Maximizing Aggregated Revenue in Sensor Networks under Deadline Constraints

Srikanth Hariharan and Ness B. Shroff

Abstract—We study the problem of maximizing the aggregated revenue in sensor networks with deadline constraints. Our model is that of a sensor network that is arranged in the form of a tree topology, where the root corresponds to the sink node, and the rest of the network detects an event and transmits data to the sink over one or more hops. We assume a time-slotted synchronized system and a node-exclusive (also called a primary) interference model. We formulate this problem as an integer optimization problem and show that the optimal solution involves solving a Bipartite Maximum Weighted Matching problem at each hop. We propose a polynomial time algorithm based on dynamic programming that uses only local information at each hop to obtain the optimal solution. Thus, we answer the question of when a node should stop waiting to aggregate data from its predecessors and start transmitting in order to maximize revenue within a deadline imposed by the sink. Further, we show that our optimization framework is general enough that it can be extended to a number of interesting cases such as incorporating sleep-wake scheduling, minimizing aggregate sensing error, etc.

I. INTRODUCTION

A Wireless Sensor Network is a wireless network consisting of a number of sensors that are distributed in a region in order to cooperatively monitor certain physical or environmental conditions. These networks are used in a number of civilian and military applications, such as environment and habitat monitoring, battlefield surveillance, and traffic control. Due to size and cost constraints, sensor nodes have limited energy, processing power, memory, and bandwidth resources. Typically, these nodes sense a desired aspect of the region in which they are deployed and occasionally report the sensed data to those sinks that have subscribed for that data. The sensed data is prone to error due to resource constraints and environmental factors. Therefore, sinks cannot rely on the data sensed by a single sensor. On the other hand, there is redundancy in the data sensed by different sensors. In many applications, the sinks only desire an aggregated form of the data sensed by different sensor nodes. Examples include finding the average temperature in a region, determining whether pressure in a region is below a certain value and determining the average location and velocity of a target. It is known that when sinks require an aggregated form of the sensed data, performing in-network computation greatly reduces the communication overhead [1].

One of the key issues in data aggregation in sensor networks is the tradeoff between delay and energy. This can be viewed as follows. Consider a data aggregation tree in which each parent aggregates data from all its children before forwarding it to the next hop. Assuming error-free links and no collisions, each parent then needs to make at most one transmission. However, each parent will have to wait until all its children send their data. On the other hand, if each parent decides to transmit every time it receives a packet from one of its children, then this results in excessive transmissions, defeating the purpose of data aggregation.

Thus, there is a delay energy tradeoff that needs to be carefully considered depending on the level of delay an application can tolerate. Moreover, the quality of data reported at the sink is also important. In particular, we are interested in finding the maximum revenue that can be obtained when a deadline is imposed by the sink after which the data is no longer useful and when there exist energy constraints. Currently, there exist a number of techniques addressing the formation of data aggregation trees ([2],[3],[4]). Our goal in this paper is to maximize the revenue in a given data aggregation tree with the sink as the root. For error-free links, "revenue" can be thought of in a number of ways. For example, if each packet has a priority associated with it, we can maximize the sum of the priorities of packets accounted for at the sink. If each node senses data with a certain accuracy, we can think of maximizing the accuracy of the aggregated data at the sink. For simplicity, we will first define revenue as the number of nodes whose packets have been accounted for at the sink within the imposed deadline.

