Latency-minimizing data aggregation in wireless sensor networks under physical interference model

Hongxing Li, Chuan Wu, Qiang-Sheng Hua, Francis C.M. Lau

Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

A R T I C L E   I N F O
Article history:
Received 20 May 2011
Received in revised form 20 September 2011
Accepted 6 December 2011
Available online 20 December 2011

Keywords:
Data aggregation
Wireless sensor networks
Physical interference model
Minimum-latency

A B S T R A C T
Minimizing latency is of primary importance for data aggregation which is an essential application in wireless sensor networks. Many fast data aggregation algorithms under the protocol interference model have been proposed, but the model falls short of being an accurate abstraction of wireless interferences in reality. In contrast, the physical interference model has been shown to be more realistic and has the potential to increase the network capacity when adopted in a design. It is a challenge to derive a distributed solution to latency-minimizing data aggregation under the physical interference model because of the simple fact that global-scale information to compute the cumulative interference is needed at any node. In this paper, we propose a distributed algorithm that aims to minimize aggregation latency under the physical interference model in wireless sensor networks of arbitrary topologies. The algorithm uses \( O(K) \) time slots to complete the aggregation task, where \( K \) is the logarithm of the ratio between the lengths of the longest and shortest links in the network. The key idea of our distributed algorithm is to partition the network into cells according to the value \( K \), thus obviating the need for global information. We also give a centralized algorithm which can serve as a benchmark for comparison purposes. It constructs the aggregation tree following the nearest-neighbor criterion. The centralized algorithm takes \( O(\log n) \) and \( O(\log^3 n) \) time slots when coupled with two existing link scheduling strategies, respectively (where \( n \) is the total number of nodes), which represents the current best algorithm for the problem in the literature. We prove the correctness and efficiency of our algorithms, and conduct empirical studies under realistic settings to validate our analytical results.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Data aggregation is a habitual operation of many wireless sensor networks, which transfers data (e.g., temperature) collected by individual sensor nodes to a sink node. The aggregation typically follows a tree topology rooted at the sink. Each leaf node would deliver its collected data to its parent node. Intermediate sensor nodes of the tree may optionally perform certain operations (e.g., sum, maximum, minimum, mean, etc.) on the received data and forward the result. Because the wireless medium is shared, transmissions to forward the data need to be coordinated in order to reduce interference and avoid collision. The fundamental challenge can be stated as: How can the aggregation transmissions be scheduled in a wireless sensor network such that no collision may occur and the total number of time slots used (referred to as aggregation latency) is minimized? This is known as the Minimum-Latency Aggregation Scheduling (MLAS) problem in the literature [1–5].

The MLAS problem is typically approached in two steps: (i) data aggregation tree construction and (ii) link transmission scheduling. For (i), we assume the simplest mode
in which every non-leaf node in the tree will make only one transmission, after all the data from its child nodes have been received. A correct solution to the MLAS problem requires that no concurrent transmissions interfering with each other should take place. If steps (i) and (ii) are carried out simultaneously in a solution, we have a “joint” design.

To model wireless interference, existing literature mostly assume the protocol interference model, in which a transmission is successful if and only if its receiver is within the transmission range of its transmitter and outside the interference range of any other concurrent transmitters. The best results known for the MLAS problem or similar problems ([2–5]) under the protocol interference model bound the aggregation latency in \( O(\Delta + R) \) time slots, where \( R \) is the radius of the sensor network in hops and \( \Delta \) is the maximal node degree (i.e., the maximum number of nodes in any node's transmission range). The protocol interference model however has been found to be too simplistic and cannot serve as an accurate abstraction of wireless interferences. Instead, the physical interference model [6], which captures the reality more accurately, is becoming more popular. Little research however has so far been done to address the MLAS problem under the physical interference model.

The protocol interference model considers only interferences within a limited region, whereas the physical interference model tries to capture the cumulative interference due to all other concurrently transmitting nodes in the entire network. More precisely, in the physical interference model, the transmission of link \( e_{ij} \) can be successful if the following condition regarding the Signal-to-Interference-Noise-Ratio (SINR) is satisfied:

\[
\frac{P_{ij}/d_{ij}^\alpha}{N_0 + \sum_{k \neq l \neq j} P_{kl}/d_{kl}^\alpha} \geq \beta. \tag{1}
\]

Here \( A_{ij} \) denotes the set of links that transmit simultaneously with \( e_{ij} \). \( P_{ij} \) and \( P_{kl} \) denote the transmission power at the transmitter of link \( e_{ij} \) and that of link \( e_{kl} \), respectively. \( d_{ij} \) (\( d_{kl} \)) is the distance between the transmitter of link \( e_{ij} \) (\( e_{kl} \)) and the receiver of link \( e_{ij} \). \( \alpha \) is the path loss ratio, whose value is normally between 2 and 6. \( N_0 \) is the ambient noise. \( \beta \) is the SINR threshold for a successful transmission, which is at least 1.

We give an example, in Fig. 1, to demonstrate the correctness and efficiency of our algorithms. We show that the distributed algorithm Cell-AS achieves a worst-case aggregation latency bound of \( O(K) \) (where \( K \) is the link length diversity), and the centralized algorithm NN-AS achieves worst-case bounds of \( O(\log n) \) and \( O(\log^2 n) \) when coupled with the link scheduling strategies in [8,9], respectively (where \( n \) is the total number of sensor nodes). In addition, we derive a theoretically optimal lower bound for the MLAS problem under any interference model—\( \log(n) \). Given this optimal bound, the approximation ratios are \( O(K/\log n) \) with Cell-AS, \( O(1) \) with NN-AS and the link scheduling in [8], and \( O(\log^2 n) \) with NN-AS and the link scheduling in [9]. We also compare our distributed algorithm with Li et al.’s algorithm in [7] both analytically and experimentally. We show that both algorithms have an \( O(n) \) latency upper bound in their respective worst cases, while Cell-AS

![Fig. 1. A data aggregation example.](image-url)
can be more effective, with latency $O(\log n)$, when applied to Li et al.’s worst case examples. Our experiments under realistic settings demonstrate that Cell-AS can achieve up to a 35% latency reduction as compared to Li et al.’s. Besides, we have found that in uniform topologies, the aggregation latencies for NN-AS (with the link scheduling in [9]) and Li et al.’s algorithm can be reduced to $O(\log^2 n)$ and $O(\log^4 n)$, respectively, while Cell-AS’s latency is between $O(\log^3 n)$ and $O(\log^6 n)$.

The contribution of this paper can be summarized as follows:

- We investigate the Minimum-Latency Aggregation Scheduling (MLAS) problem under the physical interference model for arbitrary topologies, and propose a distributed algorithm, Cell-AS, to avoid the need of global information about interference with a latency bound of $O(K)$, where $K$ is the link length diversity (the logarithm of the ratio between the lengths of the longest and the shortest links).
- We also propose a centralized algorithm, NN-AS, for completeness and to serve as a benchmark in the performance comparison. The worst-case latency bounds of the centralized algorithm can be $O(\log n)$ and $O(\log^2 n)$ when coupled with the link scheduling strategies in [8,9], respectively (where $n$ is the total number of sensor nodes).
- A theoretically optimal lower bound for the MLAS problem under any interference model is derived—$\log (n)$. Given this optimal bound, the approximation ratios are $O(K/\log n)$ with Cell-AS, $O(1)$ with NN-AS and the link scheduling strategy in [8], and $O(\log^2 n)$ with NN-AS and the link scheduling strategy in [9]. Thus, our centralized algorithm, NN-AS, with link the scheduling strategy in [8] achieves an asymptotically optimal latency performance, which is the current best result in the literature.
- Both analytical and experimental comparisons are conducted between our distributed algorithm and Li et al.’s algorithm in [7] to demonstrate the efficiency of our proposed algorithm.

