7564ms, respectively. The energy costs are 68 and 103, meaning that Opportunistic Flooding saves about 34% in energy while providing a similar flooding delay.

7.3 Why Opportunistic Flooding is Better

This section presents insights on why Opportunistic Flooding significantly improves performance over Improved Traditional Flooding.

7.3.1 Observation on Delay Distribution

To investigate how a flooding packet propagates over a network, we recorded the receiving time stamps of individual packets and plotted the Cumulative Distribution Function of delay with both flooding methods in Fig.18(a). The experiment is done with a duty cycle of 1%. As seen in the figure, 80% of the nodes receive the flooding packet quickly within 10 seconds. However, it takes significantly more time to deliver the flooding packet to the other 20%. This indicates that the total flooding delay is severely affected by only a few nodes. This figure also shows that Opportunistic Flooding has a comparable delay to that of Improved Traditional Flooding for reaching individual nodes during the flooding process. Although it reaches 80% of the nodes more slowly, it reaches 100% of the nodes more quickly.

7.3.2 Observation on Energy Distribution

In addition to flooding delay, we also recorded the energy consumption at each individual node when the network operated at a 1% duty cycle and compared the distribution of single-node energy consumption in Fig.18(b). As seen in the figure, opportunistic flooding outperforms Improved Traditional Flooding at any given percentile. For example, about 70% of the nodes in Opportunistic Flooding transmit the flooding packets only 3 times, in contrast to 5 for Improved Traditional Flooding. Also, in order to reach the last a few nodes, especially the last 10%, the number of transmissions increase significantly due to poor link quality and connectivity. Again, this implies that the flooding delay is dominated by the last few nodes.

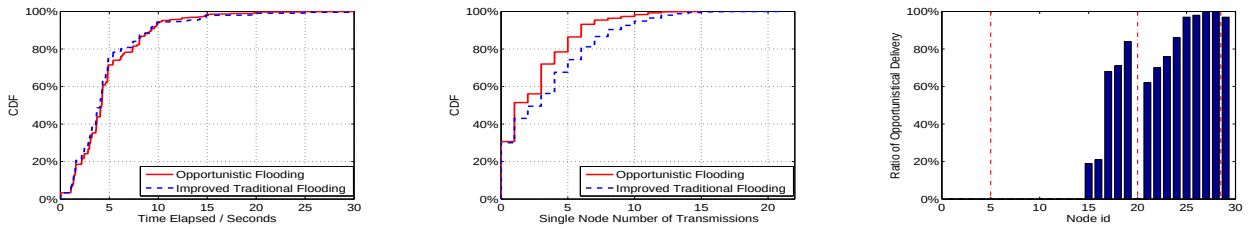
7.3.3 Observation on Opportunistic Ratio

We studied how Opportunistic Flooding helps reduce the flooding delay. Fig.18(c) plots the percentage of opportunistic flooding packets received at each node. The nodes are sorted according to their hop counts, and three vertical lines show the separation of nodes with different hop counts. We observe that as hop count increases, the chance of receiving opportunistic early packets increases significantly. For example, while no hop-one nodes receive any opportunistic early packets, about one-third of the nodes at hop-two receive opportunistic early packets. At hop three, almost every node receives opportunistic early packets, and the average percentage of such packets is around 80% of the total flooding packets received. This observation indicates that Opportunistic Flooding design is very effective in reducing flooding delay, especially when the network scale becomes large.

8 Related Work

As essential operations for wireless networks, multicasting [13], [32] and flooding [14], [21], [23], [29], [38] have been extensively studied in the literature. RMAC [37] presents a reliable multicast service at the MAC layer using the busy tone mechanism. Mobicast [13] and GARUDA [32] provide reliable data delivery from a sink to the sensors in specified delivery regions. PBBF [29] proposes a MAC layer solution for flooding and investigates tradeoffs among reliability, latency and energy consumption. Aimed at ameliorating message implosion, Smart Gossip [21] adaptively determines the forwarding probability for received flooding messages at individual sensor nodes based on previous knowledge and network topology. By exploiting network density and maintaining reliable bridge links among dense clusters of nodes, RBP [38] demonstrates its energy efficiency and high reliability for flooding. For services such as network reprogramming, protocols such as Deluge [14] and Trickle [23] also propose techniques for efficiently propagating code to nodes in the network. Recently, Zhu et al. explores link correlation in flooding service [51]. All these works assume there are usually multiple neighbors available at the same time to receive the broadcast message sent by a sender, which does not hold in low-duty-cycle networks.

