This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Delay Performance of Scheduling with Data Aggregation in Wireless Sensor Networks

Changhee Joo, Jin-Ghoo Choi, and Ness B. Shroff

***Abstract*—In-network aggregation has become a promising technique for improving the energy efficiency of wireless sensor networks. Aggregating data at various nodes in the network results in a reduction in the amount of bits transmitted over the network, and hence, saves energy. In this paper, we focus on another important aspect of aggregation, i.e., delay performance. In conjunction with link scheduling, in-network aggregation can reduce the delay by lessening the demands for wireless resources and thus expediting data transmissions. We formulate the problem that minimizes the sum delay of sensed data, and analyze the performance of optimal scheduling with in-network aggregation in tree networks under the node-exclusive interference model. We provide a system wide lower bound on the delay and use it as a benchmark for evaluating different scheduling policies. We numerically evaluate the performance of myopic and non-myopic scheduling policies, where myopic one considers only the current system state for a scheduling decision while non-myopic one simulates future system states. We show that the one-step non-myopic policies can substantially improve the delay performance. In particular, the proposed non-myopic greedy scheduling achieves a good tradeoff between performance and implementability.**

I. Introduction

Wireless sensor networks can be used to monitor physical or environmental conditions, and collect and transmit sensed data. These networks can serve as an infrastructure for a number of applications including surveillance, medical monitoring, agricultural cultivation, facility monitoring, and entertainment. For a number of applications, sensor nodes could be deployed in a vast area or in harsh environments. Hence, they need to operate with a limited power source. To efficiently manage these limited resources, efficient resource allocation for sensing and communications is of great importance.

In sensor applications, aggregation techniques have been investigated to facilitate robust operations, extend coverage, and improve accuracy and system reliability [1]. In wireless sensor networks, the techniques have been further exploited for energy-efficient communications [2], [3]. Since many applications of wireless sensor networks are intended to compute a specific function from the sensed data, there is often substantial redundancy, and it is possible to aggregate the data at intermediate nodes in the network without affecting the final function value. Thus, efficiency is improved by compressing the information inside the network. This also improves delay performance by reducing competition for scheduling resources, which expedites end-to-end data transmission. On the other hand, additional delays can also result due to aggregation. In general, the longer an intermediate node holds data before forwarding, the more likely it is to improve the throughput efficiency via greater aggregation, however, this could result in longer delays. Therefore, in-network aggregation is strongly coupled with scheduling, and should be carefully designed so that end-to-end delay is actually improved [4].

Most previous works on data aggregation have mainly focused on energy saving issue. Building an energy-efficient topology (usually a tree construction) has been investigated in [5]–[8]. Tree topologies are often employed for wireless sensor networks not only because their structure is suitable for a network with one or a few sink nodes, but also because their simplicity is very attractive when network resources are limited. It has been shown that finding the optimal aggregation tree that minimizes the number of transmissions or maximizes the network lifetime is an NP-Hard problem [5], [8], and efficient approximation algorithms have been provided for constructing such trees. This is not the focus of our paper. Instead, we assume that an aggregation tree has been provided. Given a tree topology, we study the impact of in-network aggregation on the delay performance of wireless scheduling.

Another group of works have investigated the energy-latency tradeoff [4], [9], [10]. They have dealt with energy efficiency of wireless sensor networks constrained on the maximum delay of sensed data. Given a deadline, they minimize overall energy dissipation of sensor nodes [9], minimize the maximum energy consumption [10], or minimize the amount of missed data [4].

In this paper, we consider applications for which the *sum delay* of the sensed data is important instead of the maximum delay. For example, suppose that sensor nodes measure their environments (e.g., location of a target, denoted by X) with time-varying errors (or measurement-dependent errors). Each node generates measurement X_i and the error statistics E_i, e.g., the variance. The errors are independent and have a zero mean, and the sink can improve the measurement accuracy by collecting data from multiple sensor nodes. Assume that there are n measurements over the network, and the sink collects k data up to time t. Then, the sink can estimate the measurement $\bar{X}(t) = \frac{1}{k}\sum_{i=1}^{k} X_i$ with improved accuracy $\bar{E}(t) = \frac{1}{k^2}\sum_{i=1}^{k} E_i$. Let t_k denote the time when the sink collects k data. The task of the network is to obtain an accurate measurement $\bar{X}(t_k)$ while minimizing the delay t_k subject to $\bar{E}(t_k) \leq \epsilon$ for some given ϵ. Under this scenario, the deadline based constraints are not useful since it is unclear a priori how much data will have to be collected (i.e., what the value of k should be) and how long it will take to collect it. Rather, we formulate the problem by minimizing the total delay of sensed

This work was supported in part by NSF Award CNS-0721434 and ARO MURI Award W911NF-07-10376 (SA08-03).

C. Joo, J. Choi, and N. B. Shroff are with the Ohio State University, e-mail: {cjoo, choiji, shroff}@ece.osu.edu.

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

data, and thus reduce the average delay. Associated with in-network aggregation, scheduling also needs to be considered since one has to determine which node forwards data, and which node holds data for aggregation, which might improve the performance in the future by reducing the amount of bits in-flight. The problem is further exacerbated by interference in the wireless environment. Our main contributions are as follows:

- We formulate the scheduling problem that minimizes the delay sum of data in wireless sensor networks, and qualify the gain of in-network aggregation techniques for tree networks.
- We study the characteristics of optimal scheduling with aggregation, and analyze the optimal performance by providing a lower bound.
- We evaluate the performance of scheduling policies against the delay bound, and show that the performance can substantially improve by considering the network state one-step ahead, i.e., by estimating the network state of the next time slot.

The paper is organized as follows. We describe our network model and formulate the problem in Section II. We clarify the benefit of in-network aggregation in Section III by comparing the performance bounds with and without the aggregation techniques. In Section IV, we illustrate the difficulties with scheduling, study the structure of the optimal scheduler, and analyze its performance. In Section V, we numerically evaluate the analysis and several scheduling policies via simulations for binary trees. We conclude our paper in Section VI.

II. Network Model

We consider a tree network $T(V, E)$, where V denotes the set of n wireless nodes and E denotes the set of wireless links. One sink node denoted by v_s is located at the root of the tree, and collects data from $(n-1)$ sensor nodes. The task of the network is to compute a function from the sensed data, and to deliver the function value through the sink node. Data is forwarded via hop-by-hop communications to save energy, which costs a unit of transmission time per forwarding. The function should allow in-network aggregation from a partial collection of data without changing the final results or increasing transmission cost (time) for forwarding aggregated data. Common examples of such functions include min, max, average, histogram, etc.

