Reducing Idle Listening during Data Collection in Wireless Sensor Networks

Aram Rasul
Department of Computer Science
University of Leicester
Email: amrr1@le.ac.uk

Thomas Erlebach
Department of Computer Science
University of Leicester
Email: te17@le.ac.uk

***Abstract*—Data collection is one of the predominant operations in wireless sensor networks. This paper focuses on the problem of efficient data collection in a setting where some nodes may not possess data each time data is collected. In that case, idle listening slots may occur, which lead to a waste of energy and an increase in latency. To alleviate these problems, successive-slot schedules were proposed by Zhao and Tang (Infocom 2011). In this paper, we introduce a so-called extra-bit technique to reduce idle listening further. Each packet includes an extra bit that informs the receiver whether further data packets will follow or not. The extra-bit technique leads to significantly reduced idle listening and improved latency in many cases. We prove that every successive-slot schedule is also an extra-bit schedule. We then consider the special case of linear networks and prove that the optimal length of a successive-slot schedule (or extra-bit schedule) is $4N - 6$ time slots, where $N \geq 3$ is the number of nodes excluding the sink. Then the proposed extra-bit technique is compared with the successive-slot technique with respect to the expected amount of idle listening, and it is shown that the extra-bit technique reduces idle listening substantially.**

I. Introduction

Advances in technology have led to the invention and development of small wireless devices, including sensors [1] which have the capability of sensing, processing, computing and communication. Wireless Sensor Networks (WSNs) have captured considerable attention in the research community recently for their numerous applications in various areas [2], ranging from military applications to fire detection, healthcare, environmental monitoring, habitat monitoring and others [3], [4]. WSNs are inherently multi-hop radio networks [1] that are comprised of hundreds or thousands of sensors [3], which are dispersed either randomly in an inhospitable area or deployed manually in a specific area. During deployment, sensors configure themselves in an ad-hoc fashion, which is a common mode of operation in WSNs [1], without installing any infrastructure, and communication happens either via single-hop or multi-hop dissemination, depending on the distance between the sensors.

The requirements for data gathering are application-dependent. In some applications, such as when the sink needs to determine the maximum temperature in a specific area, it is not important that each data packet is delivered to the sink individually. In this case, the sink does not care about all the data; if all nodes send their data to the base station, the result is an over-consumption of energy due to the number of transmissions. In this case, aggregation can be used to forward only aggregated values to the sink [4]. However, there are also many applications in which all packets are important individually. For instance, when sensors are deployed for structural monitoring or leak detection, packets need to be collected from all sensors (collection without aggregation) to learn the conditions at each individual sensor location, otherwise the exercise could be a failure [5]. In this paper, we consider the latter setting, i.e., data collection without aggregation.

Sensors are tiny devices that are resource-constrained, for example, having limited power, small memory, relatively slow processors, or limited transceivers. Clearly, WSNs are likely to be powered by disposable batteries [3]; when these are exhausted, they cease to function. Therefore, careful control of energy consumption is a very significant factor in WSNs and poses many challenges.

In the literature, several reasons have been pointed out as the cause for wasted energy. The first one is collisions, which occur when two or more nodes try to transmit their data to the same destination node simultaneously. Indeed, the packets collide with each other and the destination node cannot receive either one correctly [6]. As a result, retransmission is needed, which leads to the use of more energy to send out the packets again. Moreover, both primary and secondary conflicts which cause collisions must be avoided. Primary conflicts happen either when a node transmits and receives at the same time, or several nodes send out their packets simultaneously to the same destination. Secondary conflicts occur when a receiver is within the transmission range of its sender and other simultaneous senders at the time of collision. When this happens, the receiver cannot receive the correct packet successfully. The second source of energy wastage is overhearing, which happens when a node hears a packet which is destined for another node. The third case where energy can be wasted, and the most relevant to our work, is idle listening. Idle listening occurs when a node is listening to the channel to receive a packet but there is the possibility that the sending node does not have data and remains silent [6], [7], [8].

Hence, several issues should be addressed and considered carefully when sensors are deployed, such as reducing schedule length, reducing interference, and saving energy. To this end, many algorithms and techniques have been proposed and examined recently.

The contributions of this paper are as follows:

- Extending earlier work on successive-slot schedules [9], [10], we propose an optimization technique,

called the extra-bit technique, which reduces idle listening further and also minimizes latency.

- We prove that the optimal number of time slots for data collection in a chain using successive-slot or extra-bit schedules is $4N - 6$, where $N \geq 3$ is the number of nodes in the network excluding the sink.
- We show how to calculate the expected amount of idle listening for extra-bit schedules and successive-slot schedules in chains and trees where each node has data with a fixed probability, and we demonstrate that the expected amount of idle listening is significantly smaller with the extra-bit technique.

The rest of this paper is organized as follows. Section II discusses related work, while Section III defines our system model, gives the problem definition, and describes the previous successive-slot technique. In Section IV, we present the extra-bit technique, show that successive-slot schedules and extra-bit schedules are the same family of schedules, and present and analyze an optimal scheduling algorithm for chains. After that, a comparison is presented between the extra-bit technique and the previous approach in Section V. Finally, Section VI concludes the paper.

