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

Shuo Guo, Yu Gu, Bo Jiang and Tian He

Department of Computer Science and Engineering, University of Minnesota
{sguo,yugu,tianhe}@cs.umn.edu

ABSTRACT

Intended for network-wide dissemination of commands, configurations and code binaries, flooding has been investigated extensively in wireless networks. However, little work has yet been done on low-duty-cycle wireless sensor networks in which nodes stay asleep most of 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. Combined with unreliable links, flooding in low-duty-cycle networks is a new challenging issue.

In this paper, we introduce Opportunistic Flooding, a novel design tailored for low-duty-cycle networks with unreliable wireless links and predetermined working schedules. The key idea is to make probabilistic forwarding decisions at a sender based on the delay distribution of next-hop nodes. Only *opportunistically early* packets are forwarded using links outside the energy optimal tree to reduce the flooding delay and redundancy in transmission. To improve performance further, we propose a forwarder selection method to alleviate the hidden terminal problem and a link-quality-based backoff method to resolve simultaneous forwarding operations. We evaluate Opportunistic Flooding with extensive simulation and a test-bed implementation consisting of 30 MicaZ nodes. Evaluation shows our design 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 in various low-duty-cycle network settings.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols

[*]This work was supported in part by NSF grants CNS-0917097, CNS-0845994 and CNS-0626609. We also receive the facility support from MSI and DTC at University of Minnesota. Bo Jiang is a Ph.D student at University of Massachusetts, Amherst.

General Terms

Performance, Design

Keywords

Wireless Sensor Networks, Flooding, Low Duty Cycle Networks

1. INTRODUCTION

Wireless Sensor Networks have been used for many long-term applications such as military surveillance [11], infrastructure protection [44] and scientific exploration [33]. After a sensor node is miniaturized into a cubic centimeter package and deployed without wired power, its available energy is severely limited. On the other hand, there is a growing need for sustainable deployment of sensor systems [36, 44, 48] 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 at a very-low-duty-cycle (e.g., 1% or less), in which a sensor node schedules itself to be active for only a very brief period of time and then stays dormant for a long time. 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 communication designs in low-duty-cycle networks. While pioneering projects have been proposed for low-duty-cycle unicasts [8, 24, 47], research is surprisingly inadequate for low-duty-cycle flooding, an important function for disseminating network-wide commands, alerts and configurations [11], time synchronization [25], and code binaries [14].

Beside sleep latency, two additional features make flooding in low-duty-cycle networks a new and challenging issue. First, a flooding packet cannot be received by multiple nodes simultaneously as it is in an always-awake network. To broadcast a packet, a sender may have to transmit the same packet multiple times if its receivers do not wake up at the same time. Essentially, flooding in such a network is realized by a number of unicasts. Second, unlike wired networks, wireless communication is notoriously unreliable [46]. A transmission might be repeated if the previous transmissions are not successful because of the low link quality. The combination of low-duty-cycle operation and unreliable links makes the problem of flooding different from that found in wired networks and always-awake wireless networks.

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*. A straightforward solution could be to make use of a routing tree to flood a packet. Yet this type of solution has been shown [16, 29] to be very fragile, since the failure of a parent node prevents all its subtree nodes from receiving flooding messages, even if the network is actually connected. Furthermore, existing tree-based solutions could be made energy efficient, only at the cost of possibly very long delays, as they only forward packets via a single route.

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*, in which nodes forward a packet with a higher probability if the packet arrives *opportunistically earlier*. This is achieved by comparing the delay of individual packets with the statistic packet delay distribution (i.e., probability mass function *pmf*) at next-hop nodes. Specifically, our contributions are as follows:

- To the best of our knowledge, this is the first distributed flooding method designed for wireless sensor networks that considers the effect of both low-duty-cycle and unreliable wireless links.
- This work is the first to propose *delay-driven opportunistic forwarding*. We propose a recursive and distributive method to compute the distribution of forwarding delays (*pmf*) at each node along an energy-optimal tree. The resultant *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 to overhear each other with good link quality. We also propose a link-quality-based back-off method to resolve simultaneous transmission 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, followed by its evaluation in Sections 5 and 6. Section 7 discusses the related work and Section 8 concludes the paper.

2. MOTIVATION

To bridge the growing gap between the lifetime requirements of sensor applications [36, 44, 48] and the limited availability of energy through fixed-budget batteries or energy harvesting [22], 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 [39]; (ii) link-layer optimization for better connectivity, reliability, and stability [3,31,40]; (iii) network-layer enhancement for better forwarders and routes [5, 12, 43]; and (iv) application-layer improvements for both content-agnostic and content-centric data aggregation and inference [10, 26]. Although these solutions are highly diverse, they all assume a wireless network in which nodes are *ready to receive* packets and focus mainly on the transmission side, a topic of interest that has resulted in hundreds of publications.

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 ChipCon CC2420 radio [37] draws 19.7mA when receiving or idle listening, which is actually larger than the 17.4mA used when transmitting. More importantly, packet transmission time is usually very brief (e.g., 1.3 milliseconds to transmit a TinyOS packet using a CC2420 radio), while the duration of idle listening for reception can be orders of magnitude longer. For example, most environmental applications, such as Great Duck Island [36] and Redwood Forest [38], 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 most of the energy cost in communication. To reduce the energy penalty in idle listening, a node has to run at a low-duty-cycle (e.g., 5%, 1% or less) and turn off its radio most of time.

2.2 The Need for a New Flooding Design

The traditional flooding method as well as many advanced versions [14, 17, 20, 27, 35] have proven their performance in terms of delivery ratio, delay and energy cost in many always-awake network settings. We argue, however, that these solutions suffer severe performance degradation (in both energy and time efficiency) if directly applied to low-duty-cycle networks. In those flooding methods, a node starts broadcasting a packet (immediately or with a certain probability) when it first receives a flooding packet from its previous-hop node. For a low-duty-cycle network in which two neighbors seldom wake up at the same time, a broadcasting packet cannot be received by many nodes simultaneously. In addition, the delivery ratio of traditional flooding methods becomes even worse when unreliable links and collisions are taken into account. To confirm this empirically, we conducted a series of simulations by decreasing the duty-cycle of a network from 100% to 1%. Fig.1 shows how performance degrades as the duty-cycle becomes lower and that even under ideal conditions (i.e., no collisions and perfect links), a network can only deliver less than 5% of packets running at a 2% duty-cycle, a clear indication that the traditional flooding method is not suitable for low-duty-cycle networks, if used directly.

