A Delay-Efficient Algorithm for Data Aggregation in Multihop Wireless Sensor Networks

Xiaohua Xu, *Student Member*, *IEEE*, Xiang-Yang Li, *Senior Member*, *IEEE*, Xufei Mao, *Student Member*, *IEEE*, Shaojie Tang, *Student Member*, *IEEE*, and Shiguang Wang, *Student Member*, *IEEE*

Abstract—Data aggregation is a key functionality in wireless sensor networks (WSNs). This paper focuses on data aggregation scheduling problem to minimize the delay (or latency). We propose an efficient distributed algorithm that produces a collision-free schedule for data aggregation in WSNs. We theoretically prove that the delay of the aggregation schedule generated by our algorithm is at most $16R + \Delta - 14$ time slots. Here, R is the network radius and Δ is the maximum node degree in the communication graph of the original network. Our algorithm significantly improves the previously known best data aggregation algorithm with an upper bound of delay of $24D + 6\Delta + 16$ time slots, where D is the network diameter (note that D can be as large as $2R$). We conduct extensive simulations to study the practical performances of our proposed data aggregation algorithm. Our simulation results corroborate our theoretical results and show that our algorithms perform better in practice. We prove that the overall lower bound of delay for data aggregation under any interference model is $\max\{\log n, R\}$, where n is the network size. We provide an example to show that the lower bound is (approximately) tight under the protocol interference model when $r_I = r$, where r_I is the interference range and r is the transmission range. We also derive the lower bound of delay under the protocol interference model when $r < r_I < 3r$ and $r_I \geq 3r$.

Index Terms—Wireless networks, aggregation, scheduling, delay, sensor.

1 Introduction

Wireless sensor networks (WSNs) have drawn a considerable amount of research interest for their omnipresent applications such as environmental monitoring, spatial exploration, and battlefield surveillance. To design and deploy successful WSNs, many issues need to be resolved such as deployment strategies, energy conservation, routing in dynamic environments, localization, and so on. All the issues essentially correlate to collecting data from a set of targeted wireless sensors to some sink node(s) and then performing some further analysis at sink node(s), which can be termed as many-to-one communication. In-network data aggregation [16] is one of the most common many-to-one communication patterns used in these sensor networks; thus, it becomes a key field in WSNs and has been well studied in recent years.

We consider the problem of designing a schedule for data aggregation from within networks to sink node(s) with minimum time slot delay. Some of previous research works on in-network aggregation did not consider the collision problem and left it to the MAC layer. Resolving collisions in MAC layer could incur a large amount of energy consumption and a large delay during aggregation. Thus, in this paper, we mainly concentrate on the TDMA scheduling problem above the MAC layer. To define the problem formally, consider a WSN G formed by n wireless nodes $V = \{v_1, \ldots, v_n\}$ deployed in a two-dimensional region. $v_s \in V$ is the sink node that will collect the final aggregation result. Each node v_i has a transmission range r and interference range $r_I = \Theta(r)$. A node v_i can send data correctly to another node v_j if and only if 1) v_j is within v_i's transmission range and (2) v_j is not within interference range r_I of any other transmitting node. Every node v_i has an ability to monitor the environment, and collect some data (such as temperature), i.e., v_i has a set of raw data A_i. Let $A = \cup_{i=1}^{n} A_i$ and $N = |A|$ be the cardinality of the set A. Then $\langle A_1, A_2, \ldots, A_i, \ldots, A_n \rangle$ is called a distribution of A at sites of V. Data aggregation is to find the value $f(A)$ at the sink node v_s for a certain function f, such as *min*, *max*, *average*, *variance*, and so on with minimum time delay.

The data aggregation scheduling problems have been extensively studied recently. Huang et al. [11] proposed a centralized scheduling algorithm with the delay bound of $23R + \Delta + 18$ time slots, where R is the network radius and Δ is the maximum node degree. However, the interference model used in [11] is a simple primary interference model: no node can send and receive simultaneously. Under the Protocol Interference Model, Yu et al. [3] proposed a distributed scheduling method generating collision-free schedules with delay at most $24D + 6\Delta + 16$ time slots, where D is the network diameter.

The main contributions of this paper are as follows: We propose efficient algorithms that will construct a data aggregation tree and a TDMA schedule for all links in the tree such that the delay of aggregating all data to the sink node is approximately minimized. For simplicity of

• *The authors are with the Department of Computer Science, Illinois Institute of Technology, 10 West 31st ST, Chicago, IL 60616. E-mail: {xxu23, swang44, xmao3, stang7}@iit.edu, xli@cs.iit.edu.*

Manuscript received 26 Feb. 2009; revised 5 June 2009; accepted 4 Sept. 2009; published online 5 Apr. 2010.
Recommended for acceptance by S. Olariu.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2009-02-0089.
Digital Object Identifier no. 10.1109/TPDS.2010.80.

analysis, we use the protocol interference model and assume that $r_I = r$. As an illustration, we first present an efficient centralized algorithm that will build a TDMA schedule of nodes based on the aggregation tree, which is build distributively. Our schedule uses a bottom-up approach: schedule nodes level by level starting from the lowest level. We theoretically prove that the delay of the aggregation schedule generated by our algorithm is at most $16R + \Delta - 14$ time slots. Note that for general r_I, our algorithm will produce a collision-free schedule for aggregation whose delay is at most $\Theta((\frac{r_I}{r})^2 R + \Delta)$ time slots. We then present an efficient distributed algorithm that builds an aggregation tree and gives a schedule for each node. For simplicity, our distributed method assumes that the clocks of all nodes are synchronized. Unlike our centralized algorithm, our distributed algorithm will *not* explicitly produce a schedule for nodes in the aggregation tree. The schedule for nodes is implicitly generated in the process of data aggregation. Our distributed scheduling algorithm thus works well in dynamic networks, as long as the constructed backbone of the network by our algorithm remains unchanged. Obviously, when $r_I = r$, for a network G with radius R and the maximum node degree Δ, the delay by *any* data aggregation algorithm is at least R. This implies that our algorithm is within a small constant factor of the optimum. We then conduct extensive simulations to study the practical performance of our proposed data aggregation method. The simulation results corroborate our theoretical results and show that our method performs better in practice. We find that data aggregation by our distributed method has delay close to R. Besides, we prove that the overall lower bound of delay for data aggregation under any interference model is $\max\{\log n, R\}$. We provide an example to show that the lower bound is (approximately) tight under the protocol interference model when $r_I = r$. We also analyze the lower bound of delay under the protocol interference model when $r < r_I < 3r$ and $r_I \geq 3r$.

The rest of the paper is organized as follows: Section 2 formulates the problem. We present our centralized and distributed scheduling algorithms in Section 3 and analyze their performance and prove the overall lower bound in Section 4. Section 5 discusses the results in other interferences models. Section 6 presents the simulation results. Section 7 outlines the related work. Finally, Section 8 concludes the paper.

2 System Models

2.1 Network Model

We consider a WSN consisting of n nodes V, where $v_s \in V$ is the sink node. Each node can send (receive) data to (from) all directions. For simplicity, we assume that all nodes have the same transmission range r such that two nodes u and v form a communication link whenever their euclidean distance $\|u - v\| \leq r$. In the rest of the paper, we will assume that $r = 1$, i.e., normalized to one unit. Then the underlying communication graph is essentially a unit disk graph (UDG).

Let $A, B \subset V$ and $A \cap B = \emptyset$. We say that data are aggregated from A to B in one time slot if all the nodes in A transmit data simultaneously in one time slot and all data are received by some nodes in B without interference. We will define interference at the end of this section. Then a data aggregation schedule with delay l can be defined as a sequence of sender sets $S_1, S_2, \ldots, S_l$ satisfying the following conditions:

1. $S_i \cap S_j = \emptyset$, $\forall i \neq j$;
2. $\cup_{i=1}^{l} S_i = V \setminus \{v_s\}$;
3. Data are aggregated from S_k to $V \setminus \cup_{i=1}^{k} S_i$ at time slot k, for all $k = 1, 2, \ldots, l$, and all the data are aggregated to the sink node v_s in l time slots.

Note that here $\cup_{i=1}^{l} S_i = V \setminus \{v_s\}$ is to ensure that every datum will be aggregated; $S_i \cap S_j = \emptyset, \forall i \neq j$ is to ensure that every datum is used at most once. To simplify our analysis, we will relax the requirement that $S_i \cap S_j = \emptyset, \forall i \neq j$. When the sets S_i, $1 \leq i \leq l$ are not disjoint, in the actual data aggregation, a node v, which appears multiple times in S_i, $1 \leq i \leq l$, will participate in the data aggregation only once (say, the smallest i when it appears in S_i), and then it will only serve as a relay node in the following appearances.

The distributed aggregation scheduling problem is to find a schedule $S_1, S_2, \ldots, S_l$ in a distributed way such that l is minimized. This problem is proved to be NP-hard in [4]. This paper proposes an approximate distributed algorithm with delay $16R + \Delta - 14$ time slots, where R is the network radius and Δ is the maximum node degree.

Interference model. We assume that a node cannot send and receive data simultaneously. In the protocol interference model [9], we assume that each node has a transmission range r and an interference range $r_I \geq r$. A receiver v of a link uv is interfered by another sender p of a link pq if $\|p - v\| \leq r_I$. As [4], [11], we first assume that $r_I = r$, which is scaled to 1. We will later study the more general case $r_I \geq r$.