The remainder of this paper is organized as follows. We discuss related work in Section 2 and formally present the problem model in Section 3. The Cell-AS and NN-AS algorithms are presented in Sections 4 and 5, respectively. An extensive theoretical analysis is given in Section 6. We report our empirical studies of the algorithms in Section 7. Finally, we conclude the paper in Section 8.

2. Related work

2.1. Data aggregation

Data aggregation is an important problem in wireless sensor network research. There exist a lot of exciting work investigating the problem [1–5,7,10,11], among which minimizing aggregation time via transmission scheduling is a common topic.

To the best of our knowledge, all except one paper [7] assume the protocol interference model. Chen et al. [1] propose a data aggregation algorithm with a latency bound of $(\Delta + 1)R$, where $R$ is the network radius in hop count and $\Delta$ is the maximal node degree. The NP-hardness proof of the MLAS problem is also presented. The current best contributions [2–5,10] bound the aggregation latency by $O(\Delta + R)$.

[2] is the first work that converts $\Delta$ from a multiplicative factor to an additive one. The algorithm is built on the basis of maximal independent set, which is also used in [5]. The latter work provides a distributed solution to the problem.

In [3], the MLAS problem is dealt with in the context of multi-hop wireless networks and with the assumption that each node has a unit communication range and an interference range of $\rho \geq 1$. Xu et al. [4] propose a distributed aggregation schedule and prove a lower bound of $\max(\log n, R)$ on the latency of data aggregation under any graph-based interference model, where $n$ is the network size. Different from the above work, where connected dominating sets or maximal independent sets are employed, a novel approach of distributed aggregation with latency bound $O(\Delta + R)$ is introduced in [10]. Here, $R$ is the inferior network radius satisfying $R < r < D < 2R$, where $D$ is the network diameter in hop-count.

The MLAS problem is extended to the case with multiple sinks in [11] with a latency bound of $O(\Delta + kR)$, where $k$ is the number of sinks.

The only solution to the MLAS problem under the physical interference model is by Li et al. [7]. They propose a distributed aggregation scheduling algorithm with constant power assignment, which can achieve a latency bound of $O(\Delta + R)$ when the transmission range is set as $\delta r$. $0 < \delta < 1$ is a configuration parameter and $r$ is the maximum achievable transmission range under the physical interference model with power assignment $p$ and $\frac{P_{\text{trans}}}{P_{\text{rec}}} = \beta$. No deterministic latency bound can be derived when the transmission range is changed to $r$, for which probabilistic analysis has been conducted. The efficiency of Li et al.’s algorithm may not be guaranteed when applied to arbitrary topologies, which is a consequence of constant power assignment.

A detailed comparison of data aggregation algorithms is given in Table 1.

2.2. Link scheduling under the physical interference model

The physical interference model has received increasing attention in recent years, as a more realistic abstraction of wireless interferences [6]. It has also been shown that it can significantly improve the network capacity [9,12–15], as compared to the protocol interference model. An important track of existing studies focuses on the Minimum Length link Scheduling (MLS) problem [9,14–18], which is to find the minimum amount of time to schedule the transmissions in a given link set without collision. The MLS problem is closely related to the link scheduling step of the MLAS problem.

Moscibroda et al. are the first to formally define and investigate the link scheduling complexity over a connected structure in wireless networks [14]. They further study topology control for the MLS problem under the
physical interference model and obtain a theoretical upper bound on the scheduling complexity in arbitrary wireless network topologies [15].

In [9], Moscibroda proposes a link scheduling algorithm for connected structures, with a scheduling complexity of $O(\log^2 n)$. The scheduling complexity of the connected structure is further reduced to $O(\log n)$ in [8]. Hua et al. [19] extend the MLS problem for connected structures to ultra-wideband networks and derive a scheduling algorithm with complexity $O(\log (n/m) \cdot \log^3 n)$, where $m$ is the processing gain. They further [20] solve the MLS problem at the cost of moderately exponential time.

Halldórsson et al. [21] give a distributed solution to the MLS problem with $O(\log n)$ approximation. They then present a constant-factor approximation for the MLS problem with any given link set and length-monotone, sub-linear power assignment in [22]. A unified algorithmic framework is built to develop approximation algorithms for link scheduling with or without power control under the physical interference model in [23]. Wan et al. [24] show a constant-approximation in the simplex mode. Kesselheim et al. [25] propose another constant approximation in fading metrics and an $O(\log n)$ approximation in the general metric space.

In [16], a new measurement called “disturbance” is proposed to address the difficulty of finding a short schedule. Goussevskaia et al. [17] make the milestone contribution of proving the NP-completeness of a special case of the MLS problem. In [18], Fu et al. extend the MLS problem by introducing consecutive transmission constraints. An NP-hardness proof is provided for this extended problem.

### 3. The problem model

We consider a wireless sensor network of $n$ arbitrarily distributed sensor nodes, $v_0, v_1, \ldots, v_{n-1}$, and a sink node, $v_s$. Let directed graph $G = (V, E)$ denote the tree constructed for data aggregation from all the sensor nodes to the sink, where $V = \{v_0, v_1, \ldots, v_n\}$ is the set of all nodes, and $E = \{e_{ij}\}$ is the set of transmission links in the tree with $e_{ij}$ representing the link from sensor node $v_i$ to its parent $v_j$.

Our problem at hand is to pick the directed links in $E$ to construct the tree and to come up with an aggregation schedule $S = \{S_0, S_1, \ldots, S_{T-1}\}$, where $T$ is the total time span for the schedule and $S_i$ denotes the subset of links in $E$ scheduled to transmit in time slot $t$, $\forall t = 0, \ldots, T-1$. A correct aggregation schedule must satisfy the following conditions. First, any link should be scheduled exactly once, i.e., $u^x_{i,j} \subseteq S_i$. Hua et al. prove the NP-hardness of this problem.

### 4. Distributed aggregation scheduling

Our main contribution is an efficient distributed scheduling algorithm called Cell Aggregation Scheduling (Cell-AS) for solving the MLS problem with arbitrary distribution of sensor nodes. Our distributed algorithm features joint tree construction, link scheduling, and power control, and executes in a phase-by-phase fashion to achieve the minimum
aggregation latency. In contrast, the tree construction and link scheduling are disjoint steps in [7]. We first present the key idea behind our algorithm and then discuss important techniques to implement the algorithm in a fully distributed fashion.

4.1. Design idea

Initially, the entire area can be seen as being divided into many small areas. Our distributed algorithm first aggregates data from sensor nodes in each small area, where the transmission links are short, and then aggregates data in a larger area by collecting from those small ones with longer transmission links; this process repeats until the entire network is covered by one large area.

We divide the lengths of all possible transmission links in the network into \(K+1\) categories: \([3^0, 2 \cdot 3^0], (2 \cdot 3^0, 2 \cdot 3^1], \ldots, (2 \cdot 3^{k-1}, 2 \cdot 3^k]\), where \(K\) is bounded by the maximum node distance \(D\) in the network with \(2 \cdot 3^{k-1} < D < 2 \cdot 3^k\). A link from node \(v_i\) to node \(v_j\) falls into category \(k\) if the Euclidean distance between these two nodes lies within \((2 \cdot 3^{k-1}, 2 \cdot 3^k]\) with \(k = 1, \ldots, K\), or \([3^0, 2 \cdot 3^0]\) with \(k = 0\). We refer to \(K\) as the link length diversity which is proportional to the logarithm of the ratio between the lengths of the longest and the shortest links in the network. In our design, aggregation links in category \(k\) are treated and their transmissions are scheduled (to aggregate data in the smaller areas) before links in category \(k + 1\) are processed (to aggregate data in the larger areas).

