Efficient Data Collection in Interference-Aware Wireless Sensor Networks

Min Kyung An and Hyuk Cho

Abstract—In this paper, we study the Minimum Latency Collection Scheduling (MLCS) problem in Wireless Sensor Networks (WSNs) adopting the two interference models: the graph model and the more realistic physical interference model known as Signal-to-Interference-Noise-Ratio (SINR). The main issue of the MLCS problem is to compute the minimum latency schedule, that is, to compute a schedule with the minimum number of timeslots, such that all data from all nodes can be collected to a sink node without any collision or interference. First, we describe an approximation algorithm with $O(1)$-approximation ratio that works in both the interference models by yielding a schedule whose latency is bounded by $O(n)$, where $n$ is the number of nodes in the network. Then, we validate the latency performance of the proposed algorithm with various simulated networks, demonstrate the robustness of the algorithms with different network parameters, and discuss the effect of the interference models’ parameter values on the latency.

Index Terms—data collection, gathering, approximation algorithm, collision-free, interference-free, wireless sensor network

I. INTRODUCTION

Wireless Sensor Networks (WSNs) consist of a number of sensor nodes deployed in a plane. Each node monitors nearby environmental conditions periodically, and the gathered data from all nodes is collected to a designated destination called a sink node. This type of application is commonly known as data collection (or data gathering) in the literature, and it is one of the most crucial applications of WSNs. As the small-sized sensors collect and send data periodically using their limited energy, researchers have focused on the issue of prolonging the network lifetime by reducing energy consumption, which can be achieved by avoiding the unnecessary retransmission of the gathered data. An interesting approach is to assign timeslots to sensor nodes to obtain a good schedule by which all data can be collected without any collision or interference with other nodes on their way to the sink node. Since the data collection occurs periodically, reducing the latency of the schedule, that is, constructing a schedule with a minimum number of timeslots, has been a fundamental issue.

The data collection problem has been widely investigated by researchers in the two interference models: the graph model and the physical interference model. Furthermore, the graph model has two models: collision-free graph model and collision-interference-free graph model. While the collision-free model concerns collision only, the collision-interference-free concerns both collision and interference. In the graph model, given a transmission range $r(u)$ (i.e., the radius of the broadcasting disk covered by the signal sent by node $u$ using its transmission power $p(u)$) for every node $u$, the interference range of $u$ is defined as $\rho \cdot r(u)$, where $\rho \geq 1$ is the interference factor. Note that when $\rho = 1$, it is called a collision-free graph model, and when $\rho \geq 1$, it is called a collision-interference-free graph model.

In the graph model, assuming all nodes have the maximum transmission range equal to one unit (i.e., $r(u) = 1$ for every $u$), Florens et al. [1] proposed 3-approximation algorithm with $\rho = 1$ for tree networks. Later, Bermond et al. [2] proved the NP-hardness, and proposed a 4-approximation algorithm with any value of $\rho \geq 1$. Bonifaci et al. [3] also proposed a 4-approximation algorithm with $\rho > 1$, and 3-approximation algorithm with $\rho = 1$. Then, Coleri et al. [4] proved its NP-completeness and introduced two heuristic algorithms with $\rho = 1$. Recently, Bermond et al. [5] studied the problem in grid networks, and proposed a 3-approximation algorithm with $\rho = 1$. Their algorithm also yields an approximation ratio of 4 when $\rho = 2$. Bermond et al. [6] investigated the problem in tree networks with $\rho \geq 2$ and proposed a closed formula for the data collection of the optimal schedule. It also considered the problem of determining the computational complexity of data collection in general graphs and showed that the problem is NP-complete, and designed a $(1 + \frac{\rho}{2})$-approximation algorithm. Then, Bermond et al. [7] addressed the problem on the specific case where the network is a path with the sink at an end vertex of the path (i.e., linear topology), and proposed an algorithm which is optimal when $r(u) = 2, 3$, and 5. Kowalski et al. [8] studied the problem in linear topologies, and proposed a 2-approximation algorithm with $\rho = 1$. Note that the results in [1]–[8], however, apply to special topologies or general graphs only where the distance between two nodes on a given communication graph is defined as the length (i.e., the number of edges) of a shortest path connecting them.

Although the graph model has been used in many studies, it is not an adequate realistic model since cumulative interference caused by all other concurrently...
transmitting nodes is ignored. Therefore, researchers have
started investigating problems of WSNs with the more
realistic physical interference model, also known as the
Signal-to-Interference-Noise-Ratio (SINR) model, since
its introduction by Gupta et al. [9]. In the SINR model,
Wan et al. [10] studied the data collection problem and
proposed a O(1)-approximation algorithm.

In this paper, we continue the study of the data collec-
tion problem in both the graph and SINR models. We
first introduce an approximation algorithm with O(1)-
approximation ratio in the graph model with ρ ≥ 1, and
then use the result in [11] to show that the proposed
algorithm works in the SINR model as well.

The rest of this paper is organized as follows. In
Section II, we describe our network models, introduce the
definitions used in this paper, and then define the Mini-
um Latency Collection Scheduling (MLCS) problem. In
Section III, we propose a constant factor approximation
algorithm for the MLCS problem that works in both graph
and SINR models. In Section IV, we analyze the proposed
algorithm, and in Section V we evaluate the latency
performance of the algorithm with simulated networks.
We conclude with some remarks in Section VI.

II. PRELIMINARIES

A. Network Models

In this paper, we consider that Wireless Sensor Network
(WSNs) consist of a set V of sensor nodes deployed in
a plane. Each node u ∈ V is assigned a transmission
power level p(u), and the transmission range r(u) of u is
defined as the radius of the broadcasting disk covered by
the signal sent by u using p(u). Accordingly, a directed
edge (u, v) exists from node u to node v, if v resides in
r(u), i.e., d(u, v) ≤ r(u), where d(u, v) denotes the
Euclidean distance between u and v.