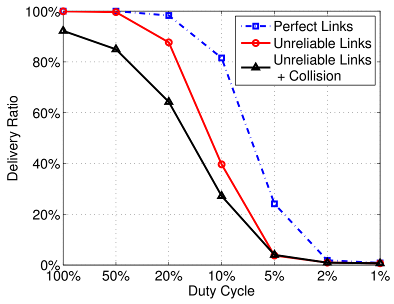


Figure 1: Traditional Flooding in Low-Duty-Cycle Networks

One could argue that traditional flooding methods could be adapted to low-duty-cycle networks by permitting (i) multiple transmissions of the same packet based on the neighbors' active schedules and (ii) ARQ-based retransmission to deal with unreliable links, but this still has a number of problems. First, it suffers from a high energy cost due to collisions. When a node wakes up, many of its neighbors attempt to start a transmission simultaneously. If the links are perfect, collisions can be reduced through media access control. Unfortunately, unreliable wireless links [31, 46] increase collisions when nodes cannot sense each other's transmission. Second, even without collisions, the number of redundant transmissions is still large especially when the network density is high. Due to these limitations in traditional flooding methods, it is necessary to have a tailored design such as the one presented in this paper.

3. PRELIMINARIES

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

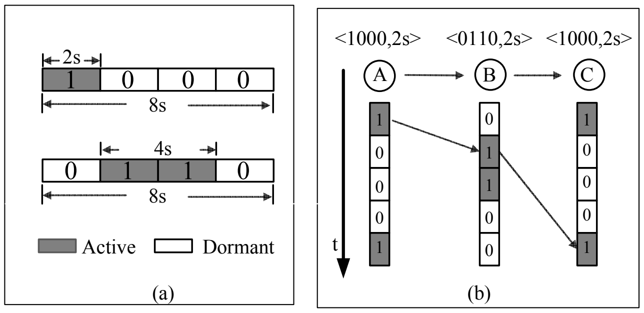


Figure 2: A Low-Duty-Cycle Network Model

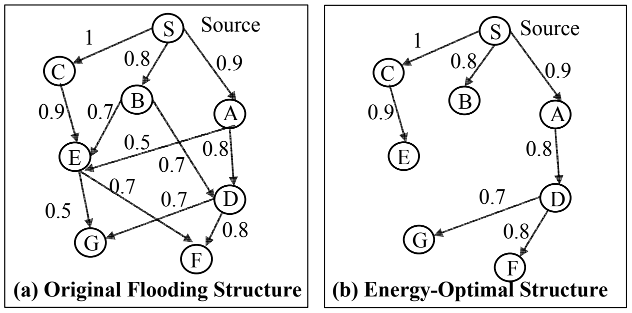


Figure 3: DAG-Based Flooding Structure

3.1 Network Model

Suppose there is a wireless sensor network of N sensor nodes. Each sensor 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 all its function modules off except a 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 temporal-correlation sensing data [9, 42]. A dormant node wakes up when (i) it is scheduled to switch to the receiving state, or (ii) it has some packets to send to a neighbor that is active at that time. *In short, a node can transmit a packet at any time, but can only receive a packet when it is active.* Formally, the network can be denoted by a time-dependent graph $G(t) = (V, E(t))$ where V is the complete set of the N nodes in the network and $E(t)$ is the set of directed edges at time t. A directed edge $e_{ij}(t)$ belongs to $E(t)$ if node i and j are neighbors and at time t, node j is active so that it is able to receive packets from node i.

For sensing purposes, the working schedules of sensor nodes are normally periodic [11, 36, 38]. 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 transmission time. Then each node picks one or more time units as its active state. (For those active states that are long enough for two or more round-trips, consecutive time units are selected.) Suppose the active states and the dormant states are denoted as '1's and '0's, 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 unit. 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(a) shows the example of $< 1000, 2s >$ and $< 0110, 2s >$ where T is $8s$ and is divided into 4 time units, each of which is $2s$ long. As shown in the figure, a node with schedule $< 1000, 2s >$ is active during the first $2s$ and dormant during the rest $6s$. Using this model, the delay of a packet can be easily computed and represented by the number of time units. As shown in the example in Fig.2(b), node A tries to forward a packet to C via B. Since a node can only receive a packet when it is active (the corresponding time units in the figure are shadowed), the packet is delayed while a sender waits for its receiver to become active. If both links are perfect, the total delay of this packet from A to C is 4 time units ($4\tau = 8s$) according to B and C's working schedules.

3.2 Assumptions

Suppose the source nodes have flooding packets to be sent throughout the whole network. Based on the network model, we make several assumptions as follows:

1. A node sets up its working schedule $< w_i, \tau >$ and shares it with all its neighbors when it joins the network, a process normally referred to as low-duty-cycle rendezvous [7]. After rendezvous, a node knows all its immediate (one-hop) neighbors' working schedules. A node only changes its schedule after the new schedule is updated to its neighbors.
2. We assume the existence of unreliable links and collision. If two or more ongoing transmissions are within the communication range, none of them will succeed. We also assume the link quality changes noticeably over time, however, research [23] has indicated that the rate of change is slow, therefore measurements of the link quality can be updated at a very low cost (i.e., once every ten minutes) or through low-cost piggybacking on regular data traffic.
3. The network is locally synchronized so that a node knows when it can send a packet to every neighbor given their working schedules. Local synchronization can be achieved by using a MAC-layer time stamping technique, as described in FTSP [25], 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 2000μs to 20,000μs, the accuracy of 2.24μs is sufficient.
4. An integer (hop count) is used to denote the minimum number of hops from a node to the source. For example, the source node is marked 0. All the source's immediate neighbors are marked 1, all these nodes' unmarked immediate neighbors are marked 2, and so on. A node will only forward a flooding packet to nodes with larger hop count to ensure the network's loop-free property.

In a low-duty-cycle network, the probability that a node's two neighbors have 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 unit 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.

4. MAIN DESIGN

The objective of our flooding design is to reduce delay and redundancy. This section presents first an overview of a three-step design, followed by a detailed description of each step in 4.2, 4.3 and 4.4, respectively.