2.2 Related Terminology

For simplicity, we present our distributed methods in a synchronous message passing model in which time is divided into slots. In each time slot, a node is able to send a message to one of its neighbors. Note that at the cost of higher communication, our methods can be implemented in asynchronous communication settings using the notions of synchronizer.

In a graph $G = (V, E)$, a subset S of V is a *dominating set* (DS) if for each node u in V, it is either in S or is adjacent to some node v in S. Nodes from S are called dominators, whereas nodes not in S are called dominatees. A subset C of V is a connected dominating set (CDS) if C is a dominating set and C induces a connected subgraph. The dominatees in C are also called *connectors*. Consequently, the nodes in C can communicate with each other without using nodes in $V \setminus C$. A CDS is also called a backbone here.

3 Distributed Aggregation Scheduling

Our Improved data Aggregation Scheduling (IAS) algorithm consists of two phases: 1) aggregation tree construction and 2) aggregation scheduling. As an illustration of our methods, we first present a centralized version of our data aggregation scheduling. We adopt an existing method for

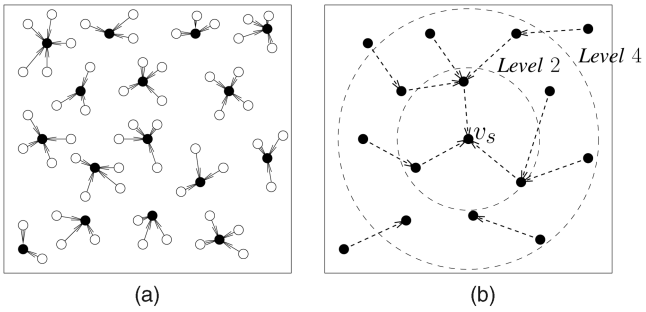


Fig. 1. The overall approach: the black nodes are dominators and white nodes are dominatees. (a) Phase I. (b) Phase II.

the first phase and the second phase is the core of our algorithm. We will present these two phases in the following two sections. At the end of the section, we present a distributed implementation based on our centralized aggregation scheduling algorithm.

3.1 The Overall Approach

In this section, we describe our overall approach for data aggregation. As a prestep, we will construct a CDS as a backbone for the network. Then, our aggregation scheduling algorithm can be divided into two phases as follows:

Phase I. Every dominator aggregates the data from all its dominatees (as shown in Fig. 1a).

Phase II. Dominators aggregate their data to the sink node v_s level by level (as shown in Fig. 1b).

For each level in the second phase, the process can be further divided into two subphases:

- All the dominators aggregate their data to its corresponding connectors.
- All the connectors transmit their data to the dominators in the upper level.

3.2 Dominating Set Construction

As our algorithm is aggregation-tree-based, in the first phase, we construct an aggregation tree in a distributed way using an existing method [19]. We employ a CDS in this phase since it can behave as the virtual backbone of a sensor network. A distributed method of constructing a CDS has been proposed by Wan et al. [19]. In their algorithm, a special dominating set is constructed first and then a CDS is constructed to connect dominators and the other nodes. This CDS tree can be used as the aggregation tree in our scheduling algorithm with a small modification as follows:

1. We choose the *topology center* of the UDG as the root of our BFS tree. Note that previous methods have used the sink node as the root. Our choice of the topology center enables us to reduce the delay to a function of the network radius R, instead of the network diameter D proved by previous methods. Here, a node v_0 is called the *topology center* in a graph G if $v_0 = \arg\min_v\{\max_u d_G(u,v)\}$, where $d_G(u,v)$ is the hop distance between nodes u and v in graph G. $R = \max_u d_G(u,v_0)$ is called the *radius* of the network G. Note that in most networks, the topology center is different from the sink node.
2. After the topology center gathered the aggregated data from all nodes, it will then send the aggregation result to the sink node via the shortest path from the topology center v_0 to the sink node v_s. This will incur an additional delay $d_G(v_0,v_s)$ of at most R.

Algorithms 1 and 2 briefly review the methods for selecting a dominating set and a CDS in [19]. In Algorithm 1, the *rank* of a node u is $(level, ID(u))$, where *level* is the hop distance of u to the root. The ranks of nodes are compared using lexicographic order. After execution of Algorithm 2, all black nodes form a dominating set. For each gray node, either it is a leaf or its children in the aggregation tree are black nodes. In the second case, a gray node plays the role of connecting two black nodes. The root is a node in the dominating set (a black node) and all its neighbors in G are its children in BFS.

Algorithm 1. Distributed Dominators Selection [19]

1: Determine the topology center of the UDG as v_0;
2: Construct a BFS (breadth-first-search) tree rooted at v_0 with height R, the radius of the original network;
3: Every node colors itself white;
4: Root node v_0 changes its color to black and broadcasts a message; BLACK to its one-hop neighbors in G;
5: **for** each white node u received a message BLACK **do**
6: u colors itself gray and broadcasts a message GRAY to its one-hop neighbors in G;
7: **if** a white node w receives GRAY from all its lower ranked neighbors **then**
8: w colors itself as black and sends message BLACK to all its one-hop neighbors in G;
9: All black nodes form a dominating set.

Algorithm 2. Distributed Construction of Aggregation Tree T

1: Select a set of dominators as in Algorithm 1;
2: Root node v_0 sends a message GRAY-JOIN to its one-hop neighbors in G;
3: **if** an unmarked gray node not in T received a message GRAY-JOIN **then**
4: Join T with the sender as its parent;
5: Send a message BLACK-JOIN to its one-hop neighbors;
6: Mark itself;
7: **if** an unmarked black node not in T received message BLACK-JOIN **then**
8: Join T with the sender as its parent;
9: Send a message GRAY-JOIN to its one-hop neighbors;
10: Mark itself;
11: Return T.

3.3 Centralized Algorithm

The second phase is aggregation scheduling, which is the core of the whole algorithm. It is based on the aggregation tree constructed in the first phase. As an illustration, we first present an efficient centralized algorithm. We will then present our distributed scheduling implementation in Section 3.4.

Algorithm 3 shows how the data from the dominatees are aggregated to the dominators. At every time slot, the set

of dominators will gather data from as many dominatees (whose data have not been gathered to a dominator yet) as possible. Note that since the maximum degree of nodes in the communication graph is Δ, our method guarantees that after at most Δ time slots, all the dominatees' data can be gathered to their corresponding dominators successfully without interferences, which will be proved in Lemma 4. The basic idea is as follows: each dominator will randomly pick a dominatee whose data are not reported to any dominator yet. Clearly, these selected dominatees may not be able to send their data to corresponding dominators in one time slot due to potential interferences. We then reconnect these dominatees to the dominators (and may not schedule some of the selected dominatees in the current time slot), using Algorithm 4, such that these new links can communicate concurrently.

Algorithm 3. Aggregate Data to Dominators

1: **for** $i = 1, 2, \ldots, \Delta$ **do**
2: Each dominator randomly chooses a neighboring dominatee, whose data are not gathered yet, as transmitter. The set of such chosen links form a link set L.
3: Apply Algorithm 4 to L, assume the output link set is S;
4: All the output links in S now transmit simultaneously;
5: $i = i + 1$;

Algorithm 4. Reconnect Dominatees to Dominators

Input: a set of links L;
Output: a set of conflict-free links S;
1: $S = L$;
2: **while** (exist a pair of conflicting links in S) **do**
3: Let u_iz_i and u_jz_j be one of the pairs of conflicting links.
4: Find the sets D_i and D_j based on their definitions;
5: **if** ($|u_iz_j| \leq 1$ and $|u_jz_i| > 1$) **then**
6: If $D_j = \emptyset$, remove the link u_jz_j.
7: If $D_j \neq \emptyset$, replace u_jz_j by a link $u_jz_{j_0}$, for a random $z_{j_0} \in D_j$.
8: **else if** ($|u_jz_i| \leq 1$ and $|u_iz_j| > 1$) **then**
9: If $D_i = \emptyset$, remove link u_iz_i.
10: If $D_i \neq \emptyset$, replace u_iz_i with $u_iz_{i_0}$, for a random $z_{i_0} \in D_i$.
11: **else if** ($|u_jz_i| \leq 1$ and $|u_iz_j| \leq 1$) **then**
12: If $D_i = \emptyset$, remove the link u_iz_i; else if $D_j = \emptyset$, remove the link u_jz_j.
13: If $D_i \neq \emptyset \wedge D_j \neq \emptyset$, replace u_iz_i and u_jz_j by two new links $u_iz_{i_0}$, $u_jz_{j_0}$, for a random $z_{i_0} \in D_i$ and a random $z_{j_0} \in D_j$.

Suppose that two directed links u_iz_i and u_jz_j interfere with each other (see Fig. 2a), where the dominatees u_i and u_j are transmitters in these two links, respectively, and z_i and z_j are dominators. For each dominatee v, let $D(v)$ be the set of neighboring dominators. Obviously, $|D(v)| \leq 5$ for any node v. Let $D(u_i) = D(u_i) \setminus \{z_i\}$, $D(u_j) = D(u_j) \setminus \{z_j\}$. Note that here $D(u_i)$ and $D(u_j)$ may be empty, or $D(u_i) \cap D(u_j)$ may not be empty.