1) Graph Model: In the graph model, let R_p(u) =
\{v | v ∈ V, d(u, v) ≤ r(u)\} denote the set of nodes that
reside in the transmission range r(u) of u with p(u).
Then, two nodes u and v can communicate each other
if u ∈ R_p(u) and v ∈ R_p(u). In addition, we consider the
collision (or conflict) and interference as follows. Given
a transmission power level p(u) of u, let I_p(u) = \{v | v ∈
V, d(u, v) ≤ ρ · r(u)\} denote the set of nodes that reside
in the interference range ρ · r(u) of u, where ρ ≥ 1 is the
interference factor. Then, the collision is said to occur at
node w if there exist other concurrently sending nodes u
and v such that w ∈ R_p(u) ∩ I_p(u), where ρ = 1 (i.e.,
I_p(u) = R_p(u)). On the other hand, interference is said
to occur at w if there exist other concurrently sending nodes u
and v such that w ∈ R_p(u) ∩ I_p(u), where ρ > 1. In the
literature, the graph model concerning only collision
(i.e., when ρ = 1) is called the collision-free graph model,
whereas the graph model concerning both collision and
interference (i.e., when ρ ≥ 1) is called the collision-
interference-free graph model.

In the graph model, the communication graph can be
modeled as a directional disk graph G = (V, E), where