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

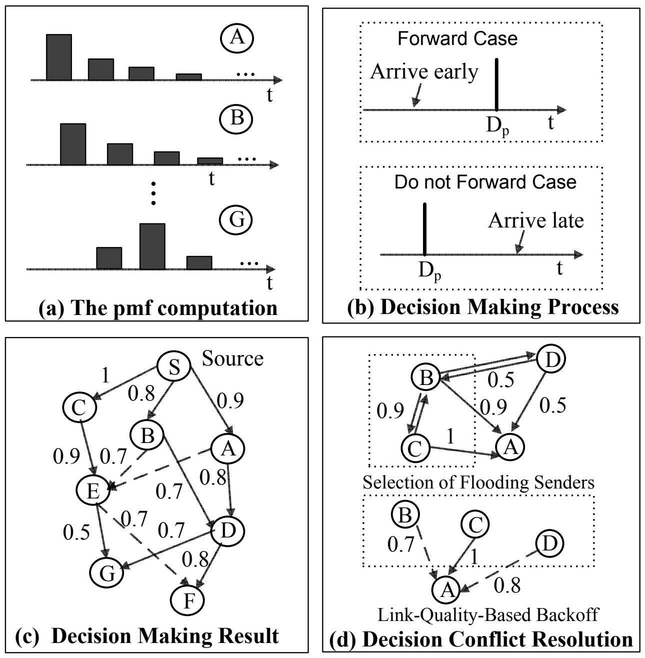

A
B
G
t
Forward Case
Arrive early
Dp
Do not Forward Case
Arrive late
(a) The pmf computation
(b) Decision Making Process
Source
S
C
B
A
E
D
G
F
1
0.8
0.9
0.7
0.5
(c) Decision Making Result
Selection of Flooding Senders
Link-Quality-Based Backoff
(d) Decision Conflict Resolution

Figure 4: Design Overview of Opportunistic Flooding

ones. As a result, the flooding structure of the network is a directed acyclic graph (DAG) of N vertices, as shown in Fig.3(a). The weights of the directed edges are the corresponding link quality. Based on the DAG, a tree structure can be constructed by assigning each node an incoming link with the best link quality among all incoming links, as shown in Fig.3(b). As we show later, it can be easily proved that this tree structure is the energy-optimal one for flooding among all tree structures generated from the DAG. In other words, if a flooding packet is forwarded based on this tree, (i.e., a node only receives a flooding packet from its parent), the expected total number of transmissions is minimized.

We observe, however, that flooding via the energy-optimal tree may have a long flooding delay, since a node's parent may not receive the flooding packet as early as its other neighbors, due the opportunistic nature of wireless communication. Based on this observation, the key idea of opportunistic flooding is to *utilize opportunistic links outside an energy-optimal tree if the transmissions via these links have a high chance of making the receiving node receive the packet "statistically earlier" than its parent.*

Clearly, the flooding structure of our design is dynamically changing, where a node decides to forward its received 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/redundant one. In other words, the packet to be forwarded *opportunistically* shall be *statistically earlier* than the packet that is otherwise 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:

1. **The *pmf* Computation**: Due to unreliable links, the delay of a flooding packet arriving at each node from its parent through the energy-optimal tree is a random variable. In our design shown in Fig.4(a), the probability mass function (*pmf*) of this delay is first derived for each node to guide the decision making process. From the *pmf*, each node computes its p-quantile delay D_p as the statistically significant threshold and shares this with all its pervious-hop nodes.
2. **Decision Making Process**: As shown in Fig.4(b), a packet is forwarded opportunistically via the links outside of the energy-optimal tree only if this forwarding can significantly reduce the delay (Note that p-quantile delay D_p 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**: Since each node makes its forwarding decision in a purely distributed manner, it would be the case that multiple nodes decide to forward the same packet to a common neighbor, which is called *decision conflict*. Two conflict resolution techniques are designed to avoid collisions and save energy further, as shown in Fig.4(d).

With dynamic decisions per packet, our design permits a packet to travel along an opportunistically-fast route instead of a fixed one via the energy-optimal tree. At the same time, late packets are not forwarded to reduce redundancy and save energy. Detailed designs are shown in the following subsections.

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

This section computes the packet delay distribution (*pmf*) via an energy-optimal tree. By comparing the delay along the energy-optimal tree, a node can decide whether opportunistic forwarding via links outside of the energy-optimal tree is needed or not.

4.2.1 About Energy Optimality

As discussed in 4.1, an energy-optimal flooding tree of a low-duty-cycle network can be constructed by assigning each node a neighbor that has a smaller hop count and the best link quality. The energy-optimality of this tree among all trees generated from the DAG can be easily proved by contradiction: In an energy-optimal tree, if there exists a node whose incoming link is not the one that has the best link quality among all incoming links, a new tree can be constructed by replacing this link with the best incoming link, making this tree more energy-efficient than the previous one given that the network has a low duty cycle and flooding is realized by a number of unicasts. We note that if multiple nodes wake up simultaneously, finding an optimal flooding structure is equivalent to finding a Minimum Connected Dominating Set (MCDS), which is proven NP-hard [6]. Since the multiple-receiver scenario is rare in low-duty-cycle networks, the aforementioned flooding tree is a good approximation of energy optimality.

4.2.2 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 $t_l(i)$, 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 spreads throughout the network level by level. Initially, level-0 node (the source) is always awake and the probability that it receives the packet with delay 0 is 100%. In other words, the pmf of the source is {(0,100%)}. Then a level-1 node calculates its *pmf* based on its level-0 parent node's *pmf*. Similarly, a level-$(l+1)$ node calculates its *pmf* based on its level-l parent's *pmf*. Given the *pmf* of this level-l node ($t_l(i)$ and $p_l(i)$ for any i), 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 unit as follows:

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

where q is the corresponding 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 $t_{l+1}(j)$. The term $p_l(i)q(1-q)^{n_{ij}}$ is the probability that the packet that 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.

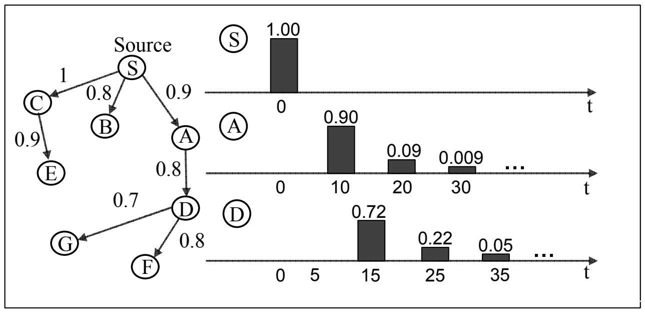


Figure 5: The *pmf* Computation

Fig.5 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 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) the probability that node A receives the packet at time 20 and succeeds in the first transmission, which is $0.9\times(1-0.8)\times0.8+0.09\times0.8 = 0.22$. Similarly, all the nodes within the network compute their *pmf* as long as their parents' *pmf* becomes available.

4.2.3 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 infinite 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.5, node A's *pmf* contains only two entries: (10, 0.9) and (20,0.09). In this case, $M = 2$.

Calculating pmf based on Eq.1 directly requires $O(M^2)$ computation. But if we reformulate it with this recursive form:

$$p_{l+1}(j) = p_{l+1}(j-1)(1-q) + \sum_{i:t_{l+1}(j-1)\le t_l(i)<t_{l+1}(j)} p_l(i)q \quad (2)$$

The computation of pmf becomes a simple recursion with a linear complexity of $O(M)$.

