Efficient Scheduling of Data-Harvesting Trees

Bastian Katz*, Steffen Mecke**, and Dorothea Wagner

Universität Karlsruhe (TH)
{katz,mecke,wagner}@ira.uka.de

Abstract. Many applications in sensor networks demand for energy and time optimal routing of data towards a sink. In this work we present mechanisms to set up energy and time efficient TDMA schedules for a given routing tree under very strict limitations: Nodes have only a constant size memory and must agree on a schedule using only a minimum of communication for set up: Each node is only allowed to send a single message to each of its neighbors.

We propose and analyze solutions in two different interference models. We show that, despite these tight restrictions, it is possible to compute energy optimal schedules which are almost time optimal and time optimal schedules which are almost energy optimal in the total interference model and we describe a 4-approximative algorithm in the k-local interference model.

We also show how to extend these mechanisms to settings with packet loss, while still guaranteeing bounds on energy consumption.

1 Introduction

Data-harvesting applications in sensor networks gather bulk data from the data field and collect them at a central repository or sink. Examples are data archiving or surveillance applications that periodically sample snapshots at high rates, storing or analyzing them centrally outside the network [1]. Another class are scenarios in which network nodes store measured data until from time to time a user requires access to the data stored in a large number of nodes [2]. In low-power networks, these applications demand highly optimized communication management to keep the network operable for as long as possible.

This paper considers sensor networks where sensor nodes are distributed over a geographic area and measure values in regular time intervals. At certain times, the stored data must be routed through the network and collected at a central location, the *sink*, usually along a routing tree rooted at the sink. Since the radio communication dominates the energy consumption, minimizing the cost of wireless communication is crucial to maximize the lifetime of a sensor network.

* Partially supported by the German Research Foundation (DFG) within the Research Training Group GRK 1194 "Self-organizing Sensor-Actuator Networks".

** Partially supported by the "BW-FIT" project "ZeuS" by the Landesstiftung Baden-Württemberg.

S. Fekete (Ed.): ALGOSENSORS 2008, LNCS 5389, pp. 43–56, 2008.

Power consumption in this scenario can be reduced in several ways. First by compressing the data, second by improving the routing tree to prevent that single nodes are overly burdened, and third by avoiding all unnecessary power consumption of the radio. Although these three issues are not fully independent, it makes sense to analyze them separately. Data reduction is an application-specific problem that is largely orthogonal to the other two problems. The construction of routing trees is subject to many other practical restrictions such as link quality in real-life networks. We will thus focus on the problem of avoiding all unnecessary power consumption by the radio for a fixed routing tree provided by some arbitrary protocol.

If communication patterns are known in advance, schedule-based ("TDMA") protocols outperform contention-based ("CSMA") protocols, because they do not waste energy due to idle listening and collisions. Their drawback however is an increased protocol overhead for schedule set-up. Therefore, we develop and analyze efficient, schedule-based protocols that require only a minimum of set-up communication.

The problem addressed in this paper can be summarized as follows: Given a routing tree in which all nodes store data that is to be collected at a sink, we allow each node to pass only one packet to each of its neighbors in that tree. Is it possible to agree on an energy-optimal TDMA schedule, i. e., a schedule that allows to transport all stored packets to the sink and in which all nodes know exactly when to send and when to listen? Is it possible to agree on a schedule of minimum length?

We will contribute to this problem by providing lower bounds and algorithms for two reasonable interference models. In the total interference model, we assume that at any time, only one single transmission is allowed throughout the whole network. We will provide two transmission schedules that comply with the above restrictions, the first being time-optimal at the price of adding a small protocol overhead to routed packets, the second being energy-optimal, but not time-optimal. We also provide a third transmission schedule for routing trees that is both, energy- and time-optimal, at increased set-up costs. Furthermore, we prove that in this interference model, an energy- and time-optimal transmission schedule cannot route single data packets with minimum delay. We conjecture that it is impossible to set up an energy- and time-optimal schedule under our restrictions at all.

In the k-local interference model we assume that transmissions do only interfere with transmissions within some constant neighborhood. We will show how to set up a transmission schedule that is energy-optimal and constant-factor time-approximative for the given tree.

This paper is organized as follows: In the remainder of this section, we will discuss related work and give a formal problem definition. In Section 2, we cover the problem of finding schedules for a routing tree with total interference. In Section 3, we do the same for local interference models. Section 4 discusses a method to handle the problem of unreliable links. We conclude in Section 5.

1.1 Related Work and Overview

There is a plethora of algorithms for finding topologies (or routing information) for the data gathering problem in sensor networks. There are several goals for optimization, for example throughput, latency, reliability, security and energy consumption, the last one being the most important in sensor networks. Almost all of these different approaches, however, construct one or sometimes several routing *trees*.