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>The path loss exponent (α ≥ 2)</td>
</tr>
<tr>
<td>β</td>
<td>The SINR threshold (β ≥ 1)</td>
</tr>
<tr>
<td>C</td>
<td>The candidate set: A set of nodes having their own messages to send messages to send</td>
</tr>
<tr>
<td>d(u, v)</td>
<td>The Euclidean distance between nodes u and v</td>
</tr>
<tr>
<td>E</td>
<td>A set of edges</td>
</tr>
<tr>
<td>F</td>
<td>The forwarding set: A set of nodes having messages to relay</td>
</tr>
<tr>
<td>G</td>
<td>A directed graph (Section II)</td>
</tr>
<tr>
<td>G_I</td>
<td>An undirected graph (Section III and IV)</td>
</tr>
<tr>
<td>r(u)</td>
<td>The transmission range of node u</td>
</tr>
<tr>
<td>T</td>
<td>The number of messages which node u has received from other nodes which u has to forward (i.e., relay)</td>
</tr>
<tr>
<td>p(u)</td>
<td>Uniform transmission power level</td>
</tr>
<tr>
<td>parent(t(u))</td>
<td>A parent node of u on T</td>
</tr>
<tr>
<td>I_p(u)</td>
<td>The set of nodes in u’s interference range</td>
</tr>
<tr>
<td>ℓ(u)</td>
<td>The level of u on T</td>
</tr>
<tr>
<td>I(u)</td>
<td>The set of nodes at level ℓ(u)</td>
</tr>
<tr>
<td>N</td>
<td>The number of nodes having messages which u has received from other nodes which u has to forward (i.e., relay)</td>
</tr>
<tr>
<td>F</td>
<td>The forwarding set: A set of nodes having messages to relay</td>
</tr>
<tr>
<td>G</td>
<td>A directed graph (Section II)</td>
</tr>
<tr>
<td>G_I</td>
<td>An undirected graph (Section III and IV)</td>
</tr>
<tr>
<td>r(u)</td>
<td>The transmission range of node u</td>
</tr>
<tr>
<td>p(u)</td>
<td>Uniform transmission power level</td>
</tr>
<tr>
<td>parent(t(u))</td>
<td>A parent node of u on T</td>
</tr>
<tr>
<td>I_p(u)</td>
<td>The set of nodes in u’s interference range</td>
</tr>
<tr>
<td>ℓ(u)</td>
<td>The level of u on T</td>
</tr>
<tr>
<td>I(u)</td>
<td>The set of nodes at level ℓ(u)</td>
</tr>
<tr>
<td>N</td>
<td>The number of nodes having messages which u has received from other nodes which u has to forward (i.e., relay)</td>
</tr>
<tr>
<td>F</td>
<td>The forwarding set: A set of nodes having messages to relay</td>
</tr>
<tr>
<td>G</td>
<td>A directed graph (Section II)</td>
</tr>
<tr>
<td>G_I</td>
<td>An undirected graph (Section III and IV)</td>
</tr>
<tr>
<td>r(u)</td>
<td>The transmission range of node u</td>
</tr>
<tr>
<td>p(u)</td>
<td>Uniform transmission power level</td>
</tr>
<tr>
<td>parent(t(u))</td>
<td>A parent node of u on T</td>
</tr>
<tr>
<td>I_p(u)</td>
<td>The set of nodes in u’s interference range</td>
</tr>
<tr>
<td>ℓ(u)</td>
<td>The level of u on T</td>
</tr>
<tr>
<td>I(u)</td>
<td>The set of nodes at level ℓ(u)</td>
</tr>
<tr>
<td>N</td>
<td>The number of nodes having messages which u has received from other nodes which u has to forward (i.e., relay)</td>
</tr>
<tr>
<td>F</td>
<td>The forwarding set: A set of nodes having messages to relay</td>
</tr>
<tr>
<td>G</td>
<td>A directed graph (Section II)</td>
</tr>
<tr>
<td>G_I</td>
<td>An undirected graph (Section III and IV)</td>
</tr>
<tr>
<td>r(u)</td>
<td>The transmission range of node u</td>
</tr>
<tr>
<td>p(u)</td>
<td>Uniform transmission power level</td>
</tr>
<tr>
<td>parent(t(u))</td>
<td>A parent node of u on T</td>
</tr>
<tr>
<td>I_p(u)</td>
<td>The set of nodes in u’s interference range</td>
</tr>
<tr>
<td>ℓ(u)</td>
<td>The level of u on T</td>
</tr>
<tr>
<td>I(u)</td>
<td>The set of nodes at level ℓ(u)</td>
</tr>
<tr>
<td>N</td>
<td>The number of nodes having messages which u has received from other nodes which u has to forward (i.e., relay)</td>
</tr>
<tr>
<td>F</td>
<td>The forwarding set: A set of nodes having messages to relay</td>
</tr>
<tr>
<td>G</td>
<td>A directed graph (Section II)</td>
</tr>
<tr>
<td>G_I</td>
<td>An undirected graph (Section III and IV)</td>
</tr>
<tr>
<td>r(u)</td>
<td>The transmission range of node u</td>
</tr>
<tr>
<td>p(u)</td>
<td>Uniform transmission power level</td>
</tr>
<tr>
<td>parent(t(u))</td>
<td>A parent node of u on T</td>
</tr>
<tr>
<td>I_p(u)</td>
<td>The set of nodes in u’s interference range</td>
</tr>
<tr>
<td>ℓ(u)</td>
<td>The level of u on T</td>
</tr>
<tr>
<td>I(u)</td>
<td>The set of nodes at level ℓ(u)</td>
</tr>
<tr>
<td>N</td>
<td>The number of nodes having messages which u has received from other nodes which u has to forward (i.e., relay)</td>
</tr>
<tr>
<td>F</td>
<td>The forwarding set: A set of nodes having messages to relay</td>
</tr>
<tr>
<td>G</td>
<td>A directed graph (Section II)</td>
</tr>
<tr>
<td>G_I</td>
<td>An undirected graph (Section III and IV)</td>
</tr>
<tr>
<td>r(u)</td>
<td>The transmission range of node u</td>
</tr>
<tr>
<td>p(u)</td>
<td>Uniform transmission power level</td>
</tr>
<tr>
<td>parent(t(u))</td>
<td>A parent node of u on T</td>
</tr>
<tr>
<td>I_p(u)</td>
<td>The set of nodes in u’s interference range</td>
</tr>
<tr>
<td>ℓ(u)</td>
<td>The level of u on T</td>
</tr>
<tr>
<td>I(u)</td>
<td>The set of nodes at level ℓ(u)</td>
</tr>
<tr>
<td>N</td>
<td>The number of nodes having messages which u has received from other nodes which u has to forward (i.e., relay)</td>
</tr>
<tr>
<td>F</td>
<td>The forwarding set: A set of nodes having messages to relay</td>
</tr>
<tr>
<td>G</td>
<td>A directed graph (Section II)</td>
</tr>
<tr>
<td>G_I</td>
<td>An undirected graph (Section III and IV)</td>
</tr>
<tr>
<td>r(u)</td>
<td>The transmission range of node u</td>
</tr>
<tr>
<td>p(u)</td>
<td>Uniform transmission power level</td>
</tr>
<tr>
<td>parent(t(u))</td>
<td>A parent node of u on T</td>
</tr>
<tr>
<td>I_p(u)</td>
<td>The set of nodes in u’s interference range</td>
</tr>
<tr>
<td>ℓ(u)</td>
<td>The level of u on T</td>
</tr>
<tr>
<td>I(u)</td>
<td>The set of nodes at level ℓ(u)</td>
</tr>
<tr>
<td>N</td>
<td>The number of nodes having messages which u has received from other nodes which u has to forward (i.e., relay)</td>
</tr>
<tr>
<td>F</td>
<td>The forwarding set: A set of nodes having messages to relay</td>
</tr>
<tr>
<td>G</td>
<td>A directed graph (Section II)</td>
</tr>
<tr>
<td>G_I</td>
<td>An undirected graph (Section III and IV)</td>
</tr>
<tr>
<td>r(u)</td>
<td>The transmission range of node u</td>
</tr>
<tr>
<td>p(u)</td>
<td>Uniform transmission power level</td>
</tr>
<tr>
<td>parent(t(u))</td>
<td>A parent node of u on T</td>
</tr>
<tr>
<td>I_p(u)</td>
<td>The set of nodes in u’s interference range</td>
</tr>
<tr>
<td>ℓ(u)</td>
<td>The level of u on T</td>
</tr>
<tr>
<td>I(u)</td>
<td>The set of nodes at level ℓ(u)</td>
</tr>
<tr>
<td>N</td>
<td>The number of nodes having messages which u has received from other nodes which u has to forward (i.e., relay)</td>
</tr>
<tr>
<td>F</td>
<td>The forwarding set: A set of nodes having messages to relay</td>
</tr>
</tbody>
</table>

Therefore, the set of nodes in G is defined as

E = \{(u, v) | u, v ∈ V, d(u, v) ≤ r(u) and d(v, u) ≤ r(v)\}.

2) SINR Model: In the SINR model, if a node u
transmits with its power level p(u), then the received
power at a receiver v is p(u) · d(u, v)^{-α}, where α > 2
is the path loss exponent. In order that the receiver v can
receive the data transmitted by the sender u, the ratio
of the received power at the receiver v to the interference
caused by all the other concurrently transmitting nodes
and background noise must be beyond an SINR threshold
β ≥ 1. Formally, node v can successfully receive data via
the communication edge (u, v) only if

SINR(u,v) = \frac{p(u)}{N + \sum_{w ∈ X(u,v)} \frac{p(w)}{d(w,v)^\alpha}} \geq \beta ≥ 1

(1)

where N > 0 is the background noise, and X is the set of
other concurrently transmitting nodes.

As u can send its data to the nodes within the dis-

© 2015 ACADEMY PUBLISHER
tance \( \left( \frac{P(u)}{N^2 \rho} \right)^2 \) (i.e., \( r(u) = \left( \frac{P(u)}{N^2 \rho} \right)^2 \)), the communication graph can be modeled as a directional disk graph \( G = (V, E) \), where \( E = \{(u,v) | d(u,v) \leq \left( \frac{P(u)}{N^2 \rho} \right)^2 \} \) and \( d(v,u) \leq \left( \frac{P(v)}{N^2 \rho} \right)^2 \).