4.3 Decision Making Process

As discussed in 4.1, only opportunistically early packets are forwarded by a node to its next-hop nodes 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 should be considered *Needed*. Otherwise it only consumes more energy and should be considered *Redundant*.

From the delay distribution computed in 4.2, a node finds its p-quantile delay (denoted as D_p) as its threshold delay and shares this threshold with its previous-hop nodes. Based on definition, D_p is a threshold delay such that if a flooding packet arrives at this node later than delay of D_p, the probability that it has already received this packet from its parent is greater than p. Then, for each new flooding packet and each next-hop node, a node computes the expected packet delay (*EPD*) and makes a forwarding decision based on the comparison between *EPD* and D_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. If the transmission from A to B is started, B's *EPD* from A can be computed using the following equation:

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

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 geometric series, which can be calculated with a close form. To reduce the number of floating-point calculations, an alternative way to calculate *EPD* 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. Based on this, A checks B's working schedule and finds B's $\lceil \frac{1}{q} \rceil$th active time unit after $t_l(i)$ (the time that this packet arrives at A) and uses this value for B's *EPD*.

After the *EPD* is computed, A compares this value with B's threshold delay D_p to decide if this transmission (from A to B) is opportunistically needed, compared with delivery via the energy-optimal tree. 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. As a result, this packet is considered *Needed*. If *EPD*$> D_p$, B has more than p percentile of chance that it has already received this packet. As a result, this packet is considered *Redundant* and will not be forwarded to the next-hop B.

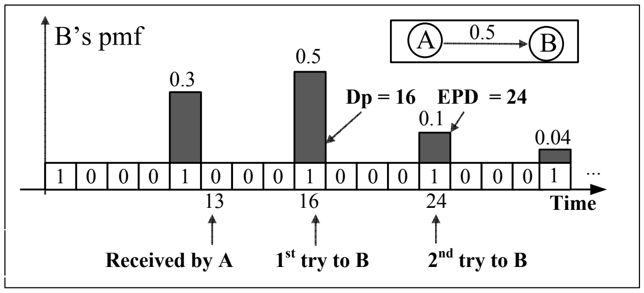


Figure 6: An example of Decision Making

Fig.6 shows a simple walkthrough of decision making. Suppose $p = 0.8$, the corresponding p-quantile delay D_p can be calculated using the discrete quantile function:

$$F^{-1}(p) = \min\{x \in R : \Pr(t \le x) \ge p\} \quad (4)$$

Then, $D_p = F^{-1}(0.8) = 16$, because $\Pr(t \le 16) = 0.3 + 0.5 = 0.8$. During the initialization, B shares the D_p value with A.

Suppose a packet arrives at node A at time 13. Since the link quality is 0.5, A is expected to transmit twice in order to forward the packet to B successfully. As shown in Fig.6, the first try is at time 16 (which is the first active time unit of B after time 13) and the second try is at time 24. Therefore, the *EPD* of B computed by A is 24. Since $EPD = 24 > D_p = 16$, in this particular case, the decision that A makes for B regarding this packet is $Redundant$ and hence it is not forwarded.

We note that p is a control parameter to balance the delay with energy consumption. On one hand, as p becomes larger, a sender is likely to mark a packet as *Needed*. Hence more packets are sent opportunistically, increasing the chance of fast delivery if collision is handled appropriately. It also increases the number of transmissions, leading to more energy consumption. 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 later.

4.4 Decision Conflict Resolution

This subsection solves the problem that occurs when two or more nodes make a decision to start a transmission toward the same node. The design includes two components: 4.4.1 selects the flooding sender set, and 4.4.2 resolves the decision conflicts within the set.

4.4.1 The Selection of Flooding Senders

In wireless communication, a certain percentage of collisions are caused by the Hidden Terminal Problem, where two nodes are trying to forward a packet to the same node without knowing each other. If this happens, both of them will keep sending but neither of them will succeed. One possible solution might be 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 note that the size of the sender set strikes a balance between delay and collision. On the one hand, we need a large sender set to increase the chance of opportunistic early packets. The more nodes there are in a node's sender set, the shorter the delay in which the node could expect to receive the packet from the set. On the other hand, including more nodes into the sender set increases the chance of collision. Since links among senders are unreliable, the more nodes there are in the same set, the more nodes that will possibly send at the same time and the greater the chance that a transmission is not sensed by all the other nodes, leading to a collision.

Based on the analysis, the selected nodes should have a certain link quality between them so that a transmission has a high probability of being sensed by all the other nodes within the same sender set. This criterion also alleviates the hidden terminal problem, since two nodes can never be in the same sender set if one can never sense the other's transmission. 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 better link quality than l_{th}.

The selection process goes as follows: First, a node only receives flooding packets from nodes that have a smaller hop count. These nodes are the candidates for the flooding senders. The selection process starts with the candidate that has the best link quality. This candidate is always included in the sender set. The next candidate is the neighbor with the second best link quality. If the link quality between this candidate and the already selected senders are all 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 quality.

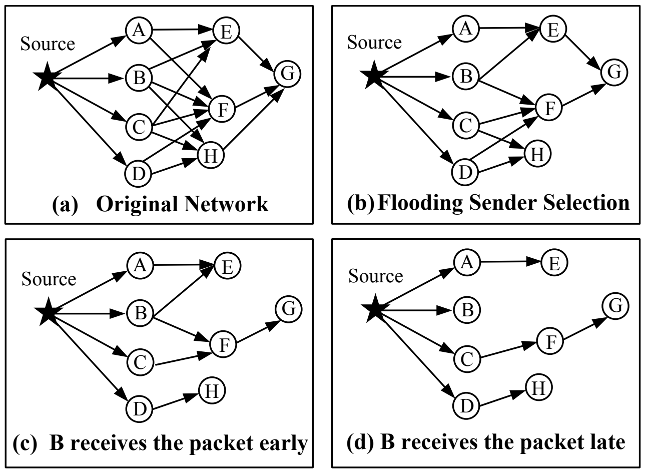

Source
A
B
C
D
E
F
G
H
(a) Original Network
(b) Flooding Sender Selection
(c) B receives the packet early
(d) B receives the packet late

Figure 7: Different Flooding Structures

4.4.2 Link-Quality-Based Backoff

Once a sender set is formed, we need to resolve the conflicts within the set. Ideally, a node with the best link quality has the highest priority to grab the channel and start a transmission with no collision. Selecting the best link always means the least number of transmissions is expected so that both the expected next-hop delay and energy cost are the smallest.

