Taming Collisions for Delay Reduction in Low-Duty-Cycle Wireless Sensor Networks

Long Cheng*†, Yu Gu‡, Jianwei Niu*, Ting Zhu§, Cong Liu¶, Qingquan Zhang§ and Tian He‖
*State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China
†Virginia Tech, USA ‡IBM Watson Health, USA §University of Maryland, Baltimore County, USA
¶University of Texas at Dallas, USA ‖University of Minnesota, Twin Cities, USA

***Abstract*—Many-to-one data collection is a fundamental operation in wireless sensor networks (WSNs). To support long-term deployment of WSNs, sensor nodes normally operate at low-duty-cycles. However, the low-duty-cycle operation significantly reduces the communication chance between nodes. Consequently, the risk of data collisions significantly increases when multiple senders transmit packets to a receiver during its very short active period. Data collision not only results in wasted packet transmissions, but also incurs a large delivery latency. Under such conditions, collision-free medium access is more appealing than recovering after collision for low-duty-cycle WSNs.**

In this work, we propose an incast-collision-free data collection protocol, named iCore, to address the *many-to-one collision* problem in low-duty-cycle WSNs. iCore employs the dynamic forwarding technique and establishes a non-conflicting schedule for delay reduction. Specifically, we design efficient forwarder assignment and forwarding optimization algorithms that ensure low *end-to-end latency* under diverse data traffic types. Through comprehensive performance evaluations, we demonstrate that, compared with the state-of-the-art protocol, iCore effectively minimizes the end-to-end delay by $25\% \sim 57\%$ and maintains high delivery ratio and energy efficiency for different many-to-one convergecast scenarios.

I. Introduction

Wireless sensor networks (WSNs) for many long-term applications, such as military surveillance and field monitoring, are expected to last for a long time with limited energy supplies. To ensure such long-term operations, a sensor network normally operates at a low-duty-cycle mode (e.g., 5% or less) [1], where nodes stay in dormant state most of time for energy saving. For all WSNs applications, data collection (also known as many-to-one convergecast) is an important communication paradigm commonly used to gather information from individual nodes to the sink node.

Typically, data collection in multihop wireless networks operates over a tree-based routing topology [2], where a sender only forwards data packets to a single and fixed parent in the tree. This works well if wireless links are reliable and the duty-cycle operation is not employed. However, experimental results have revealed that wireless connectivity is unreliable especially for low-power embedded devices [3]. The low-duty-cycle operation combined with the unreliable links pose several challenges for efficient data collection in WSNs. First, since unreliable wireless links inevitably incur data retransmissions and the interval between consecutive retransmissions is large, low-duty-cycle WSNs suffer from a large delivery latency. While data collection in WSNs could be delay-sensitive such as for emergency response and safety-critical applications. Therefore, *reducing the end-to-end (E2E) delay* becomes a critical design objective [1] in low-duty-cycle networks. Second, the risk of data collisions increases since multiple senders have to contend for the channel in order to transmit data packets to the same receiver during its very short active state [4]. This problem further causes data retransmissions and long delivery delay, making the situation even worse. Specifically, we term such *many-to-one* data collisions as the *incast-collisions* in low-duty-cycle WSNs.

To solve the incast-collision problem in low-duty-cycle WSNs, one straightforward approach is to employ the TDMA technique [5] based on a data collection tree, where multiple senders that share a common parent node in the tree transmit data packets at different time slots to the parent node. However, scheduled medium access over a data collection tree such as traditional TDMA approach normally gives lower channel utilization and higher delays [6], especially in low-duty-cycle WSNs. On the other hand, reactive collision resolving techniques based on CSMA/CA [4], [7] cannot be applied to solve the problem, since the length of a receiver being in active state is normally very limited in low-duty-cycle WSNs. Therefore, to ensure time-efficient data collection, we are in need of a design that is able to prevent collisions while achieve *low E2E delivery latency* in low-duty-cycle WSNs.

In this paper, we present a novel design tailored for low-duty-cycle WSNs to solve the incast-collision problem. The key idea is to employ the dynamic forwarding technique while proactively avoid the incast-collisions to achieve efficient data collection in low-duty-cycle WSNs. The major contributions of this work are listed as follows:

- To address the *incast-collision* problem in general data collection scenarios, we propose the incast-collision-free data collection (iCore) protocol to proactively establish a non-conflicting schedule for efficient many-to-one convergecast in low-duty-cycle WSNs.
- We define the min-max fair assignment problem in iCore, design a recursive forwarder assignment algorithm that ensures low latency and good fairness, and develop an efficient forwarding optimization technique to minimize the expected E2E delay.

978-1-4673-9953-1/16/$31.00 ©2016 IEEE

- To improve the channel utilization, the proposed iCore allows senders to opportunistically transmit data packets to secondary (i.e., non-primary) forwarders with lower priority to access the channel once detecting idled slots. Based on the analysis of key properties of dynamic forwarding, we propose an efficient secondary forwarder assignment algorithm to further optimize the E2E latency.

The rest of the paper is organized as follows: Section II presents the network model and assumptions. Section III describes the motivation behind this work. Section IV elaborates the design overview of iCore. From Section V to Section VIII, we present the detailed description of each key components of iCore. We provide the evaluation results in Section IX. Section X surveys the related work. Finally, conclusions are drawn in Section XI.

II. Preliminaries

A. Network Model

1) Low-Duty-Cycle Model: We consider a *connected stationary multi-hop* WSN. For the sake of simplicity, we assume a single sink node is deployed for data collection purpose. However, our solution can be extended to scenarios with multiple sink nodes. Each sensor node operates at a low-duty-cycle, where the duty-cycle is defined as the percentage of time a node is active in one working cycle. All nodes have their own periodic duty-cycle schedules. These schedules are shared with one-hop neighboring nodes, so that a sender knows the rendezvous time for exchanging data with an intended receiver. A dormant node switches to the active state when (i) it is scheduled to switch to the active state, or (ii) it has some packets to transmit to a receiver that is active at that time.

The working cycle T is equally divided by fixed length time slot τ, which is normally very short due to the low-duty-cycle operation. In this work, we set the length of τ to be large enough to complete a round-trip packet transmission (including data and ACK packets). The working schedule of a node i is denoted as $\{t_1^i, t_2^i, \cdots, t_k^i\}$, where t_k^i represents the k^{th} active slot in one working cycle.