We now briefly examine the related work in this area. In [5], Krishnamachari et. al., show that forming the optimal data aggregation tree, when the goal is to minimize the number of transmissions, is NP-Hard. They propose three sub-optimal schemes and compare their performance. Rajagopalan and Varshney [6] perform an extensive survey on data aggregation protocols. Bouhis et. al., [7] study tradeoffs between energy and data accuracy in data aggregation trees. In [8], Yu et. al., study tradeoffs between energy and latency in data aggregation trees assuming a time-slotted synchronized system. As mentioned in [8], enforcing the latency constraint requires time-synchronization schemes such as [9]. Interference is not a part of the optimization framework in any of these works. In this work, we propose an optimization framework that can be used to study tradeoffs between revenue, energy and latency, in a time-slotted system, under...
We model the system as a graph \( G(V, E) \) where \( V \) is the set of nodes and \( E \) is the set of links. The system has \( N \) nodes and a sink. When an event occurs, nodes sense some desired quantity and send an aggregated form of the data to the sink. A node may or may not be a source for a particular event. Every node can perform in-network computation of the data that it receives. We assume that the system is time-slotted and synchronized. During each slot, a node can perform only one of the following: sending a packet, receiving a packet, or remaining idle. We assume that the aggregation delay (the time required to aggregate data from different sensors) is negligible. We assume that the capacity of each link is fixed and equal to 1. However, it is straightforward to extend the proposed algorithm to links with arbitrary fixed capacities. We also assume that links are error-free. By error-free links, we mean that channels are error-free. Packets may still get lost due to interference. The sink imposes an event-dependent deadline within which it should receive data from the sensor nodes. We assume that events are static events in which each source node associated with that event knows that some parameter needs to be sensed at a particular time. For example, sensor nodes could periodically sense temperature in a region and report some aggregated form of the data to the sink. We discuss about dynamic events, such as tracking an object, in Section 6.

For each event, we assume that all the source nodes that are associated with the event, sense this event and are ready to transmit their observation at time zero. Each node then waits for a certain time to aggregate data from its predecessors. The sink must receive the aggregated data within a deadline. We assume that the next event occurs only after the deadline for the current event expires. We consider the one-hop or node exclusive interference model. In the one-hop interference model, any two links that share a node cannot be active at the same time.

The aggregation function we consider, can be any divisible function [12]. Divisible functions are those that can be computed in a divide and conquer fashion. For example, assume that the sink desires the function \( f(x_1, x_2, \ldots, x_N) \), where \( x_1, x_2, \ldots, x_N \) are the raw data measurements of the \( N \) sensor nodes. Let \( S \) denote the set \( \{x_1, x_2, \ldots, x_N\} \) and let \( f(S) \) denote \( f(x_1, x_2, \ldots, x_N) \). The function \( f \) is divisible if, given any partition of \( S \), \( P(S) = \{S_1, \ldots, S_j\} \), there exists a function \( g^{P(S)} \), such that \( f(S) = g^{P(S)}(f(S_1), f(S_2), \ldots, f(S_j)) \). The complete definition can be found in [12]. Examples of divisible functions include MIN, MAX, Sum etc. Consider the MAX function for instance. Suppose the sink desires \( MAX(1, 2, 3, 4, 5) \). Then, given a partition, (say) \( \{\{1, 2\}, \{3, 4\}, \{5\}\} \) of \( \{1, 2, 3, 4, 5\} \), \( MAX(1, 2, 3, 4, 5) = MAX(MAX(1, 2), MAX(3, 4), MAX(5)) \). While Average is not a divisible function, it can also be computed if the total number of source nodes is known at the sink.

III. Problem Formulation - Error-Free Links

Consider a wireless sensor network with \( N \) nodes and a sink. Before we formulate our problem in this section, we first make some general observations regarding the complexity of forming a data aggregation tree when the set of source nodes is arbitrary. As mentioned before, Krishnamachari et al., [5] have shown that finding a data aggregation tree that minimizes the number of transmissions from an arbitrary set of source nodes to a sink, is NP-Hard. We observe that this result holds even when the sink imposes a deadline. Propositions 1 and 2 show this result.

**Proposition 1:** For a network with \( N \) nodes and a sink, it takes at most \( N \) time slots to send data from every source node to the sink when data aggregation is performed.

**Proof:** We will assume that the network is connected so that there exists a path from any node to the sink. Construct a spanning tree with the sink as the root. Consider a data aggregation policy, in which, apart from the sink, every other node in this spanning tree aggregates data from all its children before transmitting to its parent. In this policy, a node makes a transmission at most once.