When a node intends to start a transmission, it first backs off for a period of time. The duration of the backoff depends on the link quality between the sender and the receiver. The better the link quality, the shorter the backoff duration. When multiple nodes within communication range make their decisions to send towards the same node, they back off first before transmission and the one with the best link quality starts first. Other nodes, after backing off for enough time, listen to the channel first and can catch the ongoing transmission. They will then abort their own transmission and mark transmission to this node as *Redundant*.

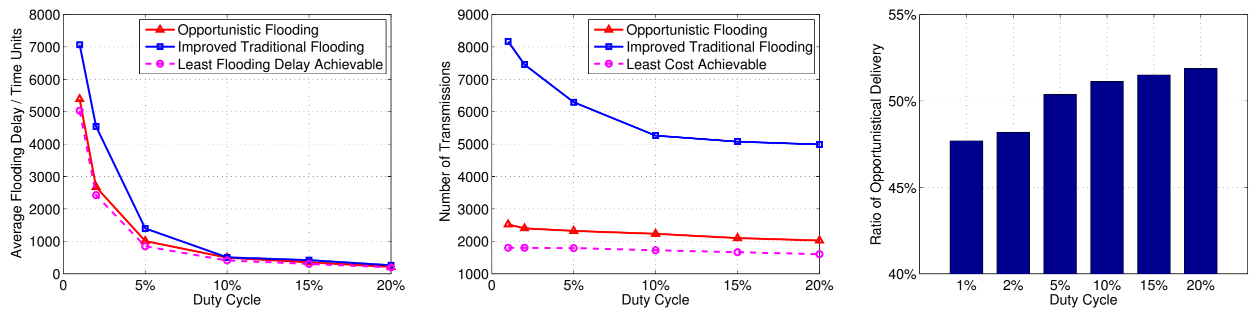
Specifically, suppose the backoff time bound is $T_{backoff}$ and the maximum size of sender set is W. $T_{backoff}$ can be divided into W slots for different backoff durations. A sender could compute its backoff duration $t_{backoff}$ using the following equation:

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

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

By using the link-quality-based backoff method, we reduce not only collisions but also the chance that a packet is forwarded via a very weak link, since the winner must have a relatively good enough link quality to start early.

Fig.7 shows an example of the whole design of opportunistic flooding. The original network is shown in (a). After the selection of flooding senders, all the nodes in the same sender set should have good-enough links between them. After the flooding sender selection, the selected links of each node are shown in (b), where some links are disabled compared to (a). For different flooding packets, each node makes forwarding decisions for different neighbors; (c) and (d) are two examples of different decision making results according to B's different packet delay.



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

Figure 8: Flooding Performance in Networks with Different Duty Cycles

5. EVALUATION

This section simulates the performance of opportunistic flooding. Specifically, we compare the flooding delay and the energy cost in our design with those of traditional flooding. We also show that the performance of our design is very close to optimal performance, which is the best performance achievable by any flooding design. In Section 6, we provide further evaluation through a physical testbed consisting of 30 MicaZ nodes.

5.1 Simulation Setup

We use randomly-generated network graphs to evaluate our design. In the simulation, the network size varies from 200 nodes to 1000 nodes. The links between these nodes are wireless path loss channels combined with shadowing effects as in [49]. Unless otherwise explicitly specified, the parameters are set as $l_{th} = 0.7$ for flooding 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. The simulation results are based on 10 network topologies and 1000 flooding packets for each topology.

5.2 Performance Metrics

We measure the *flooding delay* 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 some inherent randomness and can be considered a random variable. Here we propose to use the average flooding delay as a measure of network performance.

Energy consumption is measured by the total number of transmissions for a single flooding packet. 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 we compare different flooding designs under the same duty-cycled schedules.

5.3 Baseline I: Optimal Performance Bounds

We compare our opportunistic flooding method with the best performance achievable by any possible flooding design. For energy costs, the optimal solution (the one with the least number of transmissions) is obtained when the flooding packets are only forwarded via the energy optimal tree. For flooding delay, the optimal solution (the one with the least flooding delay) is obtained by pure flooding with an oracle collision-free media access control. We note that the optimal energy and optimal delay are achieved by two different flooding methods, respectively. In other words, neither of them can achieve optimal delay and energy simultaneously.

5.4 Baseline II: Improved Traditional Flooding

To our knowledge, there is no distributed flooding algorithm specially designed for low-duty-cycle networks; therefore, besides comparing our design with optimal performance as described above, we have to compare our design with a variation of traditional flooding. We remember that, as shown in Section 2, the performance of traditional flooding deteriorates dramatically when the duty cycle of a network is significantly reduced.

To make comparison as fair as possible, we modify the traditional flooding method to make it more efficient for low-duty-cycle networks and name it *improved traditional flooding*. First, it uses the same link-quality-based backoff method as our design to avoid collisions when multiple senders are within communication range. This modification also ensures that the nodes with the best link quality have the greatest chance to start a transmission and that the energy is optimized. Second, a node stops the transmission to a certain neighbor if another node with a better link quality grabs the channel. This modification reduces a great amount of redundant transmission. Third, the hidden terminal problem is alleviated by using a p-persistent backoff scheme after a fixed number of retries. By adding these three techniques, the traditional flooding resolves a greater percentage of collisions itself, saves redundant energy cost, and recovers from the hidden terminal problem quickly.

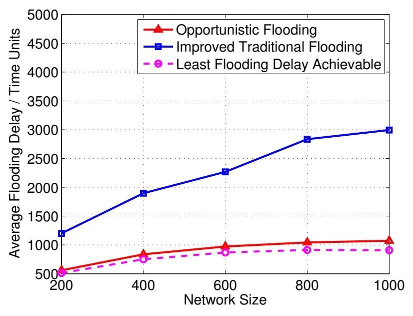
5.5 Performance Comparison

This section compares opportunistic flooding with optimal performance bounds and improved traditional flooding.

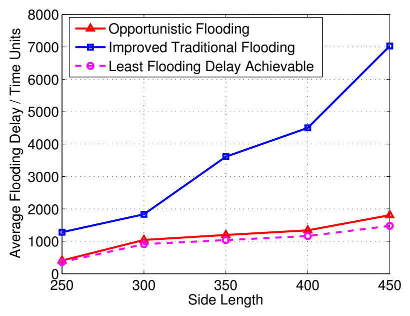
5.5.1 Different Duty Cycles

We first evaluate the performance in networks with different duty cycles. In this simulation, 800 nodes are generated randomly on a $300m \times 300m$ field.