For every other active transmitter v, $v \neq u_i$ and $v \neq u_j$, we delete all dominators from $D(u_i)$ (and also from $D(u_j)$) that

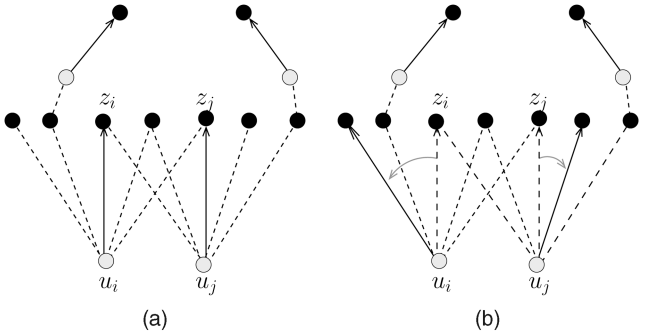


Fig. 2. (a) An interference between two links. The dashed line means that the endpoints are within interference ranges of each other. (b) A state after rescheduling.

are within the transmission range of v. Note that we can discard these dominators since their degrees are already decreased by at least 1 because of the existence of some active transmitter v. We also delete the dominators that are within the transmission range of both u_i and u_j from $D(u_i)$ and $D(u_j)$. Note that we can do this because these dominators' degree will be decreased by 1 since our rescheduling can guarantee at least one transmitter of u_i and u_j will remain as an active transmitter, as we will show later.

Let D_i (resp. D_j) be the set of remaining dominators in $D(u_i)$ (resp. $D(u_j)$).

Fig. 2b illustrates one possible state after the preceding two deletions of dominators from $D(u_i)$ and $D(u_j)$. Note that

1. The distance between u_i and any member of D_j is greater than 1. The distance between u_j and any member of D_i is greater than 1.
2. It is possible that D_i or D_j or both could be empty.

Algorithm 4 shows how to reconnect dominatees to dominators to avoid the interference.

After all the data in the dominatees have been aggregated to dominators, our next step is to aggregate all the intermediate results in the dominators to the root.

We can see that in each layer of the BFS tree, there are some dominator(s) and some dominatee(s). For every dominatee, it has at least one dominator neighbor in the same or upper level. Thus, every dominator (except the root) has at least one dominator in the upper level within two hops. Using this property, we can ensure that all the data in the dominators can reach the root finally if every dominator transmits its data to some dominator in upper level within two hops. From another point of view, considering dominators in the decreasing order of their levels, a dominator u in level L aggregates data from all dominators in level $L + 1$ or $L + 2$ that are within two hops of u. This will ensure that all the data will be aggregated to the root. Algorithm 5 presents our method in detail.

Algorithm 5. Centralized-IAS

Input: BFS tree with root v_0 and depth R, and a distributive aggregation function f (Please see the definition of aggregation function in [23]), data A_i stored at each node v_i.

1: Construct the aggregation tree T' using Algorithm 2. Remove the redundant connectors to ensure that each dominator uses at most 12 connectors to connect itself

to all dominators in lower level and is within two-hops. Here a connector node x (a dominatee of a dominator u) is said to be *redundant* for the dominator u, if removing x will *not* disconnect any of the two-hop dominators of u from u.

Let T be the final data aggregation tree.

2: **for** $i = R-1, R-2, \ldots, 0$ **do**
3: Choose all dominators, denoted by B_i, in level i of the BFS tree.
4: **for** every dominator $u \in B_i$ **do**
5: Find the set $D_2(u)$ of unmarked dominators that are within two-hops of u in BFS, and in lower level $i+1$ or $i+2$.
6: Mark all nodes in $D_2(u)$.
7: Every node w in $D_2(u)$ sends $f(A_w, X_1, X_2, \ldots, X_d)$ to the parent node (a connector node) in T. Here A_w is the original data set node w has, and X_1, X_2, $\ldots$, X_d are data that node w received from its d children nodes in T.
8: Every node z that is a parent of some nodes in $D_2(u)$ sends $f(X_1, X_2, \ldots, X_p)$ to node u (which is the parent of z in T). Here X_1, X_2, $\ldots$, X_p are data that node z received from its p children nodes in T.
9: $i = i - 1$
10: The root v_0 sends the result to the sink using the shortest path.

In Algorithm 5, we only concentrate on communications between dominators. The algorithm runs from lower level to upper level in aggregation tree, every dominator will remain silent until the level where it locates begins running. When it is its turn, the dominator will try to gather all the data from other dominators in lower levels that have not been aggregated. If a dominator's data have been collected before, then it is unnecessary to be collected again. Actually, we have to guarantee that every datum should be and only be used once. Our algorithm implements this by discarding the dominators after their data have been gathered to upper levels.

Note that in our algorithm after we process dominators B_i (all dominators in level i), there may still have some dominators in B_{i+1} whose data are not aggregated. This could happen because a dominator in B_{i+1} could be within two hops of some dominator in B_{i-1}, but not within two hops of *any* dominator from B_i. We conclude that after the execution of all the dominators in B_i , the data from all dominators in B_{i+2} have already been aggregated.

3.4 Distributed Implementation

Now we present a distributed implementation for our data aggregation scheduling. The distributed implementation consists of three stages:

1. Every dominatee transmits its data to the neighboring dominator with the lowest level.
2. Data are aggregated from dominators in lower levels to dominators in upper levels, and finally, to the root of the aggregation tree, which is the topology center of the network.
3. Topology center then transmits the aggregated data to the original sink via the shortest path.

The distributed implementation differs from the centralized one in that the distributed one seeks to transmit greedily: we will try to allocate a node v a time slot to transmit whenever v has collected the aggregated data from all its children nodes in the data aggregation tree T. Thus, the first two phases may interleave in our distributed implementation. The interleaving will reduce the delay greatly since it increases the number of simultaneous transmissions. Later, we will provide the simulation result of our distributed method, which shows that our distributed implementation is quite close to $(1+\varepsilon)R + \Delta + \Theta(1)$, where ε is a small positive constant. Therefore, we conjecture that the data aggregation delay by our distributed implementation indeed has a theoretical performance guarantee of $(1+\varepsilon)R + \Delta + \Theta(1)$. It will be interesting if we can prove or disprove this conjecture, which is left as future work.

To run our algorithm, every node v_i should maintain some local variables, which are as follows:

1. Leaf indicator: $\mathrm{Leaf}[i] \in \{0, 1\}$, to indicate whether the node v_i is a leaf node in the data aggregation tree.
2. Competitor Set: $\mathrm{CS}[i]$, the set of nodes such that for each $j \in \mathrm{CS}[i]$, nodes v_i and v_j cannot transmit simultaneously to their parents due to interference. In other words, if $j \in \mathrm{CS}[i]$, we have either the parent $p_T(i)$ of node v_i in the data aggregation tree T is within the interference range of node v_j; or the parent $p_T(j)$ of node v_j in the data aggregation tree T is within the interference range of node v_i; or both. Note that under the interference model studied in this paper, each node in $\mathrm{CS}[i]$ is within a small constant number of hops of i.
3. Ready Competitor Set: $\mathrm{RdyCS}[i]$, which is the set of nodes that collides with i and it is ready to send data to its parent, i.e., it has received the data from all its children nodes.
4. Time Slot to Transmit: $\mathrm{TST}[i]$, which is the assigned time slot that node v_i indeed sends its data to its parent.
5. Number of Children: $\mathrm{NoC}[i]$, which is the number of children nodes of v_i in the data aggregation tree T.

Observe that here, at some time, if we let Rdy be the set of nodes which are ready to transmit (i.e., $v \in Rdy$ iff v has collected the aggregated data from all its children nodes in the data aggregation tree T), and let F denote all the nodes which have finished their transmission, then $\mathrm{RdyCS}[i] = \mathrm{CS}[i] \cap Rdy - F$. The TST of all nodes is initialized to 0. The details of our distributed method are shown in Algorithm 6.

Algorithm 6. Distributed Data Aggregation Scheduling
Input: A network G, and the data aggregation tree T;
Output: $\mathrm{TST}[i]$ for every node v_i

1: The node v_i initializes the value $\mathrm{NoC}[i]$, and $\mathrm{Leaf}[i]$ based on the constructed aggregation tree T.
2: Initializes the set $\mathrm{CS}[i]$ based on the tree T and the original interference relation,
3: $\mathrm{RdyCS}[i] \leftarrow \mathrm{CS}[i] \cap \{j \mid j \text{ is a leaf in } T\}$.
4: $\mathrm{TST}[i] \leftarrow 0$; DONE $\leftarrow$ FALSE;
5: Node v_i randomly selects an integer r_i. Then we say $(r_i, i) < (r_j, j)$ if (1) $r_i < r_j$ or (2) $r_i = r_j$ and $i < j$.
6: **while** (not DONE) **do**

7: **if** $\mathrm{NoC}[i] = 0$ **then**
8: Send message $\mathrm{READY}(i, r_i)$ to all nodes in $\mathrm{CS}[i]$.
9: **if** $(r_i, i) < (r_j, j)$ for each $j \in \mathrm{RdyCS}[i]$ **then**
10: Send message $\mathrm{FINISH}(i)\&TST[i]$ to all nodes in $\mathrm{CS}[i]$;
11: DONE $\leftarrow$ TRUE;
12: **if** Node v_i received a message $\mathrm{FINISH}(j)\&TST[i]$ **then**
13: Delete j from $\mathrm{RdyCS}[i]$;
14: $\mathrm{TST}[i] \leftarrow \max\ \{\mathrm{TST}[i], \mathrm{TST}[j] + 1\}$;
15: **if** j is a child of i **then**
16: $\mathrm{NoC}[i] \leftarrow \mathrm{NoC}[i] - 1$;
17: **if** Node v_i received a message $\mathrm{READY}(j, r_j)$ **then**
18: **if** j is in $\mathrm{CS}[i]$ **then**
19: Add j to $\mathrm{RdyCS}[i]$.
20: Node v_i transmits data based on the time slot in $\mathrm{TST}[i]$.
21: The topological center transmits aggregated data to the sink.