Consider any general interference model and assume that a central scheduler decides the scheduling policy. In the worst case, no two nodes in the spanning tree can transmit simultaneously. In this case, it takes at most \( N \) slots to send data from every source node to the sink, since the number of links in the spanning tree is \( N \).

**Proposition 2:** Finding a data aggregation tree that minimizes the number of transmissions from an arbitrary set of source nodes to the sink, when the sink has imposed a deadline, is NP-Hard.

**Proof:** Consider the special case when the deadline is \( N \). From Proposition 1, it is possible to send the aggregated data from every node to the sink within this deadline. Therefore, the result follows from Proposition 1 and the...
fact that finding a data aggregation tree that minimizes the number of transmissions from an arbitrary set of sources to the sink is NP-Hard [5].

We now consider a network, modeled as a tree with the sink being the root, and formulate the problem of maximizing revenue at the sink when the sink imposes a deadline.

We first describe some notations and definitions.

- \( V \) - Set of \( N \) sensor nodes and a sink, \( S \).
- \( E \) - Set of edges.
- \( T_i \) - Denotes whether node \( i \) is a source for a particular event. Thus, it denotes whether node \( i \) has its own packet to send for a particular event.
- \( n_{ij} \) - The number of transmissions from node \( i \) to its parent node \( j \) for a particular event, \( n_{ij} \in \{0, 1\} \).
- \( W_i \) - The time for which node \( i \) waits to aggregate packets from its predecessors for a particular event. After \( W_i \), node \( i \) will no longer accept packets from its predecessors. Also, until \( W_i \) expires, node \( i \) will not transmit any aggregated packet to its parent. As mentioned before, we assume that the event is sensed by each source node at time zero.
- \( V_L \) - Set of all leaf nodes in the tree.
- \( M_i \) - The number of source nodes accounted for by node \( i \) for a particular event. \( M_i \) is defined recursively as follows:

\[
M_i = \begin{cases} 
T_i, & \text{if } i \text{ is a leaf node} \\
T_i + \sum_{j : (j, i) \in E} M_j n_{ji}, & \text{otherwise} 
\end{cases} \quad (2)
\]

- \( D \) - The deadline by which packets must reach the sink.

Now the optimization problem can be framed as follows.

We call this problem \( Y \).

**Problem \( Y \):**

\[
\begin{align*}
\text{max} & \quad M_S = \sum_{j : (j, S) \in E} M_j n_{jS} \quad (3) \\
\text{s.t.} & \quad n_{ij} \in \{0, 1\} \quad \forall (i, j) \in E \\
& \quad \text{For each } i \in V \setminus V_L: \forall C \subseteq \{(j, i) : (j, i) \in E\}, \\
& \quad \sum_{j : (j, i) \in C} n_{ji} \leq W_i - \min_{j : (j, i) \in C} W_j \quad (4) \\
& \quad W_i \in \{0, 1, ..., D\} \quad \forall i \in V \setminus \{S\} \quad \text{and} \quad W_S = D \quad (5)
\end{align*}
\]

The goal of problem \( Y \) is to determine the control variables, \( n_{ij} \), for each link \( (i, j) \in E \), and \( W_i \), for each node \( i \in V \setminus \{S\} \), such that the revenue, \( M_S \), is maximized. Note that we have made two important assumptions while formulating the problem (which will be shown to not affect the optimal solution).

1) Our data aggregation policy does not allow any node to transmit more than once.
2) In our data aggregation policy, a node cannot accept packets once its waiting time expires. For example, consider a node \( i \) with waiting time \( W_i \). Then \( W_i \) serves as a deadline by which source nodes in the sub-tree rooted at node \( i \) should send their packets to \( i \).

We made these assumptions only in order to assist in formulating the optimization problem.

**Problem \( Z \):**

Consider an optimization problem, \( Z \), whose objective is to maximize the number of source nodes accounted for at the sink, within a deadline imposed by the sink, under the one-hop interference model. Let \( Z \) allow for multiple transmissions and also allow a node to accept packets irrespective of the current time.