We assume a time-slotted system with a single frequency channel for the entire system. Channels are assumed error free. However, there is wireless interference between channels, which is modeled by the well-known node-exclusive interference model[1]. Under this model, two links sharing a node cannot transmit at the same time. The model is suitable for Bluetooth and FH-CDMA networks [11], [12]. At each time slot, a link can transmit a single packet, and multiple links can transmit simultaneously only when they are greater than one hop away from each other.

[1]It is also denoted by the 1-hop interference model or the primary interference model.

The overall system operations are described as follows. Assume that a measurement event occurs at time $t = 0$. Each node senses its environment and generates data, which is denoted by a message. All messages from sensor nodes have to be delivered to the sink. During the first time slot, each message m generated at node v 1) is contained in a packet for transmission and forwarded to the parent node of v, 2) stays at node v due to transmissions from an interfering neighbor, or 3) waits at node v for further aggregation. The parent node that receives the packet may forward the message to its own parent node in the next time slot. After the first time slot, some of the nodes may have multiple messages. Under certain sensor network applications, nodes with multiple messages can compress the data and put them into a single packet without affecting the final results. This procedure is called as *in-network aggregation*. For example, suppose that the task of the network is to compute an average of the messages obtained from sensor nodes. If we format the content of a packet in a tuple of (the average of messages, the number of averaged messages), an intermediate node can compute an average of all the messages that it has received or generated, and package the computation result into a single tuple without losing any information. We assume that aggregation is permanent and messages that are aggregated will not split later. At each time slot, the process of transmission and aggregation is repeated until the sink collects all the messages. Clearly, aggregation will improve scheduling efficiency by reducing the number of packet transmissions. We assume that if in-network aggregation is allowed, it can be done with a negligible cost of energy or computing power. If aggregation is not allowed, we assume that one packet can contain only one message.

Our objective is to minimize the total delay in collecting all the messages related to a particular measurement event. Let $\hat{V}$ denote the set of all messages of the event. Each message is generated at a sensor node during $t = 0$ and no message is generated for $t > 0$. Messages do not need to measure the same physical environment, and may have a different meaning with a different importance, in which case, we quantify the importance of each message m using a weight $w_m > 0$. Let $f_m(t)$ denote the node where message m is located *at*[2] time t. We denote the system state at time t by its vector $f(t) := \{f_m(t)\}$. Also let $S(t)$ denote the schedule during time slot t, which is the set of nodes that transmit during time slot t. Note that each node $v \in S(t)$ will transmit a packet to its parent node $p(v)$, and those set of active links $(v, p(v))$ will correspond to a matching under the node-exclusive interference model. Let $\mathcal{S}$ denote the set of all possible schedules. Given a system state, let $\hat{H}(v)$ denote the set of messages that can be transmitted when node v is scheduled. Without aggregation, the set $\hat{H}(v)$ will have a single message, which might be the one with the largest weight among the messages located at node v. (See Fig. 1.) With aggregation, we assume that $\hat{H}(v)$ includes all messages at node v.

[2]Throughout the paper, we use the term 'at time t' to imply 'at the beginning of time slot t', and the term 'after time t' to imply 'at the end of time slot t' (after completion of scheduling).

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

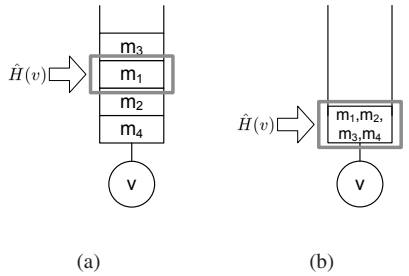


Fig. 1. Messages in the queue of node v with and without aggregation. Without aggregation, when node v is scheduled, the message m_1 will be forwarded. With aggregation, all messages are aggregated into a single packet and will be forwarded simultaneously. (a) Without in-network aggregation. (b) With in-network aggregation.

After time slot t, the system state will change as follows:

$$f_m(t+1) = \begin{cases} p(f_m(t)) & \text{if } f_m(t) \in S(t) \\ & \text{and } m \in \hat{H}(f_m(t)), \\ f_m(t) & \text{otherwise.} \end{cases} \quad (1)$$

Let t_m denote the time when message m arrives at the sink v_s, i.e., $t_m := \min\{t \mid f_m(t+1) = v_s\}$. Multiple messages, if aggregated, can arrive at the sink simultaneously and have the same arrival time. Our objective is to solve the following problem:

$$\underset{\{S(t)\in\mathcal{S}\}}{\text{minimize}} \quad \sum_{m\in\hat{V}} w_m t_m. \quad (2)$$

We rewrite the above problem using the delays that each message experiences. Let $\hat{S}(t)$ denote the set of messages that are scheduled during time slot t, i.e., $\hat{S}(t) := \{m \mid f_m(t) \in S(t), m \in \hat{H}(f_m(t))\}$. Further, let $h(v, v')$ denote the number of hops between two nodes v and v', and let h_m denote the number of hops between the initial location $f_m(1)$ of message m and the sink v_s, i.e., $h_m := h(f_m(1), v_s)$. Note that h_m is the least number of time slots for m to arrive at the sink when there is no other message. When multiple messages coexist in the network, two packets that contain a different message can compete with each other for scheduling resources. While a node is transmitting a packet, its neighboring nodes cannot transmit simultaneously due to wireless interference and the messages in those nodes have to be delayed staying at the current location. Let D_m denote the delays experienced by each message m in the network until it arrives at the sink, i.e., $D_m := t_m - h_m$. Since $\sum_{m\in\hat{V}} h_m$ is a constant, (2) is equivalent to

$$\underset{\{S(t)\in\mathcal{S}\}}{\text{minimize}} \quad \sum_{m\in\hat{V}} w_m D_m. \quad (3)$$

At each time slot, we have to determine which message m has to be forwarded, and which should stay for aggregation. Hence, in-network aggregation results in a tradeoff between scheduling efficiency and delays.

Remark: The problem (3) is similar to the scheduling for minimum delay shown in [13], where the delay optimal scheduler for linear networks has been provided. Although there is a similarity in the objective, the problem becomes much more complex here because the topology is richer (a tree topology), and we have to deal with aggregation. Hence, the solution of [13] is not applicable.

III. Impact of Aggregation

In this section, we study the effectiveness of in-network aggregation in terms of the total delay. Given a tree network of n nodes including the sink, we consider a scenario in which each sensor node generates a message, and all $(n-1)$ messages are equally important. Specifically, all weights w_m are set to 1. We show that without aggregation, the optimal delay is lower bounded by $O(n^2)$, while it is upper bounded by $O(n \log n)$ with aggregation. Therefore, the gain is quite substantial $O(n/\log n)$.