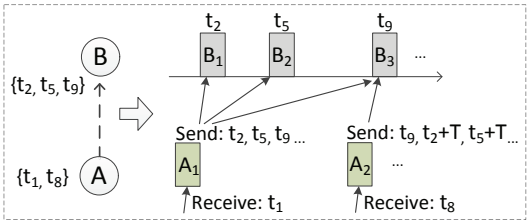


Fig. 1. Example of the time-expanded network in low-duty-cycle WSNs

2) Time-Expanded Network: To facilitate the presentation of data forwarding process in low-duty-cycle WSNs, we use the time-expanded network to represent a WSN with low-duty-cycle operations [1]. For node i with k receiving slots in one working cycle, e.g., $\{t_1^i, t_2^i, \cdots, t_k^i\}$, it can be represented by k time-expanded nodes $\{i_1, i_2, \cdots, i_k\}$, where i_k only wakes up at t_k^i to receive data. For example, in Fig. 1, node A will be extended to two time-expanded nodes A_1 and A_2. We use the terms node (or forwarder) and time-expanded node interchangeably in the rest of the paper. Then the low-duty-cycle network in one working cycle can be represented as a time-expanded network by replacing the physical nodes with the corresponding time-expanded nodes. An example of a two-node network is shown in Fig. 1. A and B's working schedules is $\{t_1^A, t_8^A\}$ and $\{t_2^B, t_5^B, t_9^B\}$, respectively. The resulting time-expanded network includes 5 time-expanded nodes, where each node corresponds to an active wake-up slot. If A_1 receives a packet at t_1, then it can forward the packet to nodes $\{B_1, B_2, B_3\}$ at time slots $\{t_2, t_5, t_9\}$ in one working cycle, respectively. Each physical node in the network is aware of its minimum number of hops away from the sink. Nodes with l hops from the sink are considered as level-l nodes. A node only forwards packets to up-level nodes (i.e., with smaller hopcount) to avoid routing loops.

B. Assumptions

1) Unreliable wireless links: Due to many factors such as interference, attenuation, and fading, wireless communication links between neighboring nodes are unreliable. We assume the MAC layer provides the link quality estimation service. There have been a lot of existing works on efficient and accurate measuring wireless link quality [8]. Since link quality is affected by many factors, it may change over time. However, it has been empirically studied in [9] that the changing rate is slow. Therefore, the measurements of the link quality can be updated at a very low frequency.

2) Local time synchronization: Our design only needs local synchronization among nodes in one-hop neighborhood to ensure the communication rendezvous between neighboring nodes. Low-cost and accurate time synchronization for WSNs has been extensively studied in the literature, e.g., GLOSSY [10] can achieve synchronization error around $0.4\mu s$, which is more than sufficient for our design.

III. Motivation

To reduce the E2E delivery delay in low-duty-cycle WSNs, the concept of *dynamic forwarding* [1], [11]–[13] has been proposed. Instead of waiting for a fixed forwarder to wake up again after a transmission failure, dynamic forwarding involves multiple forwarders to transmit a packet, so that the waiting time spent on forwarding a packet at each hop is significantly reduced. Therefore, such technique is desirable in low-duty-cycle WSNs to achieve short delivery delay. However, existing dynamic forwarding protocols [1], [11]–[13] did not consider the concurrent transmission scenarios, and thus suffering from the severe incast-collision problem. To reveal this phenomenon, we compare the modeling/estimated and actual E2E delays when applying DSF [1] for many-to-one data collection traffic scenario in 1% duty-cycle WSNs, where every node sends a packet to the sink node in each data reporting period (*e.g.*, every 10 minutes [14]). Details of the simulation setup can be found in Section IX.

Fig. 2 shows there exists a large gap between the modeling and actual delays under many-to-one data traffic. The incast-

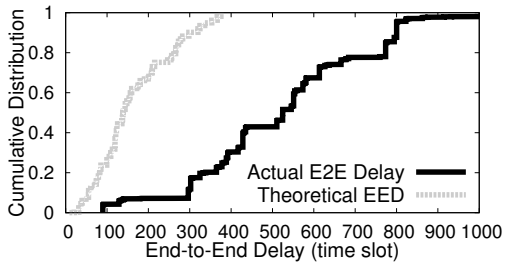


Fig. 2. Dynamic forwarding (DSF) suffers the incast-collision problem for many-to-one data collection scenarios

collisions incur increased E2E delay by 300% compared with the theoretical EED. Moreover, our initial simulation studies indicate that incast-collisions result in less than 70% delivery ratio in DSF. To address the incast-collision problem, in this work, we propose a novel dynamic forwarding protocol with proactive collision resolving for time-efficient data collection in low-duty-cycle networks.

IV. Design Overview

Overall, the design of iCore consists of the following four components, as illustrated in Fig. 3.

* **Dynamic Forwarding with Sequence** (Section V). Data forwarding in low-duty-cycle WSNs is essentially a discrete process. Each node maintains a *forwarding sequence* sorted in the order of wake-up time of potential forwarders, where a forwarder corresponds to one wake-up slot. Fig. 3(a) shows an example of the data collection topology. The corresponding time-expanded network is shown in Fig. 3(b), where B has been extended to two time-expanded nodes B_1 and B_2. The forwarding sequence of B_1 is shown in Fig. 3(c). Suppose B_1 receives a packet at t_2 and then forwards it to the sink. It has a sequence of communication chances to up-level nodes $\{D_2, C_1, C_2, D_1, \cdots\}$ at time slots $\{t_3, t_7, t_8, t_0{+}\mathrm{T}, \cdots\}$.
* **Primary Forwarder Assignment** (Section VI). When a forwarder appears in the forwarding sequences of multiple senders, there exists a potential possibility of incast-collision. In order to provide performance guarantee, we distribute available forwarders (i.e., forwarding chances) to different senders to achieve a collision-free transmission schedule. If node i can transmit packets to a forwarder with the highest priority, we call this forwarder as a *primary forwarder* of i, where the priority is implemented by adjusting the backoff time for channel accessing [6]. For example in Fig. 3(d), we assign primary forwarders $\{C_1\}$ and $\{C_2, D_1, D_2\}$ to $\{A_1\}$ and $\{B_1, B_2\}$, respectively, where B_1 and B_2 are associated with the same physical node B. In other words, $\{A_1\}$ and $\{B_1, B_2\}$ are the primary owners of forwarding slots $\{t_7\}$ and $\{t_0, t_3, t_8\}$, respectively.
* **Optimizing Primary Forwarding Sequence** (Section VII). After the primary forwarder assignment, a node owns a sequence of non-conflicting primary forwarders. We optimize the primary forwarding sequence so that the expected E2E delay is minimized. In the example in Fig. 3(e), B_1 will ignore forwarder D_1 (i.e., slot t_0) from the optimal forwarding subsequence to optimize the latency.
* **Secondary Forwarder Assignment** (Section VIII). If the primary owner of a slot has no data to transmit, to improve channel utilization in the time domain, our design allows other neighbors to opportunistically transmit data once detecting the slot is unused. If a sender can only transmit packets to a forwarder with a low priority, we call it as a *secondary* (i.e., non-primary) forwarder to this sender. In particular, we propose an effective secondary forwarder assignment algorithm to further reduce the E2E latency at each node. Fig. 3(f) shows an example of the secondary forwarder assignment. If involving C_1 in B_1's optimal forwarding subsequence can potentially decrease B_1's E2E delay, C_1 will be assigned to B_1 as a secondary forwarder. As a result, B_1 can opportunistically access the channel if detecting A_1 has no data to transmit to C_1 at slot t_7. Finally, B_1 obtains an optimal non-conflicting forwarding subsequence, including the *primary and secondary* forwarders.