When a node v_i finishes its scheduling, it sends a message FINISH to all nodes in its competitor set $\mathrm{CS}[i]$. When a node v_i received a message FINISH, it sets its $\mathrm{TST}[i]$ to the larger one of its original $\mathrm{TST}[i]$ and $\mathrm{TST}[j] + 1$. When all the children of node v_i finished their transmission, the node v_i is ready to compete for the transmission time slot and it will send a message $\mathrm{READY}(i, r_i)$ to all nodes in its competitor set. When a node v_i received a message READY from another node v_j, it will add the sender j to its ready competitor set $\mathrm{RdyCS}[i]$ if j is in $\mathrm{CS}[i]$. When the scheduling ends, all nodes will transmit their data based on $\mathrm{TST}[i]$. In the end, the topology center aggregates all the data and sends the result to the sink node via the shortest path.

4 Performance Analysis

In this section, we first theoretically prove that the delay of the data aggregation based on our scheduling is at most $16R + \Delta - 14$, where R is the radius of the network and Δ is the maximum node degree in the original communication graph. We conjecture that the theoretical performance of our centralized and distributed algorithms could actually be much better than $16R + \Delta - 14$, which is supported by our extensive simulations. On the other hand, we also present a network example to show that our centralized algorithm cannot achieve a delay lower than $4R + \Delta - 3$. It remains as future work to find bad network examples to show that our distributed methods could perform worse than $(1 + \varepsilon)R$ for a sufficient small constant $\varepsilon > 0$. At last, we present a general lower bound on the delay of data aggregation for any algorithm.

4.1 Performance of Our Algorithm

First, we show that after every time slot of Algorithm 4, for each dominator, the number of neighboring dominatees whose data are not collected is decreased by at least 1.

Claim 1. *All the output links in S in Step 4 of Algorithm 3 are conflict-free. In addition, after all the links transmit, for each dominator, the number of neighboring dominatees whose data are not collected is decreased by at least 1.*

Proof. We first check the origin of these links. As shown in Algorithm 3, each dominator u chooses a dominatee randomly from its neighbors and lets the chosen dominatee transmit to u. We call all chosen dominatees as *active transmitters* for later references. Assume that there are n_d dominators, then we have a set L of (at most) n_d chosen links. We input L to Algorithm 4 and assume that the output is the set S.

We define a **Loop Invariant** for Algorithm 4 as: for each dominator, the number of neighboring dominatees whose data are not collected is decreased by at least 1. Initially, since each dominator u chooses a neighboring dominatee to transmit to u, the loop invariant is true.

If these links in L do not conflict with each other, Algorithm 4 will skip the execution of the while loop and output a set of links which are the same as the input. Clearly, the output links are conflict-free and the loop invariant remains true.

Else, there exist interferences among links in L, then Algorithm 4 will execute the loop body. In each loop, Algorithm 4 adjusts a pair of conflicting links. By Lemma 2, after one round of adjustment , we solve the interferences caused by the pair of conflicting links, and the loop invariant remains true. Algorithm 4 repetitively adjusts a pair of conflicting links when interferences exist. Observe that due to the recursive nature of our adjustment algorithm, we must prove that Algorithm 4 will terminate in a finite number of rounds. Clearly, when it terminates, there is no pair of conflicting links and the loop invariant remains true.

To show that Algorithm 4 terminates, we define a **Potential Function** for a schedule as the cardinality of the set $\mathcal{C} = \{(x_1, x_2) \mid x_1, x_2$ are active transmitters and their corresponding links x_1y_1, x_2y_2 are conflicting links$\}$. We call the pair $(x_1, x_2) \in \mathcal{C}$ a pair of conflicting transmitters. Clearly, the initial cardinality of the set $\mathcal{C}$ is at most $n_d(n_d - 1)/2$. After one round of rescheduling, the interferences between at least one pair of conflicting transmitters are resolved. By Lemma 3, our adjustment will not introduce any new pair of conflicting transmitters. Thus, the potential function will be decreased by at least 1 after one round, which means that Algorithm 4 will terminate after at most $\frac{n_d(n_d-1)}{2}$ rounds of execution of the while loop in Algorithm 4.

Therefore, Algorithm 4 will terminate, which means that there exists no conflict among the output links in S. In addition, the loop invariant is true after Algorithm 4 terminates. Thus, claim 1 holds. □

Lemma 2. *After one round of adjustment (the loop body of Algorithm 4), we solve the interferences caused by the pair of conflicting links, and the loop invariant (every dominator's degree will be decreased by at least 1) remains true.*

Proof. We prove the claim for each of the complementary cases separately.

In Case 1 ($|u_iz_j| \leq 1$ and $|u_jz_i| > 1$), first, we prove that the interferences are solved. If $D_j = \emptyset$, since we remove one link, the interferences are clearly solved. Else, $D_j \neq \emptyset$, by definition, the distance between any dominator in D_j and u_i is greater than 1, thus, $|u_iz_{j_0}| > 1$. At the same time, $|u_jz_i| > 1$, thus, the output adjusted links

$u_iz_i, u_jz_{j_0}$ are conflict-free, the interferences are solved. Next, we prove that the loop invariant remains true. All other dominators in D_i are not affected by the adjustment, thus, we only need to prove that for the dominators in $D_i \cup D_j$, the number of their neighboring dominatees whose data are not collected is decreased by at least 1. If $D_j = \emptyset$, we only need to for every dominator in D_i, the number of their neighboring dominatees whose data are not collected is decreased by at least 1. This is straightforward since u_i transmits their data. Else, $D_j \neq \emptyset$, since both u_i, u_j transmit their data, thus, for every dominator in $D_i \cup D_j$, the number of their neighboring dominatees whose data are not collected is decreased by at least 1. Case 2 ($|u_jz_i| \leq 1$ and $|u_iz_j| > 1$) is similar to Case 1.

In Case 3 ($|u_jz_i| \leq 1$ and $|u_iz_j| \leq 1$), we first prove that the interferences are solved. If $D_j = \emptyset$ or $D_j = \emptyset$, since we remove one link, the interferences are clearly solved. Else, by definition of D_i, D_j, $|u_iz_{j_0}| > 1, |u_jz_{i_0}| > 1$, thus, the output adjusted links $u_iz_{i_0}, u_jz_{j_0}$ are conflict-free, the interference are solved. Then we prove that the loop invariant remains true. Similar to Case 1, we only need to prove that for the dominators in $D_i \cup D_j$, the number of their neighboring dominatees whose data are not collected is decreased by at least 1. If $D_i = \emptyset$, we only need to for every dominator in D_j, the number of their neighboring dominatees whose data are not collected is decreased by at least 1. This is straightforward since u_j transmits their data. The proof is similar for $D_i = \emptyset$. Else, both u_i, u_j transmit their data, thus, for every dominator in $D_i \cup D_j$, the number of their neighboring dominatees whose data are not collected is decreased by at least 1. □

Lemma 3. *The adjustment in one round of Algorithm 4 will not introduce any new pair of conflicting transmitters.*

Proof. We prove by contradiction. Suppose after an adjustment for a pair of links (u_iz_i, u_jz_j) to $(u_iz_{i_0}, u_jz_{j_0})$, Algorithm 4 introduces a new pair of conflicting transmitters (u, v). Since our adjustment only reconnects either u_i or u_j to a new dominators while does not change the links for other transmitters, one transmitter in (u, v) must be u_i or u_j. Assume that u is u_i, and the corresponding receiver of v is z_k. Since u_i and v conflict, either $(1)|u_iz_k| \leq 1$ or $(2)|vz_{i_0}| <= 1$. In Case 1, u_i and v is a pair of conflicting transmitters before the adjustment, which causes contradiction. Case 2 also causes contradiction since $z_{i0} \in D_i \subseteq D(u_i)$, by the definition of $D(u_i)$, the distance between other active transmitter $v, v \neq u_i$, and $v \neq u_j$ is greater than 1. (Refer to the first sentence in the second paragraph, right column, Page 4: For every other active transmitter $v, v \neq u_i$ and $v \neq u_j$, we delete all dominators from $D(u_i)$ (and also from $D(u_j)$) that are within the transmission range of v). □

Lemma 4. *Given a communication graph G of a network, under the assumption that the interference range r_I is the same as the transmission range r, Algorithm 3 (aggregating data from dominatees to dominators) costs at most Δ time slots, where Δ is the maximum node degree in G.*

Proof. Each dominator has at most Δ neighboring dominatees. We define a dominator's unaggregated-node-degree as the number of the neighboring dominatees whose data have not been aggregated to dominators yet. At first, each dominator's unaggregated-node-degree is bounded by Δ. By Claim 1, after one time slot, each dominator's unaggregated-node-degree is decreased by at least 1. Thus, Algorithm 3 costs at most Δ time slots. □

We now bound the number of connectors that a dominator u will use to connect to all dominators within two hops. Our proof is based on a technique lemma implied from lemmas proved in [20].