A. Performance without aggregation

For a given tree network, let $\tilde{d}$ and W denote the maximum depth and the maximum width of the tree, respectively. At time t, a message m is said to be located at depth d if $h(f_m(t), v_s) = d$. The following proposition provides a bound on the optimal delay performance.

Proposition 1: Without in-network aggregation, the delay is lower bounded by

$$\sum_{m\in\hat{V}} D_m \geq O(n^2),$$

when $\tilde{d} = O(\log n)$.

Proof: Since messages cannot be aggregated, each packet has to include only one message. Then, at the root, the sink can receive at most one packet during each time slot under the node-exclusive interference model. Since it takes at least $(n-1)$ time slots to receive all $(n-1)$ messages, we have that $\sum_{m\in\hat{V}} t_m \geq 1 + 2 + \cdots + (n-1) = n(n-1)/2$. Using the fact that $\sum_{m\in\hat{V}} h_m \leq 2\tilde{d}W^{\tilde{d}+1}$, we obtain

$$\sum_{m\in\hat{V}} D_m \geq \sum_{m\in\hat{V}} t_m - \sum_{m\in\hat{V}} h_m = O(n^2),$$

when $\tilde{d} = O(\log n)$. ■

B. Performance with aggregation

In-network aggregation can significantly reduce the delays by integrating multiple messages into a single packet. Since the reduction in the number of transmissions implies less interference, messages can be forwarded faster. We assume that messages can be aggregated into a single packet with no cost if they are located in the same node.

Proposition 2: With in-network aggregation, the delay is upper bounded by

$$\sum_{m\in\hat{V}} D_m \leq O(n \log n),$$

when $\tilde{d} = O(\log n)$.

Proof: We provide a scheduling policy based on link coloring and show its delay bound, which will serve as an upper bound on the delay of the optimal scheduler.

Since each node has at most $(W+1)$ links, we can color links using $(W+2)$ different colors such that two links with

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

the same color are not adjacent to each other [14]. Then we can schedule links with the same color simultaneously. We choose a color in a time slot in a round robin manner. Note that under this policy, each node transmits a packet to its parent every $(W+2)$ time slots. We can estimate the maximum delay that a message experiences before it arrives at the sink.

We group $(W+2)$ time slots as a period. By scheduling links based on their color, each node can transmit one packet to its parent during a period. Then any message located at depth d at time t will be located at depth $(d-1)$ or lesser after a period. This implies that all the messages initially located at depth d arrive at the sink by the end of the d-th period, i.e., $t_m \le d(W+2)$ if $h_m = d$. Since there are at most W^d nodes at each depth d, we have at most W^d messages with $h_m = d$. Hence, we obtain

$$\sum_{m\in\hat{V}} t_m \le \sum_{d=1}^{\tilde{d}} d(W+2)\cdot W^d \le 4\tilde{d}\cdot W^{\tilde{d}+2} = O(n\log n),$$

when $\tilde{d} = O(\log n)$. ■

IV. The Optimal Scheduling Policy and Its Performance Analysis

In this section, we are interested in finding an optimal scheduling policy and its performance with in-network aggregation.

Finding optimal schedules that minimize the total delay (3) is a difficult problem. Let us consider a simple example with three sensor nodes as shown in Fig. 2. Sensor nodes v_a, v_b, v_c each generate a message m_1, m_2, m_3, respectively, at $t=0$. All messages have to be delivered to the sink node v_s. Messages can be aggregated if they are located at v_a. Note that only one node can be scheduled at a time slot due to interference.

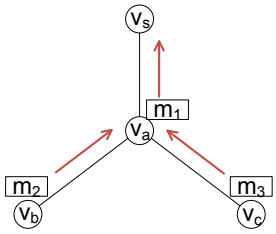


Fig. 2. A tree network with three sensor nodes. When all weights are equal to 1, the optimal schedule with aggregation is $\{v_b, v_a, v_c, v_a\}$, which achieves the total delay of 3.

There are four possible combinations of three schedules[3]. We do not take into account the cases that can be obtained by exchanging the initial location of m_2 and m_3. Let $\{S(1), S(2), \dots\}$ denote a sequence of schedules in time order, and let (x_1, x_2, x_3) represent the delay increment for each message m_1, m_2, m_3, respectively. For example, if $S(1) = v_b$, then while m_2 is forwarded from v_b to v_a, other messages m_1 and m_3 stay at v_a and v_c, respectively. Hence, the delay increments can be written as $(1,0,1)$. At the second time slot, since m_1, m_2 are now located at v_a and m_3 is at v_c, we can schedule either v_a or v_c. If $S(2) = v_c$, then m_1 and m_2 have to stay at v_a, and we have the delay increments $(1,1,0)$. Finally, at $t=3$, we can only schedule v_a, and forward all messages at the same time resulting in the delay increments $(0,0,0)$. After the third time slot, the sink collects all messages. The resultant delays are $(2,1,1)$ for the sequence of schedules $\{v_b, v_c, v_a\}$ and the delay is 4 when all weights are 1. Similarly, we can evaluate other schedules. Table I summarizes all the four sequences of schedules and their delay increments at each time slot. The best performance is achieved by $\{v_b, v_a, v_c, v_a\}$ with the total delay of 3. The example illustrates that scheduling the largest number of messages at each time slot may not result in the best performance when the objective is to minimize the sum delay, and that there are difficulties in finding optimal schedules even in a simple network graph. In the following, we further study the structure of the optimal scheduling policy.

[3]We exclude the schedules for which no packet is transmitted.