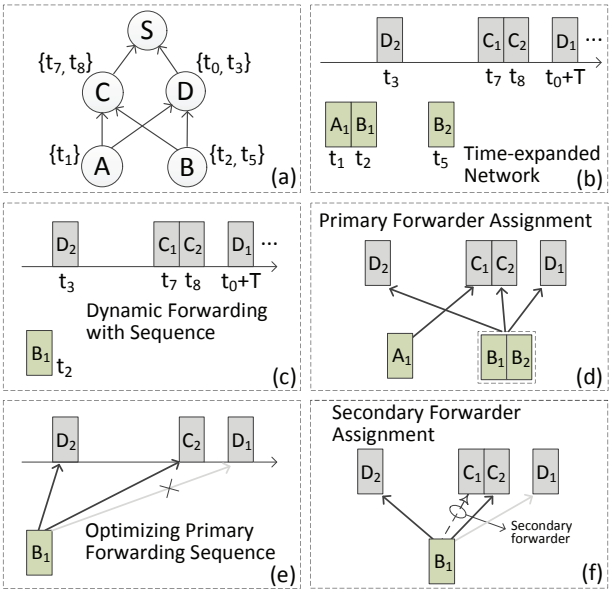


Fig. 3. Design overview of iCore

V. Dynamic Forwarding with Sequence

A. Basic Procedure

The dynamic forwarding procedure is as follows: node i has a sequence of neighbours $S^i = \{j_1, j_2, \cdots\}$ that can forward packets from i to the sink. We call S^i as the *forwarding sequence* of i. Once node i starts to send a packet, it turns on the radio and tries to send the packet at the time intervals when nodes in S^i wake up in sequence. Under practical low-duty-cycle WSNs, it is necessary to avoid that a sender retransmits a packet endlessly. Therefore, we set the maximum time bound T_{max} (a tunable system parameter) for a sender to transmit a specific packet at each hop. As a result, we have a finite forwarding sequence, in which every forwarder wakes up before the deadline T_{max} from the moment a sender begins to transmit the packet. The transmission is repeated until node

i receives an ACK from one of the forwarders or it reaches the end of the sequence and then drops the packet.

B. Dynamic Forwarding Modeling

For a sending node i, let EDR^i denote the expected packet delivery ratio (EDR), and EED^i denote the expected E2E delay (EED) for the packets sent by node i and received by the sink. Given a known forwarding sequence S^i ($|S^i| = n$) and the corresponding link qualities (i.e., packet reception ratio), we have the probability that forwarder j_k in S^i successfully receives the packet (after k-1 failed attempts), denoted by P_k^i.

$$P_k^i = p_{ij_k} \cdot \prod_{m=1}^{k-1}(1 - p_{ij_m}) \quad (1)$$

where p_{ij_k} is the bi-directional link quality (i.e., the probability of a successful round-trip transmission of data and ACK) between sender i and forwarder j_k, $k \in [1, n]$.

Let EDR^{j_k} and EED^{j_k} denote j_k's EDR and EED, respectively. Given the EDR and EED values associated with each forwarder in S^i, we can derive EDR^i and EED^i as Eq. 2 and Eq. 3, respectively.

$$\text{EDR}^i = \sum_{k=1}^{n} P_k^i \cdot \text{EDR}^{j_k} \quad (2)$$

$$\text{EED}^i = \sum_{k=1}^{n} \frac{P_k^i \cdot \text{EDR}^{j_k}}{\text{EDR}^i} \cdot (d_k + \text{EED}^{j_k}) \quad (3)$$

where $\frac{P_k^i \cdot \text{EDR}^{j_k}}{\text{EDR}^i}$ is the conditional probability that a packet forwarded by j_k finally arrives at the sink. d_k is the *sleep latency* between sender i and forwarder j_k, which is the time that a sender spends on waiting for a forwarder to wake up.

C. Obtaining Forwarding Sequence

The process of obtaining the forwarding sequence starts from the sink (i.e., the level-0 node) and propagates throughout the network *level by level*. The sink node s is always awake and its forwarding sequence is empty, we have $\text{EDR}^s = 100\%$ and $\text{EED}^s = 0$. Once level-1 nodes receive the broadcasted EDR^s and EED^s from the sink, they individually calculate and then broadcast their EDR and EED values to level-2 neighbors. Similarly, once a level-l node has received messages announcing the EDR and EED values from all level-$(l-1)$ neighbors, it calculates its own EDR and EED according to Eq. 2 and Eq. 3, respectively, and then broadcasts its EDR and EED values to level-$(l+1)$ neighbors. This recursive calculation of EDR and EED can be implemented at individual nodes in a distributed manner. Finally, each node obtains a *full forwarding sequence* and the corresponding EDR and EED values of each forwarder in the sequence.

VI. Primary Forwarder Assignment

A. Design Objective

To avoid the performance degradation due to incast-collisions, a non-conflicting transmission schedule based on dynamic forwarding is needed for time-efficient many-to-one data collection. In our design, each node in level-l will be assigned to a specific level-$(l+1)$ node. Since there is a one-to-one relationship between a node and a slot in time-expanded networks, *the forwarder assignment is equivalent to the slot assignment*. Note that multiple time-expanded nodes may share the network interface of one physical node. Each forwarder assigned to a time-expanded node is also owned by other time-expanded nodes who share the same physical node.

Given a finite forwarding sequence, there exists a subset of the forwarders that can achieve the *optimal/minimal* EED, where the optimization technique will be detailed in Section VII. The primary forwarder assignment operation inevitably increases a node's achievable optimal EED, since the size of forwarding sequence is decreased thus reducing the searching space for the optimal EED. When assigning primary forwarders, we need to consider the following factors. For many WSN applications, *reliable and timely delivery* of data packets is crucial to guarantee the network performance. It is desirable to minimize the EED with a certain EDR constraint in a low-duty-cycle WSN, where the EDR constraint is a tunable system parameter. In allocating forwarders, *fairness* is also a key concern. Therefore, the *design goal* of our primary forwarder assignment is to ensure low EED and good fairness among nodes under a certain EDR constraint.