There has been previous work on minimzing the *time* for data gathering. In [3] a problem similar to ours is studied, a 4-approximation algorithm is given and NP-hardness of the problem shown. A problem with variable release times is studied in [4]. Unlike this previous work, however, we focus on distributed algorithms and take set-up cost into account. Also, we do not concentrate so much on time optimality but on *energy* optimality.

There are two main kinds of medium access methods: contention based ("CSMA") protocols and scheduled ("TDMA") protocols (see [5] for a partial overview of MAC protocols for sensor networks). There are also a few hybrid forms. The strength of contention based protocols include simplicity, low protocol overhead and flexibility, but they suffer from energy waste caused by collisions, overhearing and idle listening. TDMA-based protocols do not have any of these drawbacks (at least in theory) but they require more communication to establish the scheme, time synchronization is usually more of an issue and they are less flexible in case of topology changes.

One of the very few contention-based MAC protocols that take advantage of the tree topologies present in data-archiving systems is [6]. The wakeup times of nodes are staggered on paths towards the sink, which reduces latency. Measures are employed to reduce interference among packets travelling along the same paths. Additionally, special "More-to-Send" packets are used to further synchronize wakeup times and thus increase throughput. However, this protocol is designed for very low data rates. It is not energy-optimal and there are no special mechanisms to reduce congestion and ensure fairness.

Flexible Power Scheduling (FPS) is described in [7]. In FPS parents are responsible for assigning time slots to their children. FPS reduces contention but does not guarantee collision-free communication. Therefore, an underlying MAC layer is still required. Fairness among children in different branches is not ensured.

MPS (Multi-Flow Power Scheduling) and HPS (Hybrid Power Scheduling) are enhancements on FPS introduced in [8]. MPS is closely related to our k-layer interference protocol in Section 3, but performance is only evaluated experimentally and there is no theoretical analysis of the protocols. Also, the interference model is not described explicitely.

The authors of [2] address the problem of congestion, fairness and robustness during the transport of high volumes of sampled data. They use the total interference model (cf. Section 2). Their approach is based on a slot distribution scheme. Nodes that have no more packets to send can pass their slots back to their parent. Every second slot that a node receives from its children is passed on to its parent. This aims at distributing slots more fairly: Nodes with high

loads get more slots. In [9] further refinement of slot distribution strategies are developed. But even with these refinements, the channel usage is still fairly low and decreases with growing network size. In the same paper, there is another scheme (similar to ours in Section 2) which, however, leads to unlimited buffer size and the control message overhead is not analyzed.

DOZER ([10]) is an approach that tries to solve the problems of medium access, tree construction and scheduling together. The authors employ a local TDMA scheme which reduces requirements on clock synchronization but does not ensure fairness. Collisions are reduced by letting schedules of interfering node pairs "drift apart" through randomization. This approach is designed for scenarios with very low data rates but causes problems when there are higher volumes of data to deliver.

In contrast to these approaches, we focus on detailed theoretical analysis of throughput and protocol overhead. Our approaches are also designed for arbitrarily large networks and high data load.

1.2 Problem Definition and Network Model

Throughout this work, we assume that we are running a sensor network with one node serving as a sink that is connected to some infrastructure or monitor. The task now is to set-up a transmission schedule that collects sensor data from all nodes at the sink without aggregation. This task naturally divides into the following subtasks or *stages*:

Topology stage: decide on a topology to collect data. We assume that the resulting topology is a tree routed at the sink.
Set-up stage: perform the communication necessary to agree on a schedule that guarantees delivery of all sensor data to the sink and complies with an interference model.
Collection stage: run the schedule as long as data is to be collected.

Energy consumption is our primary concern. We want to minimize it during both the set-up stage and the collection stage. Energy use during the collection stage can be minimized if each node knows exactly in which slots to listen and when to send and if there is no idle listening or failed transmission. Nodes that are neither sending nor trying to receive can go into sleep mode or at least turn off their radio, dramatically reducing the energy consumption. We concentrate on solutions which achieve this with a minimalistic kind of set-up communication: Just one convergecast and one broadcast. It is impossible to let all nodes know about the schedule's length, let alone a first point in time to transmit or receive with less communication.

More formally, we restrict solutions to the following model:

1. Each node v in the network must transmit a (possibly individual) number of own data packets, $\sigma(v)$ to the sink.
2. Within the network, a spanning tree T, rooted at the sink r is provided by some standard protocol, i. e., every node knows its parent and a list of its children.