A. Optimal Scheduling

Let $\hat{A}(t)$ and $\hat{T}(t)$ denote the set of messages that have arrived at the sink, and are transient in the network, respectively, at time slot t. That is, $\hat{A}(t) = \{m \mid f_m(t) = v_s\}$, and $\hat{T}(t) = \{m \mid f_m(t) \ne v_s\}$. Clearly, we have $\hat{A}(1) = \emptyset$ and $\hat{T}(1) = \hat{V}$. Let $D_m(t)$ denote the delays experienced by m during $[0,t]$. Hence,

$$D_m(t) = \begin{cases} D_m(t-1) & \text{if } m \in \hat{S}(t)\cup\hat{A}(t), \\ D_m(t-1)+1 & \text{otherwise,} \end{cases}$$

and $D_m(0) = 0$ for all m. We define $\|\cdot\|$ as the sum of weights of messages in a set. For example, $\|\hat{V}\| = \sum_{m\in\hat{V}} w_m$ is the total weight of messages to be delivered. Then, letting[4] $t_{max} = \max_{m\in\hat{V}} t_m$, the objective of (3) can be rewritten as

$$\sum_{m\in\hat{V}} w_m D_m = \sum_{t=1}^{t_{max}} \sum_{m\in\hat{V}} w_m (D_m(t) - D_m(t-1)). \quad (4)$$

Note that $\sum_{m\in\hat{V}} w_m(D_m(t) - D_m(t-1))$ is the (weighted) delay increments due to the messages m that are transient at time slot t, but not scheduled during time slot t (i.e., $m \in \hat{T}(t)$, but $m \notin \hat{S}(t)$). Hence, it is equivalent to $\|\hat{T}(t)\backslash\hat{S}(t)\|$. Then we have that

$$\begin{aligned} \sum_{m\in\hat{V}} w_m D_m &= \sum_{t=1}^{t_{max}} \|\hat{T}(t)\backslash\hat{S}(t)\| \\ &= \sum_{t=1}^{t_{max}} \left(\|\hat{V}\| - \|\hat{S}(t)\| - \|\hat{A}(t)\| \right). \end{aligned} \quad (5)$$

The last equality holds since $\|\cdot\|$ is a linear function and $\hat{T}(t) = \hat{V}\backslash\hat{A}(t)$.

Since the schedule $\hat{S}(t)$ changes $\hat{T}(t+1)$ and $\hat{A}(t+1)$, an optimal scheduler requires future knowledge to determine the current schedule. For example, we can compute $\sum_{k=t}^{t_{max}} \|\hat{T}(k)\backslash\hat{S}(k)\|$ for each feasible schedule $\hat{S}(t)$ provided $\sum_{k=t+1}^{t_{max}} \|\hat{T}(k)\backslash\hat{S}(k)\|$. This approach can be implemented in

[4]Note that t_{max} depends on the sequence of schedules. However, in this paper, we do not make use of any properties of t_{max}, and use it for notational convenience to denote the time when all messages arrive at the sink. Hence, it can be replaced with an arbitrary large time.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

TABLE I
POSSIBLE SCHEDULES FOR THE EXAMPLE OF FIG. 2 AND THE DELAY INCREMENTS.

Schedules $\{S(1), S(2), \dots\}$	$t=1$	$t=2$	$t=3$	$t=4,5$	(D_1, D_2, D_3)	$\sum D_i$
$\{v_b, v_c, v_a\}$	(1,0,1) +	(1,1,0) +	(0,0,0) =		(2,1,1)	4
$\{v_b, v_a, v_c, v_a\}$	(1,0,1) +	(0,0,1) +	(0,0,0) +	(0,0,0) =	(1,0,2)	3
$\{v_a, v_b, v_c, v_a\}$	(0,1,1) +	(0,0,1) +	(0,1,0) +	(0,0,0) =	(0,2,2)	4
$\{v_a, v_b, v_a, v_c, v_a\}$	(0,1,1) +	(0,0,1) +	(0,0,1) +	(0,0,0) =	(0,2,2)	4

a recursive manner, which, however, requires an extremely high computational complexity since all possible sequences $\{\hat{S}(t), \hat{S}(t+1), \dots\}$ have to be examined. Because of these reasons, instead of analyzing the optimal scheduler, we focus on obtaining a system wide lower bound on the optimal performance.

B. Performance of optimal scheduling

In this section, we analyze the performance of the optimal scheduling scheme for tree networks. Let $\{S^*(t)\}$ denote a sequence of optimal schedules with $1 \le t \le t_{max}$. For each $S^*(t)$, we denote the corresponding set of scheduled messages by $\hat{S}^*(t)$, i.e., $\hat{S}^*(t) = \{m \mid f_m(t) \in S^*(t)\}$[5]. Also let M_1^* denote the maximum weighted matching in the first time slot, and let $\hat{M}_1^*$ denote the set of messages scheduled by M_1^*, i.e.,

$$\begin{aligned} M_1^* &:= \operatorname*{argmax}_{S \in \mathcal{S}} \sum_{m \in \hat{V}} w_m \cdot \ {}_{\{f_m(1) \in S\}}, \\ \hat{M}_1^* &:= \{m \mid f_m(1) \in M_1^*\}, \end{aligned} \tag{6}$$

where ${}_{\{\cdot\}}$ denotes the indicator function. We have the following lemma.

Lemma 1: The delays during the first time slot satisfies that

$$\sum_{m \in \hat{V}} w_m D_m(1) \ge \|\hat{V}\| - \|\hat{M}_1^*\|.$$

Proof: Since $\sum_{m \in \hat{V}} w_m D_m(1) = \|\hat{V}\| - \|\hat{S}^*(1)\| - \|\hat{A}(1)\|$, the result follows from $\|\hat{S}^*(1)\| \le \|\hat{M}_1^*\|$ by (6) and the fact that $\hat{A}(1) = \emptyset$. ■

Let C denote the set of children nodes of the sink, i.e., $C := \{v_p \mid h(v_p, v_s) = 1\}$. Let T_p denote the subtree rooted at $v_p \in C$. Let V_{pq} denote the subset of nodes in T_p with depth q, i.e., $V_{pq} := \{v \mid h(v, v_s) = q, h(v, v_p) = q-1\}$. Let $\hat{V}_{pq}$ denote the set of messages generated within V_{pq}. We have the following lemma for $t \ge 2$.

Lemma 2: The delays satisfy that for each p, q,

$$\sum_{m \in \hat{V}_{pq}} w_m (D_m(t_{max}) - D_m(1)) \ge \|\hat{V}_{pq}\| - \max_{m \in \hat{V}_{pq}} w_m. \tag{7}$$

The lemma implies that after the first schedule $S^*(1)$, each message generated in V_{pq} will still compete for scheduling resources with at least one other message generated in V_{pq} before it arrives at the sink.

Proof: The basic idea of the proof is that if there are two messages located at the same depth at time t, they should have the same receiver at some time $t' \ge t$ unless one of them is delayed in the meantime. Hence, at least one of them must experience a delay during $[t, t']$.

