Neither Shortest Path Nor Dominating Set: Aggregation Scheduling by Greedy Growing Tree in Multihop Wireless Sensor Networks

Chen Tian, *Member, IEEE*, Hongbo Jiang, *Member, IEEE*, Chonggang Wang, *Senior Member, IEEE*, Zuodong Wu, Jinhua Chen, and Wenyu Liu, *Member, IEEE*

***Abstract*—Data aggregation is a fundamental task in multihop wireless sensor networks. Minimum-latency aggregation scheduling (MLAS) seeks to minimize the number of scheduled time slots to perform an aggregation. In this paper, we present the first work on a solvable mathematical formulation of the MLAS problem. The optimal solution of small example networks suggests that an optimal scheduling can be neither shortest path nor dominating set based. Instead of yet another theoretical analysis with provable bounds, our work focuses on reducing the average latency of general random topologies. Inspired by backward induction theory, we propose to schedule the aggregation in a reverse order, and the tree construction of Greedy Growing Tree (GGT) is directly guided by the scheduling algorithm in a step-by-step way. By following priority rules when we schedule candidate $\langle sender, receiver \rangle$ pairs, the opportunity of parallel transmission is maximized. As a result, the aggregation latency can be minimized. Our extensive evaluation results demonstrate the superiority of the GGT algorithm: For sparse networks, the resultant latency is comparable with the best practice, whereas for high-degree networks, the latency is only half of that using state-of-art competitors.**

***Index Terms*—Greedy algorithm, scheduling, wireless sensor networks.**

I. Introduction

In MULTIHOP wireless sensor networks, a fundamental task is to gather data from all sensors to a distinguished $sink$ node [1]. In general, each intermediate node merges its received data with its own record according to some aggregation functions (e.g., taking the maximum or minimum of them) into a single packet with fixed size. This type of application is called *data aggregation*; its communication pattern is called *convergecast* [2]. The goal of our study is to minimize the average data aggregation latency of the convergecast process.

Naive aggregation approaches, which purely rely on medium-access-control layer mechanisms (e.g., carrier-sense multiple access), could result in latency that is too high to be practical due to the existence of mutual transmission interference [2]–[4]. Synchronized aggregation scheduling is necessary, where all transmissions proceed in synchronous time slots. Such an aggregation scheduling includes both a spanning inward tree rooted at the *sink* and an edge scheduling of this spanning routing tree under three conditions.

1) Each node transmits at most one packet of a fixed size in its allocated time slot.
2) A node cannot transmit until all its children complete the transmissions to itself.
3) All transmissions assigned in the same time slot should be interference free.

The latency is the required number of time slots of the whole aggregation convergecast process. To minimize the latency, this problem is often named minimum-latency aggregation scheduling (MLAS) [5].

Previous works can construct feasible interference-free schedules for the MLAS problem. Mostly, they solve the problem in two consecutive yet independent phases, i.e., a tree construction phase followed by an edge-scheduling phase [3]–[5]. Specifically, a spanning tree rooted at a *sink* is constructed first from the adjacency graph. In the second phase, the scheduling algorithm iteratively constructs smaller and smaller residual trees, which span all residual nodes possessing data of interest. In each round, all nodes in the residual tree are divided into two groups, i.e., the leaf nodes and the nonleaf nodes. The interference-free scheduling is separately applied to leaf nodes and nonleaf nodes. The selected nodes become $senders$ of the corresponding aggregation step. They are then removed from the residual tree before entering the next round.

The assumption behind these two-phase approaches is that a "well-chosen" tree would consequently lead to "good" scheduling. Chen *et al.*'s approach [3] is based on the shortest path tree (SPT), whereas the work of Huang *et al.* and Wan *et al.* [4], [5] is based on the dominating set tree (DST). Hereby, we emphasize the problems of two-phase approaches: First,

Manuscript received January 22, 2011; revised May 11, 2011; accepted July 5, 2011. Date of publication July 18, 2011; date of current version September 19, 2011. This work was supported in part by the National Natural Science Foundation of China under Grant 60803115, Grant 60873127, and Grant 61073147; by the National Natural Science Foundation of China-Microsoft Research Asia under Grant 60933012; by the Youth Chenguang Project of Wuhan City under Grant 201050231080; by the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry); and by the Program for New Century Excellent Talents in University (State Education Ministry) under Grant NCET-10-408. The review of this paper was coordinated by Prof. V. W. S. Wong.