B. Problem Definition

We define the Minimum Latency Collection Scheduling (MLCS) problem as follows. Given a set of nodes for a network in a plane, we assign every node timeslots such that nodes assigned a same timeslot can send data to their receivers simultaneously without any collision or interference. A schedule is defined as a sequence of such timeslots. Formally, at each timeslot \( t \), we have an assignment set \( \pi_t = \{(s_{t_1}, r_{t_1}), \ldots, (s_{t_m}, r_{t_m})\} \), where \((s_{t_i}, r_{t_i})\), \(1 \leq i \leq m\), is an edge in the communication graph \( G \), and \( m \) denotes the number of nodes scheduled at timeslot \( t \). Therefore, in each \( \pi_t \), every \( s_{t_i} \) is assigned the same timeslot \( t \) to simultaneously send data with its power level \( P(s_{t_i}) \) to a receiver \( r_{t_i} \), and the following condition is satisfied:

- (Graph Model) Neither collision nor interference occurs at any receiver \( r \).
- (SNIR Model) The SNIR Inequality (1) is satisfied for every receiver \( r \).

A schedule \( \Pi = (\pi_1, \pi_2, \ldots, \pi_M) \) has its latency \( M \), which is the length of the schedule. \( \Pi \) is successful if all data of every node \( v \in V \) is collected to a sink node \( s \in V \).

The MLCS problem is formally defined as follows:

**Input:** A set \( V \) of nodes for a network, transmission power level \( P(u) \) for every node \( u \in V \), and a sink node \( s \in V \)

**Output:** A successful minimum latency schedule \( \Pi \)

III. CONSTANT FACTOR APPROXIMATION ALGORITHM

In this section, we describe a constant factor approximation algorithm called the Hop-Based Collection Scheduling (HBCS) algorithm for the MLCS problem that can be applicable to both the graph model and the SNIR model. We assume the uniform power model where all nodes are initially assigned a uniform power level \( P \), i.e., \( P(v) = P \), for every \( v \in V \).

A. Graph Model

For the graph model, we set the maximum link length (i.e., the maximum transmission range \( r \)) to be the given \( P \), and assume that the communication graph, modeled as a undirected unit disk graph \( G = (V, E) \) whose \( E = \{(u,v) | d(u,v) \leq r \} \), is connected and its interference factor \( \rho \geq 1 \).

The HBCS algorithm (Algorithm 1) starts by constructing an interference graph \( G_I = (V, E_I) \), where \( E_I = \{(u,v) | d(u,v) \leq \rho \cdot r \} \). Then it assigns timeslots to nodes based on a collection tree \( T \), which is a breadth-first-search (BFS) tree (See [12]) on \( G \) rooted at the sink node \( s \), as well as an interference tree \( T_I \), which is a BFS

---

**Algorithm 1** Hop-Based Collection Scheduling (HBCS)

**Input:** \( V \) (whose every \( u \) is assigned \( P \)) including \( s \)

**Output:** Schedule \( \Pi \)

1: Construct an interference graph \( G_I = (V, E_I) \) where \( E_I = \{(u,v) | d(u,v) \leq \rho \cdot r \} \).
2: Construct a collection tree \( T \) of \( G \), and an interference tree \( T_I \) of \( G_I \).
3: for each \( v \in V \setminus \{s\} \) do
4: \( \ell(v) \leftarrow \) level of \( v \) on \( T \)
5: \( \ell_I(v) \leftarrow \) level of \( v \) on \( T_I \)
6: \( n_m(v) \leftarrow 1 \)
7: \( n_f(v) \leftarrow 0 \)
8: \( L_i \leftarrow L_i \cup \{v\} \) where \( i = \ell_I(v) \)
9: end for
10: for each \( z \in L_i, 1 \leq i \leq h_I \) do
11: Construct subtree(z).
12: for each descendant \( w \) of \( z \) on subtree(z) do
13: \( \text{subroot}(w) \leftarrow z \)
14: end for
15: end for
16: \( t \leftarrow 1 \)
17: repeat
18: for \( j = 3 \) downto 1 do
19: \( L \leftarrow \{L_i | i \% 3 = j \% 3, 1 \leq i \leq h_I \} \)
20: \( t_{start} \leftarrow t \)
21: \( t_{max} \leftarrow t \)
22: for each non-empty \( L_i \in L \) do
23: \( F \leftarrow \{w | n_f(w) \neq 0, w \in L_i \} \)
24: if \( F = \emptyset \) then
25: \( C \leftarrow \{w | n_m(w) \neq 0, w \in L_i \} \)
26: if \( C \neq \emptyset \) then
27: \( t \leftarrow t_{start} \)
28: Pick \( w \in C \) whose \( \ell(w) \) is largest.
29: \( \pi_i \leftarrow \pi_i \cup \{w, \text{parent}(w)\} \)
30: \( n_m(w) \leftarrow 0 \)
31: \( n_f(\text{parent}(w)) \leftarrow n_f(\text{parent}(w)) + 1 \)
32: \( t \leftarrow t + 1 \)
33: end if
34: else if \( F \neq \emptyset \) then
35: \( t \leftarrow t_{start} \)
36: Pick a node \( w \in F \) whose \( \ell(w) \) is largest.
37: \( \pi_i \leftarrow \pi_i \cup \{w, \text{parent}(w)\} \)
38: \( n_f(w) \leftarrow n_f(w) + 1 \)
39: \( n_f(\text{parent}(w)) \leftarrow n_f(\text{parent}(w)) + 1 \)
40: \( t \leftarrow t + 1 \)
41: end if
42: end if
43: if \( w \in L_i \) and \( \text{parent}(w) \in L_i \) then
44: \( t \leftarrow \text{Subtree-Scheduling}(w, \text{parent}(w), t) \)
45: end if
46: if \( t_{max} < t \) then \( t_{max} \leftarrow t \) end if
47: end for
48: end for
49: until \( n_m(v) = 0 \) for every \( v \in V \setminus \{s\} \)
50: \( M \leftarrow t - 1 \)
51: return \( \Pi = (\pi_1, \pi_2, \ldots, \pi_M) \)
Algorithm 2 Subtree-Scheduling