II. Related Work

WSNs have penetrated into various fields for different purposes. During data collection many problems arise due to resource constraints and pose numerous challenges. Therefore, several algorithms and techniques have been presented to tackle these challenges from different angles.

The most relevant articles which pertain to our work are [9], [10] and [11]. Zhao and Tang [9], [10] consider data collection in trees in a setting where not all sensors have data in each round, the schedule must be independent of which sensor nodes have data, and the goal is to reduce idle listening and latency. They aim to conserve energy and extend the lifespan of the network. They present a technique called the *successive-slot technique*, in which all transmissions of a node must be made in successive slots starting from the first slot during which the node is scheduled to transmit by the schedule. In particular, parent nodes cannot transmit before their child nodes, and often parents need to listen to their child nodes one more time after their last transmission, in order to detect that no more data will come from these child nodes. Then the parent node can be switched off for all the slots during which it is scheduled to listen to these child nodes for the remainder of the data collection.

In [11], Gandham et al. consider minimum latency scheduling for data collection in trees, in a setting where all nodes have data. They show that the minimum schedule length for data collection in linear networks is $3N-3$, where N is the number of nodes in the chain, excluding the sink. Their schedule is not a successive-slot schedule. Furthermore, they also show an upper bound of $\max\{3n_k - 1, N\}$ on the schedule length for multi-line networks, where n_k is the length of the longest line connected to the sink, and N is the total number of nodes in the network. For tree networks, they also show an upper bound of $\max\{3n_k - 1, N\}$, where n_k is the number of nodes in the largest one-hop sub-tree of the root. Similarly, scheduling for data collection has also been addressed by Choi et al. [12]. They show a lower bound of $3(N-2)$ time slots for collecting data in a chain, which matches the result from [11] because they take N as the number of nodes including the sink.

Dai et al. [13] also address data collection, considering a multi-sink setting for multi-lines and using a variable interference model, compared to constant interference distance of two hops as in the above-mentioned articles. Incel et al. [5] explain that if all interference is mitigated between nodes then data collection can be performed in $\max\{2n_k - 1, N\}$ time slots. They also suggested using different frequencies for data converge-cast. Haibo et al. [14] address data collection scheduling in a model with different frequencies. Moreover, they prove a lower bound of $2N - 1$ slots for the chain and show that the maximum number of frequencies that are required is $N/2$.

III. System Model, Arbitrary Schedules, Successive-Slot Schedules

A. System Model and Arbitrary Schedules

Consider a sensor network with a tree topology. The tree network is represented as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges in the network. In other words, V represents sensors and E represents communication links between these sensors in the tree network. The sink is the root of the tree.

We assume that all nodes have a single omnidirectional transceiver, and all communication among sensors is performed over a single unique frequency channel. Furthermore, a node cannot send and receive a packet at the same time. It cannot receive a packet successfully when it hears several packets simultaneously. That is to say, both primary and secondary conflicts must be avoided for successful transmission. We consider TDMA schedules, where time is divided into a number of slots of equal length. In each slot several packets can be scheduled, but no conflicting transmissions can be scheduled in the same time slot.

We want to collect data from the sensor nodes in a way that minimizes the total number of time slots, while also reducing idle listening as much as possible. The aim is to achieve fast data collection and conserve energy. We assume that not every sensor will have data to be collected in each round of data collection. Nevertheless, we require a schedule that is independent of which nodes have data: the same schedule must be followed no matter whether all nodes have data or only some nodes have data. If a transmission from node v to its parent p is scheduled in a particular time slot but no data is available at v to be sent to p, there will either be idle listening (i.e., p listens for a transmission from v, but v remains silent) or, if p already knows that no data will be sent from v in this time step, the transceivers of v and p can be switched off (and energy saved).

For a node $v \in V$, we denote by T_v the set of nodes of the subtree rooted at v, and by $|T_v|$ the cardinality of that set. The set of children of node v is denoted by $C(v)$.

A schedule S of length (or latency) K is a mapping of time-slots $1, 2, \ldots, K$ to sets of transmitting nodes, where $S(t)$ is the set of nodes scheduled for transmitting in time slot t. As

every transmission is from a node to its parent, the schedule does not need to specify the receivers of the transmissions. The sink will never be a transmitting node. A schedule S is *feasible* if it satisfies the following conditions:

- (C1) For every t, the nodes in $S(t)$ can transmit simultaneously without conflict. This means that no two nodes in $S(t)$ have distance two or less in the tree G.
- (C2) Every node v (apart from the sink) is scheduled exactly $|T_v|$ many times for transmission.
- (C3) For $i > 1$, if the i-th transmission of node v is scheduled in time slot t, then at least $i-1$ transmissions of children of v must have been scheduled before time slot t.

Condition (C1) models interference constraints. Condition (C2) is required because each node v must transmit its own data and the data of all the other nodes in its subtree T_v. Condition (C3) expresses that node v cannot send more data than its own data and the data it has already received from its children.

We refer to such a feasible schedule as an *arbitrary schedule* as no further restrictions are imposed.