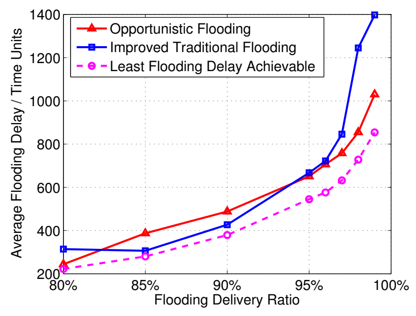
Fig.8(a) and (b) plot the flooding delay and energy cost of the opportunistic flooding, improved traditional flooding and optimal solutions. In Fig.8(a), the average flooding delay of opportunistic flooding is only around 80% of that of the improved traditional flooding and is very close to the optimal solution. In Fig.8(b), our design costs less than 50% of 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 only 0.5 redundant packets on average. Fig.8(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 tra-



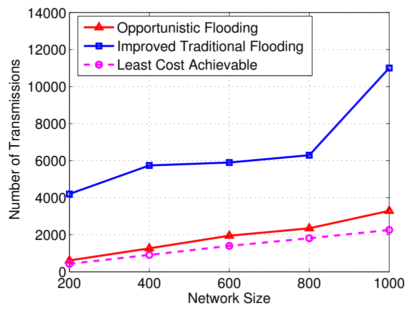
(a) Flooding Delay vs. Network Size



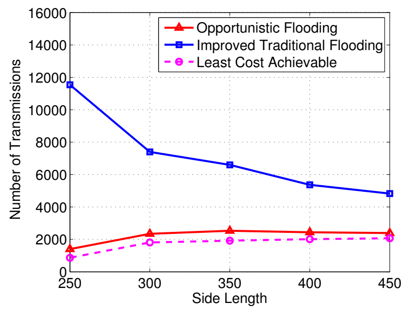
(a) Flooding Delay vs. Density



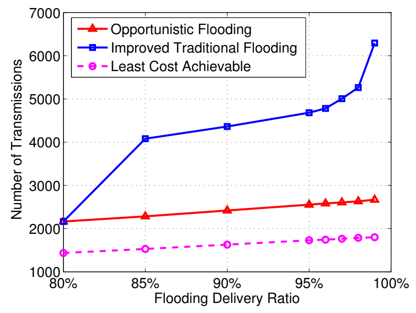
(a) Flooding Delay vs. Delivery Ratio



(b) Energy Cost vs. Network Size



(b) Energy Cost vs. Density



(b) Energy Cost vs. Delivery Ratio

Figure 9: Network Size

Figure 10: Network Density

Figure 11: Flooding Delivery Ratio

ditional 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.

5.5.2 *Different Network Sizes and Densities*

We also evaluate the performance of our design in different network settings as shown in Fig.9 and Fig.10. For different network sizes from 200 to 1000, the side length of the area changes from $200m$ to $400m$ to keep a similar density. For different network densities, the network size is fixed at 800 while the side length of the area changes from $250m$ to $450m$.

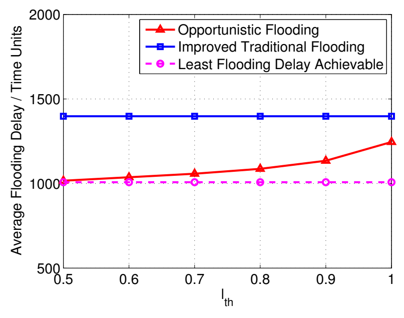
In Fig.9, the average flooding delay and energy cost increases as the network size increases, as expected. Again, the opportunistic flooding outperforms the improved traditional flooding and saves about 40% of flooding delay and 50% of energy cost. It is very close to the optimal performance, with around 10% more delay and energy cost. For different network densities as shown in Fig.10, the average flooding delay of both methods increases. This is because with less density, the average number of neighboring nodes decreases. Also the link quality becomes worse since the average distance between neighboring nodes is longer. This also explains why opportunistic flooding has a higher energy cost when density decreases. An interesting observation regarding Fig.10(b) is that the energy cost of the improved traditional flooding decreases when the network is more sparse as traditional flooding suffers from a great number of collisions in dense networks.

5.5.3 *Different Delivery Ratio Requirements*

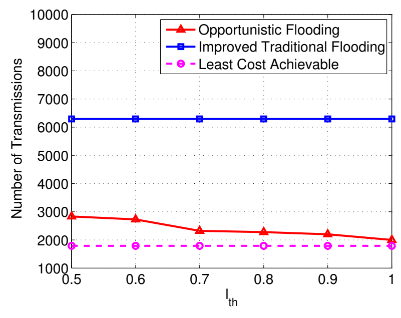
We compare the performance when a different flooding delivery ratio is required. The flooding delivery ratio is the percentage of nodes that have received a flooding packet. Again, the 800-node network with a 5% duty-cycle is used. The flooding delay and energy cost are recorded when only a certain percentage of nodes have received the packet.

The average flooding delay and energy cost are plotted in Fig. 11. As shown in Fig.11(a), when the flooding delivery ratio is below 95%, the average flooding delay of both methods is very close to the optimal delay and that of the traditional flooding is slightly better. However, when the required delivery ratio goes to 99%, the delay for traditional flooding grows significantly. The flooding delay of opportunistic flooding, on the other hand, grows but is still very close to the least possible delay. The flooding delay of traditional flooding grows quickly when the required flooding delivery ratio increases because in a randomly generated topology, some nodes may have a very low degree (i.e., they have very few neighboring nodes). When the duty-cycle is low and the links are not perfect, delivering a flooding packet to these nodes takes a much longer time. Similarly, when many nodes are trying to communicate with the same node and the links between them are unreliable, it is very difficult for the traditional flooding to recover from collisions. That is why the flooding delay in traditional flooding becomes much longer than that of opportunistic flooding when a high delivery ratio is required, although in a lower delivery ratio it performs well (ignoring its higher energy cost).

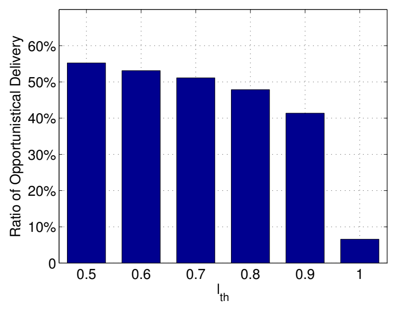
Fig.11(b) shows the corresponding energy cost, noting that this energy cost represents only part of the energy cost in a whole flooding process since the counting of transmissions stops when certain percentage of nodes have received the packet. In this figure, the opportunistic flooding still consumes less than half of the energy of improved traditional flooding. In the figure, the energy cost of opportunistic flooding grows linearly when the delivery ratio approaches 1 while the energy cost of traditional flooding grows exponentially due to collisions.