Input: Node \( w \), node \( z \), timeslot \( t \)

Output: Timeslot \( t \)

1: while \( z \neq \text{subroot}(w) \) and \( z \neq s \) do
2: \( \pi_t \leftarrow \pi_t \cup \{(z, \text{parent}(z))\} \)
3: \( n_f(z) \leftarrow n_f(z) + 1 \)
4: \( n_f(\text{parent}(z)) \leftarrow n_f(\text{parent}(z)) + 1 \)
5: \( t \leftarrow t + 1 \)
6: \( z \leftarrow \text{parent}(z) \)
7: end while

The third submain iteration examines \( \mathcal{L} = \{L_1, L_4, L_7, \cdots \} \).

Each main iteration repeats the aforementioned three submain iteration steps until all the messages from all nodes are received at the sink \( s \), i.e., \( n_m(v) = 0 \) for every \( v \in V \setminus \{s\} \) (Steps 17-49 in Algorithm 1).

Lastly, the algorithm returns the schedule \( \Pi \) whose latency is \( M \) (Step 51 in Algorithm 1).

In what follows, we explain the details of the timeslot assignments at each of the submain iterations (Steps 19-47 in Algorithm 1): For each non-empty \( L_i \in \mathcal{L} \), the algorithm first computes a forwarding set \( F \) of nodes which have messages to forward (i.e., to relay) (Steps 22-23 in Algorithm 1).

1) If \( F = \emptyset \) (i.e., all nodes in \( L_i \) do not have messages to forward), then it computes a candidate set \( C \) of nodes which do have their own messages to transmit (Steps 24-25 in Algorithm 1). After then, if \( C \neq \emptyset \) (Step 26 in Algorithm 1), a node \( w \in C \) whose \( \ell(w) \) is largest is first selected and then scheduled to transmit its message to \( \text{parent}(w) \) at timeslot \( t \), and its \( n_m(w) \) is set to be 0. Also, \( \text{parent}(w) \)'s \( n_f(\text{parent}(w)) \) and the timeslot \( t \) are incremented (Steps 28-32 in Algorithm 1).

Else if \( F \neq \emptyset \) (i.e., there exist some nodes in \( L_i \) which have received messages to forward (Step 34 in Algorithm 1)), then a node \( w \) whose \( \ell(w) \) is largest is first selected and then scheduled to forward the message to \( \text{parent}(w) \) at timeslot \( t \), and its \( n_f(w) \) is decremented (Steps 36-38 in Algorithm 1). (Notice that in this case, \( w \) has been forwarded a message from its descendant in an earlier iteration, and its \( n_f(w) \) has been incremented at that time (Step 4 in Algorithm 2). Also, \( \text{parent}(w) \)'s \( n_f(\text{parent}(w)) \) and the timeslot \( t \) are incremented (Steps 39-40 in Algorithm 1).

2) After then, during the Subtree-Scheduling (Algorithm 2) subroutine, \( \text{parent}(w) \) forwards the message to \( \text{parent}(\text{parent}(w)) \) at the timeslot \( t + 1 \) that has been incremented at Step 32 in Algorithm 1 (or at Step 40 Algorithm 1). The subroutine repeats until the \( \text{subroot}(w) \) finally receives the message from \( w \). During the subroutine, whenever a node \( z \) forwards the message to \( \text{parent}(z) \) at a timeslot \( t \), \( n_f(z) \) is decremented, and \( n_f(\text{parent}(z)) \) and \( t \) are incremented (Steps 2-5 in Algorithm 2).

For example, in Figure 5, the message of \( w_{12} \in L_3 \) is delivered to \( z_1 \) based on \( \text{subtree}(z_1) \). If the starting timeslot is \( t \) for current iteration (Step 22 in Algorithm 1), \( w_{12} \) is scheduled to transmit its own data (or rely the forwarded data) to \( w_{11} \) at timeslot \( t + 1 \) at Step 29 (or at Step 31) in Algorithm 1, and \( w_{11} \) is scheduled to forward the message to \( z_1 \) at timeslot \( t + 1 \) at Step 2 in Algorithm 2. In parallel, nodes in \( L_6 \) are scheduled similarly as follows. In \( L_6 \), \( w_{23} \)'s message is delivered to \( z_2 \) based on \( \text{subtree}(z_2) \). The starting timeslot is identical to the one used for \( L_3 \), i.e., \( t \). So, \( w_{23} \) is scheduled to transmit its own data (or rely the forwarded data) to \( w_{22} \) at timeslot...
Figure 1: When $\rho = 1$

Figure 2: When $\rho = 2$

Figure 3: When $\rho = 3$

Figure 4: When $\rho = 4, 5$

$t$ in Step 29 (or at Step 31) in Algorithm 1, and $w_{22}$ is scheduled to forward the message to $z_2$ at timeslot $t + 1$ at Step 2 in Algorithm 2. Like this, as we want to use the same starting timeslot $t$ for each node selected first from each level set, $t_{\text{start}}$ is used (Steps 20, 27 and 35 in Algorithm 1) to store the starting timeslot for the level sets examined during the iteration. Also, after every $L_i \in \mathcal{L}$ is examined at this iteration, $t$ is set with the largest timeslot
used for those level sets (Steps 21, 45 and 47 in Algorithm 1) for the next iteration.