Lemma 5. *Suppose that dominator v and w are within two hops of dominator u, v' and w' are the corresponding connectors for v and w, respectively. Then either $|wv'| \leq 1$ or $|vw'| \leq 1$ if $\angle vuw \leq 2\arcsin\frac{1}{4}$.*

Lemma 6. *In Algorithm 5, a dominator requires at most 12 connectors to connect to all dominators within two hops.*

Proof. Consider any dominator u, let $I_2(u)$ be the set of dominators within two hops of u in the original communication network G. Assume that we have already deleted all the redundant connectors for node u. Let C be the set of connectors left for a dominator u. Then for each remaining connector $x \in C$, there is at least one dominator (called a *nonsharing dominator*) that can only use this connector to connect to u (otherwise, connector x is redundant, and thus, will be removed). Assume that there are 13 connectors in C. Then there are at least 13 nonsharing dominators in $I_2(u)$. From pigeonhole principle, we know that there must be two dominators v_1 and v_2 such that $\angle v_1uv_2 \leq 2\pi/13 < 2\arcsin(\frac{1}{4})$. Thus, using Lemma 5, v_1 and v_2 will share a common connector in C, which contradicts to the selection of v_1 and v_2. □

In the rest of the proof, for a dominator u, we use $C(u)$ to denote the set of connectors used to connect all dominators in $D_2(u)$.

Lemma 7. *In Algorithm 5, a dominator u in level i can receive the data from all neighboring dominators $D_2(u)$ in at most 16 time slots.*

Proof. Each dominator u will collect the aggregated data from all dominators within two hops in lower level. Any connector in $C(u)$ has at most four other neighboring dominators, besides u. Similar to the proof of Lemma 4, we can show that it takes at most four time slots for each connector to collect data from those neighboring dominators other than u. Recall that at most 12 connectors are needed for u to reach all dominators in $D_2(u)$. Thus, it will take at most 12 time slots for the dominator u to collect data from all these connectors. Consequently, within at most $12 + 4 = 16$ time slots, every dominator u can collect the aggregated data from all the dominators in $D_2(u)$. □

Theorem 8. *By using Algorithm 5, the sink can receive all the aggregated data in at most $17R + \Delta - 16$ time slots.*

Proof. Every dominatee's data can be aggregated to a dominator within Δ time slots from Lemma 4. Observe that every dominator, except the root of the data aggregation tree T, connects to at least one dominator in the upper level within two hops. Then Algorithm 5

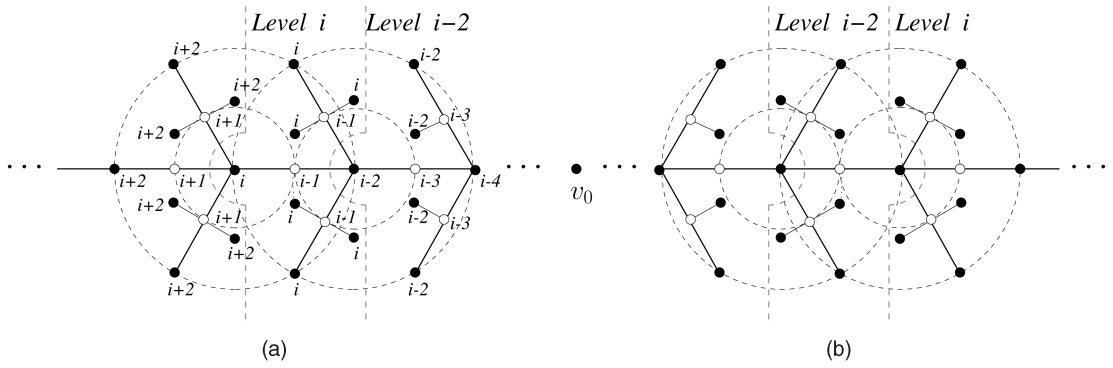


Fig. 3. A network example to show the lower bound of our algorithm.

ensures that every dominator's data can be aggregated at the root finally. For each level of the BFS tree, every dominator u including the root of data aggregation tree T can collect aggregated data from all dominators in $D_2(u)$ within at most 16 time slots by Lemma 7. Since there is no dominator in Level 1, after at most $16(R-1)$ time slots, every dominator's data can be aggregated to the root. The root then uses at most R time slots to transmit data to the original sink node via the shortest path. Therefore, within $17R+\Delta-16$ time slots, all the data can be aggregated to the sink node. □

Next, we provide a revised schedule that only needs 15 time slots for dominators in level i ($i \geq 2$) to aggregate data from some dominators within two hops, which can also ensure that data will be aggregated to the root finally. This means that we can reduce our delay by $R-2$ time slots totally.

For a dominator u other than the root, we denote all dominators within two hops of u by $B_2(u)$. Note that $B_2(u)$ includes at least one dominator v located in upper level of u. By Lemma 6, u needs at most 12 connectors to connect to $B_2(u)$, we denote the set of at most 12 connectors by $C(u)$. There must exist a connector $w \in C(u)$, which connects u to v. Then all dominators in $B_2(u)$ that are connected to w are also two-hop neighbors of the dominator v, we denote the set of these dominators by $B_2'(u)$, thus, $B_2'(u) \subset B_2(v)$. Clearly, all data in $B_2'(u)$ can be collected by v, it is not necessary for them to be collected by u. So we let u only collect the data in $B_2(u) \setminus B_2'(u)$. It requires at most 11 connectors (all the connectors in $C(u) \setminus \{w\}$) to connect to the dominators in $B_2(u) \setminus B_2'(u)$. So at most 15 $(=4+11)$ time slots are required for u to aggregate the data from $B_2(u) \setminus B_2'(u)$. If every dominator u other than the root aggregates the data from $B_2(u) \setminus B_2'(u)$, all the data can be aggregated to the root.

Theorem 9. *By using Algorithm 5, the sink can receive all the aggregated data in at most $16R+\Delta-14$ time slots.*

Proof. Similar to the proof of Theorem 8, we need Δ time slots for dominators to aggregate data from dominatees. After that, for each level of the BFS tree, every dominator u, other than the root of the data aggregation tree T, can collect aggregated data from all dominators in $B_2(u) \setminus B_2'(u)$ in at most 15 time slots as stated above. Thus, it costs at most $15(R-2)$ for data to be aggregated to the dominators in level 2. The root r_T can collect the aggregated data from dominators in level 2 within 16 time slots. Thus, within $15(R-2)+16$ time slots, every dominator's data can be aggregated to the root. The root then transmits the result to the original sink node in R time slots. In total, within $16R+\Delta-14$ time slots, all the data can be aggregated to the sink node. □

Observe that although our analysis is based on the centralized method, it is easy to show that all results carry to the distributed implementation (Algorithm 6). Thus, we have the following theorem:

Theorem 10. *By using Algorithm 6, the sink can receive all the aggregated data in at most $16R+\Delta-14$ time slots.*

4.2 Lower Bound of Our Algorithm

The lower bound of our algorithm is the delay for data aggregation in the worst input case. It is an important measurement to estimate the tightness of the upper bound of our algorithm derived in Section 4.1. In the following context, we present a network example, and show that when applying our algorithm to it, the delay can be as bad as $4R+\Delta-3$.

In Fig. 3, the root v_0 (which is the topology center) has two children, which means that there are two symmetric branches, each branch is symmetric with respect to the horizontal axis. For some nodes in the left branch, we mark their corresponding levels beside them. We use black nodes to denote dominators and white nodes to denote connectors. For each black node on the horizontal axis, we draw two cocentric circles with radii r and $2r$, respectively; all its three neighboring connectors are located on the inner circle. We omit all leaf nodes in the figure. The original sink v_s is located in the rightmost of the right branch.

Lemma 11. *When applying a centralized algorithm to the example shown in Fig. 3, the delay is $4R+\Delta-3$ time slots.*

Proof. First, aggregating data from dominatees to dominators costs Δ time slots by Lemma 4.

Second, both branches aggregate data from lower to upper levels. Between level i and level $i+2$ as shown in Fig. 3, it costs three time slots to aggregate data from the seven dominators in level $i+2$ to three connectors in level $i+1$ and costs another three time slots to aggregate data from three connectors in Level $i+1$ to a dominator in

level i. So it costs $(3+3)\cdot\frac{R-2}{2}$ time slots to gather data from dominators in level R toward dominators in level 2. After that, it costs one time slot to gather data from dominators in level 2 to connectors in Level 1 and then two time slots to the topology center v_0. Finally, v_0 transmits the aggregated data to the sink node, which will cost another R time slots. Therefore, we need $\Delta+(3+3)\cdot\frac{R-2}{2}+1+2+R=4R+\Delta-3$ time slots in total. □

4.3 Overall Lower Bound

In this section, we give the overall lower bound on the delay for data aggregation. Here, overall lower bound refers to the minimum time slots needed to finish the data aggregation by any possible algorithm.