On the other hand, data forwarding in low-duty-cycle networks have acquired a lot of attentions in recent years [2], [7]. ZigBee provides a specification for a suite of high level



(a) Delay CDF (b) Energy CDF (c) Opportunistic Delivery Ratio

Fig. 18. System Insights of Opportunistic Flooding

communication protocols for low-power devices. However, it does not provide a solution to multi-hop broadcasting. TDMA-based approaches [18], [27] have proposed scheduling methods for either collision-free sender-side transmissions or energy-efficient sleep schedules. While these works successfully save the energy cost caused by collisions and idle listening, they do not consider redundant transmissions and unreliable wireless links. Scheduling also becomes less efficient with the existence of wireless loss. A number of works have proposed routing solutions for duty-cycled networks [1], [8], [19], [31], [34], [35], [48]. However, the point-to-point routing model makes them not suitable to be applied in broadcasting/flooding. For broadcasting service in duty-cycled sensor network, RBS [44] proposes a solution and shows its efficiency in terms of delay and energy cost. ADB [39] proposes an asynchronous MAC protocol by distributing the coverage information in broadcasting process. Jiao [15], [16] proposes scheduling algorithms for multi-hop broadcasting without considering unreliable links. Lai [22] proposes a broadcasting solution that saves energy by letting a node decide how long to wait for more receivers to wake up, but requires relatively high duty cycle. None of these solutions investigates the low-duty-cycle operation and unreliable links at the same time. Different from all these previous works, Opportunistic Flooding provides a promising solution for multi-hop broadcasting while considering the problems caused by both low-duty-cycle operation and unreliable wireless links simultaneously.

9 Conclusion

In this paper, we present Opportunistic Flooding: a delay-driven flooding method that is particularly designed for low-duty-cycle wireless sensor networks. Each node makes probabilistic forwarding decisions based on the delay distribution of next-hop nodes. Only opportunistically early packets are forwarded via the links outside the energy-optimal tree to reduce the flooding delay and the level of redundancy. To resolve decision conflict, we build a reduced flooding sender set to alleviate the hidden terminal problem. Within the same sender set, we use a link-quality-based backoff method to resolve and prioritize simultaneous forwarding operations. Extensive simulations and test-bed experiments show that Opportunistic Flooding approaches the optimal performance and achieves a shorter average flooding delay with less than half of the energy cost of Improved Traditional Flooding in various network settings. In the future, we shall extend this work into the scenario where working schedules can be flexibly changed to provide better flooding performance.

Acknowledgement

This work was supported in part by SUTD SRG ISTD 2010 002, SUTD-ZJU/RES/03/2011, US National Science Foundation (NSF) grants CNS-0845994, CNS-1117438, CNS-0917097, and the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive & Digital Media Programme Office, Media Development Authority.

References

[1] M. Brzozowski, H. Salomon, and P. Langendoerfer. Completely Distributed Low Duty Cycle Communication for Long-Living Sensor Networks. In *CSE(2)'09*, 2009.

[2] Z. Cao, Y. He, and Y. Liu. L2: Lazy forwarding in low duty cycle wireless sensor networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1323 – 1331, march 2012.

[3] D. Couto, D. Aguayo, J. Bicket, and R. Morris. A High Throughput Path Metric for Multi-Hop Wireless Routing. In *MobiCom'03*, 2003.

[4] B. Das and V. Bharghavan. Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets. In *ICC '97*, 1997.

[5] P. Dutta and D. Culler. Practical Asynchronous Neighbor Discovery and Rendezvous for Mobile Sensing Applications. In *SenSys'08*, 2008.

[6] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with glossy. In *IPSN'11*, pages 73–84, 2011.

[7] G. Ghidini and S. Das. An energy-efficient markov chain-based randomized duty cycling scheme for wireless sensor networks. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 67 –76, june 2011.

[8] Y. Gu and T. He. Dynamic switching-based data forwarding for low-duty-cycle wireless sensor networks. *IEEE Transactions on Mobile Computing*, 10(12):1741–1754, 2011.

[9] S. Guo, Y. Gu, B. Jiang, and T. He. Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links. In *MobiCom'09*, 2009.

[10] H. Gupta, V. Navda, S. Das, and V. Chowdhary. Efficient Gathering of Correlated Data in Sensor Networks. *ACM Trans. on Sensor Networks*, 4(1), 2008.

[11] T. He, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher. AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks. *ACM Trans. on Embedded Computing System, Special issue on Dynamically Adaptable Embedded Systems*, 2004.

[12] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *ACM Trans. on Sensor Networks*, February 2006.

[13] Q. Huang, C. Lu, and G.-C. Roman. Spatiotemporal Multicast in Sensor Networks. In *SenSys'03*, 2003.

[14] J. W. Hui and D. Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In *SenSys'04*, 2004.

[15] X. Jiao, W. Lou, J. Ma, J. Cao, X. Wang, and X. Zhou. Duty-Cycle-Aware Minimum Latency Broadcast Scheduling in Multi-Hop Wireless Networks. In *ICDCS'10*, 2010.