B. SINR Model

From the SINR Inequality (1), we can compute the maximum link length as $r_{max} = \left( \frac{P}{\kappa} \right)^{\frac{1}{\gamma}}$. Notice that the links whose length is $r_{max}$ are not considered, as only node $u$ can be a sender to send its data to some node $v$ over the link $(u,v)$ whose $d(u,v) = r_{max}$, while other nodes cannot transmit concurrently. Therefore, we are interested in links $(u,v)$ whose $d(u,v) \leq r_{max}$ for some constant $\gamma > 1$ as in [11]; thus, we set $\gamma$ to be $\left( \frac{P}{\kappa} \right)^{\frac{1}{\gamma}}$ for the SINR model. We also make the assumption that the undirected communication graph $G = (V,E)$, where $E = \{(u,v) | d(u,v) \leq r\}$, is connected and $\alpha > 2$ as suggested in [9].

Huang et al. [11] showed that the SINR model can be translated to the collision-free-interference model if $\rho = \frac{P}{\kappa} \cdot \frac{1}{\alpha} \cdot \frac{1}{\gamma} + 3$, accordingly, The HCBS algorithm works for the SINR model as well if one sets $\rho = \left( \frac{2}{\gamma - 1} \right)^{\frac{1}{\gamma}} (\frac{2}{\gamma - 1} + \frac{1}{\gamma - 1} + 3)$.

IV. ANALYSIS

In this section, we analyze the HCBS algorithm (Algorithm 1) and bound the latency of the data collection schedules produced by it.

Lemma 1: When $\rho = 1$, any two nodes $u$ and $v$ can send data simultaneously without interference if $|\ell(u) - \ell(v)| \geq 3$, where $\ell(u)$ and $\ell(v)$ are the levels of $u$ and $v$ on $T$, respectively.

Corollary 2: When $\rho \geq 1$, any two nodes $u$ and $v$ can send data simultaneously without interference if $|\ell(u) - \ell(v)| \geq 3$, where $\ell(u)$ and $\ell(v)$ are the levels of $u$ and $v$ on $T_1$, respectively.

Lemma 3: Consider $\text{subtree}(z)$ of $z \in V$. The maximum height of $\text{subtree}(z)$ is $[4\pi \rho (\rho + 1) - 4\pi + 1]$.

Proof: First, let $\text{sec}(z, \theta, r)$ denote a sector with an angle of $\theta$ radian of a circle centered at node $z$ with the radius of $r$. Then, the collision area and the interference area can be defined by $\text{sec}(z, 2\pi, r)$ and $\text{sec}(z, 2\pi, \rho r) - \text{sec}(z, 2\pi, r)$, respectively.

Next, let us partition the sectors, where $3 \leq q \leq (\rho - 1)$, similarly. Then, the last sector $\text{sec}(z, 0.5, \rho r) - \text{sec}(z, 0.5, (\rho - 1)r)$ is partitioned into $2\rho$ cells such that the length of every line segment connecting two points is $\leq 0.5r$, and the length of every arc connecting two points is $\leq 0.5r$. After partitioning, the maximum distance in each cell is $\leq 0.5r$.

Next, let us bound the number of hops on $T$ from the root $z$ to its one of descendants, whose level on $\text{subtree}(z)$ is largest. It is obvious that $\text{sec}(z, 2\pi, 0.5r)$ can contain at most one hop. The sector $S := \text{sec}(z, 2\pi, r) - \text{sec}(u, 2\pi, 0.5r)$ can contain at most $4\pi \cdot 1$ cells and thus, it can contain at most $4\pi \cdot 1$ hops, if the whole area of the sector $S$ is partitioned as explained above. Similarly, $\text{sec}(z, 2\pi, 2r) - \text{sec}(u, 2\pi, r)$, $\text{sec}(z, 2\pi, 3r) - \text{sec}(u, 2\pi, 2r)$, $\ldots$, and $\text{sec}(z, 2\pi, \rho r) - \text{sec}(u, 2\pi, (\rho - 1)r)$ can contain at most $4\pi \cdot 4$ hops, $4\pi \cdot 6$ hops, $\ldots$, and $4\pi \cdot 2\rho$ hops, respectively.

Therefore, we can conclude that in the area in $z$’s interference range, there exist at most $1 + 4\pi + \sum_{i=2}^{\rho} 4\pi \cdot 2i = 4\pi \rho (\rho + 1) - 4\pi + 1$ hops. Hence, the maximum height of $\text{subtree}(z)$ is $[4\pi \rho (\rho + 1) - 4\pi + 1]$.

Figure 6: Partitioning areas in $z$’s collision and interference ranges into several cells such that the maximum distance in each cell is $\leq 0.5r$. Radius of the innermost circle (disk) having the smallest area is $0.5r$, radius of the circle having the second smallest area is $r$, radius of the circle having the third smallest area is $1.5r$, ... and radius of the outermost circle having the largest area is $3r$.

Lemma 4 (Lower Bound): Every data collection schedule has at least $n - 1$ timeslots, where $n$ is the number of nodes.

Proof: As a sink $s$ has to receive $n - 1$ distinct messages, any data collection schedule needs at least $n - 1$ timeslots.

Theorem 5 (Graph Model): The HCBS algorithm produces a successful schedule whose latency is bounded by $3 \cdot \left\lceil 4\pi \rho (\rho + 1) - 4\pi + 1 \right\rceil (n - 1) = O(n)$, and it is therefore a constant-factor approximation with the factor of $3 \cdot \left\lceil 4\pi \rho (\rho + 1) - 4\pi + 1 \right\rceil$.