Theorem 12. *Under any interference model, the overall lower bound of delay for data aggregation by* any *method is* $\max\{R,\log n\}$ *time slots, where* R *is the network radius and* n *is the number of nodes in the network.*

Proof. The lower bound R immediately follows from the fact that no matter what algorithm is implemented and no matter what interference model we will use, it costs at least R time slots for the farthest node v to transmit its data to the sink node v_s.

Next, we prove that $\log n$ is a lower bound for any valid schedule under any interference model. Here, a valid schedule is defined in Section 2.1, which is denoted by a sequence of sender sets $S_1,S_2,\ldots,S_l$. Then for any set of senders S_{l-i}, its receivers must be inside $\{v_s\}\cup(\bigcup_{j=0}^{i-1}S_{l-j})$. Consequently, $|S_{l-i}|<1+\sum_{j=0}^{i-1}|S_{l-j}|$ since different senders in S_{l-i} must have different receivers. Thus, we have

$$\begin{cases}|S_l|\le 1=2^0\\ |S_{l-1}|<1+|S_l|\le 2=2^1\\ |S_{l-2}|<1+|S_l|+|S_{l-1}|\le 4=2^2\\ \cdots\\ |S_{l-i}|<1+\sum_{j=0}^{i-1}|S_{l-j}|\le 2^i\\ \cdots\\ |S_1|\le 2^{l-1}.\end{cases}$$

Therefore, we have $\sum_{i=1}^{l}|S_i|\le 2^l-1$. From the precondition for a valid schedule that $\cup_{i=1}^{l}S_i=V\setminus\{v_s\}$, we get $n-1\le\sum_{i=1}^{l}|S_i|\le 2^l-1$. Therefore, $l\ge\log n$, which means that we need at least $\log n$ time slots for any schedule. Thus, $\log n$ time slots are a general overall lower bound, which finishes the proof. □

Under the protocol interference model when $r_I=r$, the communication graph is a UDG. Using area argument, we can get $n=O(\Delta\cdot R)$, where Δ is the maximum degree in UDG. Thus, $\max\{R,\log n\}=\max\{R,\log O(\Delta\cdot R)\}=\max\{R,\log\Delta\}$. By Theorem 12, $\max\{R,\log\Delta\}$ is also a lower bound under the protocol interference model when $r_I=r$. In Theorem 13, we will construct an example to show that the lower bound of both $\max\{R,\log\Delta\}$ and $\max\{R,\log n\}$ can be (approximately) achievable.

Theorem 13. *Under the protocol interference model when* $r_I=r$, *there is a placement of nodes such that the delay of data aggregation is only* $2\log\Delta(=2\log\frac{n+1}{2})$. *In other words, the overall lower bound provided in Theorem 12 is (approximately) tight in this model.*

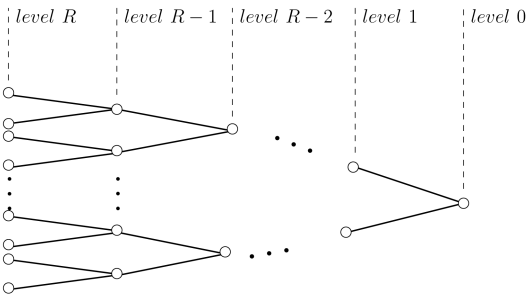


Fig. 4. An overall lower bound example.

Proof. We prove by construction. In Fig. 4, we construct a network example like a complete binary tree. There are R levels and level i has 2^i nodes. The distance between all nodes in level R is at most r. Thus, the degrees of all nodes in level R reach Δ. We order all nodes in level i from the highest to the lowest, it means that a node with order j is the j-highest among all nodes in level i (we note the node by $v_{(i,j)}$). The sink node is located on level 0, which is the root of the binary tree. The distance between any corresponding pair of nodes located in two consecutive levels is r, such as the pair of $v_{(i,2j-1)}$ and $v_{(i-1,j)}$ or the pair of $v_{(i,2j)}$ and $v_{(i-1,j)}$. The distance of any other pair of nodes located in two different levels is greater than r, such as $v_{(i,k)}$ and $v_{(i-1,j)}$ when $k\ne 2j-1$ and $k\ne 2j$.

We produce a valid schedule for the network example as follows: For $i=R,R-1,\ldots,1$

1. All links of $v_{(i,2j-1)}v_{(i-1,j)}(1\le j\le 2^{i-1})$ transmit simultaneously.
2. All links of $v_{(i,2j)}v_{(i-1,j)}(1\le j\le 2^{i-1})$ transmit simultaneously.

From the schedule, we can see that we only need two time slots to aggregate data from level i to level $(i-1)$. This implies that totally we need $2R$ time slots to aggregate data from all nodes to the sink. Since $R=\log\Delta=\log(n+1)/2$, this finishes the proof. □

Now we provide the overall lower bound under the protocol interference model when $r<r_I<3r$ and $r_I\ge 3r$.

Theorem 14. *Under the protocol interference model when* $r<r_I<3r$, *the overall lower bound of data aggregation is* $\max\{R,\frac{\Delta}{\phi}\}$, *where*

$$\phi=\frac{2\pi}{\lfloor\arcsin\frac{\gamma-1}{2\gamma}\rfloor}$$

and $\gamma=\frac{r_I}{r}$; *when* $r_I\ge 3r$, *the overall lower bound is* $\max\{R,\Delta\}$.

Proof. By Theorem 12, R is a lower bound.

Assume that node u has Δ neighbors. Since every neighbor of u needs to transmit at least once to report its data, we try to compute the maximum number of u's neighbors that can transmit simultaneously without interference, which implies a lower bound.

When $r < r_I < 3r$, assume that two neighbors p, s of u transmit simultaneously, and q, t are their corresponding receivers. From [22, Lemma 5], $\angle qut$ must be no larger than $\theta = \arcsin\frac{\gamma-1}{2\gamma}$ to ensure that links pq and st are interference-free with each other. So the maximum number of u's neighbors that can transmit simultaneously is $\phi = \lfloor\frac{2\pi}{\theta}\rfloor$. Therefore, $\frac{\Delta}{\phi}$ is an overall lower bound. Thus, the overall lower bound of delay is $\max\{R, \frac{\Delta}{\phi}\}$ when $r < r_I < 3r$.

When $r_I \geq 3r$, if one of u's neighbors is transmitting to the node w, the distance between w and any other neighbor of u is smaller than $3r$, thus, smaller than r_I. So the maximum number of u's neighbors that can transmit simultaneously is only one. Therefore, Δ is an overall lower bound. Thus, the overall lower bound of delay is $\max\{R, \Delta\}$ when $r_I \geq 3r$. This finishes the proof. □

5 Other Network Models

To schedule two links at the same time slot, we must ensure that they are interference-free with each other. Previous studies on stable link scheduling mainly focused on the protocol interference model in which the transmission and interference ranges are the same. In addition to the protocol interference model, several different interference models have been used to model the interference. We briefly review these models below.

k-hop interference model. A sending node u (with receiver p) is said to cause interference at another receiving node w if w is within k-hops of the node u, i.e., the hop distance between u and w in the communication graph G is at most k.

RTS/CTS model. For every pair of transmitter and receiver, all nodes that are within the interference range of either the transmitter or the receiver cannot transmit. In this case, we assume that node u will interfere the receiving of another node w from another sender v if either v or w is in the transmission range of u. Although RTS/CTS is not the interference itself, for convenience of our notation, we will treat the communication restriction due to RTS/CTS as *RTS/CTS interference* model.

Now we discuss data aggregation in other interference models. Similar to the algorithms in Section 3, we apply a scheme in which all the data in the dominatees are aggregated to the dominators first, then dominators transmit their data toward the root level by level until all data reach the root. As already shown in Lemma 4, all the data can be aggregated to the dominators by at most Δ time slots (here, Δ is the maximum degree in the interference graph instead of communication graph). The only difference is how to collect data from all dominators to the root. We still use the scheme similar to Algorithm 5. To analyze the performance, we need to count the maximum number of dominators in $k+1$ hops. Observe that here RTS/CTS model is essentially two-hop interference model. We first discuss two-hop model, the other models are similar.

Theorem 15 (Wegner Theorem [10]). *The area of the convex hull of any $n \geq 2$ nonoverlapping unit-radius circular disks is at least $2\sqrt{3}(n-1) + (2-\sqrt{3})\lceil\sqrt{12n-3} - 3\rceil + \pi$.*

Lemma 16. *There are at most 41 independent nodes within any disk of radius 3.*

Proof. Fix a disk D_2 centered at a point u. Let S denote the set of independent nodes in D_2. If for each node in S we consider a disk of radius 0.5 centered at this node, then all of those disks must be disjoint. Therefore, the convex hull of S must be contained in the disk of radius 3.5 centered at u. By applying Wegner Theorem with proper scaling, we have $2\sqrt{3}(|S|-1) + (2-\sqrt{3})\lceil\sqrt{12|S|-3} - 3\rceil + \pi < 49\pi$. Straightforward calculation shows that the maximum integer to make the above inequality hold is $|S| = 41$. □

Thus, similar to Theorem 8, we have the following theorem on the delay of our data aggregation method under two-hop interference model:

Theorem 17. *Under two-hop interference model, the sink can receive all the aggregated data in at most $O(R+\Delta)$ time slots.*

Note that under two-hop interference model, any two senders x and y cannot be communication neighbors (otherwise, x will cause interference at the receiver of y). Thus, given Δ neighbors of a node, we need at least $\Delta/5$ time slots to just let every of these Δ neighbors transmit once. Thus, the below theorem follows:

Theorem 18. *Under two-hop interference model, for any data aggregation method, it will take at least $\max(R, \Delta/5)$ time slots for the sink to receive the aggregated data.*

For k-hop interference model, where $k \geq 3$, then any two nodes x and y that are neighbors of a node u clearly cannot transmit simultaneously. Thus, Δ is a lower bound on delay of data aggregation. For general k-hop interference model, we are also able to prove the following:

Theorem 19. *Under k-hop interference model ($k \geq 3$), the sink can receive all the aggregated data in at most $O(k^2)(R+\Delta)$ time slots. For any data aggregation method, it will take at least $\max(R, \Delta)$ time slots for the sink to receive all the aggregated data.*

6 Simulation Results

In this section, we present the simulation results, which evaluate our Distributed Data Aggregation Algorithms (Algorithm 6).