[16] X. Jiao, W. Lou, X. Wang, J. Ma, J. Cao, and X. Zhou. Interference-Aware Gossiping Scheduling in Uncoordinated Duty-Cycled Multi-Hop Wireless Networks. In *WASA'10*, 2010.

[17] A. Kamra, V. Misra, and D. Rubenstein. CountTorrent: Ubiquitous Access to Query Aggregates in Dynamic and Mobile Sensor Networks. In *SenSys'07*, 2007.

[18] A. Kanzaki, T. Uemukai, T. Hara, and S. Nishio. Dynamic TDMA Slot Assignment in Ad Hoc Networks. In *AINA'03*, 2003.

[19] D. Kim and M. Liu. Optimal stochastic routing in low duty-cycled wireless sensor networks. In *WICON*, 2008.

[20] J. H. Kim and J. K. Lee. Capture Effects of Wireless CSMA/CA Protocols in Rayleigh and Shadow Fading Channels. *IEEE Trans. on Vehicular Technology*, 48(4), 1999.

[21] P. Kyasanur, R. R. Choudhury, and I. Gupta. Smart Gossip: An Adaptive Gossip-based Broadcasting Service for Sensor Networks. In *MASS'06*, 2006.

[22] S. Lai and B. Ravindran. On Multihop Broadcast over Adaptively Duty-Cycled Wireless Sensor Networks. In *DCOSS'10*, 2010.

[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 *NSDI'04*, 2004.

[24] K. Lin, J. Yu, J. Hsu, S. Zahedi, D. Lee, J. Friedman, A. Kansal, V. Raghunathan, and M. Srivastava. Heliomote: Enabling Long-Lived Sensor Networks through Solar Energy Harvesting. In *SenSys'05*, 2005.

[25] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel. Delay Efficient Sleep Scheduling in Wireless Sensor Networks. In *INFOCOM'05*, 2005.

[26] G. S. M. Maroti, B. Kusy and A. Ledeczi. The Flooding Time Synchronization Protocol. In *SenSys'04*, 2004.

[27] J. Ma, W. Lou, Y. Wu, X.-Y. Li, and G. Chen. Energy Efficient TDMA Sleep Scheduling in Wireless Sensor Networks. In *INFOCOM'09*, 2009.

[28] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *Operating Systems Design and Implementation*, 2002.

[29] M. J. Miller, C. Sengul, and I. Gupta. Exploring the Energy-Latency Trade-Off for Broadcasts in Energy-Saving Sensor Networks. In *ICDCS'05*, 2005.

[30] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis Diffusion for Robust Aggregation in Sensor Networks. In *SenSys'04*, 2004.

[31] W. Pak, K.-T. Cho, and S. Bahk. Energy efficient routing protocol for wireless sensor networks with ultra low duty cycle. In *PIMRC*, 2009.

[32] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz. GARUDA: Achieving Effective Reliability for Downstream Communication in Wireless Sensor Networks. *IEEE Trans. on Mobile Computing*, 2008.

[33] J. Polastre and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *SenSys'04*, 2004.

[34] N. Ramanathan, M. Yarvis, J. Chhabra, N. Kushalnagar, L. Krishnamurthy, and D.Estrin. A Stream-Oriented Power Management Protocol for Low Duty Cycle Sensor Network Applications. In *EmNets*, 2005.

[35] A. Ruzzelli, G. O'Hare, and R. Jurdak. MERLIN: Cross-layer Integration of MAC and Routing for Low Duty-Cycle Sensor Networks. In *Elsevier Ad Hoc Networks Journal*, 2008.

[36] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter. LUSTER: Wireless Sensor Network for Environmental Research. In *SenSys '07*, 2007.

[37] W. Si and C. Li. RMAC: A Reliable Multicast MAC Protocol for Wireless Ad Hoc Networks. In *ICPP'04*, 2004.

[38] F. Stann, J. Heidemann, R. Shroff, and M. Z. Murtaza. RBP: Robust Broadcast Propagation in Wireless Networks. In *SenSys '06*, 2006.

[39] Y. Sun, O. Gurewitz, S. Du, L. Tang, and D. B. Johnson. ADB: An Efficient Multihop Broadcast Protocol Based on Asynchronous Duty-Cycling in Wireless Sensor Networks. In *SenSys'09*, 2009.

[40] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. An Analysis of a Large Scale Habit Monitoring Application. In *SenSys'04*, 2004.

[41] Texas Intruments. *2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. B)*, 2007. Available at http://focus.ti.com/docs/prod/folders/print/cc2420.html.

[42] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, S. Burgess, D. Gay, P. Buonadonna, W. Hong, T. Dawson, and D. Culler. A Macroscope in the Redwoods. In *SenSys'05*, 2005.