Proof: First note that there are $n - 1$ messages that the sink node $s$ must receive. In the HCBS, $s$ receives a message at most every 3 timeslots when $\rho = 1$ (Lemma 1), and $3 \cdot \left\lceil 4\pi \rho (\rho + 1) - 4\pi + 1 \right\rceil$ timeslots (Lemma 3) when $\rho > 1$ until it receives all messages from all the other nodes without any collision or interference (Lemma 1 and Corollary 2). Therefore, it takes at most $3 \cdot (n - 1)$ timeslots when $\rho = 1$, and $3 \cdot \left\lceil 4\pi \rho (\rho + 1) - 4\pi + 1 \right\rceil (n - 1)$
timeslots when \( \rho > 1 \) for the sink node \( s \) to collect all data.

Therefore, by Lemma 4, the HBCS algorithm produces a data collection schedule, and it is an approximation algorithm with the constant factor of 3 when \( \rho = 1 \), and \( 3 \cdot [4\pi \rho (\rho + 1) - 4\pi + 1] \) when \( \rho > 1 \).

**Corollary 6 (SINR Model):** For SINR threshold \( \beta \geq 1 \), path loss exponent \( \alpha > 2 \), and some constant \( \gamma > 1 \), the HBCS algorithm produces a successful schedule whose latency is bounded by \( [4\pi \rho (\rho + 1) - 4\pi + 1] \cdot (n - 1) = O(n) \), where \( \rho = (\frac{24\gamma\beta}{\gamma - 1} \left( \frac{2}{\alpha - 2} + \frac{1}{\alpha - 1} + 3 \right))^\frac{1}{2} \), and it is therefore a constant-factor approximation with the factor of \( 3 \cdot [4\pi \rho (\rho + 1) - 4\pi + 1] \).

V. Simulation

In this section, we examine the performance of our proposed algorithm, the HBCS (Algorithm 1). For our simulation, we generate a set \( G = \{G_n | n \in \{50, 100, 150, 200, \cdots, 500\} \} \), where \( G_n = \{G_i | 1 \leq i \leq 100 \} \) consists of 100 different networks, \( G_1^n, G_2^n, \cdots, G_{100}^n \), each of which has \( n \) nodes. All networks are generated randomly in the Euclidean plane of dimension 500 \( \times \) 500. For each \( G_n \in G \), we average the latencies produced by the HBCS algorithm over all the 100 networks.

A. Simulation Setup

1) Graph Model: For the initial power assignment of nodes in a network, we first compute a minimum spanning tree \( T_{\text{MST}} \) using edge weights defined as the distance between two nodes. Then, we set the maximum transmission range \( r \) to be the distance of the longest edge in \( T_{\text{MST}} \), and the uniform power level \( P \) to be \( r \). Notice that \( P \) is the minimally required power level to get a connected graph. Given the initial power assignment, we obtain the initial graph \( G = (V, E) \), where \( E = \{(u, v) | d(u, v) \leq r \} \).

We then set the interference factor \( \rho \) as follows:

- Choice of \( \rho \): We use \( \rho \in \{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0 \} \). It is practically assumed that \( \rho \in [3, 5] \) for the collision-interference-free graph model [11]. Given \( \rho \), we obtain the interference graph \( G_I = (V, E_I) \), where \( E_I = \{(u, v) | d(u, v) \leq \rho \cdot r \} \).

2) SINR Model: For the initial power assignment of nodes in a network, if we set \( P \) to be \( r_{\text{max}} \) defined as the distance of the longest edge in \( T_{\text{MST}} \), then only node \( u \) can be a sender to send data to some node \( v \) over the link \( (u, v) \) with \( d(u, v) = r_{\text{max}} \), while other nodes cannot transmit concurrently. Therefore, we set the maximum transmission range \( r \) to be \((\gamma N \beta \cdot (r_{\text{max}})^\alpha) \) (See Section III-B) thereby setting \( P \) to be \( r \). Notice again that \( P \) guarantees the network connectivity and it is also the minimally required power level for at least two nodes to send data at the same time. Given the initial power assignment, we obtain the initial graph \( G = (V, E) \), where \( E = \{(u, v) | d(u, v) \leq r \} \). To apply the HBCS algorithm to the SINR model, we compute \( \rho \) as discussed in Section III-B:

\[
\rho = \left( \frac{24\gamma\beta}{\gamma - 1} \left( \frac{2}{\alpha - 2} + \frac{1}{\alpha - 1} + 3 \right) \right)^\frac{1}{2}.
\]

Given the computed \( \rho \), we obtain the interference graph \( G_I = (V, E_I) \), where \( E_I = \{(u, v) | d(u, v) \leq \rho \cdot r \} \).

The SINR parameters including the pass loss exponent \( \alpha \), the SINR threshold \( \beta \), some constant \( \gamma \), and the background noise \( N \) are set as follows:

- Choice of \( \alpha \): We use \( \alpha \in \{2.1, 3.0, 4.0, 5.0, 5.9 \} \) as it is typically assumed that \( 2 < \alpha < 6 \) [14].
- Choice of \( \beta \): We use \( \beta \in \{1.0, 1.5, 2.0, 2.5, 3.0 \} \).
- Choice of \( \gamma \): We use \( \gamma \in \{1.0, 2.0, 3.0, 4.0, 5.0 \} \).
- Choice of \( N \): Noticing that \( \rho \) is independent on \( N \) in the Equation (2), we simply set \( N = 1 \).

B. Simulation Results

1) Graph Model: Figure 7(a) shows the uniform power level \( P \) values (dots) and the corresponding averages (lines) for each \( n \), and we can observe that as \( n \) increases, the minimally required power for connectivity decreases. It is because as \( n \) increases, the distance of the longest edge in \( T_{\text{MST}} \) decreases. Figure 7(b) shows the interference factor \( \rho \) values that generate complete interference graph \( G_I \), and we can observe that as \( n \) increases, \( \rho \) values also increase. This is because as \( n \) becomes larger (i.e., as \( P \) becomes smaller), \( \rho \) values need to become larger so that interference ranges (i.e., \( \rho \cdot r = \rho \cdot P \)) cover a whole network.