**Proposition 3:** Any optimal solution to problem \( Y \) is also an optimal solution to problem \( Z \).

**Proof:** Suppose that in the optimal solution to problem \( Z \), a node \( Q \) makes \( k \) “useful” transmissions. By “useful”, we mean that the transmitted packet reaches the sink within the deadline \( D \). Specifically, let node \( Q \) transmit packet \( p_i \) at slot \( W_i \), \( 1 \leq i \leq k \) \( (W_1 < W_2 < \ldots < W_k) \).

Since the packets \( p_1, \ldots, p_k \) can be aggregated into a single packet, the same optimal solution could have been achieved if \( Q \) had simply aggregated these packets and transmitted the aggregated packet at slot \( W_k \) instead of making \( k \) separate transmissions. However, this is equivalent to \( Q \) having a deadline \( W_k \) and transmitting it exactly once.

Therefore, any optimal solution to \( Z \) can be transformed into a solution satisfying the constraints of \( Y \). Therefore, if \( M^*(Z) \) denotes the maximum revenue obtained by an optimal solution to problem \( Z \) and \( M(Y) \) denotes the revenue obtained by the transformed solution to problem \( Y \), then \( M^*(Z) = M(Y) \). Let \( M^*(Y) \) denote the maximum revenue obtained by an optimal solution to problem \( Y \). Then \( M^*(Z) \leq M^*(Y) \). We know that any optimal solution to problem \( Y \) is a feasible solution to problem \( Z \).

Thus, any optimal solution to problem \( Y \) is also an optimal solution to problem \( Z \).

Therefore, it is enough to solve Problem \( Y \) in order to obtain an optimal solution to problem \( Z \). We will now explain the constraints in problem \( Y \). Consider the constraint in (5). Equation (5) explains the relationship between interference and delay, in data aggregation trees. Under the one-hop interference model, a parent node can only receive packets from one of its children nodes during a particular slot. However, when a child node transmits to its parent, the other children (of the same parent) can receive data from their children (by the definition of the one-hop interference model). For example, consider Figure 1(a) with node \( P \) receiving data from its children \( C_1, C_2 \) and \( C_3 \). This figure represents a single hop in a large data aggregation tree. During a slot in which \( C_1 \) transmits to \( P \), \( C_2 \) and \( C_3 \) can receive data from their children. However, no two children of \( P \) can transmit to \( P \) in the same slot. Let node \( P \) have a waiting time \( W \), and \( C_1, C_2 \) and \( C_3 \) have waiting times \( W_1, W_2 \) and \( W_3 \), respectively. Let \( W_1 < W_2 < W_3 < W \). Then, the total number of transmissions that can be made from all the children nodes to the parent \( P \) is limited by the
difference between $W$ and $W_1$ (since the first transmission can occur only after $W_1$ and the last transmission can occur only before $W$, by the definition of waiting time). Also, the total number of transmissions that can be made from $C_2$ and $C_3$ to $P$ is limited by $W-W_2$. So, (5) says that for any subset of children nodes, the total number of transmissions made by this subset of nodes is bounded above by the difference between the waiting time of the parent and the waiting time of the child that has the least waiting time in the chosen subset.

IV. Solution

In this section, we provide an algorithm (Algorithm A) that solves problem $Z$ and prove that this algorithm provides an optimal solution to $Z$. Note that when $D \geq N$, the problem is trivial, since irrespective of the order of transmission, all source nodes will be accounted for at the sink, by the deadline (Proposition 1). So we will only consider the case when $D < N$.

Algorithm A:

1) Let $X[i, W]$ denote the maximum number of nodes that node $i$ can account for if its waiting time is $W$, $0 \leq W \leq D - 1$. For every leaf node $l$ and for each $W$, initialize $X[l, W] = T_l$.