C. Tian, H. Jiang (Corresponding author), Z. Wu, J. Chen, and W. Liu are with the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: hongbojiang2004@gmail.com).

C. Wang is with InterDigital Communications, King of Prussia, PA 19406 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2011.2162086

the performance of the same specific algorithm could vary greatly, depending on the tree initially constructed. Second, separated scheduling to predivided leaf nodes and nonleaf nodes is suboptimal: the opportunity of parallel transmission among nodes, particularly among nodes of different layers of the predefined tree, is decreased. There is less competition in a sparse network; hence, the SPT approaches could perform fairly well. Unfortunately, for a high-degree network, the aggregation latency is undesirably large.

Instead of yet another theoretical analysis with provable bounds, our work focuses on reducing the average latency of general random topologies. The main contributions of this paper are summarized here.

1) For the first time, we present a solvable mathematical formulation for the MLAS problem (see Section IV). By solving the optimal solutions of some small example networks, we get the insight that an optimal scheduling could be neither shortest path nor dominating set based.
2) Inspired by backward induction theory[1] [6], we propose the Greedy Growing Tree (GGT) algorithm. GGT schedules the aggregation process in a reverse order: The last aggregation step is scheduled first to choose the last $sender$ to the *sink* and then choose the $\langle sender, receiver\rangle$ set of the next-to-last step, and so on, until all nodes are scheduled. As its name suggests, GGT constructs larger and larger spanning trees rooted at the *sink*: The temporary spanning tree contains only the *sink* node at the beginning. In each round, all nonleaf nodes are candidate $receivers$, and all leaf nodes of the temporary tree are candidate $senders$. The $\langle sender, receiver\rangle$ set is selected in a manner so that the opportunity of parallel transmissions can be maximized. As a result, the whole aggregation latency could be minimized. The new $senders$ are added into the tree before the next round. In contrast to previous works, GGT has only one phase, and the tree construction is directly guided by the scheduling algorithm step by step.
3) We conduct extensive evaluations to compare the performance of different approaches. The results show that, for the sparse networks, the latency using the GGT algorithm is comparable with the state-of-art algorithm. The main achievement is that, for high-degree networks, the latency is only half of that using the other existing methods.

The rest of this paper is organized as follows: We first introduce the necessary background and formally define the aggregation scheduling model in Section II. In Section III, we review the existing literature and argue that neither SPT nor DST could guarantee a satisfactory solution to the MLAS problem. We present the first solvable mathematical MLAS formulation in Section IV. The main idea and design of GGT are presented in Section V. We evaluate the proposed scheduling algorithm by simulations in Section VI. Finally, we present our conclusion in Section VII.

[1]Backward induction is the process of reasoning backward in time, from the end, to determine a sequence of appropriate actions. It proceeds by first considering the last time a decision might be made and choosing what to do in any situation. Using this information, one can then determine what to do at the second-to-last time of decision. This process continues backward until the actions at every point in time have been determined.

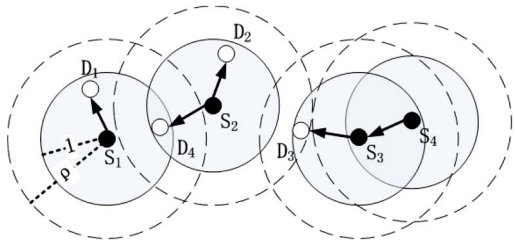


Fig. 1. Protocol interference model: Each node has a unit transmission radius and an interference radius $\rho \geq 1$.

II. Background

A. System Model

In this paper, we study the MLAS problem under the protocol interference model in multihop wireless networks. All nodes are located in an Euclidean plane and are equipped with omnidirectional antenna. Each node has a fixed transmission radius, which is normalized to 1 and an interference radius $\rho \geq 1$. The communication range and the interference range of a node v are the two disks centered at v of radius 1 and ρ, respectively (see node S_1 in Fig. 1).