3. During the set-up stage, every node can send at most one packet of size $O(\log N)$ bits to each of its neighbors in T, N being the number of packets in the network. We assume that during this stage a different medium access scheme is used to establish parameters for the TDMA-based scheme of the collection stage.
4. Each node (except for the sink) has only a limited, constant amount of memory for buffering packets and storing information about the slots to be active.

As a secondary criterion, we want to minimize the length of the schedule, i. e., the time until all packets have been delivered.

1.3 Definitions and Notation

We will denote a node's distance from the sink in T by $h(\cdot)$ and the height of the tree by h. We will talk of *children*, *descendants*, *parents* and *ancestors* in the usual sense. We will refer to the set of a node v's descendants including v as $D(v) := \{w \mid w \text{ is descendant of } v\} \cup \{v\}$. The set of children of v is denoted by $C(v)$. We will assume that there is a fixed ordering among the children of every node. A child v is said to be *left* of w if it preceeds w in this order. We will also refer to the pre- and postorder number of nodes as $\mathrm{pre}(v)$ and $\mathrm{post}(v)$ (with respect to these orderings) in the usual sense. As defined above, let $\sigma(v)$ denote the number of own packets a node has to deliver. We will assume that every node (except the sink, for which we assume $\sigma(r) = 0$) has at least one data packet. We will shortcut $\sum_{v \in V'} \sigma(v)$ by $\sigma(V')$ and $\sigma(V)$ with N. We will also write $\pi(V') := \sum_{v \in V'} h(v)\sigma(v)$ for the number of transmissions a set of nodes causes and $\bar{\pi}(v) := \pi(D(v))$.

2 Scheduling a Tree with Total Interference

This section covers the case of *total interference*. That is, no two nodes are ever allowed to transmit in the same time slot.

2.1 Infeasibility

We conjecture that it is not possible to find a scheme that achieves optimal time *and* energy using only one concast and convergecast in the total interference model under the assumptions given in section 1.2. We have no proof for this claim, but if we restrict ourselves to certain kinds of schedules, it follows quite easily:

Proposition 1. *Let $\sigma(v) \equiv 1$, every node can store at most one packet (at the beginning every node's memory is filled with its own packet), during the set-up stage only one convergecast and one broadcast of ($\log N$)-sized messages is allowed and no additional information can be attached to the packets during the collection stage.*

Under these restriction it is not possible to compute and run a time-optimal

schedule in which each packet is immediately passed to the sink once it has started from its originating node.

Proof. Let r be the sink node. During the set-up stage, r has only received $|C(v)|c \log N$ bits for some constant c. For every packet arriving at r during the collection stage, the height of the originating node can be determined by the time since the arrival of the previous packet. As r is only allowed to be awake when packets arrive, it must know the times of arrival in advance. This is equivalent to knowing the heights $\{h_1, h_2, \ldots, h_n\}$ of the packets in advance. There are, however, an exponential[1] number of such sequences, each one requiring a different schedule at r. But these cannot be identifed by the $C(v)c \log N$ bits.

Most of the assumptions can be relaxed without invalidating Proposition 1, but it is not apparent if different kinds of schedules (which, for example, buffer incoming packets) could achieve more. The rest of the paper will show what is possible if some of the assumptions are relaxed.

2.2 An Optimal Scheme with Increased Packet Size

In our first approach we will achieve a time and energy optimal scheme by allowing slightly increased packet size during the collection stage.

We will proceed as follows: The packets are transported to the sink in preorder. Each packet is immediately passed down to the sink once it has started. A node v never has to wake up before all nodes which are further left than v but not descendants of v have transmitted their packets to the sink. Let's call the first time slot in which v has to transmit a packet $T(v)$. We have

$$T(v) = T(\text{parent}(v)) + \pi\,(\text{parent}(v)) \qquad (1)$$

if v is the leftmost child of its parent. Else, we have

$$T(v) = T(s_l(v)) + \bar{\pi}\,(s_l(v)) \quad . \qquad (2)$$

The $\sigma\,(v)$ packets of v are then transmitted in time slots $T(v), T(v) + h(v), \ldots$ until $T(v) + (\sigma\,(v) - 1)h(v)$. After transmitting its own packets, the next time to wake up for v is when its leftmost child w transmits its packet.