(a) Flooding Delay vs. l_{th}

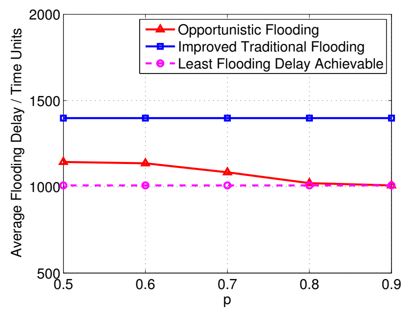


(b) Energy Cost vs. l_{th}

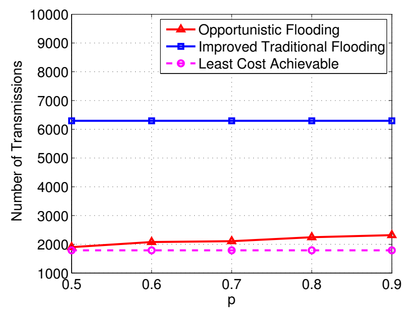


(c) Opportunistic Delivery Ratio

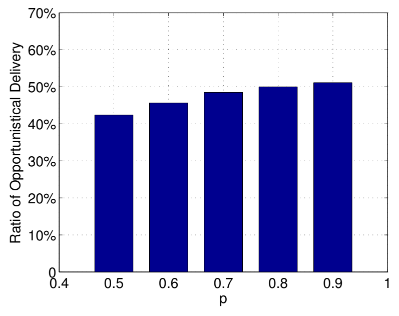
Figure 12: Flooding Performance in Networks with Different l_{th}



(a) Flooding Delay vs. p



(b) Energy Cost vs. p



(c) Opportunistic Delivery Ratio

Figure 13: Flooding Performance in Networks with Different p

5.6 Investigation on System Parameters

In the opportunistic forwarding design, 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.

5.6.1 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 5% duty-cycle. p is still fixed to 0.9 while l_{th} is changed from 0.5 to 1.0. (Noting that in order 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 average flooding delay, energy cost and opportunistic delivery ratio. 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. The choice of l_{th} in this range is a trade-off between flooding delay and 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.

5.6.2 The Quantile Probability p

We study the impact of p, the threshold to decide whether a packet is opportunistically early or not. Again, randomly generated networks consisting of 800 nodes are used in the simulation. 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, energy cost and opportunistic delivery ratio, respectively. As discussed before, 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.8, 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 comparison, we conclude that the opportunistic flooding design approaches the optimal bounds. It outperforms the 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. IMPLEMENTATION AND EVALUATION

In addition to large-scale simulations, we implemented our opportunistic flooding and the improved traditional flooding described in Section 5 on the TinyOS/Mote platform in nesC with 30 MicaZ motes to further validate our design in practice.

6.1 Experiment Setup

We randomly deployed 30 MicaZ nodes on an in-door test-bed as shown in Fig.14. The transmission power at MicaZ motes 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 stand-alone base station node and corresponding GUI

Figure 14: Experimental Setup

interface. 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 count ready and a neighbor table built with working schedules from all neighboring nodes. Followed by neighbor discovery, a node begins to measure the pair-wise link quality between itself and each neighboring node in its neighbor table and exchange this information with neighboring nodes. Consequently, the neighbor table of each node would contain both incoming and outgoing link qualities for all neighboring nodes. With such information, the *pmf* is computed and the $p-$quantile is shared with neighbors. After such an initialization phase, all nodes transit to low-duty-cycle mode. They turn on and turn off their radios based on their working schedules. Specifically, in this experiment we set the unit time as $50ms$.

6.2 Performance Comparison

In this section, we compare the empirical flooding delay and energy consumption for both flooding methods. For each specified duty cycle, the source sends 100 flooding 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.

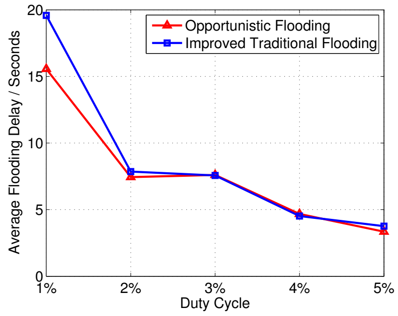


Figure 15: Flooding Delay vs. Duty Cycle

6.2.1 *Delay Performance*

We investigated the impact of duty-cycle on delay as shown in Fig.15. 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.15 does not show the similar significant delay reduction that we observed in the 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 effectiveness of opportunistic flooding in terms of delay. Second, a pure-flooding algorithm is considered delay-optimal when a network is not congested. In a four-hop network of 30 nodes, the congestion level is not as significant as that observed in the simulation.

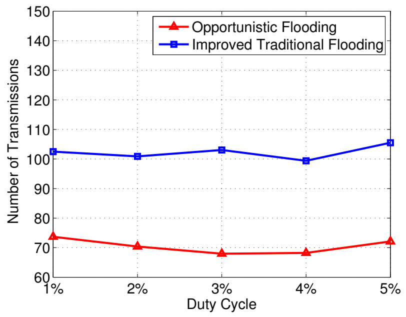


Figure 16: Avg. Energy vs. Duty Cycle

6.2.2 *Energy Performance*

Fig.16 compares average energy consumption from which we can see that our design has a much lower energy cost than improved traditional flooding. For example, at a duty cycle of 3%, the average flooding delay for our design and improved traditional flooding are 7603ms and 7564ms, respectively. The energy costs, however, are 68 and 103, which means our design saves about 34% in energy cost while providing a similar average flooding delay.

6.3 Why Opportunistic Flooding is Better

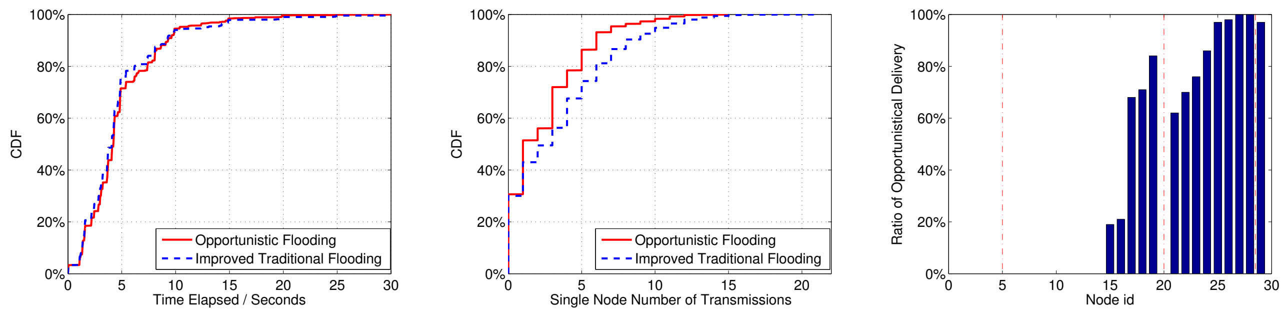
This section presents some insights into why opportunistic flooding significantly improves performance over the improved traditional flooding.