We prove the lemma by contradiction. Suppose that there is $\hat{V}_{pq}$ such that (7) does not hold. Note that if all messages in $\hat{V}_{pq}$ experience exactly one delay for $t \ge 2$, we have that $\sum_{m \in \hat{V}_{pq}} w_m (D_m(t_{max}) - D_m(1)) = \|\hat{V}_{pq}\|$. Hence, the inequality $\sum_{m \in \hat{V}_{pq}} w_m (D_m(t) - D_m(1)) < \|\hat{V}_{pq}\| - \max_{m \in \hat{V}_{pq}} w_m$ implies that there are at least two messages $m_1, m_2 \in \hat{V}_{pq}$ that arrive at the sink with no delay. For such $m_1, m_2 \in \hat{V}_{pq}$, we have that $D_{m_1}(t) - D_{m_1}(t-1) + D_{m_2}(t) - D_{m_2}(t-1) = 0$ for all $2 \le t \le t_{max}$. In the following, we show that two messages m_1, m_2 have to interfere with each other during some time slot $t' \ge 2$, which leads to a contradiction to $D_{m_1}(t') - D_{m_1}(t'-1) + D_{m_2}(t') - D_{m_2}(t'-1) = 0$.

Let v_a and v_b denote the node that generates m_1 and m_2, respectively. They have the same depth q. The location of m_1 and m_2 after the first schedule can be different based on whether each of v_a and v_b is scheduled or not. There are four possibilities: i) $(f_{m_1}(2), f_{m_2}(2)) = (v_a, v_b)$, ii) $(p(v_a), v_b)$, iii) $(v_a, p(v_b))$, and iv) $(p(v_a), p(v_b))$. We consider the first two cases in this proof. The results for the others can follow using the same line of analysis.

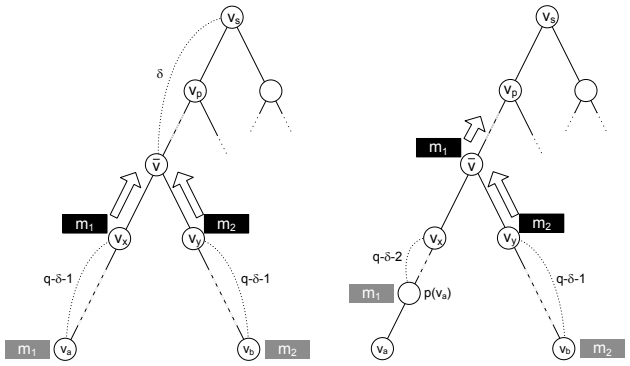
Case i) Since $v_a, v_b \in V_{pq}$, two paths $(v_a, \dots, v_p)$ and $(v_b, \dots, v_p)$ have a set of common nodes. Let $\bar{v}$ denote the farthest node from v_p among those common nodes in the two paths. (See Fig. 3(a).) Let $\delta := h(\bar{v}, v_s)$. Also let v_x and v_y denote the child of $\bar{v}$ on the path from v_a and v_b, respectively. Then since $D_{m_1}(t) - D_{m_1}(t-1) + D_{m_2}(t) - D_{m_2}(t-1) = 0$ for all $2 \le t \le t_{max}$, messages m_1 and m_2 will be continuously scheduled, and arrive at v_x and v_y, respectively, after time $q - \delta$, i.e., $f_{m_1}(q - \delta + 1) = v_x$ and $f_{m_2}(q - \delta + 1) = v_y$. From our construction, we have that $v_x \ne v_y$ and both nodes are a child of $\bar{v}$. Since two links $(v_x, \bar{v})$ and $(v_y, \bar{v})$ cannot be scheduled at the same time under the node-exclusive interference model, at least one of two messages have to be delayed for the transmission of the other during time slot $t' = q - \delta + 1$, which contradicts our assumption.

Case ii) Using the similar argument, we have that $f_{m_1}(q - \delta + 1) = \bar{v}$ and $f_{m_2}(q - \delta + 1) = v_y$. Note that $\bar{v}$ belongs to the subtree T_p, and hence $\bar{v} \ne v_s$. Since $\bar{v}$ is not the sink, messages m_1 and m_2 have to be forwarded simultaneously at the next time slot to $p(\bar{v})$ and $\bar{v}$, respectively as shown in Fig. 3(b). However, a node cannot be a sender and a receiver at the same time due to the interference constraint. Hence, we arrive at a contradiction. ■

Let m_{pq} denote the message in $\hat{V}_{pq}$ with the largest weight, i.e., $m_{pq} = \operatorname{argmax}_{m \in \hat{V}_{pq}} w_m$. If there are multiple messages

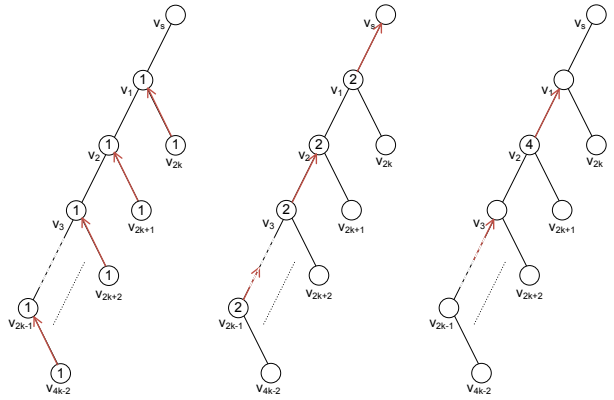
[5]We omit the condition $m \in \hat{H}(f_m(t))$ since, under our aggregation assumption, all messages in node v are included in $\hat{H}(v)$.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.



(a) Case 1 (b) Case 2

Fig. 3. Case 1 shows that two messages that were located at the same depth q at time $t = 2$ (messages colored in gray) compete for scheduling at their first common parent $\bar{v}$ at some time t' (messages colored in black), assuming that both are not delayed in the meantime. Case 2 is for when one of them was located at depth $q - 1$ due to the schedule during $t = 1$.



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$

Fig. 4. Example network and the schedules that achieve the equality of (8). Circles denote nodes and lines denote wireless links. At each time slot, the sum weight of messages in a node is presented in the circle, and the schedules are depicted as a set of arrows. Delays increase by the unscheduled messages. For $t \geq 3$, there is no unscheduled message.

with the same weight, we choose one arbitrarily. Let $\hat{X}_p := \{m_{pq}\}$ denote the set of such messages that are located at all depth q nodes in the subtree T_p at time $t = 1$. Clearly, $\hat{X}_p$ includes at most $\tilde{d}$ messages. Let $\hat{X} := \cup_{p\in C}\hat{X}_p$. We can obtain the following performance bound on the delay.

Proposition 3: Under a tree network, we have that

$$\sum_{m\in\hat{V}} w_m D_m \geq 2\|\hat{V}\| - \|\hat{M}_1^*\| - \|\hat{X}\|, \tag{8}$$

and the bound is tight for certain topologies.