B. Problem Formulation

Given an EDR constraint, let EED^i_{ideal} denote the optimal EED of node i before its primary forwarder assignment, and EDR^i_{ideal} denote the corresponding EDR. (The calculation of EED^i_{ideal} with a given forwarding sequence will be introduced in Section VII). EED^i_{ideal} is calculated without considering any potential incast-collision, and thus it can be considered as node i's lower bound EED. We define EED *Inflation Rate* of node i (denoted as EIR^i) as the ratio of EED^i_{ach} divided by EED^i_{ideal}, where EED^i_{ach} is the achievable optimal EED based on i's available primary forwarding sequence excluding those slots that have been assigned to other nodes.

We consider the problem of min-max fair forwarder assignment, where the objective is to maintain a lower inflation rate (EIR) for any level-$(l+1)$ node when allocating level-l primary forwarders to level-$(l+1)$ nodes. Suppose there are N level-l nodes/slots indexed from j_1 to j_N to be assigned to M level-$(l+1)$ nodes indexed from i_1 to i_M. For the sake of discussion convenience, we assume that a physical node has been extended to multiple time-expanded nodes, i.e., the number of available level-l slots to be assigned is several times larger than the number of level-$(l+1)$ physical nodes. Actually, when the number of available slots is not enough to accommodate all senders even in a one-to-one basis, we just extend the working cycle length (e.g., merging two or more working cycles) so that each node has multiple active slots in one working cycle. As a result, the number of available slots will be larger than the number of senders. We have the problem formulation as follows:

$$\min \max \frac{\text{EED}^i_{ach}}{\text{EED}^i_{ideal}}, \quad (4)$$

where $\text{EIR}^i = \frac{\text{EED}^i_{ach}}{\text{EED}^i_{ideal}}$ denotes the EED inflation rate of any level-$(l+1)$ node i ($i \in [i_1, \cdots, i_M]$). Note that a level-l node can only be assigned to one level-$(l+1)$ physical node as a primary forwarder to avoid potential collisions, which serves as the constraint of this problem. Apparently, the min-max fair forwarder assignment problem can be transformed into a

mixed-integer nonlinear programming problem, which is NP-hard [15] (The detailed proof is omitted due to space limit). In the following subsection, we will present our heuristic design to solve the problem.

C. A New Metric

Let U_l denote the set of unassigned level-l forwarders, initially containing all level-l nodes. Let F^i denote the available level-l forwarders to a level-$(l+1)$ node i, including the already assigned forwarders to node i and all i's neighboring nodes in U_l. Given forwarding sequence F^i with an EDR constraint, denote F^i_{opt} as the optimal forwarding subsequence that achieves the minimal EED searching from F^i, $\text{EED}^i(F^i_{opt})$ as the optimal EED value, and $\text{EDR}^i(F^i_{opt})$ as the corresponding EDR value. Let $\text{EED}^i(\{F^i-\{j\}\}_{opt})$ represent the achievable optimal EED given F^i but excluding the forwarder j ($j \in F^i$). The corresponding EDR is denoted as $\text{EDR}^i(\{F^i-\{j\}\}_{opt})$, which is not necessarily larger than the EDR constraint. We define the *utility* of forwarder j towards node i as the change of the *normalized* EIR before and after excluding forwarder j from the available forwarder set F^i, denoted as $\text{UTIL}(i,j)$.

$$\text{UTIL}(i,j) = \frac{\frac{\text{EED}^i(\{F^i-\{j\}\}_{opt})}{\text{EDR}^i(\{F^i-\{j\}\}_{opt})} - \frac{\text{EED}^i(F^i_{opt})}{\text{EDR}^i(F^i_{opt})}}{\frac{\text{EED}^i_{ideal}}{\text{EDR}^i_{ideal}}}, \qquad (5)$$

where $F^i-\{j\} \neq \emptyset$, and the normalized EIR refers to the EIR considering respective EDR values, as shown in Eq. 5. If $F^i-\{j\}=\emptyset$, we set $\text{UTIL}(i,j)$ a large default value, in order to get higher chance for node i to obtain forwarder j.

$\text{UTIL}(i,j)$ reflects the *importance* of forwarder j towards node i, which is always a non-negative value. For example, if $\text{UTIL}(i,j)=0$, it means j is not in i's optimal forwarding subsequence. A large value of $\text{UTIL}(i,j)$ means that the normalized EIR can be reduced a lot if assigning forwarder j to i. The rationale of our design is, we assign a forwarder j in U_l which can help a level-$(l+1)$ node i to decrease its EIR value by the greatest extent in each iteration of the primary forwarder assignment.

D. The Proposed Algorithm

The primary forwarder assignment algorithm is described as following. Each forwarder in U_l calculates the utility towards level-$(l+1)$ neighboring nodes according to Eq. 5. In each iteration, the forwarder with the maximal utility value is selected and assigned to the corresponding level-$(l+1)$ node, followed by updating U_l. If two forwarders have the same UTIL values regarding one level-$(l+1)$ node, we randomly assign the level-$(l+1)$ node to one of the two forwarders. This process is repeated until all available level-l forwarders are assigned to level-$(l+1)$ nodes.

Fig. 4 illustrates an example to explain how this recursive assignment works. As shown in Fig. 4(a), assume each physical node has two active slots and thus is extended to two time-expanded nodes. Fig. 4(b) shows that level-l nodes/slots $\{j_1, j_2, j_3, j_4\}$ will be assigned to level-$(l+1)$ nodes $\{i_1, i_2, j_3, i_4\}$. In the first round as shown in Fig. 4(c), suppose $\text{UTIL}(i_2, j_3)$ has the maximal utility.

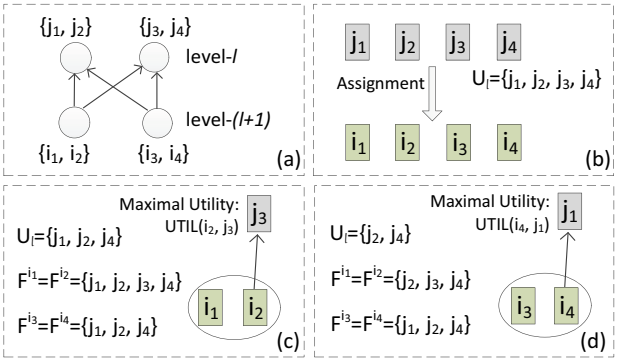


Fig. 4. Example of the primary forwarder assignment

Thus, we assign j_3 to i_2. Since i_1 and i_2 belong to the same physical node, j_3 is actually assigned to both i_1 and i_2. We update $U_l = \{j_1, j_2, j_4\}$, $F^{i_1}=F^{i_2}=\{j_1, j_2, j_3, j_4\}$ and $F^{i_3}=F^{i_4}=\{j_1, j_2, j_4\}$. As shown in Fig. 4(d), in the next round, j_1 is assigned to i_4 since $\text{UTIL}(i_4, j_1)$ has the maximal utility. The assignment process is continued until U_l is empty.