The algorithm is carried out in an iterative fashion: In round \(k (k=0, \ldots, K)\), the network is divided into hexagonal cells of side length \(3^k\). In each cell, a node with the shortest distance to the sink is selected as the head, responsible for data aggregation; the other nodes in the cell directly transmit to the head, one after another, with links no longer than \(2 \cdot 3^k\). In the next round \(k + 1\), only the head nodes in the previous round remain in the picture. The network is covered by hexagonal cells of side length \(3^{k+1}\) and a new head is selected for data aggregation in each cell. After \(K + 1\) rounds of the algorithm, only one node remains, which will have collected all the data in network, and will transmit the aggregated data to the sink node in one hop. Fig. 2 gives an example of the algorithm in a sensor network with three link length categories, in which selected head nodes are in black.

In each round \(k\) of the algorithm, links of length category \(k\) are scheduled as follows to avoid interference and to minimize the aggregation latency. We assign colors to the cells and only cells with the same color can schedule their link transmissions concurrently in one time slot. To bound the interference among concurrent transmissions, cells of the same color need to be sufficiently far apart. We use \(\frac{1}{2}X^2 + 12X + 7\) colors in total, such that cells of the same color are separated by a distance of at least \(2(X + 1)3^k\) with \(X = \left(6\beta \left(1 + \left(\frac{1}{3}\right)^{1/2} + 1\right)^{1/2}\right)^{1/2}\), as illustrated in Fig. 3. (The solid cells are of the same color. A–F are six cones to be referred to in the analysis in Section 6.) We will show in Section 6 that by using these many colors, we are able to bound the interferences and thus prove the correctness and efficiency of our algorithm. Inside each cell, the transmission links from all other nodes to the head are scheduled sequentially. Note that each round of the algorithm may take multiple time slots.

The Cell-AS algorithm is summarized in Algorithm 1, where the scheduling of links in cells of the same color is carried out according to Algorithm 2.

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notations.</td>
</tr>
<tr>
<td>Symbol</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>(V)</td>
</tr>
<tr>
<td>(E)</td>
</tr>
<tr>
<td>(v_n)</td>
</tr>
<tr>
<td>(v_i)</td>
</tr>
<tr>
<td>(e_{ij})</td>
</tr>
<tr>
<td>(S)</td>
</tr>
<tr>
<td>(S_t)</td>
</tr>
<tr>
<td>(T(S_t))</td>
</tr>
<tr>
<td>(R(S_t))</td>
</tr>
<tr>
<td>(K)</td>
</tr>
<tr>
<td>(R)</td>
</tr>
<tr>
<td>(\Delta)</td>
</tr>
<tr>
<td>(n)</td>
</tr>
<tr>
<td>(N_0)</td>
</tr>
<tr>
<td>(a)</td>
</tr>
<tr>
<td>(\beta)</td>
</tr>
<tr>
<td>(P_e)</td>
</tr>
<tr>
<td>(d_{ij})</td>
</tr>
<tr>
<td>(A_{ij})</td>
</tr>
</tbody>
</table>

**Algorithm 1.** Distributed Aggregation Scheduling (Cell-AS)

**Input:** Node set \(V\) with sink \(v_n\).

**Output:** Tree link set \(E\) and link schedule \(S\).

1: \(k:=0; t:=0; V:=V - \{v_n\}; E:= \emptyset; S:=\emptyset;\)
2: \(X := \left(6\beta \left(1 + \left(\frac{1}{3}\right)^{1/2} + 1\right)^{1/2}\right)^{1/2};\)
3: while \(|V| \neq 1\) do
4: Cover the network with cells of side length \(3^k\) and color them with \(\frac{1}{2}X^2 + 12X + 7\) colors;
5: for \(i:=1\) to \(\frac{1}{2}X^2 + 12X + 7\) do
6: \(E_i:=0\), where \(E_i\) is link set in cells of color \(i\);
7: for each cell \(j\) with color \(i\) do
8: Select node \(v_m\) in cell \(j\) closest to sink \(v_n\) as head;
9: Construct links from all other nodes in cell \(j\) to \(v_m\);
10: Add the links to \(E_i\) and \(E\);
11: Remove all the nodes in cell \(j\) except \(v_m\) from \(V\);
12: end for
13: \((PS_n, t) := \text{Same-Color-Cell-Scheduler}(E_i, t);\)
14: \(S:=S \cup PS_n;\)
15: end for
16: \(k:=k+1;\)
17: end while
18: \(v_m:=\) the only node in \(V\); Construct link \(e_{mn}\) from \(v_m\) to \(v_n\);
19: \(E:=E \cup \{e_{mn}\}; S:=S \cup \{e_{mn}\};\)
20: return \(E\) and \(S\).
Algorithm 2. Same-color-cell-scheduler

Input: Link set $E_i$ and time slot index $t$.  
Output: Partial link schedule $PS_i$ for links in $E_i$ and $t$.

1: $X := \left( 6\beta \left( 1 + \left( \frac{2}{\sqrt{3}} \right)^{2} \right) + 1 \right)^{1/\alpha}$;  
2: Define constant $c := N_0 / \alpha X^2$;  
3: $PS_i := \emptyset$;  
4: while $E_i \neq \emptyset$ do  
5: $S_i := \emptyset$;  
6: for each cell $j$ with color $i$ do  
7: Choose one non-scheduled link $e_{gh}$ in cell $j$;  
8: Assign transmission power $P_{gh} := c \times d_{gh}^\alpha$;  
9: $S_i := S_i \cup \{ e_{gh} \}$; $E_i := E_i - \{ e_{gh} \}$;  
10: end for  
11: $PS_i := PS_i \cup \{ S_i \}$; $t := t + 1$;  
12: end while  
13: return $PS_i$ and $t$.

4.2. Distributed implementation

The algorithm can be implemented in a fully distributed fashion.

4.2.1. Location and synchronization

In the bootstrapping phase, a middle position of the sensor network is assigned to be the origin (0,0). Each node is then assigned its location coordinates $(x,y)$ relative to the origin with such techniques as GPS. In fact, only a small number of nodes need to be assigned their coordinates initially, as the others can obtain their coordinates through relative positioning (e.g., [26]).

Each node in the sensor network carries out the distributed algorithm in a synchronized fashion, i.e., it knows the start of each round $k$ and each time slot $t$. Such synchronization can be achieved using one of the practical synchronization algorithms in the literature (e.g., [27]).

4.2.2. Neighbor discovery

In each round $k$, the network is divided into cells of side length $3^k$ in the manner as illustrated in Fig. 3. Each node can determine the cell it resides in the current round based on the node’s location. It can then discover its neighbors in the cell via local broadcasting [28]. The broadcasting range is $2 \cdot 3^{k+1}$, such that all nodes in the same cell can be reached.

4.2.3. Head selection

The head of a cell in round $k$ is the node in the cell closest to the sink. All the nodes are informed of the sink’s location in the bootstrapping stage of the algorithm, or even before they are placed in the field. Since each node knows the location information of all its neighbors in the same cell, it can easily identify the head.

4.2.4. Distributed link scheduling

In each round $k$, coloring of the cells is done as illustrated in Fig. 3. As each node knows which cell it resides in, it can compute color $i$ of its cell in this round. Cells of the same color are scheduled according to the sequence of their color indices, i.e., cells with color $i$ schedule their transmissions before those with color $i+1$. The head node in a cell is responsible to decide when the other nodes in its cell can start to transmit, and to announce the completion of transmissions in its cell to all head nodes within distance $2(X+1)3^k$.