Let V denote the set of sensor nodes and $G = (V, E)$ be the unit-disk graph (UDG) on V with a *sink* node $b \in V$. The communication graph of the network is a digraph $\overrightarrow{G}$ obtained from G by replacing each link (e.g., (u, v)) in G with two oppositely directed edges $u \rightarrow v$ and $v \rightarrow u$. A pair of communication edges $S_1 \rightarrow D_1$ and $S_2 \rightarrow D_2$ is said to be interference free if the two line segments (S_1, D_2) and (S_2, D_1) are both longer than ρ, as shown in Fig. 1. Otherwise, they belong to the *exclusive interference edges set* of each other and cannot be scheduled in the same time slot (e.g., $S_1 \rightarrow D_1$ and $S_2 \rightarrow D_4$). A subset of edges scheduled in the same time slot is said to be an *interference-free set* if they are pairwise interference free. Such an interference model is referred to as the *protocol interference model* [7] and is widely used because of its generality and tractability. What's more, we assume that a node works in *half-duplex* mode: It can either send or receive data at one time slot, or it can receive data correctly only if exactly one of its neighbors is transmitting at that moment. For example, when S_3 is transmitting to D_3, it cannot simultaneously receive the packet from S_4.

B. Data Aggregation Model

We consider a wireless network consisting of N sensor nodes. Throughout this paper, we always use 0 as the index of the *sink* node b; nodes $1 \ldots (N-1)$ process data of interest. A small example network is shown in Fig. 2(a); the dash lines among nodes denote the communication neighborhood relationship. Recall that the main task of sensor nodes is to collect data and transmit them back to *sink* b by convergecast; data can be "aggregated" by intermediate nodes along the path to *sink* b. In other words, if packets are received from its neighbors before its scheduled transmission time slot, a node aggregates the data

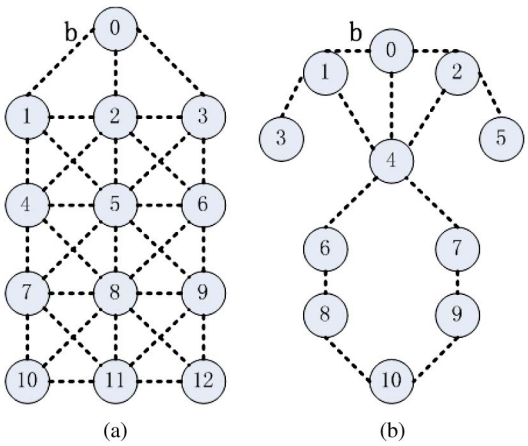


Fig. 2. Two example networks.

with its own data and only sends the aggregated packet to its parent.

We consider two subsets $C_1, C_2 \subset V$ with $C_1 \supset C_2$. If all nodes in $C_1 \setminus C_2$ can transmit packets to nodes in C_2 within one time slot and these transmissions are interference free, the packets from the nodes of $C_1 \setminus C_2$ can be simultaneously received by C_2. In this case, we say that data can be aggregated from C_1 to C_2 in one step. The *data aggregation property* means that, after $C_1 \setminus C_2$ transmissions, the data packets can be aggregated from C_1 to C_2 without interference.

A data aggregation, which is completed within K steps, can be considered as a sequence of subsets $C_1, C_2, \ldots, C_K, C_{K+1}$ ($C_k \subset V$, where $1 \leq k \leq K+1$) satisfying the data aggregation property. This sequence requires that the nodes of $C_1 \setminus C_2$ transmit to some nodes in C_2 in the first time slot, followed by the nodes of $C_2 \setminus C_3$ transmitting to some nodes in C_3 in the second time slot, and so on. If we iteratively continue this process, finally, all data packets are aggregated to a single *sink* node $b = 0$.

The formal definition of the data aggregation process is given as follows: A data aggregation schedule is a sequence of subsets $C_1, C_2, \ldots, C_K, C_{K+1}$ that satisfy three conditions.

1) $C_{k1} \supset C_{k2}, \forall 1 \leq k1 < k2 \leq K+1$.
2) $C_1 = V$, and $C_{K+1} = b$.
3) Data packets are aggregated from C_k to C_{k+1} for all $k = 1, 2, \ldots, K-1$; finally, data packets are aggregated from C_K to $C_{K+1} = b$ in time slot K.