2) Consider any node $j$ other than the sink and leaf nodes. Suppose it has $k$ children, $C_1, C_2, ..., C_k$. For every $W$, $0 \leq W \leq D - 1$, calculate $X[j, W]$ as follows.

a) If $W > k$, assign waiting times $W_1, W_2, ..., W_k$ to $C_1, C_2, ..., C_k$, respectively, such that each $W_i$, $1 \leq i \leq k$, takes a value in the set \{ $W-1, W-2, ..., W-k$ \} where no two nodes can have the same waiting time and such that the sum $\sum_{i=1}^{k} X[C_i, W_i]$ is maximized. This is a Maximum Weighted Matching(MWM) problem.

b) If $W < k$, assign waiting times from the set \{ $W-1, W-2, ..., 0$ \} to $W$ out of the $k$ children such that no two children that have been assigned a waiting time from this set have the same waiting time and such that the sum $\sum_{i\in C_i} X[C_i, W_i]$ is maximized. This is also a Maximum Weighted Matching(MWM) problem.

3) Finally, at the sink, calculate $X[S, D]$ as illustrated in Step 2.

We now prove that this algorithm provides an optimal solution to problem $Y$, and hence, to problem $Z$.

Lemma 4: Let the sink $S$ have $k$ children denoted by $C_1, ..., C_k$. Suppose that the optimal waiting times of these $k$ children are given by $W_1^*, ..., W_k^*$ and that the optimal number of transmissions are given by $n_{C_i, S}^*, ..., n_{C_k, S}^*$, respectively. $M_S$ is optimal with a deadline $D$ if and only if for each $i \in \{1, 2, ..., k\}$ with $n_{C_i, S}^* = 1$, $M_{C_i}$ is optimal with the deadline $W_i^*$.

Proof: Since $M_S = \sum_{i:n_{C_i, S}^*=1} M_{C_i}$, the result follows.

This Lemma shows that problem $Y$ exhibits an optimal substructure property. This suggests that a distributed solution may exist to the problem.

Lemma 5: For any node $i \in V$, the optimal $M_i$ cannot decrease as the waiting time of $i$ increases.

Proof: Clearly, if a node waits for a longer time to accumulate packets from its children, then it should be able to accumulate at least as many packets as it had accumulated when it had waited for a shorter period of time.

We now build the optimal solution from the leaves of the tree to the root. We will use induction to prove the optimality.

Lemma 6: Consider a node $P$ whose children are all leaf nodes. Let there be $k$ children, $L_1, ..., L_k$. If $P$ has a waiting time $W$, then at most $\min(W, k)$ children can transmit among the $k$ children within the deadline $W$ at $P$. The algorithm that allocates waiting times, $W-1, ..., W-\min(W, k)$, in the order of decreasing $T_{L_j}$ for $j = 1, 2, ..., k$, will maximize $M_P$.

Proof: The proof is obvious since leaf nodes will only account for one packet if they are sources and no packets
otherwise. Since packets are available at the leaf nodes at time zero, the result follows.

Thus, we can find the optimal solution for any deadline \( W \) at each parent having only leaf nodes as children. Now, assume that we can find the optimal solution for any deadline \( W \) at each parent node that is \( h \) hops away from the sink.

**Lemma 7**: Consider a node \( P \) that is \( h - 1 \) hops from the sink having \( k \) children \( C_1, \ldots, C_k \). Let \( P \) have a waiting time \( W \). Then, at least one of the solutions for the optimal waiting times of \( C_1, \ldots, C_k \) will satisfy the following conditions.

- If \( W \geq k \), then \( n_{C_i,P} = 1 \) \( \forall i = 1, 2, \ldots, k \), and each child \( C_1, \ldots, C_k \) transmits in one of the slots in the set \( \{W-1, \ldots, W-k\} \) where no two nodes transmit in the same slot.
- If \( W < k \), then \( n_{C_i,P} = 1 \) for exactly \( W \) children among the \( k \) children. These \( W \) children transmit in one of the slots in the set \( \{0, 1, \ldots, W-1\} \) and no two nodes transmit in the same slot.