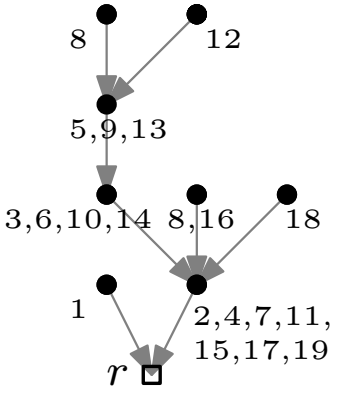


Fig. 1. Example schedule, numbers denote slots for sending

The ancestors of v also have to have information about when to receive these packets on their way to the sink. A node w on the path from v to the sink r has to receive in slot $T(v) + h(v) - h(w) - 1$ and has to send in slot $T(v) + h(v) - h(w)$ (and so on...). Therefore, v attaches

[1] Exponential in n, even disregarding isomorphisms and order.

to the payload P of the message an information t about the next time slot it is going to send in.

$$t = \begin{cases} T(v) + h(v)k & \text{if } P \text{ is the } k\text{th packet of } v \text{ and it has packets left} \\ T(c) + 1 & \text{if the next packet belongs to a child } c \text{ of } v \\ t' + 1 & \text{if } P \text{ is from a descendant of } v \text{ and } t' \neq 0 \\ 0 & \text{if } t' = 0 \text{ and no children of } v \text{ are left} \end{cases} \tag{3}$$

After all descendants of v have transmitted their packets, v can sleep for the rest of the protocol.

In summary, each node v has to compute

1. the number of packets $\sigma(D(c))$ in the subtree of each of its children c_i (counted during the convergecast),
2. its own height $h(v)$ (determined during the broadcast),
3. the weighted sizes of the subtrees $\bar{\pi}(c_i)$ (computed from $h(v)$ and information collected during the convergecast),
4. its own starting time slot $T(v)$ and the starting slots of all of its children (computed via Equations (1) and (2)).

Theorem 1. *There is a scheme that produces an optimum length schedule of length $\bar{\pi}(r)$. In this scheme, every node receives $1 + C(v)$ packets and transmits $1 + C(v)$ packets of size $\log(N)$ in the set-up stage.*

Every node receives and sends $\Theta(\sigma(D(v)) \log(h))$ additional bits during the collection stage. In total, an additional $\Theta(n\Delta \cdot \log(N) + \bar{\pi}(r) \log h)$ bits are transmitted and received (with $\Delta := \max_v C(v)$). A node needs additional memory of $O(\Delta \log(nh \max_{w \in V} \sigma(w)))$.

Remark 1. The amount of additional data can be reduced to $\Theta(D(v) \log h)$ (or $O(n \log h)$) bits per node. The total amount of additional data can be reduced to $\sum_v h(v) \log h \leq nh \log h$ bits.

Summarizing, we have described a time- and energy-optimal scheme at the price of increasing each packet by $\log(h)$ bits.

2.3 Variants

In the previous section we saw how an optimal (shortest) schedule can be constructed with relatively little (but unbounded) message overhead. With slight modifications (which include adding a memory buffer for one packet at every node), a similar scheme as in the previous section can be constructed for postorder. In this section we will propose two different ways of cheating to get a time optimal schedule.

Our first proposal needs more energy in the set-up stage and more than constant memory:

Proposition 2. *It is sufficient to send (and store) $h \log N$ bits per edge once: the number of nodes in every layer below that edge. If we use level order, we can compute t from this information.*

Finding a scheme for level order seems a little bit more involved. If, however, every node v knows the number of nodes in lower levels and for each level $\ell \geq h(v)$ the number of nodes in level ℓ to the left of v, in v's subtree and to the right of v, then it can easily compute the slots in which it has to send or receive. However, this requires additional memory per node in the order of $\Theta(h \log N)$ and the same amount additional communication *per edge*.

The second approach is a scheme which fulfills all the requirements on communication and memory but at the cost of time optimality:

Proposition 3. *If all nodes have the same number σ of packets, we can arrange an energy optimal schedule with length Nh (h is the tree height) and time approximation factor $\sqrt{n}$.*

Proof. This schedule works as follows: As before we process packets in preorder and immediately hand them down to the sink. The only difference being that $T(v) = \sigma h \cdot (\mathrm{pre}(v) - 1) + h - h(v) + 1$. The packets of v are then transmitted in slots $T(v), T(v) + h, \ldots, T(v) + (\sigma - 1)h$. If v actually has height $h(v) < h$, there are $h - h(v) - 1$ unused slots.

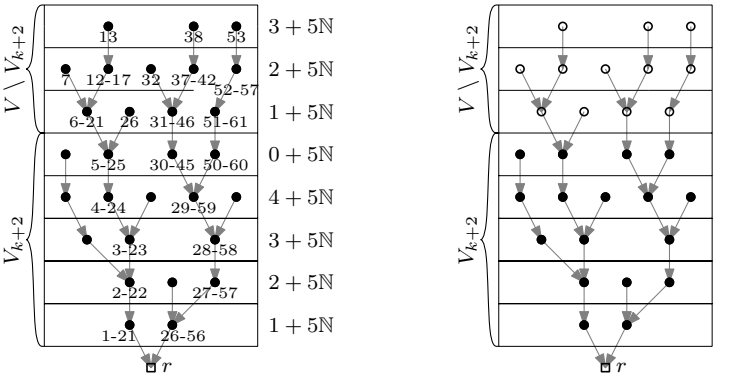
The approximation ratio of the length of this scheme is $\Theta(\sqrt{n})$: For a fixed height h the worst case is a "flower" with one path of length h and all other nodes at height 1. The approximation ratio is $\sigma nh/|S_{\mathrm{OPT}}| = nh/(h \cdot (h-1)/2 + (n-h) \cdot 1) \leq c\sqrt{n}$.