6.1 Evaluating the Worst-Case Performances

Since in our paper and all related works, the performance analysis part mainly focuses on the upper bound on latencies, which is the worst-case performances as well, we evaluate the worst-case performance of our algorithm first. Here, we compare our algorithm (which has an upper bound on delay of $16R + \Delta - 14$ time slots) with the previously known best result (which has an upper bound on delay of $24D + 6\Delta + 16$ time slots in [3]).

We can see that when the network radius R is fixed, our worst-case performances are 3 to 4.5 times better than previous best result (Fig. 5a); when the maximum node degree Δ is fixed, our worst-case performances are 1.5 to 2 times better than previous best result (Fig. 5b).

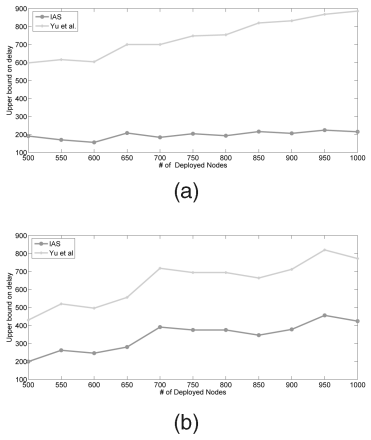


Fig. 5. Comparisons of worst-case performances for two methods. (a) Fixed R. (b) Fixed Δ.

6.2 Evaluating the Average Performances

Now we compare the average performances of three algorithms (Algorithm 6, Yu et al. [3], and Huang et al. [11]). We randomly deploy nodes (representing sensors) into a region of $200\ \mathrm{m} \times 200\ \mathrm{m}$. All nodes have the same transmission radius.

In Fig. 6a, the transmission radius of each sensor is fixed to 25 m. The figure shows the delay for aggregating data from all nodes to the sink by running three algorithms while the number of deployed nodes increases.

Fig. 6b compares the latencies for aggregating data using three algorithms when the maximum node degree varies. Here, the maximum node degree Δ is fixed to 25. It can be seen in the figure that our algorithm (nearly the same with Yu et al.'s) outperforms Huang et al.'s algorithm with much lower latencies.

6.3 Evaluations on TOSSIM of TinyOS 2.0.2

We implemented IAS on TOSSIM of TinyOS 2.0.2. We randomly deploy a number of sensor nodes in a two-dimensional square region, all nodes have the same transmission range. Each node will generate a random 16-bit nonnegative number as its own datum. The objective of the sink node is to report the aggregation result of all data (totally n data, n is the network size) correctly.

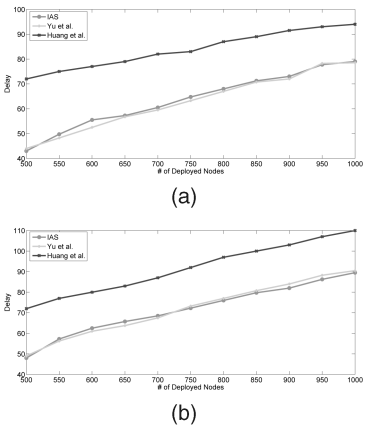


Fig. 6. Comparisons of average performances for three methods. (a) Fixed R. (b) Fixed Δ.

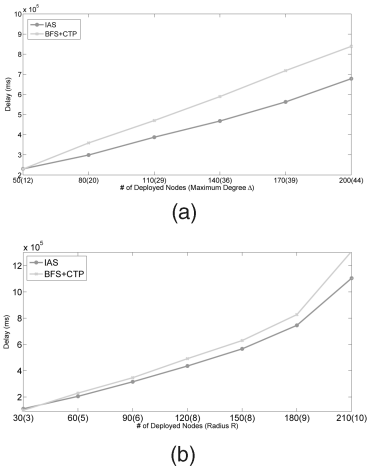


Fig. 7. Simulation results for our algorithm and BFS+CTP. (a) Fixed R. (b) Fixed Δ.

In order to evaluate the efficiency of IAS, we also implemented another data aggregation algorithm by combining BFS tree and Collection Tree Protocol (CTP, which is provided by TinyOS 2.0.2) using TOSSIM. We call this method BFS+CTP method for simplicity. The main idea of BFS+CTP method is to construct a BFS tree rooted at the sink node based on the link quality. In other words, during the procedure of constructing BFS, the link quality computed by CTP will be considered as the link weight. Note that the original CTP method (components) provided in TinyOS 2.0.2 is used to collect data to the sink node. To enable CTP to support data aggregation rather than to collect all data to the sink, we modified CTP in the upper layer such that each node will not send data to its parent (on the BFS tree) until it aggregates all necessary data from all children (on the BFS tree).

We tested and compared the latencies for IAS method and BFS+CTP method in two different cases. For the first case, we randomly generated the network topology (connected) with different network size (increasing from 30 to 210 with step 30) while ensuring the network density unchanged, i.e., the network deployment area increases with the increment of the network size. Actually, by doing this, we fixed the maximum degree Δ (in our simulation, Δ is around 22) for each case; thus, the radius of communication graph increases with the increment of network size. The delay performance of two methods, IAS and BFS+CTP, is illustrated in Fig. 7a. Note that here the definition of delay is the time duration from the time the first datum is transmitted heading for the sink node to the time the sink node reports the result finally. From Fig. 7a, we can see that when the network density is not big, the delay difference between two methods is not so big. In most cases, our IAS method has better performance than that of BFS+CTP. The radius R for each case is indicated by the value in the brackets right after the network size on x-coordinate.

For the second case, we fix the deployment area as (300×300) and continue to increase the network size from 50 to 200 with step 30 while keeping the network connected.

By doing this, we can fix the radius R and test the performance of both algorithms with the increment of network density (maximum degree Δ).

As we can see in Fig. 7b, there is a big gap between these two methods when the density (maximum degree Δ) continues increasing. This is because the interference will be greatly decreased after IAS gathers all data to dominators. Hence, the total delay decreases significantly. However, for BFS+CTP method, the number of relay nodes will continue to increase with the increment of network size such that the delay increases greatly due to the interference. From the simulation results, we can see that in most cases, IAS has better performance than BFS+CTP method. Especially, the denser the network is, the more efficient our IAS algorithm is.

7 Related Work

Data aggregation in sensor networks has been well studied recently [2], [12], [15], [25]. In-network aggregation means computing and transmitting partially aggregated data rather than transmitting raw data in networks, thus, reducing the energy consumption [16].

There are a lot of existing researches on in-network aggregation in the literature [6], [17]. Suppression scheme and model-driven methods were proposed in [5], [7] toward reducing communication cost. The trade-off between energy consumption and time delay was considered in [25]. A heuristic algorithm for both broadcast and data aggregation was designed in [1]. Another heuristic algorithm for data aggregation was proposed [18], aiming at reducing time delay and energy consumption. Kesselman and Kowalski [13] proposed a randomized and distributed algorithm for aggregation in WSNs with an expected delay of $O(\log n)$. Their method is based on two assumptions: One is that sensor nodes can adjust their transmission range without any limitation. The other is that each sensor node has the capability of detecting whether a collision occurs after transmitting data. Both assumptions pose some challenges for hardware design and are impractical when the network scales. A collision-free scheduling method for data collection is proposed in [14], aiming at optimizing energy consumption and reliability. All these works did not discuss the minimal delay aggregation scheduling problem.

In addition, the minimum delay of data aggregation problem was proved NP-hard and a $(\Delta - 1)$-approximation algorithm was proposed in [4], where Δ is the maximum degree of the network graph. Another aggregation scheduling algorithm was proposed in [11], which has a delay bound of $23R + \Delta + 18$, where R is the network radius and Δ is the maximum degree. Recently, Wan et al. [21] proposed three novel centralized data aggregation methods for networks when nodes have the same transmission radius and interference radius, which achieve schedules of latency $15R + \Delta - 4$, $2R + O(\log R) + \Delta$, and $(1 + O(\frac{\log R}{\sqrt[3]{R}}))R + \Delta$, respectively. Recently, Xu et al. [24] studied aggregation with multiple queries in WSNs. All the algorithms mentioned above are centralized. In many cases, centralized algorithms are not practical, especially when the network topology changes often in a large sensor network.