A head node in a cell with color $i+1$ waits until it has received completion notifications from all head nodes in
cells of color $i$ within distance $2(X + 1)^3$. It then schedules the transmission of all the other nodes in its cell one by one, by sending “pulling” messages. For a non-head node in the cell, it waits for the “pulling” message from the head node and then transmits its data to the head.

When the algorithm is executed round after round, only the nodes that have not transmitted (the heads in previous rounds) remain in the execution, until their transmission rounds arrive.

5. Centralized aggregation scheduling

Assuming global information is available at each sensor, then a centralized scheduling algorithm can be constructed, which can achieve the best aggregation latency for the MLAS problem. We present in the following such a centralized algorithm, Nearest-Neighbor Aggregation Scheduling (NN-AS).

Our centralized algorithm progresses in a phase-by-phase fashion, with joint tree construction and link scheduling. In each round, the algorithm finds a nearest neighbor matching among all the sensor nodes that have not transmitted their data, and schedule all the links in the matching.

The algorithm is started with all the sensor nodes in $V - \{v_n\}$. It finds for each node $v_i$ the nearest neighbor node $v_j$, where neither $v_i$ nor $v_j$ has already been included in the matching, and a directed link from $v_i$ to $v_j$ is established. For example, in Fig. 4 showing a sensor network of six nodes, the matching identified in round 0 contains two links, $1 \rightarrow 3$ and $4 \rightarrow 6$. The links in matching $M_0$ (of round 0) are then scheduled, using either the link scheduling algorithm proposed in [8] or the one in [9], both of which schedule a set of links with guaranteed scheduling correctness under the physical interference model. After all transmissions in round 0 are scheduled, all the nodes that have transmitted are removed, and the algorithm repeats with the remaining nodes. In Fig. 4b, nodes 2, 3, 5, and 6 remain, and two links are generated based on the nearest neighbor criterion and then scheduled for transmission. The process repeats until only one sensor node remains, which will transmit its aggregate data to the sink node in one hop.

The centralized algorithm is summarized as Algorithm 3, where Phase-Scheduler-1 and Phase-Scheduler-2 call upon Algorithm 4 provided in [8] and Algorithm 5 provided in [9], respectively, to generate the schedule for links in matching $M_k$ in round $k$. In Algorithm 4, $\zeta(\cdot)$ is the Riemann zeta function [29]. In Algorithm 5, the pre-processing ($M_k$) procedure assigns two values, i.e., $\tau_{ij}$ and $\gamma_{ij}$ related to link length $d_{ij}$, for each link $e_{ij} \in M_k$, while the check ($e_{ij}$,$S_t$) procedure checks whether link $e_{ij}$ can transmit concurrently with links in $S_t$ and returns a Boolean value.

![Fig. 4. The steps of NN-AS: an example with six sensor nodes.](image)

### Algorithm 3. Centralized Aggregation Scheduling (NN-AS)

| Input: | Node set $V$ with sink $v_n$. |
| Output: | Tree link set $E$ and link schedule $S$. |
| 1: | $k := 0$; $t := 0$; $E := \emptyset$; $S := \emptyset$; $V := V - \{v_n\}$; |
| 2: | while $|V| \neq 1$ do |
| 3: | $M_k := \emptyset$; |
| 4: | for each $v_i \in V$ do |
| 5: | if $v_i \notin T(M_k) \cup R(M_k)$ then |
| 6: | Find $v_j$’s nearest-neighbor $v_i \in V$; |
| 7: | if $v_j \notin T(M_k) \cup R(M_k)$ then |
| 8: | Construct link $e_{ij}$ from $v_i$ to $v_j$; $M_k := M_k \cup \{e_{ij}\}$; |
| 9: | end if |
| 10: | end for |
| 11: | $v_i :=$ the only node in $V$; Construct link $e_{in}$ from $v_i$ to $v_n$; |
| 12: | $E := E \cup \{e_{in}\}$; $S := S \cup \{e_{in}\}$; |
| 13: | return $E$ and $S$. |

### Algorithm 4. Phase-Scheduler-1 [8]

| Input: | Link set $M_k$ and time slot index $t$. |
| Output: | Partial link schedule $P_{Sk}$ for links in $M_k$ and $t$. |
| 1: | Define constant integer $b := \lceil 16^{3/2} \rceil$. |
| 2: | $\zeta(2) := \zeta(2) \cdot 3^b \cdot 2^{3b} / (2 - 2^b)$; $P_{S0} := \emptyset$; |
| 3: | Let $R_{\text{max}} := \max_{e_{ij} \in M_k} (d_{ij})$; $R_{\text{min}} := \min_{e_{ij} \in M_k} (d_{ij})$; |
| 4: | for each integer $v$ with $0 \leq v \leq b^3 - 1$ do |
| 5: | $S_{0v} := \emptyset$; |
| 6: | end for |
| 7: | for each link $e_{ij} \in M_k$ do |
| 8: | $P_{ij} := 3N\beta \cdot (R_{\text{max}})^{(2/2)/2} \cdot (d_{ij})^{(2/2)} / (2 - 2^b)$; |
| 9: | $u := [\log_2 (d_{ij}/R_{\text{min}})]; q := u \mod b; l := \lfloor \sqrt[2]{\frac{2}{2R_{\text{max}}} u} \rfloor \mod b$; |
| 10: | $S_{i} := S_{i} \cup \{e_{ij}\}$; |
| 11: | end for |
| 12: | if $S_{0v} \neq \emptyset$ then |
| 13: | $P_{Sk} := P_{Sk} \cup \{S_{0v}\}$; $t := t + 1$; |
| 14: | end if |
| 15: | end for |
| 16: | return $P_{Sk}$ and $t$. |
equal-sized non-overlapping cones. It is clear that the cells of the same color by a distance of at least 
separate the cells of the same color by a distance of at least 

Algorithm 5. Phase-Scheduler-2 [9]

| Input: | Link set \( M_k \) and time slot index \( t \). |
| Output: | Partial link schedule \( PS_k \) for links in \( M_k \), and \( t \). |
| Phase-Scheduler-2(\( M_k, t \)) | 1: pre-processing (\( M_k \)); 2: Define a large enough constant \( c_1 \); \( PS_{c1} = \emptyset \); \( \xi = 2N_0(a - 1)/(a - 2) \); 3: for \( m = 1 \) to \( \xi \) do \( \log (\xi / \beta) \) do \( \emptyset \); 4: Let \( E_m = \{ e_i \in M_k | y = m \} \); 5: while not all links in \( E_m \) have been scheduled do \( \emptyset \); 6: \( S_i = \emptyset \); 7: for each \( e_i \in E_m \) in decreasing order of \( d_i \) do \( \emptyset \); 8: \( S_i = S_i \cup \{ e_i \} \); \( E_m = E_m - \{ e_i \} \); \( P_i = d_i^2 \cdot \xi / \beta \); 9: end for \( \emptyset \); 10: \( PS_k = PS_{c1} \cup \{ S_i \} \); \( t = t + 1 \); 11: end while \( \emptyset \); 12: end for \( \emptyset \); 13: return \( PS_k \) and \( t \).
| pre-processing (\( M_k \)) | 1: Please refer to [9] for details.
| check (\( e_i, S_i \)) | 1: Please refer to [9] for details.

6. Analysis

In this section, we prove the correctness of our distributed and centralized algorithms, and analyze their efficiency with respect to the bound of aggregation latency.

6.1. Correctness

We first prove that \( \frac{16}{3}X^2 + 12X + 7 \) colors are enough to separate the cells of the same color by a distance of at least \( 2(X + 1)d \), where \( d = 3^k \) is the side length of cells in category \( k \).