The value of K is considered to be the data aggregation latency, which we want to minimize. Unfortunately, this problem is nondeterministic polynomial-time hard (NP-hard), even with the UDG model [3]. Since the ρ value only affects the calculation of exclusive edges, we focus on the case where $\rho = 1$ in this paper for the simplicity of presentation.

III. Related Work

A. SPT Based

The minimum data aggregation time problem was proven to be NP-hard [3]. All existing heuristics perform in two independent phases. A $(\Delta - 1)R$ approximation algorithm Shortest Data Aggregation (SDA) was proposed by Chen [3], where Δ is the maximum degree, and R is the radius of the network. SDA constructs an SPT in the first phase. After that, the scheduling is iteratively implemented: Each round introduces a schedule of the corresponding aggregation step. In round r, SDA picks $senders$ only from the leaf nodes guided by the interference-free principal. Subsequently, SDA eliminates the $senders$ from the tree and enters the $(r+1)$th iteration. Thus, the algorithm SDA proceeds by incrementally constructing smaller and smaller SPTs rooted at b that span nodes possessing all data of interest.

The performance of SDA varies greatly, depending on the SPT initially supplied. We take the example network in Fig. 2(a) as an illustration. Two distinct SPTs are given in Fig. 3(a) and (b). The SDA scheduling is then performed to them, respectively, and the results are shown in Fig. 3(c) and (d). It is observed that SPT 1 can be aggregated in nine time slots, whereas SPT 2 takes ten. The reason is that there are more dependency constraints in SPT 2. It is obvious that the child tree of node 5 always needs six slots to be aggregated; as the distance of node 5 from $sink$ is 2, the total aggregation time cannot be less than eight slots for SPT 2. Obviously, an SPT generated in a careless way is very likely to be suboptimal.

B. DST Based

The First-Fit algorithm is proposed by Huang *et al.* [4]. In the tree construction phase, breadth-first search is used to divide all nodes into layers. *Dominators* are then marked layer by layer. After adding some *connectors*, a connected dominating set (CDS) tree is formed; the rest nodes are *dominatees* (i.e., leaves). In the scheduling phase, the first-fit principal is adopted for a given set of edges: Edges are added into parallel transmission set one by one according to the interference-free principal; the process is repeated to find the maximal possible interference-free set for the next time slot; the procedure is iteratively applied to a set of edges until all edges are scheduled this way. All leaves are applied first; then, the scheduling is applied to nonleaf nodes layer by layer from bottom up.

Unfortunately, the interference-free judgment implementation in First-Fit is problematic, which is also found by other researchers [8]. Let S_r denote the set of $senders$ already selected in round r. For a new candidate z, First-Fit's implementation only tests if the candidate's transmission interference with the parents of S_r's nodes, whereas already scheduled transmissions could also interference with the candidate's parent, which is neglected in [4]. (Such a correct interference judgment procedure is presented later in Algorithm 2 in the Appendix.) For this reason, we focus on the work of Wan *et al.* [5] in this part.

As a successor, Wan *et al.* [5] developed a $15R + \Delta - 4$ approximation algorithm SAS. The basic framework of SAS is similar to that of First-Fit; the main difference is that the parents of leaves are no longer prefixed. Based on their proposed *minimum cover* concept, the leaf nodes' parents are dynamically determined during the scheduling process. Recently, Xu *et al.* [9] developed an approximation algorithm improved data aggregation scheduling (IAS) with bound $16R + \Delta - 14$. Similar to the work of Huang *et al.* and Wan *et al.*, it also divided the aggregation process into the tree construction