Remark 2. For "well-behaved", geometric graphs in the plane with height $\Theta(\sqrt{n})$ and $O(\ell)$ nodes in every level ℓ (like a regular, geometric grid), the schedule has length $\Theta(\sigma n\sqrt{n})$ which is within a constant factor of the optimum.

3 Scheduling with Local Interference

In this section, we assume that a transmission (u, v) is successful if and only if u is the only active sender in v's k-hop neighborhood for some constant k. We call this interference model *k-local interference*. This model is a reasonable, yet a cautious approximation of interference in dense networks, in which euclidian distance and hop-distance closely correlate.

Quite typically, routing trees for data aggregation in sensor networks are set up using some kind of request flooding, i. e., the routes between a node and the sink are shortest paths (in hops). This is not only a very lightweight, robust protocol, but also guarantees that data travels on short routes, which reduces the risk of packet loss. In such trees, a transmission (u, v) is always successful if u is the only active sender among all nodes with $|h(v) - h(u)| \leq k$. This follows from the triangle inequality. We call this property of a rooted tree *k-layer bounded interference*. We will propose a protocol, coined k-LS, to set up a schedule that is energy optimal and time approximative within a constant factor.



(a) slot assignment (b) resulting situation

Fig. 2. Slot assignment (a) and result (b) of the first phase of k-LS for $k = 3$ and $\sigma \equiv 1$. Edges are active at time t if t is in the given range and matches the modulo for the sender's level of the tree.

Theorem 2. *In the k-local interference model, there is a scheme for trees with k-layer bounded interference yielding an energy optimal schedule that is time approximative within a factor of* $\frac{k+2}{\lfloor (k+1)/2 \rfloor} \leq 4$.

Proof. The key idea of k-LS is to set up two schedule phases that are performed successively. In the first phase data is "pipelined" towards the sink ending in a state where no nodes having a height of more than $k+2$ have any packets left. In more detail, the sink's neighbors pass packets to the sink every $k+2$ slots starting with the first slot and sending one after another. Whenever a node transmits a packet, it receives a packet in the next slot from one of its children as long as their children have more packets. Every node v passes $\sigma\left(D(v) \setminus V_{k+2}\right)$ packets, i. e., as many packets as there are in its subtree in heights more than $k+2$. In the second phase data is collected from the remaining nodes using essentially the same technique as in Proposition 2. For the following formal description, we will shortcut the set of nodes with height less than or equal to some h with $V_h := \{u \in V \mid h(u) \leq h\}$.

During the convergecast phase each node v learns how many packets to relay from nodes with heights $h(v)+1, \ldots, h(v)+k+2$ and from more distant levels. Obvioulsy, $(k+3)\mathrm{ld}N$ bits, i. e. a message size of $O(\log N)$ during the concast phase is sufficient to achieve this. In the broadcast phase, each node v learns its height $h(v)$ and some starting slots to be described later. Given its height, it knows how many packets to relay that do not originate at nodes $u \in V_{k+2}$, i. e. $\sigma\left(D(v) \setminus V_{k+2}\right)$ and, if $h(v) \leq k+2$, how many nodes to relay from each of the levels $h(v)+1$ to $k+2$. For the first, the pipelining phase, every node is additionally assigned a starting slot $T(v)$ as follows: The sink assigns itself the (imaginary) starting slot $T(r) = 0$, and using messages of at most $2\mathrm{ld}N$ bits, each node v assigns start slots $T(v) + 1 + \sum_{0<j<i} \sigma\left(D(c_j) \setminus V_{k+2}\right)$ to each of its children c_i. For the second phase, nodes in V_{k+2} additionally receive the starting