Lemma 1. At most \( \frac{16}{3}X^2 + 12X + 7 \) hexagons with size length of \( d \) can cover a disk with radius \( 2(X + 1)d \).

Proof. As shown in Fig. 3, we divide the disk into six equal-sized non-overlapping cones. It is clear that the maximum number of hexagons to cover the disk is at most six times of that to cover each cone.

Take cone \( A \) for instance. We have at most \( \frac{1}{6}d \) hexagons in range of \( \frac{1}{6}d \), \( \frac{1}{6} + 1 \) hexagons in range of \( 2d \), \( \frac{1}{6} + 1 \) hexagons in range of \( \frac{2}{3}d \), etc. So it is not hard to prove by induction that we have at most \( \frac{1}{6}d + \sum_{i = 0}^{\frac{1}{6}d} \) hexagons in range of \( \frac{1}{6}d \) in one cone. So in a range of \( 2(X + 1)d \), for which \( j \leq \frac{2(X + 1)}{6} - 1 \), we have at most \( \frac{16}{3}X^2 + 12X + 7 \) hexagons in one cone, which means at most \( \frac{16}{3}X^2 + 12X + 7 \) in the disk. □

Theorem 1 (Correctness of Cell-AS). The distributed algorithm Cell-AS (Algorithm 1) can construct a data aggregation tree and correctly schedule the transmissions under the physical interference model.

Proof. Algorithm 1 guarantees that each sensor node transmits exactly once and will not serve as a receiver again after the transmission. Hence the resulting transmission links constitute a tree.

The link scheduling guarantees that a node would not transmit and receive at the same time and a non-leaf node transmits only after all the nodes in its subtree have transmitted. We next prove that each transmission is successful under the physical interference model.

In [30], a safe CSMA protocol under the physical interference model is presented. The core idea is to separate each pair of concurrent transmitters by a predefined distance, such that the cumulative interference in the network can be bounded. However, the background noise is not considered in [30]. We revise the conclusion of [30] to adapt their result to the physical interference model in this paper.

We know that any two concurrent transmitters of links in the same category \( k \) are separated by at least \( 2(X + 1)^3d^3 \), where \( X = \left( 6\beta \left( 1 + \frac{2}{\sqrt{3}} \right)^2 \right)^{1/2} \). For any scheduled link with length \( r \), the power assigned for transmission is \( P = N_0 / X^2r^2 \). According to the conclusion of [30], the cumulative interference \( I \) at any receiver of a link in category \( k \) satisfies

\[
I \leq 6 \left( \frac{1}{X} \right)^z \left( 1 + \frac{2}{\sqrt{3}} \right)^z \frac{1}{a - 2} \frac{N_0 / X^2(2 \cdot 3^k)^z}{(2 \cdot 3^k)^z} \\
= 6 \left( 1 + \frac{2}{\sqrt{3}} \right)^z \frac{1}{X^2 - 2} N_0 \beta = N_0 (X^2 - 1).
\]

So the SINR value for any scheduled link with length \( r \) should satisfy

\[
\frac{P/r^2}{N_0 + 1} \geq \frac{N_0 / X^2}{N_0 + N_0 (X^2 - 1)} = \beta.
\]

We can conclude that each link transmission is successful under the physical interference model. □

Theorem 2 (Correctness of NN-AS). The centralized algorithm NN-AS (Algorithm 3) can construct a data aggregation tree and correctly schedule the transmission under the physical interference model.

Proof. The algorithm in Algorithm 3 guarantees that each node will be removed from the node set \( V \) after selected for transmission, and hence it will be a transmitter exactly once. At the end of each round, receivers and other non-scheduled nodes remain in \( V \), and all aggregated data reside in the remaining nodes. Therefore, the generated transmission links correctly construct a data aggregation tree.

For the link scheduling, Algorithm 3 applies either one of the algorithms in [8,9], whose correctness under the physical interference model are proven. □

6.2. Aggregation latency

We now analyze the efficiency of the algorithms. We also derive a theoretically optimal lower bound of the
aggregation latency for the MLAS problem under any interference model and show the approximation ratios of our algorithms with respect to this bound.

6.2.1. Distributed Cell-AS

We now analyze the efficiency of the distributed Cell-AS algorithm.

**Theorem 3 (Aggregation Latency of Cell-AS).** The aggregation latency for the distributed algorithm Cell-AS (Algorithm 1) is upper bounded by \(12 \left(\frac{15}{2}X^2 + 12X + 7\right)\) time slots for cells of side length \(3^k\) \((k > 0)\). Since \(2 \cdot 3^k \geq D\) (the maximum node distance in the network), cells of side length \(3^k\) can cover the entire network. There can be only one cell of this size, and so at most 12 time slots are needed for scheduling its links. In summary, at most \(6 \left(\frac{15}{2}X^2 + 12X + 7\right) + 12 \left(\frac{15}{2}X^2 + 12X + 7\right)(K − 1) + 12 = 12 \left(\frac{15}{2}X^2 + 12X + 7\right)K − 32X^2 − 72X − 29\) time slots are needed to schedule all the transmissions in the data aggregation tree.

One additional time slot is required to transmit the aggregated data to the sink. Therefore the overall aggregation latency is at most \(12 \left(\frac{15}{2}X^2 + 12X + 7\right)\) time slots to transmit the aggregated data to the sink. Therefore the overall aggregation latency is \(O(K)\).

6.2.2. Centralized NN-AS

We first prove a few lemmas before analyzing the efficiency of the centralized NN-AS algorithm.

**Lemma 2.** The data aggregation tree can be constructed with at most \(\lceil \log_2 n \rceil\) rounds in NN-AS.

**Proof.** We give the proof by first showing that each node can be the nearest neighbor of at most six other nodes on a euclidean plane. We prove this claim by contradiction. Fig. 5 gives an example that one node (node 0) can be the nearest neighbor of at most six other nodes on a hexagon grid. An example is given in Fig. 5, with seven nodes in one hexagon of side length \(d = 1\).

From the above result, we know that there can be at most six links transmitting to the head node in each cell of side length \(3^0\). Each cell of side length \(3^k\) with \(k > 0\) covers at most 13 cells of side length \(3^{k−1}\) (an illustration is given in Fig. 2b and c). Therefore, at most six time slots are needed for scheduling the transmissions in a cell of side length \(3^0\), and at most 12 for the cells of side length \(3^k\) \((k > 0)\), to avoid the primary interference.

As we cover cells of the same size with \(\frac{15}{2}X^2 + 12X + 7\) colors, at most \(\frac{15}{2}X^2 + 12X + 7\) rounds are needed to schedule all the cells in the same link length category. Thus at most \(6 \left(\frac{15}{2}X^2 + 12X + 7\right)\) time slots are needed for scheduling all the cells with side length \(3^0\), and
least one node out of these seven nodes will be removed from \( V \) at the end of this round. Therefore at least \( \frac{1}{7} |V| \) nodes are removed from \( V \) in total.

From the above discussion, at most \( \frac{2}{7} |V| \) nodes are left in \( V \) after each round of the algorithm. The algorithm terminates when only one node remains in \( V \). Let \( k \) be the maximum number of rounds which the algorithm executes. We have \( \left\lfloor \frac{6}{7} n \right\rfloor = 1 \), and thus \( k = \left\lfloor \log_2 n \right\rfloor \). \( \square \)

**Lemma 3.** The link scheduling latency in each round of NN-AS is \( O(1) \) with Phase-Scheduler-1 in Algorithm 4 and \( O(\log^2 n) \) with Phase-Scheduler-2 in Algorithm 5.