The primary forwarder assignment propagates from level to level, and the process is repeated until all nodes in the network are covered. When the level-l's primary forwarder assignment to level-$(l+1)$ nodes is completed, each node in level-$(l+1)$ calculates its own EDR, EED and EIR values, then broadcasts the updated information to neighbors. The distributed version of this forwarder assignment can be implemented such as using timer-based contention and suppression level-by-level. For example, a level-$(l+1)$ node with a higher UTIL value would broadcast the request message for a specific level-l slot earlier and suppress other candidates. For the computation complexity, the assignment in one level can be accomplished with $O(N^2M)$, where N and M is the number of level-l nodes and level-$(l+1)$ nodes, respectively.

VII. Optimizing Primary Forwarding Subsequence

A. Basic Idea

After the primary forwarder assignment, node i has a sequence of allocated primary forwarders, defined as *primary forwarding subsequence* S^i_p. However, forwarding over the original primary forwarding subsequence may not lead to the minimal EED. Under a EDR constraint R, we can optimize the primary forwarding subsequence in terms of EED, to achieve timely and reliable data delivery.

Based on the analysis of the key properties of forwarding sequence, we present a dynamic programming based efficient algorithm to optimize EED in the following subsection. This algorithm is also used for the calculation of optimal EED in Section VI. The basic idea is to break down the original problem into sub-problems. Given the last node j_{last} in the optimal primary forwarding subsequence of node i, we search the optimal subsequence backwardly, denoted as $S^i_{opt_p}(j_{last})$. We try every node in S^i_p as the last node in the optimal subsequence and compare the corresponding EED values, then we choose the subsequence with the least EED as the optimal primary forwarding subsequence, denoted as $S^i_{opt_p}$.

B. The Key Property

We first elaborate the key properties of forwarding sequence, then we present the description on how to search an optimal subsequence backwardly. Without loss of generality, suppose we have selected at least one node into $S^i_{opt_p}(j_{last})$, next we decide whether a new node j_k should be included or not. Since the forwarding sequence is always sorted in the order of wake-up times, we have $t(i)<t(j_k)<\cdots<j_{last}$. Including j_k into $S^i_{opt_p}(j_{last})$ needs to meet one of the following two conditions: either improving EED optimality (i.e., $\text{EED}^i(S^i_{opt_p}(j_{last})\cup\{j_k\}) < \text{EED}^i(S^i_{opt_p}(j_{last}))$), or increasing EDR without sacrificing the optimality of EED.

Property 1: Given a forwarder j_k and the current primary forwarder subsequence $S^i_{opt_p}(j_{last})$, where j_{last} is the last node, and j_k wakes up earlier than any node in $S^i_{opt_p}(j_{last})$. j_k should be selected into $S^i_{opt_p}(j_{last})$ when meeting either condition below:

i) $d_{j_k}+\text{EED}^{j_k}<\text{EED}^i(S^i_{opt_p}(j_{last}))$; (where d_{j_k} is the sleep latency between sender i and forwarder j_k)

ii) $\text{EDR}^{j_k}>\text{EDR}^i(S^i_{opt_p}(j_{last}))$, if EED does not change.

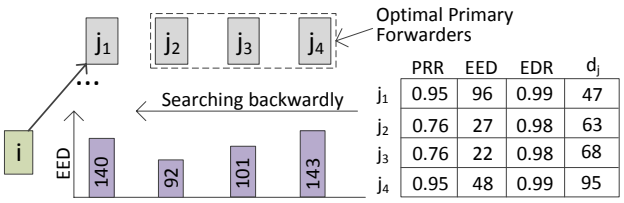
The detailed proof is omitted due to space limit, which is derived from EDR and EED's definitions.

Property 1 gives the sufficient conditions for adding j_k to $S^i_{opt_p}(j_{last})$, which can help to quickly filter out ineligible nodes when searching for the optimal primary forwarding subsequence backwardly, thus reducing the computational complexity. The optimality of this backward searching algorithm holds due to the existence of an overlapping optimal substructure $S^i_{opt_p}(j_{last})$, and the decision for including or excluding a later item in $S^i_{opt_p}(j_{last})$ does not affect the decisions made earlier.

C. The Proposed Algorithm

The overall primary forwarding subsequence optimization for node i is described as follows. We try every node in S^i_p as the last node in $S^i_{opt_p}$. We proceed to add any node that improves the subsequence's optimality according to Property 1. However, after one round searching with a fixed last node, it is possible that $\text{EDR}^i(S^i_{opt_p}(j_k))$ is less than the EDR constraint R. In this case, we keep adding nodes backwardly until meeting the EDR constraint or no node is left. We record the subsequence that yields lower EED and the corresponding EED value (denoted by EED^i_{min}) in each round. Finally, we obtain the optimal forwarding subsequence which has the lowest EED. Note that if there is no subsequence that can meet the EDR constraint, which means the constraint might be too restrictive to be achieved, node i will transmit packets using a best-effort delivery approach, e.g., using every forwarding chance to transmit a packet. The complexity of this algorithm is $O(n^2)$, where n is i's primary forwarding sequence length.

In Fig. 5, we show an example to illustrate the backward searching method. Let us suppose node i is assigned four primary forwarders $\{j_1, j_2, j_3, j_4\}$. Each forwarder's link quality (PRR), EED, EDR and sleep latency (d_j) are listed in the figure. We set the EDR constraint as 0.95. We first



	PRR	EED	EDR	d_j
j_1	0.95	96	0.99	47
j_2	0.76	27	0.98	63
j_3	0.76	22	0.98	68
j_4	0.95	48	0.99	95

Fig. 5. Example of optimizing the primary forwarding sequence

consider j_4 as the last node in the optimal subsequence, and get $\text{EED}^i(S^i_{opt_p}(j_4))=143$. Since $d_{j_3}+\text{EED}^{j_3}=90$, involving j_3 can decrease the EED, thus we add j_3 to the optimal subsequence (i.e., $S^i_{opt_p}(j_4)\leftarrow(S^i_{opt_p}(j_4)\cup\{j_3\})$). Then, we update $\text{EED}^i(S^i_{opt_p}(j_4))=101$. Similarly, j_2 is added to $S^i_{opt_p}(j_4)$, where the updated EED is 92. However, adding j_1 increases the EED from 92 to 140, thus it will be discarded. In the first round, we find an optimal subsequence candidate $\{j_2, j_3, j_4\}$ and its EDR is above 0.98. For the remaining rounds in this example, there is no other candidate providing lower EED. Therefore, the final optimal primary forwarding subsequence $S^i_{opt_p}$ will be $\{j_2, j_3, j_4\}$.