slots according to the scheme proposed in Proposition 2 restricted to nodes in V_{k+2} plus an offset of $(k+2) \cdot \sigma(V \setminus V_{k+2})$, which is known to the sink after the convergecast. In the pipelining phase, each node v but the sink now transmits in slots $T(v) + i(k+2)$ for $i = 0, \ldots, \sigma(D(v) \setminus V_{k+2})$. It receives a packet in the following slot the first $\sum_{c \in C(v)} \sigma(D(c) \setminus V_{k+2})$ times. This process is depicted in Fig. 2 for $k = 3$ and $\sigma \equiv 1$. Quite obviously, this part of the schedule has length $(k+2) \cdot \sigma(V \setminus V_{k+2}) + 1$. Nevertheless, for the analysis, we can assume a length of only $(k+2) \cdot \sigma(V \setminus V_{k+2})$ since the last transmission of this part can safely overlap with the first transmission of the second part. After completion, no node with height more than $k+2$ neighborhood has any packets left and each node v in V_{k+2} has exacly $\sigma(v)$ packets stored. Now, the rest of the packets is collected optimally as described in Proposition 2, i. e. in $\pi(V_{k+2})$ slots.

It is easy to see that the resulting schedule is energy optimal for the given tree. To show the approximation factor of $(k+2)/\lfloor (k+1)/2 \rfloor$, we observe that for $l := \lfloor (k+1)/2 \rfloor$, no two nodes in V_l can transmit at the same time. Thus, each packet originating at a node with height of more than l accounts for at least those l slots where it is the only one transmitted by a node with height of l or less in an optimal schedule. Similarly, every packet originating at a node $v \in V_l$ accounts for at least $h(v)$ slots. Hence, an optimal schedule has at least length

$$\begin{aligned} |S_{\mathrm{OPT}}| &\geq l\sigma(V \setminus V_l) + \pi(V_l) \\ &= l\sigma(V \setminus V_{k+2}) + l\sigma(V_{k+2} \setminus V_l) + \pi(V_l) \end{aligned} \tag{4}$$

The schedule produced by k-LS in turn uses

$$\begin{aligned} |S_{k\text{-LS}}| &= (k+2)\sigma(V \setminus V_{k+2}) + \pi(V_{k+2}) \\ &= (k+2)\sigma(V \setminus V_{k+2}) + \pi(V_{k+2} \setminus V_l) + \pi(V_l) \end{aligned} \tag{5}$$

The claim now follows from the fact that $\pi(V_{k+2} \setminus V_l) \leq (k+2)\sigma(V_{k+2} \setminus V_l)$.

Applying very similar arguments as above, we can observe that first, k-LS produces a time optimal schedule if the first $(k+2)$ levels of T form a single path, and second, that if T is a hop-shortest path tree in the sink's l-hop neigborhood, the produced schedule is time approximative not only for an optimal schedule of T, but for an optimal schedule of any spanning tree. For the sake of brevity, we omit the proof.

Corollary 1. *The schedule produced by k-LS is time optimal for the given tree if V_{k+2} is a path and time approximative within a factor of $\frac{k+2}{\lfloor (k+1)/2 \rfloor}$ among all schedules of all spanning trees of G if $h(v) = d_G(v, r)$ for all v with $d_G(v, r) \leq l$, i. e. if T is a hop-shortest path tree in the l-hop neighborhood of r.*

All these results can very naturally be extended to a relaxed local interference model, the k_i/k_o-local interference model, in which a transmission (u, v) is successful if u is the only active sender in v's k_o-neighborhood, but might be successful if u is the only active sender in the k_i-neighborhood of v. Then, the approximation ratio from Theorem 2 becomes $\frac{k_o+2}{\lfloor (k_i+1)/2 \rfloor}$.

4 Making Schedules Robust

In this section, we will analyze an approach to make the above schemes robust to link failures using logarithmic sized additional memory per node. To allow for more sophisticated solutions than blindly repeating transmissions, which would not lead to a robust scheme anyway, we will assume some sort of ACK to acknowledge successful transmissions and a known lower bound on reception probability $\alpha(u)$ (including the feedback) for every link $(u, v) \in T$. We further assume reception probabilities to be independent for every transmission. We will first analyze a generic approach and then discuss individual issues of the proposed schemes.

For the generic approach, we will assume a non-robust schedule in which no node receives twice in a row without sending in between like all of the above. We further assume that we we have a mechanism to allocate $t(u) \geq \lceil 2/\alpha(u) \rceil$ successive slots for every scheduled transmission (u, v) in the non-robust schedule. We will discuss later how to do this for the individual protocols. The basic idea now is to run the non-robust schedule, with $t(u)$ slots reserved for every transmission (u, v). We will show that as long as every node only uses the $t(u)$ slots allocated for a single transmission in the non-robust schedule for at most one *first* transmission attempt (and arbitrarily many retransmissions), with high probability all packets are delivered if nodes are able to buffer a logarithmic number of packets and if the schedule budgets for a slightly more than $\sigma(u)$ packets for every node, namely $\sigma'(v) := \sigma(v) + \hat{\sigma}(v)$. More specifically, let every node v have a buffer of size of $(|C(v)| + 1)b$ for $b := 3 \log_4 N$ packets, and let $\hat{\sigma}(v) := (\frac{25}{32} \ln 4 + \frac{5}{2}(|C(v)| + 1))b$. We propose a robust transmission protocol (RTP) where a node u uses the $t(u)$ slots reserved for a transmission (u, v) in the non-robust schedule as follows: First, if all packets in the sender's buffer are marked as "pending", it pushes a "fresh" own packet to the buffer if there are any. Second, if there are unmarked packets in the buffer, it then marks one of them as "pending". Third, it tries to transmit as many pending packets as possible, removing successfully transmitted packets from the buffer. The receiver v simply adds new packets to its buffer if they have not been received before[2].