If not all nodes have data, the behavior of a node is as follows. Whenever the node is scheduled to transmit in the current time slot, it checks whether it has any data (either its own data or data received from a child) that has not yet been sent to the parent. If so, it uses the current time slot to forward any such data to the parent. Otherwise, it remains silent in the current time slot. We refer to this node behavior as *local greedy*.

B. Successive-Slot Schedules

As observed by Zhao and Tang [9], [10], arbitrary schedules can cause a lot of idle listening if not all nodes have data. For example, if a node has 10 time slots for transmitting data to its parent, but only 3 nodes in its subtree have data, then there will in general be 3 time slots with transmissions and 7 time slots with idle listening. The parent usually cannot turn off its transceiver to avoid idle listening as it cannot predict whether a packet will be sent by the child in the current time slot or not. Zhao and Tang therefore propose a restricted type of schedule, which they call *successive-slot schedule*. The special property of successive-slot schedules is that all transmissions from a node to its parent will happen in successive slots starting from the first slot that is assigned to the node for transmission, provided that local greedy scheduling is used in each node. A node cannot cause idle listening at the parent in between two actual transmissions from the node to its parent. Formally, if node v is scheduled to transmit to its parent in slots $t_1, t_2, \ldots, t_{|T_v|}$ and if r of the nodes in T_v have data, the transmissions from v to its parent must be made in time slots $t_1, t_2, \ldots, t_r$. The advantage of successive-slot scheduling is that as soon as the parent detects that the child is silent in a transmission slot, it knows that no further transmissions from that child will arrive. Therefore, the parent can switch off its transceiver in all remaining time slots where that child is scheduled to transmit. Similarly, the sink will know that data collection has been completed as soon as each child v of the sink has either sent $|T_v|$ data packets or has been silent in one time slot. This means that data collection can potentially be completed earlier (before the end of the full schedule S). Therefore, successive-slot scheduling can reduce idle listening (as there can be at most one time slot with idle listening for each parent-child pair) and schedule latency.

Zhao and Tang prove in [9] that successive-slot schedules can be characterized as follows.

Lemma 1 ([9]): A feasible schedule S is a successive-slot schedule if and only if the following condition holds:

- (C3′) For each node v and each $1 \leq i \leq |T_v|$, the i-th transmission of node v is scheduled after the i-th transmission of each child c of v with $|T_c| > i$, and after the last transmission of each child c of v with $|T_c| \leq i$.

Observe that condition (C3′) implies condition (C3).

Successive-slot scheduling can be applied to data collection in any tree network. In more general networks, a data collection tree can be determined in a first step, and successive-slot scheduling can then be applied to that tree (but the interference constraints would be derived from the full network).

In a successive-slot schedule, the process of data collection starts from leaf nodes and proceeds towards the root node (sink). Generally speaking, a node must listen for transmissions from its child nodes until an idle transmission occurs. Receiving idle listening from any child node guarantees the end of transmission from that node, which allows the parent to turns off its transceiver for any further scheduled transmissions from that node. Each time the parent node has listened to all its child nodes and has received at least one packet, it can make one transmission to its own parent node. This continues until it has received idle listening from all of its child nodes (or the child nodes have completed all their transmissions). Finally, it will transmit the remaining packets to its own parent. This process continues for all nodes until the specified sink node has stopped listening to all its children, implying that all packets have been received.

We observe that if some node in the subtree T_v does not have data in the current round of data collection, the parent p of node v needs to listen to v one more time than the number of actual transmissions from v to p.

Zhao and Tang [9] propose a heuristic that aims at producing successive-slot schedules with minimum schedule length. They do not prove bounds on the worst-case schedule length produced by their algorithm compared to the optimum schedule length.

We illustrate successive-slot scheduling using the simple example shown in Figure 1. There are six nodes; the root is the sink. Suppose that only two of the leaf nodes have data, namely C and D. Firstly, B must listen to all its child nodes to check whether they have data. It is obvious that B must listen three times. In the first two time slots, B can receive data from C and D, while in the last time slot no packet is available and this results in idle listening. This means that B has only two packets. After that, A is scheduled to listen to B. In the first and second such slot, A can receive data from B. Thus, A needs to listen again to B, but in the third such slot A does not receive any packet from B; therefore, A does not have to listen to B again after the third listening and turns its

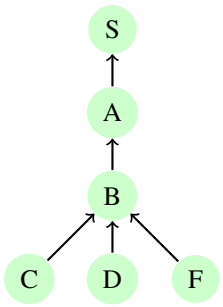


Fig. 1. Data collection, only C, D have data

transceiver off for transmissions from B in the remainder of the data collection schedule. Although in B two packets are available, A will also need to listen one more time than the number of packets, to find out whether any more packets are available. Similarly, the sink needs to listen to A three times. In the first two time slots the sink can receive two packets but in the third slot it does not receive any packet. Therefore, the sink turns its transceiver off, and the data collection is completed. In an arbitrary schedule for data collection, it can be observed that the sink should listen to A five times and A should listen to B four times. Following the successive-slot technique by Zhao and Tang [9], when a parent has listened and not received any packet from the child then it does not listen again to that child. This reduces idle listening.