**Proof**: Before we prove this lemma, note that \( n_{ij} = 1 \) does not imply that node \( i \) has a packet to send to node \( j \). It only means that if node \( i \) has a packet to send to node \( j \), then it is allowed to send it to \( j \) during the time slot allocated to it for transmission.

**Case 1**: \( W \geq k \)

Suppose \( n_{C_i,P} \neq 1 \) for some node \( C_i \). Then there is at least one slot during which there is no transmission. By scheduling node \( C_i \) to transmit in this slot, we can obtain a larger \( M_P \) than when \( C_i \) was not transmitting.

Suppose some node \( C_i \) does not transmit in one of the slots in the set \( \{W-1, \ldots, W-k\} \). Suppose \( C_i \) transmits in the slot \( W-k-\alpha, \alpha > 0 \). Then there exists at least one slot in the set \( \{W-1, \ldots, W-k\} \) during which no transmission takes place. If the node \( C_i \) had transmitted during this slot, the value of \( M_P \) that we would have obtained would be at least as much as what would have been obtained if it had transmitted during the slot \( W-k-\alpha \) (by Lemma 5).

**Case 2**: \( 0 \leq W < k \)

If \( n_{C_i,P} = 1 \) for more than \( W \) nodes, then the solution is infeasible since it is not possible to schedule more than \( W \) nodes in \( W \) slots. In particular, the constraint in (5) is violated.

On the other hand, if \( n_{C_i,P} = 1 \) for less than \( W \) nodes, then by a similar argument as made in Case 1, we can obtain a larger \( M_P \) by making \( W \) nodes transmit.

Similarly, the rest of the proof follows from the same argument as made in Case 1.

**Theorem 8**: Algorithm \( A \) results in an optimal solution to problem \( Z \).

**Proof**: We have found the set of optimal waiting times for the children \( C_1, \ldots, C_k \). We now only need to assign these slots to the \( k \) children such that \( M_P \) is maximized.

Suppose \( X[i,W] \) represents the optimal number of nodes that \( C_i \) can account for if its waiting time is \( W \), \( 0 \leq W \leq D - 1 \). Then the problem of finding the optimal \( M_P \) is a Maximum Weighted Matching (MWM) problem in a bipartite graph with node-exclusive interference. Consider the bipartite graph in Figure 2. The nodes at the top of the graph represent the children nodes and the nodes at the bottom represent waiting times. If a child node \( C_i \) (say) has a waiting time of \( W - j \), \( 1 \leq j \leq \min(W,k) \), the edge connecting \( C_i \) and \( W - j \) has a weight \( X[C_i,W-j] \). No two nodes can have the same waiting time and a particular node can be allotted at most one waiting time. The goal is to assign waiting times to the children nodes such that the sum of the weights on the edges assigned is maximized. This is clearly a Maximum Weighted Matching (MWM) problem and can be solved in polynomial time for a one-hop interference model.

![Fig. 2. Maximum Weighted Matching Solution](image)

Hence we have now solved the optimization problem for nodes that are \( h-1 \) hops from the sink. By induction, from Lemmas 6 and 7, given the waiting time of the parent node at any hop, the optimal waiting times and the optimal number of transmissions of the children nodes in that hop can be determined using algorithm \( A \). Hence, at the sink, \( X[S,D] \) can be determined. Thus, algorithm \( A \) results in an optimal solution to problem \( Y \).

**V. Computational Complexity of Algorithm \( A \)**

In this section, we analyze the computational complexity of algorithm \( A \) described in Section 4. Let the farthest node (in terms of number of hops) be \( h \) hops away from the sink. Let the in-degree of each node in the tree (apart from leaf nodes) be \( k \). By this, we mean that each non-leaf node has \( k \) children. Let \( D \) represent the deadline imposed by the sink. Let the total number of sensor nodes (not including the sink) be \( N \).