**Proof.** In each round of NN-AS, the number of links to be scheduled is equal to exactly the number of nodes removed from \( V \), i.e., at least \( \frac{1}{7} |V| \). Meanwhile, as each node can either be the transmitter or the receiver but not both in one round, the number of links to be scheduled is upper bounded by \( \frac{2}{7} |V| \). Since \( |V| \leq n \), we have \( O(n) \) links to schedule in each round. As the link set generated in each round is based on the nearest-neighbor mechanism, we can apply the link scheduling strategy proposed in [8] to schedule them with constantly bounded time slots. On the other hand, the link scheduling algorithm achieves a latency of \( O(\log^2 n) \) with \( n \) links [9]. Therefore, the link scheduling latency in each round of NN-AS is \( O(1) \) with Phase-Scheduler-1 in Algorithm 4 and \( O(\log^2 n) \) with Phase-Scheduler-2 in Algorithm 5. \( \square \)

**Theorem 4** (Aggregation latency of centralized NN-AS). The aggregation latency of the centralized algorithm NN-AS (Algorithm 3) is upper bounded by \( O(\log n) \) with Phase-Scheduler-1 in Algorithm 4 and \( O(\log^3 n) \) with Phase-Scheduler-2 in Algorithm 5.

**Proof.** From Lemmas 2 and 3, we know that NN-AS is executed for at most \( \left\lfloor \log_2 n \right\rfloor \) rounds and the link scheduling latency in each round is \( O(1) \) with Phase-Scheduler-1 in Algorithm 4 and \( O(\log^2 n) \) with Phase-Scheduler-2 in Algorithm 5. In total, NN-AS schedules the data aggregation in \( O\left( \left\lfloor \log_2 n \right\rfloor \right) \) time slots, which is equivalent to \( O(\log n) \), with Phase-Scheduler-1 in Algorithm 4 and \( O\left( \left\lfloor \log_2 n \right\rfloor \right) \) \( \log^2 n \) time slots, which is equivalent to \( O(\log^3 n) \), with Phase-Scheduler-2 in Algorithm 5. \( \square \)

6.2.3. Optimal lower bound

We next derive the optimal lower bound of the aggregation latency, and the approximation ratios of our algorithms with respect to this bound.

**Theorem 5** (Optimal lower bound of aggregation latency). The aggregation latency for the MLAS problem under any interference model is lower bounded by \( \log n \).

**Proof.** Under any interference model, as a node cannot transmit and receive at the same time, at most \( \frac{|V|}{2} \) links can be scheduled for transmission in one time slot. Since each node only transmits exactly once, at most \( \frac{|V|}{2} \) nodes complete their transmissions in one time slot.

Suppose we need \( k \) time slots to aggregate all the data. We have \( \frac{n}{k} = 1 \), and thus \( k = \left\lfloor \log_2 n \right\rfloor \), i.e., the aggregation latency under any interference model is at least \( \log n \). \( \square \)

Comparing to the optimal lower bound, our distributed Cell-AS achieves an approximation ratio of \( O(K/\log n) \), and the centralized NN-AS achieves an approximation ratio of \( O(1) \) with the link scheduling strategy in [8] and \( O(\log^3 n)/\log n \), which is equivalent to \( O(\log^2 n) \), with the link scheduling strategy in [9]. We show in Appendix A that \( O(K) \) is between \( O(\log n) \) and \( O(n) \).

6.3. Comparison with Li et al.’s Algorithm [7]

We next analytically compare our distributed Cell-AS with the distributed algorithm proposed by Li et al. [7], which is the only existing work addressing the MLAS problem under the physical interference model.
Li et al.’s algorithm has four consecutive steps:

- **Topology Center Selection**: the node with the shortest network radius in terms of hop count is chosen as the topology center.
- **Breadth First Spanning (BFS) Tree Construction**: using the topology center as the root, breadth-first searching is executed over the network to build a BFS tree.
- **Connected Dominating Set (CDS) Construction**: a CDS is constructed as the backbone of the aggregation tree with an existing approach [31], based on the BFS tree.
- **Link Scheduling**: the network is divided into grids with side length \( l = \delta r / \sqrt{2} \), where \( 0 < \delta < 1 \) is a configuration parameter assigned before execution, and \( r \) is the maximum achievable transmission range under the physical interference model with constant power assignment \( P \) and \( P_r = \beta \). The grids are colored with \( \left( \frac{4P \beta l^2}{\sqrt{2}} \right)^{1 + \sqrt{2}} \) colors and links are scheduled with respect to grid colors. Here, \( \tau = \frac{2^{1 - 2^{1 - \frac{1}{2}}} + 2^{2^{1 - \frac{1}{2}}}}{2^{1 - \frac{1}{2}}} \).

**Aggregation Latency.** Li et al.’s algorithm solves the MLAS problem in \( O(\Delta + R) \) time slots, where \( R \) is the network radius in hop count and \( \Delta \) is the maximum node degree. In the worst case, either \( R \) or \( \Delta \) can be \( O(n) \), e.g., in the examples in Figs. 7 and 8 to be discussed shortly, and \( R = O(\log n) \) in the best case. Our Cell-AS achieves an aggregation latency of \( O(K) \), which is also equal to \( O(n) \) in the worst case, e.g., in the example in Fig. 9, and \( O(\log n) \) in the best case (see Appendix A). Therefore the two algorithms have the same orders of worst-case and best-case aggregation latencies.

**Computational and Message Complexity.** Both the computational complexity and the message complexity of our Cell-AS algorithm are upper bounded by \( O(\min(Kn, 13^3)) \). Since \( K = n \) in the worst case, both are at most \( O(n^2) \). Li et al.’s algorithm has a computational complexity of \( O(n |E|) \) and message complexity of \( O(n + |E|) \). As \(|E| = n^2\) in the worst case, the computational complexity and
message complexity of Li et al.’s algorithm are $O(n^2)$ and $O(n^2)$, respectively.

We can see that Cell-AS enjoys a better computational complexity while having the same order of message complexity with Li et al.’s algorithm. More details on the analysis of the complexities of our algorithm and Li et al.’s algorithm can be found in Appendix B.

**Case Study.** We next show that Cell-AS can outperform Li et al.’s algorithm in its worst cases. The minimum link length is set to one unit in the following examples, without loss of generality.

**Fig. 7** is a worst case of Li et al.’s algorithm. Nodes are located along the line with distance $r = 1$ between neighboring nodes. The topology center should be the center of the line, which leads to $R = \frac{3}{2}$. According to the latency bound $O(\Delta + R)$, Li et al.’s algorithm takes $O(n)$ time slots to complete data aggregation.

On the other hand, the maximum node distance in Fig. 7 is $n - 1$. Therefore, the link length diversity $K$ with our algorithm should be $\log_3 \frac{n - 1}{2}$. According to the latency bound $O(K)$, the scheduling latency should be $O(\log n)$ with Cell-AS, which is better than $O(n)$.

**Fig. 8** is another worst case for Li et al.’s algorithm, in which all nodes reside on the circle with unit distance between neighboring nodes, except for node 1 in the center. The radius of the circle is $r > 1$. Therefore, node 1 has the maximum node degree $\Delta$ of $n - 1$. With respect to latency bound $O(\Delta + R)$, $O(n)$ time slots are required to complete aggregation with Li et al.’s algorithm.

Meanwhile, the maximum node distance in Fig. 8 is $2r$. Since the distance between any neighboring nodes on the circle is 1, we have $2\pi r \approx n - 1$ with large values of $n$, which is an approximation of the circle’s perimeter. Then the link diversity $K$ should be about $\log_3 \frac{n - 1}{2}$. Therefore, the aggregation latency is $O(\log n)$ with Cell-AS, which is better than $O(n)$ with Li et al.’s algorithm.