VIII. Secondary Forwarder Assignment

A. Basic Idea

The primary forwarder assignment inevitably incurs inflated EED (i.e., increased EED compared with the optimal EED with full forwarding sequence), especially for nodes with large hopcount from the sink. In order to improve channel utilization when traffic load is low or unbalanced, our design allows secondary owners of a forwarder to transmit packets in the order of predefined priority once detecting an unused slot. Although this mechanism (also called slot multiplexing) has been proposed in the literature [6] [5], in this work, our main contribution is to efficiently *select* and *prioritize* secondary owners of a forwarder to further optimize the EED.

The basic idea is that, after level-$(l+1)$ nodes finishing optimizing their primary forwarding subsequences, level-l nodes collect the EIR and optimal primary forwarding subsequence information from level-$(l+1)$ nodes. A level-l node selects and prioritizes those level-$(l+1)$ neighbors as its secondary owners if it can potentially further decrease their EED values. Note that if a node's primary forwarding subsequence is an empty set, we set its EIR as a default value which is much larger than normal values, so that this node has higher chance to be the secondary owner of a slot.

B. The Key Property

After optimizing the primary forwarding subsequence, as introduced in Section VII, node i obtains the optimal primary forwarding subsequence $S^i_{opt_p}$. Given a nonempty $S^i_{opt_p}$, we decide whether node i selects j_k ($j_k\notin S^i_{opt_p}$) as one of its secondary forwarders. The rationale is that, if node i forwarding a packet to j_k can potentially decrease i's EED without sacrificing the EDR, the secondary forwarder j_k will be assigned to i.

Without loss of generality, we assume nodes in $S^i_{opt_p}$ are indexed from j_1 to j_n, where n is number of items in $S^i_{opt_p}$.

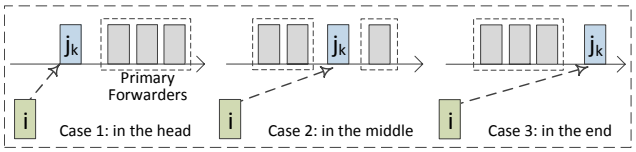


Fig. 6. Three potential positions (cases) if inserting j_k into the optimal primary forwarding subsequence $S^i_{opt_p}$ in terms of wake-up time

Since the forwarding sequence is always sorted based on the wake-up times, let $S^i_{opt_{p1}}$ denote the nodes in $S^i_{opt_p}$ that wake up earlier than j_k, and $S^i_{opt_{p2}}$ represent the nodes which wake up later than j_k. As shown in Fig. 6, there are three potential positions if inserting j_k into $S^i_{opt_p}$: 1) in the head ($S^i_{opt_{p1}}$=$\emptyset$); 2) in the middle ($S^i_{opt_{p1}}\neq\emptyset$ && $S^i_{opt_{p2}}\neq\emptyset$); or 3) in the end ($S^i_{opt_{p2}}$=$\emptyset$) of the optimal primary forwarding subsequence. Our analysis shows that the decision making *does not necessarily need* the probability that node i successfully transmits a packet to secondary forwarder j_k, which makes the secondary forwarder assignment possible without any prior knowledge of traffic loads at different nodes.

Property 2: When deciding whether j_k is a beneficial secondary forwarder to node i or not, we have following three decision making criteria:

i) For case 1, when $d_{j_k}+\text{EED}^{j_k} < \text{EED}^i(S^i_{opt_p})$ and $\text{EDR}^{j_k} \geq \text{EDR}^i(S^i_{opt_p})$, involving j_k can potentially decrease node i's EED without sacrificing the EDR (where d_{j_k} is the sleep latency between sender i and forwarder j_k);
ii) For case 2, when $d_{j_k} + \text{EED}^{j_k} < \text{EED}^i(S^i_{opt_{p2}})$ and $\text{EDR}^{j_k} \geq \text{EDR}^i(S^i_{opt_{p2}})$, j_k is a beneficial secondary forwarder to node i;
iii) For case 3, when $d_{j_k} + \text{EED}^{j_k} < \text{EED}^i(S^i_{opt_p})$, j_k should select i as its secondary owner.

We omit the proof details due to space limit, which can be derived from EDR and EED's definitions.

C. The Proposed Algorithm

The design of secondary forwarder assignment goes as follows. Let Δ denote the necessary time to detect that a channel is being used. For example, according to the current MicaZ hardware and TinyOS platform, the minimum time $192\mu s$ is achievable for detecting unused slots [5]. Assume the length of an active slot τ in our low-duty-cycle model can accommodate up to $L\cdot\Delta$ plus a round-trip packet transmission. In the last stage of the primary forwarder assignment from level-l to level-$(l+1)$, based on the gathered EIR and optimal primary forwarding subsequence information from level-$(l+1)$ nodes, a level-l forwarder will select up to L secondary owners of itself, according to the decision making criteria in Property 2. In addition, owners of a forwarder should be able to overhear from each other to avoid hidden terminal problems. After that, the selected secondary owners are prioritized *in the descending order* of their EIR values. The computational complexity of this algorithm is $O(nD)$, where n is the length of optimal primary forwarding subsequence and D represents the density of low-level neighbors.

D. Incast-collision-free Forwarding

After all the steps above, each node maintains an optimal non-conflicting forwarding subsequence, which is composed of sequential primary forwarders and secondary forwarders. Subsequently, the *incast-collisions are proactively avoided* in a low-duty-cycle WSN. When a node has data to transmit, it just follows the dynamic forwarding procedure to forward packets towards the sink as introduced in Section V.

As most transmission scheduling works with considering unreliable links, our design philosophy is that the relatively high initial overhead will be eventually compensated by reduced delivery delay and improved energy efficiency. The overhead incurred by the forwarder assignment (either in the initialization phase or the periodic link quality updating phase considering link dynamics in practice) is expected to be *amortized* over a reasonably long period of network operation [6].

IX. Evaluation

We mainly compare the performance of our iCore with DSF [1], which is one of the most cited practical forwarding protocols known to the community in low-duty-cycle WSNs. For a fair comparison, DSF is implemented with the MAC-layer binary exponential backoff collision resolution scheme.