Proposition 4. *With high probability, RTP delivers all packets.*

Before we prove this claim, we prove the following lemma:

Lemma 1. *With high probability, no buffer of any node v ever contains more than b packets marked as pending or more than $|C(v)|b$ unmarked packets.*

Proof (Lemma 1). First, we observe that in a transmission phase, the number of nodes marked as pending at the sender can increase by at most 1, if all $t(u)$ transmission attempts fail and decreases by at least one if two or more transmission attempts are successful. Since the number of transmission attempts is

[2] Which can happen due to lost ACKs.

higher than $2/\alpha(u)$, the probability that the number of pending packets descreases, p^-, is more than four times the probability that this number increases, p^+, unless there were no pending packets at the begin of the transmission phase. Modeling the buffer utilization by marked packets as a finite markov chain, we get a probability of less than $(p^+/p^-)^b$ to be in a state where the buffer contains more than b marked packets by steady state analysis. The probability to reach such a state in any node in any transmission phase is thus less than

$$1 - \left(1 - (1/4)^b\right)^{N^2} = 1 - \left(1 - 1/N^3\right)^{N^2} < 1/N$$

On the other hand it is easy to see that also the number of unmarked packets in a node v's buffer can never exceed $|C(v)|b$: Looking at a node v and its children $C(v)$, we observe that the sum of packets marked as pending in the buffers of the children and the unmarked packets in the buffer of v can only increase during a transmission phase of a child and the next transmission phase of v (no node receives twice in a row) if v had no unmarked packet in its buffer prior to the transmission phase and the transmissions all failed. But since with high probability, the first number stays below b for all children, i. e. the sum of marked packets in the children's buffers does whp. not exceed $|C(v)|b$, the sum of marked packets at the children plus the number of unmarked packets at v cannot exceed $|C(v)|b$, which proves the claim.

Proof (Proposition 4). Before the additional $\hat{\sigma}(v)$ transmission phases start, every node had enough transmission phases to shift all its own packets to the buffer. From Lemma 1, we know that whp., the buffer sizes are sufficient. It remains to show that the additional $\hat{\sigma}(v)$ transmission phases are sufficient for every node to get rid of the packets left in the buffer, i. e. at most $(C(v)+1)b$ packets. As argued above, in every transmission phase, with probability $p > 4/5$, at least one packet is transmitted. Thus, Hoeffding's inequality gives us the following upper bound on the probability that *for a single node*, less than $(|C(v)|+1)b$ of the $\hat{\sigma}(v)$ calls are successful:

$$P_{\text{snf}} \leq \exp\left(-\frac{2(\hat{\sigma}(v)\,p - (|C(v)|+1)b)^2}{\hat{\sigma}(v)}\right) < \exp\left(\frac{16(|C(v)|+1)}{5}b - \frac{32}{25}\hat{\sigma}(v)\right)$$

With b and $\hat{\sigma}(v)$ as above, we get $P_{\text{snf}} < 1/N^3$, and a probability of less than $P_{\text{fail}} < 1 - (1 - 1/N^3)^{|V|} < 1/N$ that any node has any packets left when the schedule ends.

Note that while retransmissions of failed transmissions are inevitable, the proposed scheme does waste some energy for the following reasons: First, every node accounts for additional $O(\log N)$ packets, and second, nodes do have to transmit at least once during a transmission phase even if they do not have any packet marked as pending. Both effects increase the energy consumption only by small constant factors if every node has a payload of at least $\sigma(v) > \hat{\sigma}(v)$, i. e. $\sigma(v) \in \Omega(\log N)$. If, in this case, the number of time slots $t(v)$ can be set

to exactly $\lceil 2/\alpha(v) \rceil$ for every v, the total number of time slots $2t(v)\sigma(D(v))$ reserved for v's transmission attempts is at most six times the expected number of necessary transmission attempts $\sigma(D(v))/\alpha(v)$.