IV. Extra-Bit Schedules

We propose a technique called the *extra-bit technique* that extends the successive-slot technique [9] and reduces idle listening further. In the successive-slot technique, parent nodes often have to listen to a child node one more time than the number of data packets sent by the child node, causing an idle listening slot. In the extra-bit technique, we can avoid idle listening in all cases where at least one node in the subtree has data.

The extra-bit technique adds a single extra bit to each packet, which indicates to the receiver whether this packet is the last one being transmitted by the node in the current data collection round. The value of the bit is set to 0 if the packet is the last one, indicating that no more packets will be sent by that node. This tells the receiver that it does not need to listen any more for transmissions from this node in the current round of data collection, thus avoiding idle listening. The bit is set to 1 if the packet is not the last one, which means that more packets will be sent in later time slots.

According to this technique, when a parent has listened to its child and checked the extra bit, then it can decide whether to listen again in further time slots. The process of listening will be continued until the parent receives a packet where the extra bit is 0; the parent then stops listening to its child node and switches its radio off for the rest of the schedule regarding that node, resulting in energy conservation. The only exception is if a node has no data to transmit at all; in that case, there will be one idle listening time slot for the parent.

A successive-slot schedule in which each node always has the information required to set the extra bit correctly is called an *extra-bit schedule*.

A. Equivalence of Extra-Bit and Successive-Slot Schedules

One might expect that extra-bit schedules are more restrictive than successive-slot schedules, because whenever a node sends a packet, it needs to be able to set the extra bit to a correct value. This means that the node must know whether the packet being sent is the last one or not. Somewhat surprisingly, we can show that every successive-slot schedule is an extra-bit schedule.

Theorem 1: Every successive-slot schedule S is also an extra-bit schedule.

Proof: Let S be a successive-slot schedule. By Lemma 1, S satisfies condition (C3′). We prove by induction on the height of the nodes that each node has sufficient information to set the extra bit for each transmitted packet. The claim clearly holds for leaf nodes. Now consider a node v and assume that the claim has been proved for all children of v. Consider the i-th transmission of node v, $1 \leq i \leq |T_v|$, and assume that v has a packet to transmit. By condition (C3′), each child c of v has already had at least i transmission slots if $|T_c| > i$, or has already had all its transmission slots if $|T_c| \leq i$. In the former case, node v knows whether c has at least $i+1$ packets or whether c has already transmitted all its packets. In the latter case, node v knows that c has already transmitted all its packets. Node v also knows how many packets it has already received but not yet forwarded to its parent. From this information, v can set the extra bit of the current packet correctly. ■

For any given tree network, there can be many different data collection schedules, and also many different extra-bit schedules or successive-slot schedules. We are interested in extra-bit schedules of minimum length.

B. Optimal Extra-Bit Schedules for Linear Networks (Chains)

In this section we consider WSNs where sensors are arranged as a chain, with the sink located at one end of the chain. We let N denote the number of nodes in the chain, excluding the sink. We denote the node that is i hops away from the sink by v_i, for $1 \leq i \leq N$. We also refer to v_i as the i-th node of the chain.

In the schedule we propose, a node will first wait until it receives the first packet from its child. From this time slot onward, it will make a transmission once every three steps, and nodes that are 3, 6, 9, ...hops further away from the sink will transmit simultaneously with the node. This process continues until the only nodes that still have packets are the two nodes closest to the sink. Then, these two nodes transmit their remaining packets to the sink, with only one transmission per time slot.

As an example, a chain with $N = 5$ sensor nodes and a sink node is shown together with an extra-bit schedule of length 14 in Figure 2. Note that time slot 5 is the only slot in which two transmissions take place simultaneously. For illustrative purposes, we partition the schedule into five phases, where each phase ends with a time slot in which a packet is transmitted to the sink. In the first phase, five time slots are required until the sink node receives the first packet. In the second and third phase, only three time slots are needed until the sink receives the second and third packet, respectively. Two

time slots are used in the fourth phase, and finally one time slot is used to finish this schedule.

Now suppose that only the last node E has data in a certain data collection round. Then the schedule shown in Figure 2 will complete data collection after 5 time slots at the end of the first phase (when the sink receives the packet from A with the extra-bit set to 0) and has no idle listening periods. For comparison, if the same schedule is executed as a successive slot schedule, data collection will be completed only after 8 time slots, and there will be four cases of idle listening (in steps 5 to 8).

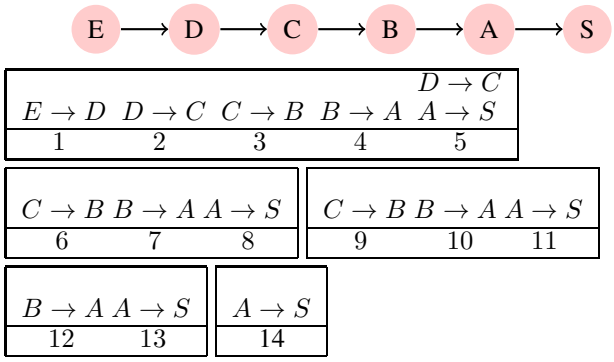


Fig. 2. Extra-bit schedule for a chain with 5 nodes

The shortest extra-bit schedules for the cases $N = 1$ and $N = 2$ can easily be seen to have lengths 1 and 3, respectively. Next, we prove that for $N \geq 3$ the optimal length of an extra-bit schedule in the chain is $4N - 6$. We first give the lower bound, and then the upper bound.