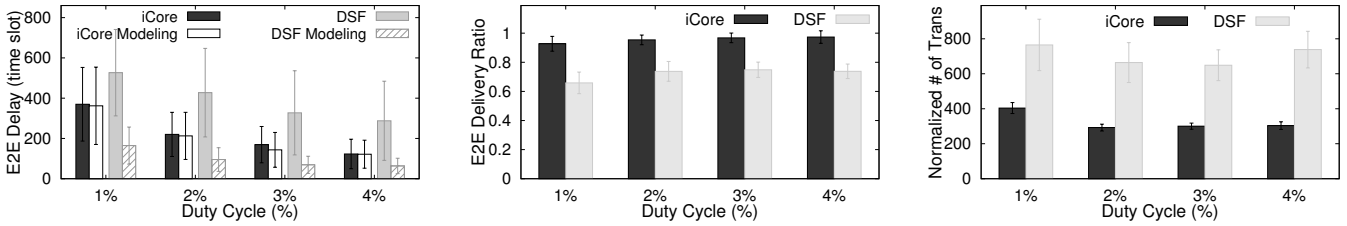
A. Simulation Setup

We deploy 100 nodes in a $150m\times150m$ area and randomly generate the network topology, using the ns-2 [16] simulator. The node transmission range is set as $30m$. Wireless links qualities are derived from the Nakagami fading model, where link quality is related with the distance between two nodes. In all simulations, the sink node is positioned at bottom left ($0m$, $0m$). We set the working cycle length T equal to 300τ. An active slot τ includes up to 5Δs and can complete a round-trip data and ACK transmissions, which means a slot can be assigned to up to 5 secondary owners. The EDR constraint is set to 0.95, and the maximum time bound for a sender to transmit a specific packet at each hop is equal to one working cycle T. The results have been averaged over 100 runs, and the related standard deviations are provided as error bars.

We mainly study the *aggregated convergecast*, where packets are fully aggregated at each hop. Such type of data collection is generally applicable when a spatial correlation exists in the data, or the goal is to collect summarized sensory data such as MIN, MAX, MEDIAN, COUNT, AVERAGE [2]. We consider two types of data traffic: 1) event-based data collection, where an event is sensed in a contiguous region centered at the event; and 2) periodic data collection from all nodes to the sink (*e.g.*, every 10 minutes [14]).

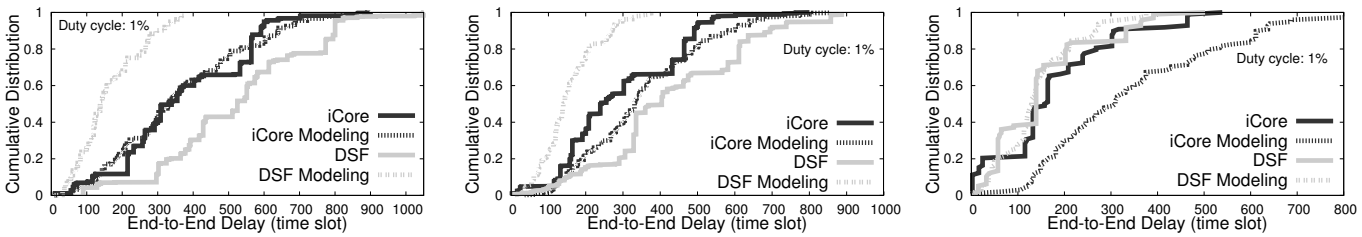
B. Performance Metrics

We use the following main metrics for comparison. *1) End-to-End Delay*: the time elapsed from a packet being sent out by the source node to the sink receives it. *2) Delivery Ratio*: the ratio of the number of packets received by the sink to the total number of packets sent by source nodes. To calculate the delivery ratio, in our simulation, the source node IDs



(a) End-to-End Delay (b) Delivery Ratio (c) Energy Efficiency

Fig. 7. Performance comparison for periodic data collection



(a) Periodic Data Collection (b) Event-based Data Collection (c) Single Source-to-Sink Communication

Fig. 8. Performance comparison in three different data traffic types

are stored in the header of an aggregated data packet. *3) Normalized Number of Transmissions*: the total number of data transmissions divided by the delivery ratio. This metric reflects the *energy efficiency* of a forwarding protocol.

C. Evaluation Results

1) Impact of Duty Cycles: In this test, we evaluate the impact of duty cycles on the performance for periodic data collection in low-duty-cycle WSNs, where each node generates a packet in the beginning of every data reporting period and transmits it to the sink node.

Fig. 7(a) illustrates the average E2E delay under different duty-cycles. iCore reduces E2E delay by $29\% \sim 57\%$ on average compared with DSF as the duty cycle varies. DSF derives the theoretical EED for single source-to-sink communication, which serves as the lower bound of E2E delay. Since the EED model in DSF does not consider the potential incast-collisions, there are large gaps between the modeling results and the actual delays. While the analytical modeling results are relatively close to the actual values in iCore. As duty cycle increases, the interval between consecutive active slots (i.e., sleep latency) is decreased. Consequently, both iCore and DSF show decreased E2E delay.

Fig. 7(b) reports delivery ratio results as duty cycle increases. iCore achieves above 95% packet delivery ratio on average. However, DSF only provides around 75% delivery ratio. The is because DSF is lack of proactive response to the incast-collisions, which could cause the *performance collapse* in convergecast scenarios with low-duty-cycle operations.

Fig. 7(c) depicts the performance comparison on normalized number of transmissions. Since the backoff collision resolution mechanism can only partially alleviate the incast-collision problem, DSF suffers lower energy efficiency in case of a high traffic load. While iCore effectively reduces retransmissions through preventing the incast-collisions.

2) Impact of Data Traffic Types: We then investigate the performance of our design in three different data traffic types. For the event-based data collection, in each simulation, the event center is randomly selected in the sensing field, and the event sensing range is set to $30m$. Once an event is detected, nodes in the event sensing range will generate messages to be sent to the sink. We also evaluate iCore for the single source-to-sink transmission scenario where the traffic load is light. In this test, the duty cycle is set to 1% in all cases.

Fig. 8(a) plots the CDF curves of the E2E delay in periodic data collection, where the traffic load is much higher than other traffic types. The gap between the actual delay and the modeling EED in DSF is very large, while iCore efficiently provides performance guarantee under high traffic load.

Fig. 8(b) shows the CDF of E2E delay in the event-based data collection with unbalanced traffic loads. Compared with Fig. 8(a), as the data traffic load decreases, the gap between the actual delay and the modeling EED in DSF becomes smaller. An interesting observation regarding Fig. 8(b) is that, the actual delay of iCore shows around 19% improvement over its modeling EED. This is because, iCore models the EED without considering the secondary forwarding chances, thus serving as a theoretical upper bound of EED. The results of iCore indicate that, *under unbalanced traffic loads, the slot multiplexing mechanism effectively improves the channel utilization*, thus providing shorter E2E delay.

Fig. 8(c) reports the E2E delay under a very light data traffic load, where only one single packet is transmitted from a randomly generated source node to the sink. It shows that DSF's EED modeling is quite accurate for the single source-to-sink communications. However, we also observe that, with slot multiplexing mechanism, iCore can achieve a comparable performance to DSF when the network is lightly loaded (*i.e.*, almost no concurrent transmissions that lead to collisions).