Proof: From (4) and the fact that $D_m(0) = 0$ for all $m \in \hat{V}$, we have

$$\begin{aligned}\textstyle\sum_{m\in\hat{V}} w_m D_m &= \textstyle\sum_{m\in\hat{V}} w_m D_m(1) \\ &\quad + \sum_{p\in C}\sum_{q=1}^{\tilde{d}}\sum_{m\in\hat{V}_{pq}} w_m(D_m(t_{max}) - D_m(1)) \\ &\geq \|\hat{V}\| - \|\hat{M}_1^*\| + \sum_{p\in C}\sum_{q=1}^{\tilde{d}}\left(\|\hat{V}_{pq}\| - \max_{m\in\hat{V}_{pq}} w_m\right) \\ &= 2\|\hat{V}\| - \|\hat{M}_1^*\| - \|\hat{X}\|,\end{aligned}$$

where the inequality comes from Lemmas 1 and 2, and the last equality comes from the definition of $\hat{X}$. ■

Remarks: Although Proposition 3 can apply to all tree networks, our main focus is trees with small width, e.g., binary tree and linear topology, since they are commonly used in practice. For trees with a large width, the bound could be further tightened. Also note that (8) can be calculated from the initial system state $f(1)$. Hence, given a system state $f(t)$, we can utilize this as the bound of the remaining delays afterward. This will be a useful tool for designing an online scheduling algorithm since we can estimate the effect of a schedule in the long run. We use this approach in Section V-B.

The bound of Proposition 3 is tight under certain network topologies. Consider the following example where the topology is a binary tree.

Example: We consider a binary tree network depicted in Fig. 4(a). There are $(4k - 1)$ nodes denoted by $v_s, v_1, \ldots, v_{4k-2}$, where v_s is the sink node and the others are sensor nodes. We denote a right leaf node by the leaf node that is the right child of its parent. Let L denote the set of right leaf nodes, i.e., $L = \{v_{2k}, \ldots, v_{4k-2}\}$. All weights are equal to 1.

It is clear that $\|\hat{V}\| = 4k-2$ and $\|\hat{M}_1^*\| = 2k-1$. Also from $\hat{X} = \{m \mid f_m(1) \in \{v_1\} \cup L\}$, we obtain that $\|\hat{X}\| = 2k$. Then from Proposition 3, we have

$$\sum_{m\in\hat{V}} w_m D_m \geq 4k - 3. \tag{9}$$

Now we consider a scheduling policy as follows:

- During the first time slot, we schedule all nodes in L as shown in Fig. 4(a). Then the delays increase for the messages out of L, and the total delay increment would be $2k - 1$.
- During the second time slot, we schedule nodes of an odd index among those with a message as shown in Fig. 4(b). The total delay increment due to unscheduled messages would be $2(k - 1)$.
- Since completion of the second time slot, all messages are aligned in a line from the sink, and no two messages will be within an 1-hop distance. Then, we can schedule all messages simultaneously as shown in Fig. 4(c) until they arrive at the sink.

Under this policy, the total delay sum can be calculated as

$$\sum_{t=1}^{t_{max}}\sum_{m\in\hat{V}} w_m(D_m(t) - D_m(t-1)) = 4k - 3.$$

Hence, the equality of (9) holds, and the bound is tight.

V. Numerical Results

We now investigate the performance of several scheduling algorithms versus our delay bound as a performance bench-

mark. We assume that the objective function, e.g., average of sensed data, conforms to our aggregation constraints, i.e., in-network computations with a partial collection of data do not change the final results and do not increase transmission cost of aggregated data. We classify scheduling policies into myopic and non-myopic ones: Myopic policies make a scheduling decision based only on the current system state. Most conventional scheduling algorithms are myopic. Non-myopic policies simulate the network with a sequence of schedules, and make a scheduling decision for time slot t based on simulation results. For example, a non-myopic policy that can simulate the network up to t_{max} steps can solve (3) by simulating $\{S(1), \ldots, S(t_{max})\}$.

In our experiments, we focus on binary tree networks. We first examine the optimal performance for full binary trees and move to binary trees that are randomly generated. We also compare the analytical results with the performance of scheduling policies when weights are assigned at random.

A. Myopic scheduling policies

For a feasible schedule $S \in \mathcal{S}$, let $\mathcal{D}(f(t), S)$ denote the total delay increment *if* S is applied to the system state $f(t)$, i.e.,

$$\mathcal{D}(f(t), S) := \sum_{m \in \hat{V}} w_m \ \ {\scriptstyle \{f_m(t) \notin S, f_m(t) \neq v_s\}}. \qquad (10)$$

Hence, if we determine S as the schedule at time slot t, we have $\sum_{m \in \hat{V}} w_m (D_m(t) - D_m(t-1)) = \mathcal{D}(f(t), S)$.

Since myopic scheduling policies make scheduling decisions based on the current state of the network, we consider the following conventional scheduling algorithms:

- Maximum Weight Matching (MWM): During each time slot t, it schedules the set of nodes that minimize the delay increment of messages for the time slot, i.e.,

  $$S(t) \leftarrow \operatorname*{argmin}_{S \in \mathcal{S}} \mathcal{D}(f(t), S).$$

  It is equivalent to $\operatorname{argmax}_{S \in \mathcal{S}} \sum_{m \in \hat{V}} w_m \ \ {\scriptstyle \{f_m(t) \in S\}}$, which is an MWM algorithm with the sum weight of messages located at a node.
- Maximum Size Matching (MSM): During each time slot t, it schedules the set of nodes that maximize the total number of scheduled nodes with a message, i.e.,

  $$S(t) \leftarrow \operatorname*{argmax}_{S \in \mathcal{S}} \sum_{v \in S} \ \ {\scriptstyle \{\exists m \text{ s.t. } f_m(t) = v\}}.$$

- Greedy Maximal Matching (GMM): At each time slot t, it sorts nodes based on the sum of message weights in each node, and includes nodes in the schedule $S(t)$ in decreasing order conforming to the inference constraints.

Note that all three policies are myopic in the sense that they determine $S(t)$ based on the system state at time slot t, i.e., $f(t)$. One may expect that MWM will achieve the best performance since it serves the largest weights during the time slot. However, since data aggregation changes the system states over time, it may cause additional delays in the future as shown in the next experiment.

We establish full binary trees with different number of nodes. For full binary trees, all nodes except leaves have two children. All weights are set to 1. Table II illustrates the results. The total delay of all messages are presented in unit of time slots. Interestingly, GMM performs slightly better than MWM, which confirms that MWM is not the best solution. Note that it does not imply that GMM outperforms MWM. Indeed, their performance depends on how a tie breaks when there are multiple schedules with the same weight sum. In this experiment, we broke a tie arbitrarily.