[43] E. Uysal-Biyikoglu, B. Prabhakar, and A. E. Gamal. Energy-Efficient Packet Transmission Over A Wireless Link. *IEEE/ACM Trans. on Networking*, 10(4):487–499, 2002.

[44] F. Wang and J. Liu. On reliable broadcast in low duty-cycle wireless sensor networks. *Mobile Computing, IEEE Transactions on*, 11(5):767 –779, may 2012.

[45] J. Wang, Y. Liu, and S. Das. Asynchronous Sampling Benefits Wireless Sensor Networks. *INFOCOM'08*, April 2008.

[46] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *SenSys'03*, 2003.

[47] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A Wireless Sensor Network for Structural Monitoring. In *SenSys'04*, 2004.

[48] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks. In *INFOCOM'04*, 2004.

[49] J. Zhao and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *SenSys '03*, 2003.

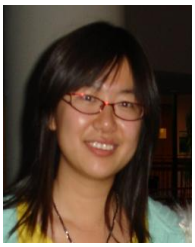
[50] J. Zhu, S. Chen, B. Bensaou, and K.-L. Hung. Tradeoff Between Lifetime and Rate Allocation in Wireless Sensor Networks: A Cross Layer Approach. In *INFOCOM'07*, 2007.

[51] T. Zhu, Z. Zhong, T. He, and Z.-L. Zhang. Exploring Link Correlation for Efficient Flooding in Wireless Sensor Networks. In *NSDI'10*, 2010.

[52] Y. Zhu and L. Ni. Probabilistic Approach to Provisioning Guaranteed QoS for Distributed Event Detection. In *INFOCOM'08*, 2008.

[53] Y. Gu, L. He, T. Zhu and T. He. Achieving Energy Synchronized Communication in Energy-Harvesting Wireless Sensor Networks. *ACM Transaction on Embedded Computing System*, to appear.

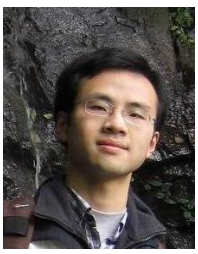
[54] M. Zuniga and B. Krishnamachari. Analyzing the Transitional Region in Low Power Wireless Links. In *IEEE SECON'04*, 2004.



Shuo Guo received her B.S. in Electronic Engineering at Tsinghua University in 2006 and is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering at the University of Minnesota, Twin Cities. Her research includes Wireless Sensor Networks, Vehicular Ad-Hoc Networks, and Real-time and Embedded Systems. She received a best paper award at IEEE MASS 2008 and has publication in many premier sensor network journals and conferences.

Liang He is currently a postdoc research fellow at Singapore University of Technology and Design. He received his B.Eng. degree in 2006 and Ph.D degree in 2011 from Tianjin University, China, and Nankai University, China, respectively. During Oct. 2009 to Oct. 2011, he worked at Panlab at University of Victoria as a visiting research student. He has been a recipient of the the best paper awards of IEEE WCSP 2011 and IEEE GLOBECOM 2011.

Yu (Jason) Gu is currently an assistant professor at Singapore University of Technology and Design. He received his Ph.D. under Prof. Tian He from University of Minnesota, Twin Cities, in 2010. He is the author and co-author of over 21 papers in premier journals and conferences. His research includes Networked Embedded Systems, Wireless Sensor Networks, Cyber-Physical Systems, Wireless Networking, Real-time and Embedded Systems, Distributed Systems, Vehicular Ad-Hoc Networks and Stream Computing Systems. Yu Gu is a member of ACM, IEEE and SIAM.



Bo Jiang Bo Jiang received his B.S. in Electronic Engineering from Tsinghua University in 2006, and his M.S. in Electrical and Computer Engineering from University of Massachusetts Amherst in 2008. He is currently a Ph.D. Student in the Department of Computer Science at University of Massachusetts Amherst. His research includes network modeling and analysis, wireless networks, and network science.

Tian He is currently an associate professor in the Department of Computer Science and Engineering at the University of Minnesota-Twin City. He received the Ph.D. degree under Professor John A. Stankovic from the University of Virginia, Virginia in 2004. Dr. He is the author and co-author of over 100 papers in premier sensor network journals and conferences with over 10,000 citations (H-Index 40). His publications have been selected as graduate-level course materials by over 50 universities in the United States and other countries. Dr. He has received a number of research awards in the area of networking, including five best paper awards. Dr. He is also the recipient of the NSF CAREER Award 2009 and McKnight Land-Grant Professorship. Dr. He served a few program chair positions in international conferences and on many program committees, and also currently serves as an editorial board member for six international journals including ACM Transactions on Sensor Networks. His research includes wireless sensor networks, cyber-physical systems, intelligent transportation systems, real-time embedded systems and distributed systems, supported by National Science Foundation, IBM, Microsoft and other agencies.