For arbitrary schedules, it has been shown in [11], [12] that $3N - 3$ time slots are required to complete a convergecast in the linear network. This lower bound can be shown by considering the three nodes closest to the sink. All transmissions by these three nodes must be scheduled in different time slots due to interference. The first node in the chain must make N transmissions, the second node $N - 1$ transmissions and the third node $N - 2$ transmissions. As a result, at least $N + (N - 1) + (N - 2) = 3N - 3$ time slots are required to complete data collection (and a schedule with $3N - 3$ time slots actually exists). We now show that extra-bit schedules require at least $4N - 6$ time slots. We remark that although extra-bit schedules are longer than the shortest arbitrary schedules, extra-bit schedules have no or substantially reduced idle listening periods if not all nodes have data.

Theorem 2: For chains with $N \geq 3$ nodes and a sink, any extra-bit schedule requires at least $4N - 6$ time slots to complete the data collection.

Proof: In the extra-bit technique, a node cannot make a transmission before it has received the first packet from its child. This implies that node v_{N-i} cannot make its first transmission before time slot $i+1$, for $0 \leq i \leq N-1$. In particular, there are $N - 3$ time slots before the first time slot in which the third node v_3 can make its first transmission. The third node must make $N-2$ transmissions, the second node $N-1$ transmissions, and the first node N transmissions (see Figure 3 for an illustration). These $3N - 3$ transmissions must all be made in different time slots, and none of them can be made

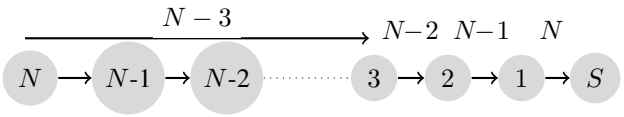


Fig. 3. Illustration of lower bound proof for chain with N nodes

Scheduling algorithm for linear network

Input: Chain with $N \geq 3$ nodes $v_1, \ldots, v_N$ and sink s
Output: $S(t)$ for $t = 1, \ldots, 4N - 6$

```
01- S(t) ← ∅ for t = 1, ..., 4N − 6
02- p(i) ← 1 for i = 1, ..., N
03- t ← 0
04- Call ScheduleFirstPart()
05- Call ScheduleRest()

06- Procedure ScheduleFirstPart()
07- For i ⟵ N down to 4
08-       t ⟵ t + 1
09-       Call parallel(i)
10- End for
11- End procedure

12- Procedure parallel(i)
13- For j ⟵ i to N increment by 3
14-       If (p(j) ≠ 0) {
                S(t) ⟵ S(t) ∪ {v_j}
                p(j) ⟵ p(j) − 1
                p(j − 1) ⟵ p(j − 1) + 1
          }
15- End for
16- End procedure

17- Procedure ScheduleRest()
18- While (p(1) ≠ 0) {
19-       For i ⟵ 3 down to 1 decrement by 1
20-             If (p(i) ≠ 0) {
                      t ⟵ t + 1
                      Call parallel(i)
                }
21-       End for
22- }
23- End procedure
```

Fig. 4. Extra-bit scheduling algorithm for linear networks

during the first $N - 3$ time slots. Therefore, the total number of time slots must be at least $(N - 3) + (3N - 3) = 4N - 6$. ■

We now present an algorithm that produces an extra-bit schedule of length $4N - 6$ for chains with N nodes and a sink. The algorithm is shown in Figure 4. First, it initializes the schedule's time slots $S(t)$ (representing the set of nodes to be scheduled at time t) to be empty, the number of packets on node v_i to $p(i) = 1$, and the current time slot t to 0. Then the procedure ScheduleFirstPart is used to schedule the first transmission of nodes from the last node to the fourth node. Whenever a node v_i is to be scheduled, t is incremented and the procedure call parallel(i) is used to schedule the node v_i as well as any nodes $v_{i+3}, v_{i+6}, \ldots$ that still have a packet. When

ScheduleFirstPart is finished, the procedure ScheduleRest is called. As long as the first node still has a packet, it repeatedly considers the nodes v_i for $i = 3, 2, 1$ and calls parallel(i) if node v_i still has a packet.

The state of the chain (i.e., the number of packets $p(j)$ stored at each node v_j) before the k-th time slot, $1 \leq k \leq 4N-8$, of the schedule produced by the algorithm is as follows:

- If $k = 4r + 1$ for $r \geq 0$: $p(j) = 0$ for $j \geq N + 1 - r$, $p(j) = 2$ for $j = N - r - 3m$ for $1 \leq m \leq \min\{r, (N - r)/3\}$, and $p(j) = 1$ for all other j.
- If $k = 4r + 2$ for $r \geq 0$: $p(j) = 0$ for $j \geq N - r$, $p(j) = 2$ for $j = N - r - 1 - 3m$ for $0 \leq m \leq \min\{r, (N - r - 1)/3\}$, and $p(j) = 1$ for all other j.
- If $k = 4r + 3$ for $r \geq 0$: $p(j) = 0$ for $j \geq N - r$, $p(j) = 2$ for $j = N - r - 2 - 3m$ for $0 \leq m \leq \min\{r, (N - r - 2)/3\}$, and $p(j) = 1$ for all other j.
- If $k = 4r + 4$ for $r \geq 0$: $p(j) = 0$ for $j \geq N - r$, $p(j) = 2$ for $j = N - r - 3 - 3m$ for $0 \leq m \leq \min\{r, (N - r - 3)/3\}$, and $p(j) = 1$ for all other j.