6.3.1 *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 (CDF) delay of both flooding methods in Fig.17(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 (note that this is a low-duty-cycle network). 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, which matches the simulation results given in Section 5.5.3, where the improved transitional flooding suffers an excessive delay for the last few nodes.

6.3.2 *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.17(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 three times. In contrast, the number of transmissions in improved traditional flooding is five at the same 70%. 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.



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

Figure 17: System Insights of Opportunistic Flooding

6.3.3 Observation on Opportunistic Ratio

We studied how opportunistic flooding helps reduce the flooding delay. Fig.17(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 our opportunistic flooding design is very effective in reducing flooding delay, especially when the network scale becomes large.

6.4 Performance Summary

Our system implementation and evaluation proves that opportunistic flooding is practical in real settings. We discovered that the delay reduction in a real testbed was not as significant as we had observed in the simulation. Energy savings, however, were significant even in a small network setting. In summary, opportunistic flooding is more suitable for heavy-traffic networks with a reasonable density and scale. In a small network with an insignificant amount of traffic and loose requirements on energy cost, traditional flooding can be used for the sake of simplicity.

7. RELATED WORK

As essential operations for wireless networks, multicasting [13, 15, 18, 19, 30] and flooding [4, 14, 17, 20, 21, 27, 35] have been extensively studied in the literature. Due to space constraints, we here focus only on reliability and efficiency in wireless sensor networks. For the multicasting service, RMAC [34] presents a reliable multicast service at the MAC layer using the busy tone mechanism. Mobicast [13] and GARUDA [30] provide reliable data delivery from a sink to the sensors in specified delivery regions. For flooding service, PBBF [27] proposes a MAC layer solution for flooding in sensor networks and investigates tradeoffs among flooding reliability, latency and energy consumption. Aimed at ameliorating message implosion, Smart Gossip [17] 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 [35] demonstrates high reliability for flooding with good energy efficiency on both testbed experiments and simulations. For services such as network reprogramming, protocols such as Deluge [14] and Trickle [20] also propose techniques for efficiently propagating code to nodes in the network. More recently, RBS [41] proposes a broadcast service for duty-cycled sensor networks and shows its effectiveness in reducing broadcast time and energy costs. All these previous works assume there are usually multiple neighbors available at the same time to receive the multicast/flooding message sent by a sender, which does not hold in low-duty-cycle networks.

On the other hand, opportunistic routing and data forwarding in low-duty-cycle networks have acquired a lot of attentions in recent years. In the area of opportunistic routing, EXOR [32], CBF [1], MRD [28] and OMS [2] have studied various techniques of exploiting a wireless broadcasting medium and multiple opportunistic paths for efficient message delivery. In the area of low-duty-cycle sensor networks, several existing works focus on providing delay guarantees and energy efficiency. Yu et al. introduce solutions on minimizing energy dissipation for sensor nodes under certain latency constraints [45]. Lu et al. propose various techniques for minimizing communication delay while maintaining energy-efficient node working schedules. Recently, DSF [8] offered a new data forwarding technique that optimizes data delivery ratio, end-to-end delay or energy consumption for data delivery in low-duty-cycle sensor networks. However, none of those low-duty-cycle solutions investigates the flooding service.

8. CONCLUSION

In this paper, we present opportunistic flooding: a delay-driven flooding method that is particularly designed for low-duty-cycle wireless sensor networks. To our knowledge, this is the first work on a flooding method that considers both low-duty-cycle and unreliable links. In our design, each node makes probabilistic forwarding decisions based on the delay distribution of next-hop nodes. Only opportunistic early packets are forwarded via the links outside of the energy-optimal tree to reduce flooding delays 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 evaluation on a running test-bed show that our opportunistic flooding method approaches the optimal performance and achieves a shorter average flooding delay with less than half of the energy cost of the improved traditional flooding method in various network settings. The opportunistic flooding method proposed in this paper is designed for the scenario where the working schedules of sensor nodes are fixed and decided by the network application. In the future, we shall extend this work into the scenario where working schedules can be flexibly changed to provide better flooding performance.

9. REFERENCES

[1] Q. Cao, T. Abdelzaher, T. He, and R. Kravets. Cluster-Based Forwarding for Reliable End-to-End Delivery in Wireless Sensor Networks. In *INFOCOM'07*, 2007.

[2] C. Cetinkaya and E. Knightly. Opportunistic Traffic Scheduling over Multiple Network Paths. *INFOCOM '04*, 2004.

[3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. SPAN: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. In *MobiCom'01*, 2001.

[4] X. Chen, M. Faloutsos, and S. Krishnamurthy. Power Adaptive Broadcasting with Local Information in Ad Hoc Networks. *ICNP '03*, 2003.

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

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

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

[8] Y. Gu and T. He. Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links. In *SenSys '07*, 2007.

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

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

[11] 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 Transactions on Sensor Networks*, February 2006.

[12] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A Stateless Protocol for Real-Time Communication in Ad Hoc Sensor Networks. In *ICDCS'03*, May 2003.

[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] J. G. Jetcheva and D. B. Johnson. Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks. In *MobiHoc '01*, 2001.

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

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

[18] S. J. Lee, W. Su, and M. Gerla. On-demand Multicast Routing Protocol in Multihop Wireless Mobile Networks. *Mobile Networks and Applications*, 7(6), 2002.

[19] B. N. Levine and J. Garcia-Luna-Aceves. A Comparison of Reliable Multicast Protocols. *Multimedia Syst.*, 6(5), 1998.

[20] 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.

[21] L. Li, R. Ramjee, M. Buddhikot, and S. Miller. Network Coding-Based Broadcast in Mobile Ad-hoc Networks. *INFOCOM '07*, 2007.

[22] 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.

[23] S. Lin, J. Zhang, G. Zhou, L. Gu, T. He, and J. A. Stankovic. ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks. In *SenSys'06*, 2006.

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

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

[26] 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.

[27] 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.

[28] A. K. L. Miu, H. Balakrishnan, and C. E. Koksal. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. In *MobiCom'05*, 2005.

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

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

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

[32] R. M. Sanjit Biswas. ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. In *SIGCOMM'05*, 2005.

[33] 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.

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

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

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

[37] 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`.

[38] 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.

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

[40] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *SenSys'03*, November 2003.

[41] F. Wang and J. Liu. RBS: A Reliable Broadcast Service for Large-Scale Low Duty-Cycled Wireless Sensor Networks. In *ICC'08*, 2008.

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

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

[44] 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.

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

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

[47] 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.

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

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