TABLE II
TOTAL DELAYS IN THE NUMBER OF TIME SLOTS UNDER FULL BINARY TREES OF n NODES.

Tree size (n)	3	7	15	31
MSM	1	11	29	103
GMM	1	7	22	60
MWM	1	7	25	66
Analysis (8)	1	6	17	42

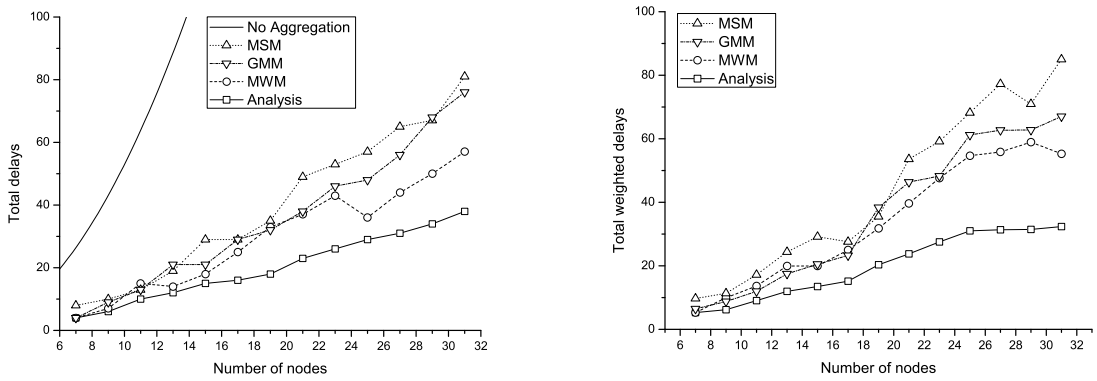
Next, we evaluate the scheduling policies in a randomly generated binary tree. Starting from the root, we generate a child for a node with probability 0.6. A node has at most two children. Nodes are added in a width-first manner. We first build a random binary tree with $n = 7$, and evaluate our analysis comparing with scheduling algorithms. Then we *add* additional nodes to the existing tree (randomly, in a width-first manner), and run the same experiment. We repeat this until $n = 31$. The reason that we construct the trees in an incremental manner is that the performance of scheduling policies depends on the underlying network topology and it is hard to directly compare the performance of a scheduling policy under two totally different topologies.

The results are depicted in Fig. 5. We simulate two cases: i) all weights are equal to 1, and ii) weights are assigned at random between $(0, 2)$. In Fig. 5(a), the (lower) performance bound on the delay of the optimal scheduling without aggregation is shown as a solid line at the left top corner. Comparing with the scheduling policies with aggregation, it confirms that in-network aggregation provides a significant gain. With aggregation, the analytical results show that the lower bound increases linearly as the tree grows. Although the bounds are tight for the trees of small size, there is an increasing gap between the analysis and the performance of scheduling policies as the network size increases, which is more evident under the random weight assignment. The gap may imply that the bound is not tight, or that scheduling policies in the experiments are not optimal, or both. The results when $n = 25$ in Fig. 5(a), however, suggest that the analytical results are close to the optimal. Comparing all myopic scheduling algorithms, MWM outperforms the others in most scenarios.

B. Non-myopic scheduling policies

In this section, we propose three non-myopic scheduling policies and evaluate their performance. Under non-myopic scheduling policies, a scheduling decision is based on the

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.



(a) All messages have the same weight 1.

(b) Weights are assigned at random between $(0.0, 2.0)$.

Fig. 5. Performance of myopic scheduling policies. The solid line at the left top in Fig. 5(a) is the (lower) performance bound of scheduling without data aggregation, which shows that the gain of in-network aggregation is significant. As for scheduling with aggregation, MWM performs best among the myopic schedulers. The gap between MWM and the analysis can reduce using non-myopic scheduling policies as shown in Fig. 6.

current and the future system states, which can be obtained by simulating the network. A higher performance is expected as further future states are taken into account. However, due to computational complexity in implementation, we consider *one-step* non-myopic policies only.

Let $\langle f(t), S\rangle$ denote the system state at time $t+1$ if the schedule S is applied to $f(t)$. One-step non-myopic policies determine the schedule $S(t)$ considering the current state $f(t)$ and the one-step future state $\langle f(t), S(t)\rangle$. To this end, we use MWM, GMM, or our analysis for estimating the delays in the future state. Specifically, we propose the following algorithms:

- Non-Myopic MWM (NM-MWM): During each time slot t, it schedules the set of nodes that will minimize the delay increments during $[t, t+1]$, i.e.,

$$S(t) \leftarrow \underset{S\in\mathcal{S}}{\operatorname{argmin}} \Big(\mathcal{D}(f(t), S) + \min_{S'\in\mathcal{S}} \mathcal{D}(\langle f(t), S\rangle, S')\Big).$$

  The second schedule S' that minimizes the delay increment during time $t+1$ can be obtained using the MWM algorithm on the future system state $\langle f(t), S\rangle$.
- Non-Myopic MWM with Analysis (NM-MWM+A): It schedules the set of nodes that minimize the future delay bound, which can be obtained using our analysis. Let $\mathcal{D}_*(f(t))$ denote the bound (8), where $\hat{V}$, $\hat{M}_1^*$, and $\hat{X}$ are accordingly calculated assuming that $f(t)$ is the initial state (i.e., $\hat{V} = \hat{T}(t)$ and so on.) Note that the calculation includes the estimation of the maximum weight matching $\|\hat{M}_1^*\|$. NM-MWM+A selects its schedule as

$$S(t) \leftarrow \underset{S\in\mathcal{S}}{\operatorname{argmin}} \Big(\mathcal{D}(f(t), S) + \mathcal{D}_*(\langle f(t), S\rangle)\Big).$$

- Non-Myopic GMM (NM-GMM): It is similar with NM-MWM except that for each feasible S, it chooses the second schedule S' using the GMM algorithm. That is,

$$S(t) \leftarrow \underset{S\in\mathcal{S}}{\operatorname{argmin}} \Big(\mathcal{D}(f(t), S) + \mathcal{D}(\langle f(t), S\rangle, S')\Big),$$

  where S' is chosen in decreasing order of the sum weight that are calculated on the future system state $\langle f(t), S\rangle$.