X. Related Work

Data forwarding in low-duty-cycle WSNs has recently attracted much attention. The existing contributions in this field mainly focus on either unicasting [1], [13], [17] or broadcasting [3], [18]–[20]. However, none of these existing solutions address the incast collision issues for general many-to-one data collection in low-duty-cycle WSNs. ORW [12] and ORPL [21] are two recent opportunistic routing protocols designed for duty-cycled WSNs. CD-MAC [22] is a contention detectable MAC designed for duty-cycled networks. These works adopt different duty-cycle models from iCore. Many solutions have been proposed to deal with packet collisions in WSNs, such as Funneling-MAC [23], ZMAC [6], TreeMAC [24] and TDMA-ASAP [5]. Unfortunately, these protocols can not be directly applied in low-duty-cycle WSNs. Strawman [4] presents a reactive contention resolution mechanism after a collision is detected at the receiver side. Nevertheless, Strawman requires large contention window length [7], which makes it unsuitable for low-duty-cycle WSNs with fixed-length short active period. More recently, mZig [25] has been proposed to enable a receiver to simultaneously decode multiple packets from different transmitters in sensor networks. This technique can be used to solve the incast collision problem in low-duty-cycle WSNs at the cost of complicated decoding process.

Unlike existing interference-free transmission scheduling works, e.g., [5], [26], this paper focuses on dealing with the *many-to-one* incast-collision problem rather than the *many-to-many interference* in low-duty-cycle WSNs. Actually, owning to the low-duty-cycle setting, the many-to-many interference happens less frequently than the many-to-one incast-collision [3]. iCore can be extended to achieve more efficient slot assignment if given prior knowledge of traffic loads at different nodes. However, obtaining accurate traffic pattern at each node is nontrivial, and may either be infeasible or incur significant overhead. Therefore, in this work, iCore is designed to be generally applicable under different traffic loads *without assuming any knowledge of traffic loads* in the network.

XI. Conclusion

In this work, we propose iCore, a novel design tailored for low-duty-cycle WSNs to solve the incast-collision problem. iCore employs the dynamic forwarding technique and proactively establishes a non-conflicting schedule for efficient data collection in low-duty-cycle WSNs. We design efficient forwarder assignment and forwarding sequence optimization algorithms that ensure low latency and good fairness. Through comprehensive performance comparisons, we demonstrate that iCore effectively minimizes E2E delay and maintains a high delivery ratio and energy efficiency in the case of a high traffic load, while achieves a comparable performance to the optimal latency when the network is lightly loaded.

Acknowledgements

This work was supported in part by the 973 Program (2013CB035503), National Natural Science Foundation of China (61300174, 61572060, 61170296, 61190125), the R&D Program (2013BAH35F01), China Postdoctoral Science Foundation (2013M530511, 2014T70026), NSF grants CNS-1503590, OISE-1427824 and CNS-1527727.

References

[1] Y. Gu and T. He, "Dynamic switching-based data forwarding for low-duty-cycle wireless sensor networks," *IEEE Trans on Mobile Computing*, vol. 10, no. 12, pp. 1741–1754, 2011.

[2] O. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Trans on Mobile Computing*, vol. 11, no. 1, pp. 86–99, 2012.

[3] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic flooding in low-duty-cycle wsns with unreliable links," in *MobiCom '09*.

[4] F. Österlind, L. Mottola, T. Voigt, N. Tsiftes, and A. Dunkels, "Strawman: resolving collisions in bursty low-power wireless networks," in *IPSN '12*.

[5] S. Gobriel, D. Mosse, and R. Cleric, "Tdma-asap: Sensor network tdma scheduling with adaptive slot-stealing and parallelism," in *ICDCS '09*.

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

[7] X. Ji, Y. He, J. Wang, W. Dong, X. Wu, and Y. Liu, "Walking down the stairs: Efficient collision resolution for wireless sensor networks," in *INFOCOM '14*.

[8] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: A survey," *ACM Trans on Sensor Networks*, vol. 8, no. 4, 2012.

[9] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "Understanding the Causes of Packet Delivery Success and Failure in Dense Wireless Sensor Networks," in *RTSS'09*, 2009.

[10] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *IPSN'11*.

[11] Y. Cao, S. Guo, and T. He, "Robust multi-pipeline scheduling in low-duty-cycle wsns," in *INFOCOM '12*.

[12] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power,low delay: Opportunistic routing meets duty cycling," in *IPSN'12*.

[13] Z. Cao, Y. He, and Y. Liu, "L2: Lazy forwarding in low duty cycle wireless sensor networks," in *INFOCOM '12*.

[14] Y. Liu, X. Mao, Y. He, K. Liu, W. Gong, and J. Wang, "CitySee: not only a wireless sensor network," *IEEE Network*, vol. 27, no. 5, pp. 42–47, 2013.

[15] I. Grossmann, "Review of nonlinear mixed-integer and disjunctive programming techniques," *Optimization and Engineering*, vol. 3, no. 3, pp. 227–252, 2002.

[16] "Network simulator." [Online]. Available: http://www.isi.edu/nsnam/ns/

[17] L. Cheng, J. Niu, Y. Gu, T. He, and Q. Zhang, "Energy-efficient statistical delay guarantee for duty-cycled wireless sensor networks," in *SECON '15*, 2015, pp. 46–54.

[18] W. Feng and L. Jiangchuan, "On reliable broadcast in low duty-cycle wireless sensor networks," *IEEE Trans on Mobile Computing*, vol. 11, no. 5, pp. 767–779, 2012.

[19] L. Cheng, Y. Gu, T. He, and J. Niu, "Dynamic switching-based reliable flooding in low-duty-cycle wireless sensor networks," in *INFOCOM '13*.

[20] J. Niu, L. Cheng, Y. Gu, J. Jun, and Q. Zhang, "Minimum-delay and energy-efficient flooding tree in asynchronous low-duty-cycle wireless sensor networks," in *WCNC '13*, 2013, pp. 1261–1266.

[21] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree bloom: Scalable opportunistic routing with orpl," in *SenSys '13*, 2013.

[22] D. Liu, X. Wu, Z. Cao, M. Liu, Y. Li, and M. Hou, "Cd-mac: A contention detectable mac for low duty-cycled wireless sensor networks," in *SECON '15*, 2015, pp. 37–45.

[23] G.-S. Ahn, S. G. Hong, E. Miluzzo, A. T. Campbell, and F. Cuomo, "Funneling-MAC: A localized, sink-oriented mac for boosting fidelity in sensor networks," in *SenSys '06*, 2006, pp. 293–306.

[24] W.-Z. Song, R. Huang, B. Shirazi, and R. LaHusen, "Treemac: Localized tdma mac protocol for real-time high-data-rate sensor networks," in *PerCom'09*.

[25] L. Kong and X. Liu, "mzig: Enabling multi-packet reception in zigbee," in *MobiCom '15*, 2015, pp. 552–565.

[26] B. Malhotra, I. Nikolaidis, and M. A. Nascimento, "Aggregation convergecast scheduling in wireless sensor networks," *Wirel. Netw.*, vol. 17, no. 2, 2011.