**Fig. 7** is a worst case of Li et al.’s algorithm. Nodes are located along the line with distance $r = 1$ between neighboring nodes. The topology center should be the center of the line, which leads to $R = \frac{3}{2}$. According to the latency bound $O(\Delta + R)$, Li et al.’s algorithm takes $O(n)$ time slots to complete data aggregation.

On the other hand, the maximum node distance in Fig. 7 is $n - 1$. Therefore, the link length diversity $K$ with our algorithm should be $\log_3 \frac{n - 1}{2}$. According to the latency bound $O(K)$, the scheduling latency should be $O(\log n)$ with Cell-AS, which is better than $O(n)$.

**Fig. 8** is another worst case for Li et al.’s algorithm, in which all nodes reside on the circle with unit distance between neighboring nodes, except for node 1 in the center. The radius of the circle is $r > 1$. Therefore, node 1 has the maximum node degree $\Delta$ of $n - 1$. With respect to latency bound $O(\Delta + R)$, $O(n)$ time slots are required to complete aggregation with Li et al.’s algorithm.

Meanwhile, the maximum node distance in Fig. 8 is $2r$. Since the distance between any neighboring nodes on the circle is 1, we have $2\pi r \approx n - 1$ with large values of $n$, which is an approximation of the circle’s perimeter. Then the link diversity $K$ should be about $\log_3 \frac{n - 1}{2}$. Therefore, the aggregation latency is $O(\log n)$ with Cell-AS, which is better than $O(n)$ with Li et al.’s algorithm.
Fig. 9 is a worst case example for both Cell-AS and Li et al.'s algorithm. In this example, the maximum node distance is $\frac{n-1}{2}$ between nodes 1 and 2 while the minimum node distance is 1 between nodes 1 and 2. Thus, $K = \log_3 \frac{n-1}{2}$ with Cell-AS. As for Li et al.'s algorithm, $\Delta = n - 1$ since the transmission range should be at least $3^{n-2}$ to maintain connectivity. Both Cell-AS and Li et al.'s algorithm will take $n - 1$ time slots to complete the data aggregation. On the other hand, our centralized NN-AS algorithm can perform better than this and achieve an aggregation latency of $O(\log n)$ or $O(\log^2 n)$ according to Theorem 4.

7. Empirical study

We have implemented our proposed distributed algorithm Cell-AS, centralized algorithm NN-AS, as well as Li et al.'s algorithm, and carried out extensive simulation experiments to verify and compare their efficiency.

It should be noted that the link scheduling algorithm in [8] achieves a worst-case latency bound of $b^2 (18 \log n) = O(\log n)$, where $n$ is the number of nodes and $b$ is a constant integer related to the path-loss-ratio $\alpha$ and the SINR threshold $\beta$. $b^2$ is the number of colors to color the grids that cover the whole network. Since the value of $b$ is too large with any $(\alpha, \beta)$ pairs, the number of required colors inhibits the application of the link scheduling algorithm proposed in [8] in typical networks of limited sizes. As a result, in the empirical study, we only implement the Phase-Scheduler-2 algorithm based on [9] in NN-AS.

In our experiments, three types of sensor network topologies, namely Uniform, Poisson and Cluster, are generated with the number of nodes $n = 100$ to 1000, which are distributed in a square area of 40,000 square meters (200 meters times 200 meters). The nodes are uniformly randomly distributed in the Uniform topologies, and are distributed with the Poisson distribution in the Poisson topologies. In the Cluster topologies [32], the centers of $n_c$ clusters are uniformly randomly located in the square and, for each cluster, $\frac{\Delta}{n_c}$
nodes are uniformly randomly distributed within a disk of radius $r_c$ at the center. We use the same settings as in [32] in our experiments, where $n_c = 10$ and $r_c = 20$. We set $N_0$ to the same constant value 0.1 as in [7] (which nevertheless would not affect the aggregation latency). The transmission power in our implementation of Li et al.’s algorithm is assigned the value that would result in a transmission range of 40 to maintain the connectivity of the respective network with high probability, while $\delta$ is set to 0.6 in compliance with the simulation settings in [7]. Since $2 < \alpha < 6$ and $\beta \geq 1$, we experiment with $\alpha$ set to 3, 4 and 5, and $\beta$ to values between 2 and 20, respectively. We implement the three algorithms in C++ and run the programs on a Solaris server with an 8-core CPU (2.6GHZ) and 8G RAM. All our results presented are the average of 1000 trials.

We first compare the aggregation latency of the three algorithms with different combinations of $\alpha$ and $\beta$ values in the three types of topologies. The results are presented in Figs. 10–12, respectively.

Fig. 10 shows that the aggregation latency with Cell-AS is larger with smaller $\alpha$, which represents less path loss of power and thus larger interference from neighbor nodes, and with larger $\beta$, corresponding to higher SINR requirement. We however observe in Fig. 11 that, with NN-AS, the latency curves tend to overlap under the same node distribution even when values of $\alpha$ and $\beta$ vary, but they show marked differences with different node distributions. This shows that network topology is the dominant influential factor in aggregation latency for NN-AS, which can be explained by the algorithm’s nearest-neighbor mechanism in tree construction and non-linear power assignment [9] for link scheduling.

For Li et al.’s algorithm, Fig. 12 shows that most of the curves produced at different $\beta$ values are straight or nearly straight lines that overlap, except in the following cases with Uniform topologies: $\beta = 2$ when $\alpha = 4$; $\beta = 2$, $\beta = 4$ and $\beta = 6$ when $\alpha = 5$. The reason behind the linearity of the lines is that each grid is scheduled one by one without any concurrency with Li et al.’s algorithm in the cases of the Poisson and Cluster topologies, as well as the Uniform topologies with smaller $\alpha$ and larger $\beta$ values. The no-concurrency phenomenon is due to the fact that since the number of colors is \[
\left(\frac{4\beta(1+\beta)}{(\sqrt{2}+\beta)^{\alpha+1}}\right)^{\frac{1}{\alpha}} + 1 + \sqrt{2}
n\] with $l = \delta r / \sqrt{2}$, $\tau = \frac{\alpha(1+\beta)}{\sqrt{\alpha}} + \frac{\alpha^2}{2\beta + \alpha}$ and $\frac{\alpha^2}{\alpha} = \beta$ (see Section 6.3 for detailed discussion of Li et al.’s algorithm), smaller $\alpha$ and larger $\beta$ values lead to a larger number of colors needed. On the other hand, in the Poisson and Cluster topologies, the nodes are not evenly distributed, thus a larger $r$ is requested to maintain the network connectivity, which leads to a smaller number of grids since the side length of each grid is $\delta r / \sqrt{2}$. In these cases, the number of required colors in the algorithm, as decided by $\alpha$ and $\beta$, is larger than the total number of grids in the network (which is proportional to $1/r$). Therefore, each grid is actually scheduled one by one. In comparison, the number of cells in our Cell-AS algorithm is only related to the link length

![Fig. 13. Aggregation latency comparison of the three algorithms in selected network settings.](image-url)
diversity, but not $r$. Therefore, our algorithm can execute with much more concurrency in link scheduling across different cells, leading to the sublinear curves in Fig. 10.

Figs. 10–12 show that concurrent link scheduling (across different cells/grids) occurs with all three algorithms only in the following four cases in the Uniform topologies: (1) $a = 4$, $b = 2$; (2) $a = 5$, $b = 2$; (3) $a = 5$, $b = 4$; and (4) $a = 5$, $b = 6$. We next compare the aggregation latencies achieved by the three algorithms in these four cases. Fig. 13 shows that our centralized NN-AS achieves a much lower aggregation latency as compared to the other two algorithms, such that the changes in its curves are almost unobservable. The performance of our distributed Cell-AS is similar to that of Li et al.’s algorithm when $n \leq 200$, but is up to 35% better than the latter when the network becomes larger.