Adapting this approach to the proposed schedulings, however, can incur additional costs. This adaption is easy if there is some reasonable lower bound on reception probability, i. e., if there is some small constant $c > 1$ for which $\max_{v \in V} \alpha(v) < c \min_{v \in V} \alpha(v)$. Then, every scheme can be made robust by choosing $t \equiv \lceil 2/\min_{v \in V} \alpha(v) \rceil$. Schemes in the total interference model can also be made robust by treating a link $(v, \text{parent}(v))$ with reception probability $\alpha(v)$ as if it was a path of length $t(v) = \lceil 2/\alpha(v) \rceil$. In this case, the height of the tree changes accordingly.

For the time-optimal scheduling scheme in the total interference model, robustness can cause higher costs: Since packets contain additional routing data, missing packets can sometimes only be compensated for by idle listening when waiting for the next packet. On the other hand, buffer sizes and packet count increase can be lowered for some of the proposed solutions: If a protocol guarantees that a node does not receive packets from children alternatingly, which does hold for all protocols but the level-order schemes, then the $|C(v)|$ can safely be replaced by 1 in the definition of b and $\hat{\sigma}(v)$.

5 Conclusion and Open Problems

We have analyzed the performance of different TDMA schemes for two interference models. Under the total interference model we have analyzed a scheme that optimizes the number of transmissions and the time to complete at the cost of increasing the packet size. We described another scheme which does not need to change the packet size but which is not time optimal. We have conjectured that there is no scheme which is time and energy optimal at the same time. The proof of this conjecture remains an open problem. It is also an open question if there are energy optimal schedules with better approximation guarantees (concerning time to complete) than $\Theta(\sqrt{n})$.

For the k-layer interference model we have proposed a scheme which is energy optimal and is a good approximation with respect to time till completion. Finally, we have shown how our schemes can be improved in order to integrate robustness mechanisms.

Our analysis has shown some lower bounds on the performance of TDMA schemes for data harvesting. It remains to evaluate the practical performance of our algorithm compared to other approaches under more realistic conditions.

Acknowledgements

The authors would like to thank Robert Görke and Reinhard Bauer for proofreading and fruitful discussions.

References

1. Xu, N., Rangwala, S., Chintalapudi, K.K., Ganesan, D., Broad, A., Govindan, R., Estrin, D.: A Wireless Sensor Network for Structural Monitoring. In: 2nd Int. Conf. on Embedded networked sensor systems (SenSys 2004), pp. 13–24. ACM Press, New York (2004)
2. Turau, V., Weyer, C.: Scheduling Transmission of Bulk Data in Sensor Networks Using a Dynamic TDMA Protocol. In: 8th Int. Conf. on Mobile Data Management (MDM 2007), pp. 321–325. IEEE Computer Society Press, Los Alamitos (2007)
3. Bermond, J.C., Galtier, J., Klasing, R., Morales, N., Perennes, S.: Hardness and Approximation of Gathering in Static Radio Networks. Parallel Processing Letters 16(2), 165–183 (2006)
4. Bonifaci, V., Korteweg, P., Marchetti-Spaccamela, A., Stougie, L.: An Approximation Algorithm for the Wireless Gathering Problem. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 328–338. Springer, Heidelberg (2006)
5. Langendoen, K., Halkes, G.: Energy-efficient medium access control. In: Zurawski, R. (ed.) Embedded Systems Handbook. CRC Press, Boca Raton (2005)
6. Lu, G., Krishnamachari, B., Raghavendra, C.S.: An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks. In: 18th Int. Parallel and Distributed Processing Symp (IPDPS 2004), p. 224a. IEEE Computer Society, Los Alamitos (2004)
7. Hohlt, B., Doherty, L., Brewer, E.A.: Flexible power scheduling for sensor networks. In: 3rd Int. Symp. on Information Processing in Sensor Networks (IPSN 2004), pp. 205–214. IEEE Computer Society, Los Alamitos (2004)
8. Yao, Y., Alam, S.M.N., Gehrke, J., Servetto, S.D.: Network Scheduling for Data Archiving Applications in Sensor Networks. In: 3rd Worksh. on Data Management for Sensor Networks (DMSN 2006), pp. 19–25. ACM Press, New York (2006)
9. Turau, V., Weyer, C.: TDMA-Schemes for Tree-Routing in Data Intensive Wireless Sensor Networks. In: 1st Int. Work. on Protocols and Algorithms for Reliable and Data Intensive Sensor Networks (PARIS), pp. 1–6. IEEE Computer Society Press, Los Alamitos (2007)
10. Burri, N., von Rickenbach, P., Wattenhofer, M.: Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In: 6th Int. Symp. on Information Processing in Sensor Networks (IPSN 2007), pp. 450–459. ACM Press, New York (2007)