**Proposition 9**: The time complexity of algorithm \( A \) is \( O(hk^2(D+k)\log k) \).

**Proof**: At every node \( i \), we need to calculate \( X[i,W], 0 \leq W \leq D - 1 \).

\[
\begin{align*}
W = 0 & \Rightarrow \text{Time} \leq 0 \\
W = 1 & \Rightarrow \text{Time} = k \\
W = 2 & \Rightarrow \text{Time} = (k+1)^2 \\
3 \leq W \leq k & \Rightarrow \text{Time} = (k+W)^2 \log(k+W) \\
k \leq W \leq D - 1 & \Rightarrow \text{Time} = (2k)^2 \log(2k) \quad (8)
\end{align*}
\]

We obtain (8) from the fact that the MWM problem in a bipartite graph can be solved in \( O(V^2 \log V + VE) \) time,
where $V$ is the number of vertices and $E$ is the number of edges. When $W \geq k$, the vertices and edges remain the same but the edge weights change.

Therefore the total time required at node $i$ is given by $k + \sum_{W=3}^{k-1} (k+W)^2 \log(k+W) + (D-k)(2k)^2 \log(2k)$. Since for $3 \leq W \leq k$, $\log(k+W) \leq \log(2k)$, the time required can be expressed as

$k^2 + \sum_{W=3}^{k-1} (k+W)^2 O(\log k) + 4k^2(D-k)O(\log k)$

$= k^2 + O(\log k)(4k^2(D-k) + \sum_{W=3}^{k-1} (k+W)^2)$

$= k^2 + O(k^2D\log k) + O(k^3\log k)$

$= O(k^2(D+k)\log k)$

Since nodes that are equal number of hops away from the sink can perform this computation in parallel and since we have $h$ hops, the complexity of algorithm $A$ is $O(hk^2(D+k)\log k)$.

VI. EXTENSIONS AND APPLICATIONS

In this section, we discuss a number of different interpretations of problem $Y$. We also discuss applications to dynamic events such as target tracking.

The optimization framework that we have described in Section 3 is very general and can be interpreted in a number of ways. A few such interpretations are listed below.

1) Priority: Suppose that each source node has a priority associated with the packet that it generates. Then, instead of maximizing the number of source nodes accounted for at the sink, we can maximize the total priority of packets accounted for at the sink within a deadline.

2) Observation errors: The data observed by sensor nodes may not be accurate. Suppose we associate a certain “confidence index” to each node’s observation, we can then maximize the data accuracy at the sink by maximizing the total confidence index.

3) Energy constraint: Suppose we have an additional constraint that the number of time slots during which a node $i$ can transmit/receive is limited to $r_i$ (say). Nodes go to sleep during other slots. Then, since each node transmits at most once, we can modify the constraint in (5) to $\sum_{j,(j,i) \in E} n_{ji} \leq \min(r_i-1,W_i - \min_{j,(j,i) \in E} W_j)$ for all $i \in V_n \cup V_L$. This problem can also be solved using algorithm $A$.

While our solution can be applied to static events, there are a number of delay-sensitive events that are dynamic events. Examples include tracking a target, reporting a fire breakout, etc. Since these events may occur in random parts of the sensor network, the source nodes for these events may not be known until the event occurs. Also, the event may spread through the network over time. So, different nodes will be sources at different instants. A dynamic event can be viewed as a sequence of events. Assuming that the rate at which the event spreads through the network is much smaller than the rate at which the sensor nodes communicate their data to the sink, we can use our optimization framework in this case as well. The challenge here, is to develop an algorithm that generates a delay efficient data aggregation tree quickly. Having multiple sinks will also help since sensors can then report to the closest sink.

VII. CONCLUSION

In this paper, we have developed a general optimization framework for solving the problem of maximizing revenue in data aggregation trees when a deadline is imposed by the sink. We considered a one-hop interference model and proposed a polynomial time algorithm that uses only local information at each hop to obtain the optimal solution. The problem can be extended to a general interference model. However, the complexity of scheduling transmissions may become high depending on the interference model used. Finally, we discussed a number of interesting applications and interpretations to our solution, such as, incorporating sleep-wake scheduling, maximizing weighted revenue and maximizing accuracy.

REFERENCES