Theorem 3: For chains with $N \geq 3$ nodes and a sink, the algorithm shown in Figure 4 computes an extra-bit schedule of length $4N - 6$.

Proof: We observe that the schedule constructed by the algorithm has the following properties:

- Every node v_i makes $N + 1 - i$ transmissions.
- Every node v_i makes its j-th transmission only after it has received j packets from its child (or all packets from the child in case $|T_{v_{i+1}}| < j$).
- The senders of simultaneous transmissions are at least three hops away from each other, so there is no interference between them.

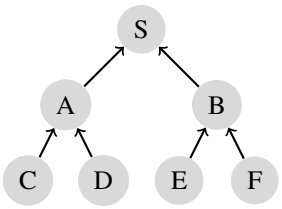
Therefore, the schedule is a feasible extra-bit schedule.

The first packet reaches the sink in time slot N. Then one packet reaches the sink every three time slots, until only the first two nodes have packets left. This requires $3(N-3)$ time slots, and at that point the first node and the second node will contain one packet each. It then requires three more time slots to transmit these to the sink. The total schedule length is therefore $N + 3(N - 3) + 3 = 4N - 6$ time slots. ■

C. Extra-Bit Schedules for Trees

Zhao and Tang [9] presented a heuristic algorithm for computing successive-slot schedules in trees. By Theorem 1, these schedules are also extra-bit schedules. We illustrate the benefits of extra-bit schedules for trees using the example in Figure 5.

In the extra-bit technique, 8 time slots are needed to complete data collection if all nodes have data. If we suppose that only C and E have data, then the beneficial impact of extra-bit scheduling can be observed: The data collection process finishes by the end of time slot 4, and there are only two occurrences of idle listening that happen when A and B listen to D and F, respectively. The sink listens to each of A and B only once and infers from receiving a packet with the extra-bit equal to 0 that no further packets will arrive. With the successive-slot technique, data collection would finish only after six time slots, and there would be four occurrences of idle listening.



C→A E→B	D→A F→B	A→S	B→S	A→S	B→S	A→S	B→S
1	2	3	4	5	6	7	8

Fig. 5. Schedule for data collection when only C, E have data

V. Idle Listening in Successive-Slot and Extra-Bit Schedules

In this section we compare extra-bit scheduling and successive-slot scheduling in terms of the amount of idle listening. We observe that the number of occurrences of idle listening in an extra-bit schedule does not depend on the particular choice of extra-bit schedule, and similarly for successive-slot schedules.

A. Idle Listening in Chains and Trees

We determine the number of occurrences of idle listening in chains and trees, both for the successive-slot technique and the extra-bit technique.

1) Chain: Consider a chain with N nodes in addition to the sink. The first node is the node closest to the sink, and the last node is the node furthest away from the sink.

With the extra-bit technique, we consider several cases for the amount of idle listening. First, if the last node has data, then there is no idle listening at all, even if some other nodes do not have data. Second, if no node has data, then there are N occurrences of idle listening. Third, if the last node has no data but some other node has data, then the amount of idle listening depends on the position of the furthest node from the sink that has data. For instance, in Figure 2, if node C has data and nodes D and E do not have data, then idle listening happens twice, once for a transmission from E to D and once for a transmission from D to C. For the general case, we can conclude that the number of occurrences of idle listening is equal to the number of nodes in the chain minus the position (distance from the sink) of the last node that has data. Let I denote the position of the last node that has data (and let $I = 0$ if no node has data). Then the number of occurrences of idle listening with extra-bit scheduling is $N - I$.

With the successive-slot technique, idle listening can be analyzed as follows. If the last node does not have data, the number of occurrences of idle listening is N as each of the N nodes will have an idle transmission to its parent. If J is the position of the last node that does not have data (or $J = 0$ if all nodes have data), then the amount of idle listening is J.

We observe that the amount of idle listening with extra-bit scheduling ($N - I$) is always less than or equal to that of successive-slot scheduling (J): If $I = N$, then extra-bit scheduling has no idle listening while successive-slot scheduling may have up to $N - 1$ occurrences of idle listening. If $I < N$, then $J = N$ and therefore $J \geq N - I$. The most extreme difference between extra-bit scheduling and successive-slot scheduling occurs if only the last node has data. In that case, extra-bit scheduling has no idle listening and successive-slot scheduling has $N - 1$ occurrences of idle listening.

2) Tree: In the extra-bit technique, idle listening happens for a transmission from a node v to its parent if and only if none of the nodes in T_v have data. The number of occurrences of idle listening is therefore equal to the number of nodes whose subtrees have no data. In the successive-slot technique, idle listening happens for a transmission from a node v to its parent if and only if at least one node in T_v does not have data. The number of occurrences is therefore equal to the number of nodes whose subtrees contain at least one node that does not have data. It is clear that idle listening for the successive-slot technique is at least the amount of idle-listening for the extra-bit technique.