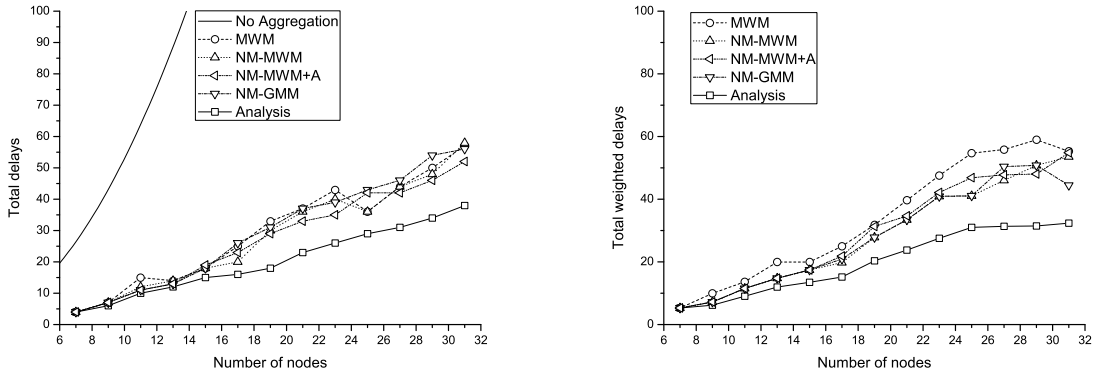
Note that NM-MWM and NM-MWM+A need to find the maximum weight matching in the simulated network, while NM-GMM selects the second schedule using the GMM algorithm. Therefore, NM-GMM has a less computational complexity than the others.

We again start our experiments with the full binary trees. As illustrated in Table III, the results show that the non-myopic scheduling policies have a comparable performance to MWM.

TABLE III
TOTAL DELAYS IN THE NUMBER OF TIME SLOTS UNDER FULL BINARY TREES OF n NODES.

Tree size (n)	3	7	15	31
MWM	1	7	25	66
NM-MWM	1	7	23	64
NM-MWM+A	1	7	23	65
NM-GMM	1	7	23	71
Analysis (8)	1	6	17	42

We next conduct experiments on the binary trees that are randomly generated as in Section V-A. We use the same topologies and the weight assignments for the comparison purpose. Figs. 6(a) and 6(b) illustrate the results when weights are assigned equally and randomly, respectively. Overall, proposed non-myopic scheduling policies outperform conventional myopic scheduling policies, and reduce the gap from the analytical lower bound. We expect that the performance will improve as we acquire more future information by simulating the network further, which however complicates the scheduling algorithm. When the weights are assigned equally, NM-MWM slightly outperforms MWM. The performance differences enlarge when the weights are assigned randomly. The gain of non-myopic scheduling policies is clear when comparing the results with the performance of myopic scheduling policies



(a) All messages have the same weight 1.

(b) Weights are assigned at random between $(0.0, 2.0)$.

Fig. 6. Performance of non-myopic scheduling policies. The performance of MWM is added for the comparison with myopic scheduling policies. Fig. 6(a) shows that non-myopic scheduling policies perform as good as MWM when all the weights are the same. Fig. 6(b) illustrates that the non-myopic policies outperform myopic policies including MWM when weights are randomly assigned. NM-GMM achieves the performance comparable to (sometimes better than) NM-MWM and NM-MWM+A.

shown in Figs. 5(a) and 5(b). Also note that the performance of NM-MWM, NM-MWM+A, and NM-GMM are comparable. Since NM-GMM is simpler than NM-MWM and NM-MWM+A, it can serve as an alternative to the optimal scheduler in practice.

VI. Conclusion

We investigate the delay performance of a sensor network with in-network aggregation. We formulate a delay minimization problem in the presence of wireless interference, and quantify the gain of the in-network aggregation for reducing the total delay in wireless sensor networks. We provide a system wide lower bound on the delay performance of the optimal joint data aggregation and scheduling policy. The specific bounds depend on the underlying network topology and the associated weight assignment. Under certain network settings, we show that the bound can be achieved.

We evaluate our analysis using binary trees with different weight assignments. The results show that our analysis is accurate, in particular when the network size is small. We also evaluate the performance of several myopic and non-myopic scheduling policies, and show that the one-step non-myopic scheduling policies achieve substantially lower delays than the conventional myopic ones. In particular, the non-myopic GMM seems to achieve good performance with moderate complexity making it attractive as a viable alternative over the optimal solution.

There are many interesting directions for future work. One can obtain analytic results when the underlying tree networks have a larger width, and develop a simple distributed algorithm with high performance. Performance of scheduling policies will depend on different interference models, e.g., matrix-based interference model or K-hop interference models, and needs to be quantified in those settings. It could also be interesting to study the scheduling performance when the energy cost of in-network aggregation itself is not negligible.

References

[1] D. L. Hall and S. A. H. McMullen, *Mathematical Techniques in Multisensor Data Fusion*. Artech House, Inc., 2004.
[2] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *the 33rd Hawaii International Conference on System Sciences*, 2000.
[3] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *ACM MOBICOM*, 2000.
[4] S. Hariharan and N. B. Shroff, "Maximizing Aggregated Revenue in Sensor Networks under Deadline Constraints," in *IEEE CDC*, 2009, to appear.
[5] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," in *ICDCSW*, 2002.
[6] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks," in *the International Conference on Networking*, August 2002.
[7] A. Goel and D. Estrin, "Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-bulk," in *ACM SODA*, 2003.
[8] Y. Wu, S. Fahmy, and N. B. Shroff, "On the Construction of a Maximum-Lifetime Data Gathering Tree in Sensor Networks: NP-Completeness and Approximation Algorithm," in *IEEE INFOCOM*, April 2008.
[9] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Energy-latency Tradeoffs for Data Gathering in Wireless Sensor Networks," in *IEEE INFOCOM*, 2004.
[10] L. Becchetti, P. Korteweg, A. Marchetti-Spaccamela, M. Skutella, L. Stougie, and A. Vitaletti, "Latency Constrained Aggregation in Sensor Networks," in *the 14th conference on Annual European Symposium*, 2006.
[11] B. A. Miller and C. Bisdikian, *Bluetooth Revealed: the Insider's Guide to an Open Specification for Global Wireless Communication*. Prentice Hall PTR, 2001.
[12] L. Tassiulas and S. Sarkar, "Maxmin Fair Scheduling in Wireless Networks," in *IEEE INFOCOM*, 2002.
[13] L. Tassiulas and A. Ephremides, "Dynamic Scheduling for Minimum Delay in Tandem and Parallel Constrained Queueing Models," *Annals of Operations Research*, vol. 48, no. 4, pp. 333–355, August 1994.
[14] R. Diestel, *Graph Theory*, 3rd ed. Springer, 2005.