To obtain a better understanding of the asymptotic performance of each algorithm, we further divide the aggregation latency in Fig. 13 by $\log^2 n$, $\log^5 n$, and $\log^6 n$, respectively, and plot the results in Fig. 14 (since the curves are similar in all four cases, we show the results obtained at $a = 4$ and $b = 2$ as being representative). Our rationale is that, if the aggregation latency of an algorithm has a higher (lower) order than $O(\log n)$, its curve in the respective plot should go up (down) with the increase of the network size, and a relatively flat curve would indicate that the aggregation latency is $O(\log n)$. From Fig. 14a and d, we infer that the average aggregation latency of NN-AS and Li et al.’s algorithm is $O(\log^2 n)$ and $O(\log^7 n)$, respectively. The curves corresponding to the Cell-AS algorithm slightly go up in Fig. 14b and slightly go down in Fig. 14c, indicating that Cell-AS achieves an average aggregation latency between $O(\log^5 n)$ and $O(\log^6 n)$.

Our analysis in Section 6 gives an aggregation latency upper bound of $O(K)$ for Cell-AS and $O(\log^3 n)$ for NN-AS with the link scheduling strategy in [9]. Our experiments have shown that the average aggregation latency under practical settings is better in the Uniform topologies with the algorithms.

8. Concluding remarks

This paper tackles the minimum-latency aggregation scheduling problem under the physical interference model. Many results for the MLAS problem under the protocol interference model have been obtained in recent years, but they are not as relevant to real networks as any solution under the physical interference model which is much closer to the physical reality. The physical interference model is favored also because of its potential in enhancing the network capacity when the model is adopted in a design [12,13,9,14,15]. Although the physical interference model makes finding a distributed solution difficult, we propose a distributed algorithm to solve the problem in networks of arbitrary topologies. By strategically dividing the network into cells according to the link length diversity ($K$), the algorithm obviates the need for global information and can be
implemented in a fully distributed fashion. We also present a centralized algorithm which represents the current most efficient algorithm for the problem, as well as prove an optimal lower bound on the aggregation latency for the MLAS problem under any interference model. Our analysis shows that the proposed distributed algorithm can aggregate all the data in $O(K)$ time slots (with approximation ratio $O(K / \log n)$ with respect to the optimal lower bound), and the centralized algorithm in at most $O(\log n)$ time slots (with approximation ratio $O(1)$, and using the link scheduling strategy in [8]) and $O(\log^2 n)$ time slots (with approximation ratio $O(\log^2 n)$, and using the link scheduling strategy in [9]).

Our empirical studies under realistic settings further demonstrate that both Cell-AS and NN-AS (using the link scheduling strategy in [9]) outperform Li et al.’s algorithm in all three topologies tested. Furthermore, the Cell-AS and NN-AS algorithms (using the link scheduling strategy in [9]) can potentially achieve an average aggregation latency of between $O(\log^2 n)$ and $O(\log^6 n)$, and $O(\log^2 n)$ in practice, respectively.

In our future work, we will investigate further reduction of the theoretical upper bound on the aggregation latency with distributed implementations and study the latency-energy tradeoff in data aggregation, e.g., the achievable asymptotic order of aggregation latency with constraint transmission power in each time slot.

**Acknowledgements**

This project is partially supported by Hong Kong RGC GRF Grants 714009E and 714311, and National Natural Science Foundation of China Grant 61103186.

**Appendix A. Analysis of the range of $K$**

Fig. 9 is a worst case example for Cell-AS. The minimum geometric node distance is 1 and the maximum geometric node distance is $\sum_{i=0}^{n-2} (3^{i-1} - 1)/2$. So $K = \log_3 3^{n-1} - 1$, which is $O(n)$ in the worst case.

Recall the existing result from [3]: suppose the entire network is a disk of radius $r = 3^k$, and the node set $V$ is a set of points with mutual distances at least 1; then we have

$$n \leq \frac{2\pi}{\sqrt{3}} r^2 + \pi r + 1 \Rightarrow n \leq \frac{2\pi}{\sqrt{3}} (3^k)^2 + \pi 3^k + 1$$

$$K \geq \log_3 \left( \frac{\sqrt{3}}{4\pi} \left( \sqrt{\pi^2 + \frac{8\pi}{\sqrt{3}} (n - 1) - \pi} \right) \right) = O(\log \sqrt{n}).$$

Since the aggregation latency lower bound is $O(\log n)$ by Theorem 5, $K$ is $O(\log n)$ in the best case instead of $O(\log \sqrt{n})$ (otherwise, the aggregation latency with Cell-AS is $O(K) = O(\log \sqrt{n})$, which contradicts with Theorem 5).

**Appendix B. More on the computational and message complexity of Cell-AS and Li et al.’s algorithm**

**B.1. Computational complexity**

Cell-AS has three main function modules, i.e., neighbor discovery, head selection, and link scheduling. During neighbor discovery in each round, each node performs exactly one local broadcast. There are $n$ nodes in round 0 and at most $\min(n, 13^{K-k+1})$ nodes in round $k > 0$. So at most $n + \sum_{k=1}^{n} \min(n, 13^{K-k+1}) = \min((K + 1)n, n + 13^{(K-1)n-1})$ local broadcast operations are involved in $K + 1$ rounds. For head selection, the total numbers of location comparisons to decide the heads in round 0 and in round $k > 0$ are at most $7n$ and $\min(13Kn, \sum_{s=1}^{K} 13^{K-s+1})$, respectively, as there are at most seven nodes in each cell in round 0, and 13 per cell in round $k > 0$. Hence the overall computational complexity for head selection throughout the algorithm is at most $7n + \min(13Kn, \sum_{s=1}^{K} 13^{K-s+1})$. Similarly, link scheduling also has a computational complexity of $7n + \min(13Kn, \sum_{s=1}^{K} 13^{K-s+1})$. In summary, Cell-AS has an overall computational complexity of $O(\min(13^Kn))$.

Li et al.’s algorithm is divided into four phases, i.e., topology center selection, breadth-first spanning (BFS) tree construction, connected dominating set (CDS) construction, and link scheduling. For topology center selection, the node with the shortest network radius in terms of hop count is chosen as the topology center. If the classical Bellman-Ford algorithm is applied to derive the routing matrix, the complexity for this phase is $O(|V||E|)$. For BFS tree construction, the complexity is $O(|V| + |E|)$. The CDS construction phase also has a complexity of $O(|V| + |E|)$. Their link scheduling phase consists of an outer iteration on the nodes and an inner iteration on the colors. Let the number of colors be $\gamma$; the computational complexity in this phase is $O(\gamma |V|)$. In summary, Li et al.’s algorithm requires a computational complexity of $O(|V||E|)$.

**B.2. Message complexity**

Cell-AS: During both the neighbor discovery and the link scheduling phase, $n$ nodes in round 0 and at most $\min(n, 13^{K-k+1})$ nodes in round $k$ send messages to their neighbors. Thus, the message complexity of either of these two functions is $\min((K + 1)n, n + 13^{(K-1)n-1})$. As head selection is conducted based on neighbor location information obtained during neighbor discovery, its message complexity is 0. Hence Cell-AS requires an overall message complexity of $O(\min(13^Kn))$.

Li et al.’s algorithm: The message complexities for topology center selection, BFS tree construction, and CDS construction all are $O(|V| + |E|)$. We are unable to analyze the message complexity of the link scheduling phase, as no implementation details are given in the paper [7].

**References**