For example, consider the tree in Figure 5 and suppose that only A, D, B, and F have data. With the extra-bit technique there are two occurrences of idle listening, one for the transmission from C to A and one for the transmission from E to B. With the successive-slot technique, however, there are four occurrences of idle listening: the same two as for the extra-bit technique, and in addition one for a transmission from A to S and one for a transmission from B to S.

B. Expected Amount of Idle Listening

Now, we consider a probabilistic model in which each node has data with probability p (which is the same for all nodes), and show how to calculate the expected amount of idle listening for extra-bit and successive-slot scheduling.

1) Chain: Consider a chain with N nodes $v_1, \ldots, v_N$, indexed in order of increasing distance from the sink.

With the extra-bit technique, there is idle listening for a transmission from v_i to v_{i-1} if and only if none of the nodes v_j with $i \leq j \leq N$ have data. The probability for this event is $(1-p)^{N-i+1}$. The expected amount of idle listening is therefore:

$$\sum_{i=1}^{N}(1-p)^{N-i+1} = \sum_{i=1}^{N}(1-p)^{i} = \frac{1-p-(1-p)^{N+1}}{p}$$

For example, consider the chain with five nodes from Figure 2. The expected contributions of the five nodes to idle listening are as follows:-

1) $1-p$ for node E
2) $(1-p)^2$ for node D
3) $(1-p)^3$ for node C
4) $(1-p)^4$ for node B
5) $(1-p)^5$ for node A

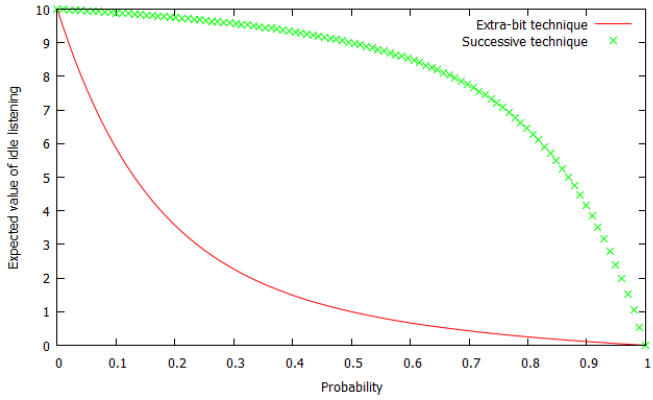


Fig. 6. Expected amount of idle listening for extra-bit and successive-slot technique in a chain with 10 nodes

The expected amount of idle listening for the chain of five nodes is therefore:

$$1-p+(1-p)^2+(1-p)^3+(1-p)^4+(1-p)^5 = \frac{1-p-(1-p)^6}{p}$$

With the successive-slot technique, there is idle listening for a transmission from v_i to v_{i-1} if and only if at least one node v_j with $i \leq j \leq N$ does not have data. The probability that all nodes v_j with $i \leq j \leq N$ have data is p^{N-i+1}. The probability for idle listening from v_i to v_{i-1} is therefore $1 - p^{N-i+1}$. The expected amount of idle listening is then:

$$\sum_{i=1}^{N}(1-p^{N-i+1}) = N - \sum_{i=1}^{N}p^{i} = N - \frac{p-p^{N+1}}{1-p}$$

For the example chain with five nodes of Figure 2, the expected amount of idle listening for the successive-slot technique is:

$$1-p+1-p^2+1-p^3+1-p^4+1-p^5 = 5 - \frac{p-p^6}{1-p}$$

Figure 6 shows how the expected amount of idle listening for extra-bit and successive-slot scheduling in a chain with 10 nodes depends on the probability p. The x-axis represents the probability p that a node has data and the y-axis represents the expected amount of idle listening. We observe that the amount of idle listening for the extra-bit technique is much smaller than for the successive-slot technique, with equality happening only for the extreme cases $p = 0$ (no node has data, 10 occurrences of idle listening) and $p = 1$ (all nodes have data, no idle listening). For a wide range of p, the extra-bit technique has a significantly lower amount of idle listening. For example, when $p = 80\%$ the expected amount of idle listening is close to zero for extra-bit scheduling but roughly 6.4 for the successive-slot technique. Starting at $p = 1$, the amount of idle listening increases sharply as p decreases with the successive-slot technique, but only slightly with the extra-bit technique.