Next, Figure 7(c) shows the latency performance of the HBCS algorithm with various \( \rho \) values. As \( \rho \) increases, the latencies also increase because the interference ranges become larger. Also, we can observe that for each \( n \), as \( \rho \) increases, the algorithm results in nearly the same performances. It is because the interference graphs started becoming nearly complete graphs in our simulation at some point with large \( \rho \) values. (See the results with some large \( \rho \) values, e.g., \( \rho = 4.0, 4.5, \) and 5.0, in Figure 7(c).) Figures 1, 2, 3 and 4 illustrate the specified examples of this explanation. To be more specific, when \( \rho = 4 \) and 5, the interference graphs become complete graphs, and thus at each timeslot, only one node can send data without any interference. Even with \( \rho = 3 \), the interference graph becomes a nearly complete graph (See Figure 3(d)).

2) SINR Model: In order to evaluate the effect of \( \alpha \), we fixed \( \beta = 1 \) and \( \gamma = 5 \). Figure 8(a) shows the latency performance of the HBCS algorithm with the fixed \( \beta \) and \( \gamma \), and various \( \alpha \in \{2.1, 3.0, 4.0, 5.0, 5.9 \} \). We can observe that as \( \alpha \) increases, the latencies decrease. To understand the reason better, let us consider the SINR Inequality (1), and a sender node \( u \) and a receiver node \( v \) over the link \( (u, v) \) whose \( d(u, v) = d_{\text{max}} \). This is the case that the SINR value of \( v \) is minimized (i.e., \( u \) suffers from maximum interference by all the other concurrently sending nodes when it sends its data to \( v \)).
Then, the SINR at the receiver $v$ is

$$SINR_{(u,v)} = \frac{p(u) \cdot d(u,v)^{-\alpha}}{N + \sum_{w \in X \setminus \{u,v\}} p(w) \cdot d(w,v)^{-\alpha} + \gamma N \beta}$$

Here, we can easily observe that increasing $\alpha$ values while fixing all the other parameter values make $SINR(u,v)$ at $v$ increase. The larger the SINR values at receivers, the more the senders can send data at the same time thereby decreasing the latencies. Thus, as $\alpha$ increases, the latencies decrease as in Figure 8(a).

Next, in order to evaluate the effect of $\beta$, we fixed $\alpha = 5$ and $\gamma = 5$. Here, $\gamma$ is fixed to be 5 because larger...
γ values result in smaller latencies. Figure 8(b) shows the latency performance of the HBCS algorithm with the fixed α and γ, and various β ∈ {1.0, 1.5, 2.0, 2.5, 3.0}. We can observe that as β decreases, the latencies also decrease. It is because having smaller β values makes more number of nodes can send data at the same time (See Equation (1)) thereby requiring the smaller number of timeslots to complete data collection.

Lastly, in order to evaluate the effect of the constant γ, we first fixed the pass loss exponent α = 5 and the SINR threshold β = 5. Note that α is fixed to be 5 because as α increases, ρ decreases (See Equation (2)). Also note that β is fixed to be 1 because as β decreases, more nodes can send data at the same time (See Inequality (1)). Figure 8(c) shows the latency performance of the HBCS algorithm with the fixed α and β, and various γ ∈ {1.0, 2.0, 3.0, 4.0, 5.0}. We can observe that as γ increases while fixing both α and β, the latencies decrease. It can be explained that as the value of γ increases (i.e., as the power P increases), the levels of many nodes decrease on T as γ increases. Accordingly, data can be collected at the sink node with less delay. Note that having a complete communication graph G gives the optimal solution (i.e., latency, n − 1) for the MLCS problem.

Based on our experiment, we infer that latency can be improved with larger α and γ values, and smaller β values.

VI. CONCLUSION

In this paper, we focused on the Minimum Latency Collection Scheduling (MLCS) problem of Wireless Sensor Networks (WSNs) in the graph model as well as the more realistic physical interference model known as Signal-to-Interference-Noise-Ratio (SINR). We proposed a O(1)-approximation algorithm that works in both interference models by yielding schedules whose latency is bounded by O(n), where n is the number of nodes in a network. We then evaluated the performance of the proposed algorithm with simulated networks and discussed experimental results.

Recent advances in networks have led to the development of new wireless sensor devices equipped with directional antenna so that interference are reduced by emitting signals within narrower area. So, for future work, we plan to study the MLCS problem in directional wireless sensor networks thereby reducing data collection latencies. We also plan to study other related problems such as broadcast and data aggregation adopting both the interference models.

REFERENCES


Min Kyung An received her Ph.D. in computer science from the University of Texas at Dallas in August 2013, and her M.S. in computer science from the University of Texas at Arlington in August 2007. During her M.S. studies, she received the Graduate Studies Abroad Program Scholarship funded by the Korean government. She is currently an assistant professor in the Department of Computer Science at Sam Houston State University, Huntsville, Texas, USA. Her major research areas include wireless ad hoc and sensor networks, social networks, design and analysis of approximation algorithms, graph theory, and program analysis.

Prof. An is a member of IEEE.
Hyuk Cho received the B.E. degree in computer engineering from Chonbuk National University, Korea, the M.A. degree in computer science from Korea University, Korea, and both the M.S. and the Ph.D. degrees in computer sciences from the University of Texas at Austin.

He is an associate professor in the Department of Computer Science at Sam Houston State University, Huntsville, Texas, USA. He is known for his work on co-clustering algorithms, their extensions, and their applications to various practical tasks in real world problems. His research interests include data mining, statistical pattern recognition, machine learning, pervasive computing, bioinformatics, and data science. Previously he worked on linear matrix inequality and soft computing, including neural networks, evolutionary computation, genetic algorithm, and fuzzy/rough set theory.

Prof. Cho is a member of IEEE/ACM and SIAM.