The distributed algorithms for convergecast scheduling were proposed in [3], [8], [13]. [8], [13] focused on the scheduling problem for data collection in sensor networks. In data collection, since data cannot be merged, the sink must receive N packets from all the nodes, where N is the number of sensor nodes in the network. Thus, the lower bound of delay is N. The upper bound of the time delay of this algorithm is $\max(3n_k - 1, N)$, where n_k is the number of nodes in the largest one-hop subtree. Bo Yu and Li [3] proposed a distributed scheduling algorithm generating collision-free schedules that has a delay bound of $24D + 6\Delta + 16$, where D is the network diameter.

8 Conclusions

Data aggregation is critical to the network performance in WSNs and aggregation scheduling is a feasible way of improving the quality. In this paper, we study the problem of distributed aggregation scheduling in WSNs and propose a distributed scheduling method with an upper bound on delay of $16R + \Delta - 14$ time slots. This is a nearly constant approximate algorithm, which significantly reduces the aggregation delay. The theoretical analysis and the simulation results show that our method outperforms the previous methods.

In addition, we provide the overall lower bound on delay for data aggregation under any interference model with formal proofs and give an example to show that the lower bound is (approximately) tight under the protocol interference model when $r_I = r$, where r is the transmission range and r_I is the interference range. We also derive the lower bound on delay under the protocol interference model when $r < r_I < 3r$ and $r_I \geq 3r$.

Acknowledgments

The research of authors is partially supported by US National Science Foundation (NSF) NSF CNS-0832120, NSF CNS-1035894, National Natural Science Foundation of China under Grant No. 60828003, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program), National Basic Research Program of China (973 Program) under grant No. 2010CB328100 and 2010CB334707, and by Tsinghua National Laboratory for Information Science and Technology (TNList).

References

[1] V. Annamalai, S. Gupta, and L. Schwiebert, "On Tree-Based Convergecasting in Wireless Sensor Networks," *Proc. IEEE Comm. and Networking Conf. (WCNC)*, 2003.

[2] J. Beaver, M. Sharaf, A. Labrinidis, and P. Chrysanthis, "Location-Aware Routing for Data Aggregation in Sensor Networks," *Proc. Geosensor Networks Workshop*, 2004.

[3] J.L. Bo Yu and Y. Li, "Distributed Data Aggregation Scheduling in Wireless Sensor Networks," *Proc. IEEE INFOCOM*, 2009.

[4] X. Chen, X. Hu, and J. Zhu, "Minimum Data Aggregation Time Problem in Wireless Sensor Networks," *Proc. First Int'l Conf. Mobile Ad-Hoc and Sensor Networks (MSN)*, vol. 133, 2005.

[5] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks Using Probabilistic Models," *Proc. Int'l Conf. Data Eng. (ICDE)*, 2006.

[6] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate Aggregation Techniques for Sensor Databases," *Proc. Int'l Conf. Data Eng. (ICDE)*, 2004.

[7] A. Deshpande, C. Guestrin, W. Hong, and S. Madden, "Exploiting Correlated Attributes in Acquisitional Query Processing," *Proc. Int'l Conf. Data Eng. (ICDE),* 2005.

[8] S. Gandham, Y. Zhang, and Q. Huang, "Distributed Minimal Time Convergecast Scheduling in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* 2006.

[9] P. Gupta and P.R. Kumar, "The Capacity of Wireless Networks," *Proc. IEEE Trans. Information Theory,* vol. 46, no. 2, pp. 388-404, Mar. 2000.

[10] G. Wegner, "Über Endliche Kreispackungen in Der Ebene," *Studia Scientiarium Mathematicarium Hungarica,* vol. 21, pp. 1-28, 1986.

[11] S. Huang, P. Wan, C. Vu, Y. Li, and F. Yao, "Nearly Constant Approximation for Data Aggregation Scheduling in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* pp. 366-372, 2007.

[12] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* vol. 22, pp. 457-458, 2002.

[13] A. Kesselman and D. Kowalski, "Fast Distributed Algorithm for Convergecast in Ad Hoc Geometric Radio Networks," *J. Parallel and Distributed Computing,* vol. 66, no. 4, pp. 578-585, 2006.

[14] H. Lee and A. Keshavarzian, "Towards Energy-Optimal and Reliable Data Collection via Collision-Free Scheduling in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* pp. 2029-2037, 2008.

[15] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks," *Proc. USENIX Symp. Operating Systems Design and Implementation,* 2002.

[16] M. Sharaf, J. Beaver, A. Labrinidis, and P. Chrysanthis, "TiNA: A Scheme for Temporal Coherency-Aware In-Network Aggregation," *Proc. Third ACM Int'l Workshop Data Eng. for Wireless and Mobile Access,* pp. 69-76, 2003.

[17] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and Beyond: New Aggregation Techniques for Sensor Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys),* pp. 239-249, 2004.

[18] S. Upadhyayula, V. Annamalai, and S. Gupta, "A Low-Latency and Energy-Efficient Algorithm for Convergecast in Wireless Sensor Networks," *Proc. IEEE Global Comm. Conf. (GlobeCom),* vol. 6, 2003.

[19] P.-J. Wan, K. Alzoubi, and O. Frieder, "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks," *Mobile Networks and Applications,* vol. 9, pp. 141-149, 2004.

[20] P.J. Wan, C.W. Yi, X. Jia, and D. Kim, "Approximation Algorithms for Conflict-Free Channel Assignment in Wireless Ad Hoc Networks," *Wiley Wireless Comm. and Mobile Computing,* vol. 6, pp. 201-211, 2006.

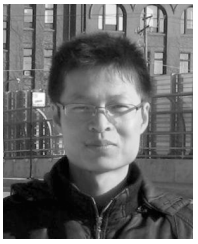
[21] P.J. Wan, S.C.-H. Huang, L.X. Wang, Z.Y. Wan, and X.H. Jia, "Minimum-Latency Aggregation Scheduling in Multihop Wireless Networks," *Proc. ACM MobiHoc,* pp. 185-194, 2009.

[22] W. Wang, Y. Wang, X.Y. Li, W.Z. Song, and O. Frieder, "Efficient Interference-Aware TDMA Link Scheduling for Static Wireless Networks," *Proc. ACM MobiCom,* pp. 262-273, 2006.

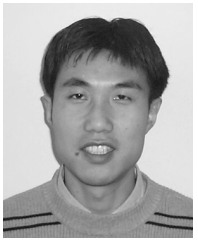
[23] X.-H. Xu, S.-G. Wang, X.-F. Mao, S.-J. Tang, and X.-Y. Li, "An Improved Approximation Algorithm for Data Aggregation in Multi-Hop Wireless Sensor Networks," *Proc. ACM MobiHoc FOWANC Workshop,* 2009.

[24] X.-H. Xu, S.-G. Wang, X.-F. Mao, S.-J. Tang, P. Xu, and X.-Y. Li, "Efficient Data Aggregation in Multi-Hop WSNs," *Proc. IEEE Global Comm. Conf. (GlobeCom),* 2009.

[25] Y. Yu, B. Krishnamachari, and V. Prasanna, "Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* vol. 1, 2004.



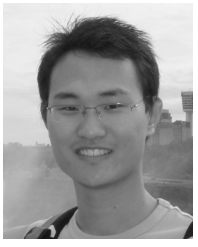
Xiaohua Xu received the BS degree from ChuKochen Honors College at Zhejiang University, P.R. China, in 2007. He is currently working toward the PhD degree in computer science at Illinois Institute of Technology. His research interests include algorithm design and analysis, optimization in mesh network, and energy efficiency and security in wireless network. He is a student member of the IEEE.



Xiang-Yang Li received the BEng degree in computer science and the bachelor's degree in business management from Tsinghua University, P.R. China, in 1995, and the MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign in 2000 and 2001, respectively. He has been an associate professor since 2006 and assistant professor of computer science at Illinois Institute of Technology from 2000 to 2006. He was a visiting professor at Microsoft Research Asia from May 2007 to August 2008. His research interests include wireless ad hoc and sensor networks, noncooperative computing, computational geometry, and algorithms. He was a guest editor of special issues for the *ACM Mobile Networks and Applications*, the *IEEE Journal on Selected Areas in Communications,* and editor of the *Networks* journal. He is a member of the ACM and a senior member of the IEEE.

Xufei Mao received the BS degree from Shenyang University of Technology, China, in 1999, and the MS degree from Northeastern University, China, in 2003. He is currently working toward the PhD degree in computer science at Illinois Institute of Technology. His research interests include design and analysis of algorithms concerning wireless networks, network security, etc. Topics include navigation problem in sensor network, Top-k query, capacity (throughput) study, channel assignment, link scheduling and TinyOS programming, etc. He is a student member of the IEEE.

Shaojie Tang received the BS degree in radio engineering from Southeast University, P.R. China, in 2006. He is currently working toward the PhD degree in computer science at Illinois Institute of Technology. His current research interests include algorithm design and analysis for wireless ad hoc network and online social network. He is a student member of the IEEE.



Shiguang Wang received the BS degree from Nanjing University, P.R. China, in 2008. He is currently working toward the PhD degree in computer science at Illinois Institute of Technology. His research interests include algorithm and system design for wireless ad hoc and sensor networks, game theoretical study of networks, and RFID system. He is a student member of the IEEE.