2) Tree: Consider a tree whose set of nodes (excluding the sink) is V. With the extra-bit technique, idle listening happens for a transmission from node v to its parent if and only if none

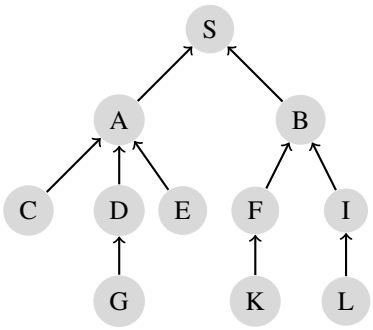


Fig. 7. Example tree for calculation of expected amount of idle listening

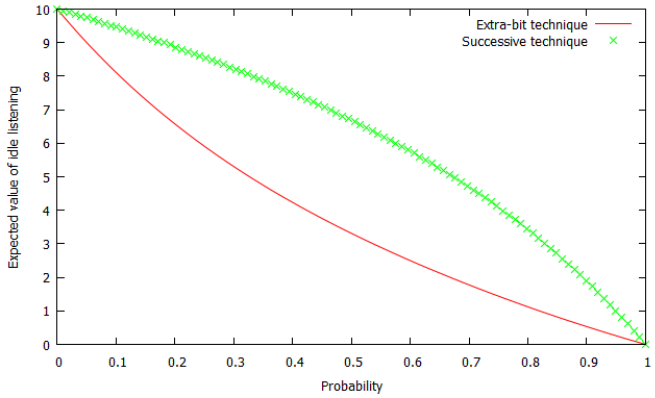


Fig. 8. Expected amount of idle listening for extra-bit and successive-slot technique in the tree of Figure 7

of the nodes in T_v have data, which happens with probability $(1-p)^{|T_v|}$. The expected amount of idle listening is therefore:

$$\sum_{v \in V} (1-p)^{|T_v|}$$

With the successive-slot technique, idle listening happens if at least one node in T_v does not have data, so the expected amount of idle listening is:

$$\sum_{v \in V} (1-p^{|T_v|})$$

For a concrete example, consider the tree shown in Figure 7. In this tree, there are five nodes with subtree size 1, three nodes with subtree size 2, and two nodes with subtree size 5. With the extra-bit technique, the expected amount of idle listening in this example is therefore $5(1-p) + 3(1-p)^2 + 2(1-p)^5$. On the other hand, the successive-slot technique has an expected amount of idle listening of $5(1-p) + 3(1-p^2) + 2(1-p^5)$.

Figure 8 shows how the expected amount of idle listening for both techniques for the tree of Figure 7 depends on p. Again, both techniques produce the same amount of idle listening only if $p = 0$ (no node has data) and $p = 1$ (all nodes have data, zero idle listening). While decreasing p from 1 to 0, the expected amount of idle listening increases more quickly for the successive-slot technique than for the extra-bit technique.

VI. Conclusion

We have proposed an optimized scheduling technique for data collection in sensor networks for scenarios where not all sensor nodes have data in each round of data collection. Our extra-bit technique is aimed specifically at increasing energy efficiency by reducing the amount of idle listening. Reduced idle listening helps to preserve energy and prolong network lifetime. Furthermore, we have shown how to construct extra-bit schedules for chains that have minimum schedule length. We have also shown how to compute the amount of idle listening in chain and tree networks for the previous successive-slot technique and for our new extra-bit technique. We have analyzed the expected amount of idle listening in a model where each node has data with probability p, showing that the extra-bit technique reduces idle listening substantially compared to the successive-slot technique.

References

[1] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *ACM Mobile Computing and Comunications Review*, vol. 6, no. 2, pp. 28–36, 2002.

[2] X. Chen, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," in *Proceedings of the First International Conference on Mobile Ad-hoc and Sensor Networks (MSN'05)*, ser. LNCS 3794. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 133–142.

[3] A. Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, and K. Jones, "Training a wireless sensor network," *Mobile Netw. Appl.*, vol. 10, no. 1-2, pp. 151–168, Feb. 2005.

[4] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

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

[6] S. Kumar and S. Chauhan, "A survey on scheduling algorithms for wireless sensor networks," *International Journal of Computer Applications*, vol. 20, no. 5, pp. 7–13, April 2011.

[7] A. Bharathidasan and V. Anand Sai Ponduru, "Sensor networks: An overview," Department of Computer Science, University of California, Davis, Tech. Rep., 2002.

[8] K. Sohraby, D. Minoli, and T. Znati, *Wireless sensor networks: Technology, protocols, and applications*. Hoboken, NJ: John Wiley & Sons, 2007.

[9] W. Zhao and X. Tang, "Scheduling data collection with dynamic traffic patterns in wireless sensor networks," in *Proceedings of INFOCOM 2011*. IEEE, April 2011, pp. 286–290.

[10] ——, "Scheduling sensor data collection with dynamic traffic patterns," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 4, pp. 789–802, April 2013.

[11] S. Gandham, Y. Zhang, and Q. Huang, "Distributed time-optimal scheduling for convergecast in wireless sensor networks," *Comput. Netw.*, vol. 52, no. 3, pp. 610–629, Feb. 2008.

[12] H. Choi, J. Wang, and E. A. Hughes, "Scheduling for information gathering on sensor network," *Wirel. Netw.*, vol. 15, no. 1, pp. 127–140, Jan. 2009.

[13] X. Dai, P. E. Omiyi, K. Bür, and Y. Yang, "Interference-aware convergecast scheduling in wireless sensor/actuator networks for active airflow control applications," *Wireless Communications and Mobile Computing*, vol. 14, no. 3, pp. 396–408, February 2014.

[14] H. Zhang, P. Soldati, and M. Johansson, "Optimal link scheduling and channel assignment for convergecast in linear WirelessHART networks," in *Proceedings of the 7th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT 2009)*. IEEE, June 2009, pp. 1–8.