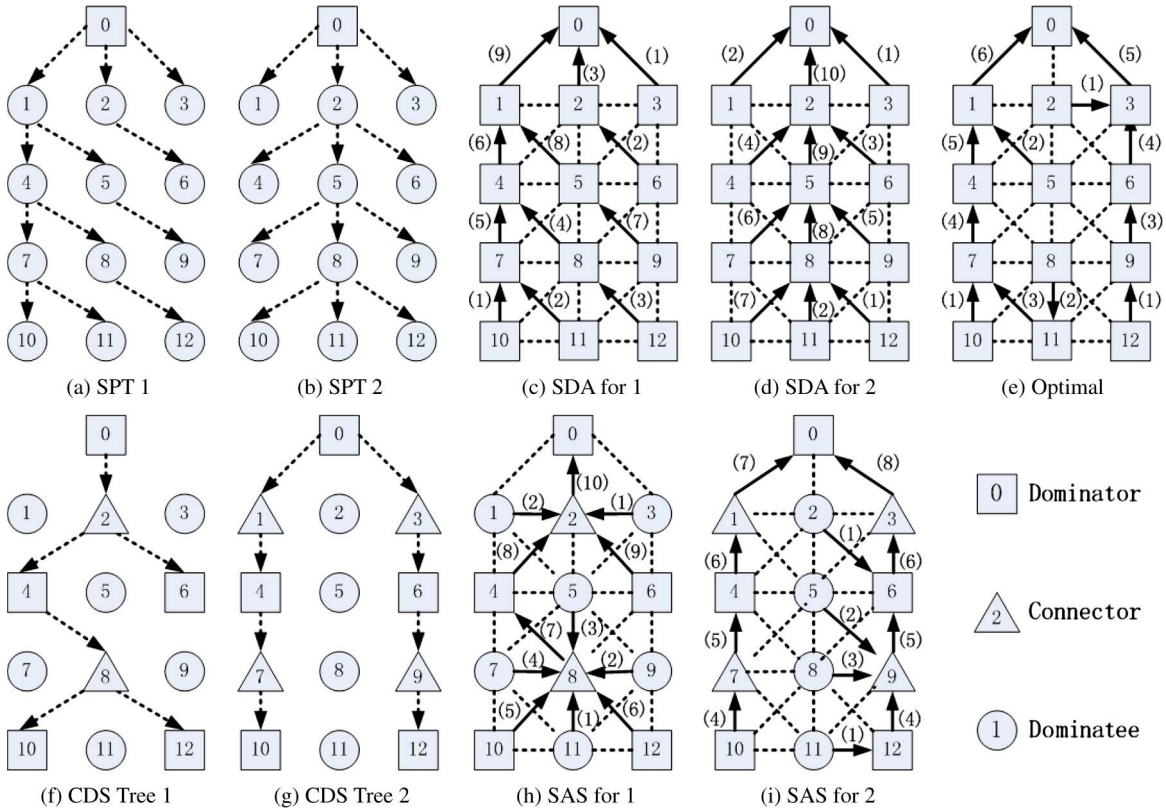


Fig. 3. (a) and (b) Two SPTs. (c) and (d) Respective SDA results for two SPTs. (e) Optimal solution. (f) and (g) Two CDS trees. (h) and (i) Respective SAS results for two CDS trees.

phase and the scheduling phase; it just confirmed a more strict bound by deleting the redundant collectors.

The performance of SAS also varies, depending on the construction of the CDS tree. We again take the example network in Fig. 2(a) for illustration. Two DSTs are given in Fig. 3(f) and (g). We deliberately use the same set of *dominators* and different sets of *connectors* for them. The results are shown in Fig. 3(h) and (i). It is observed that DST 1 can be aggregated in ten time slots, whereas DST 2 takes only eight slots. Similar to SPT, a DST generated in a careless way is likely to be suboptimal. Again, the role of each node is dominated by the pre-given tree, and the opportunity of parallel transmissions among nodes, particularly among nodes of different layers of the predefined tree, is greatly reduced.

C. Others

Yu *et al.* [8] proposed a distributed version of MLAS scheduling; Xu *et al.* [9] also presented a decentralized algorithm based on the centralized version. Pipelined approaches are presented in Wan *et al.*'s work [5]. In this paper, we limit ourselves to the primary topic of centralized simple-round-based scheduling. Possible pipelined or distributed versions of our algorithm can be extended and are left for future work.

Zhu and Hu presented a heuristic that surpassed SDA; however, geographical location of each node is required, which is a much stronger assumption than in other works and this paper [10]. Li *et al.* analyzed the problem in the context of a physical model [11]; in this paper, we still focus on the protocol model and consider the physical model as our future work.

Jia *et al.* [12] proposed the algorithm GIST for constructing an optimal data aggregation tree; this paper focused on a subgraph of the original network. In [13], the authors aimed to minimize the sum delay of sensed data rather than the maximum delay. In [14], the authors considered the problem of maximizing the lifetime of aggregation. All of these works are only loosely related to the topic of this paper.

IV. Solvable Mathematical Formulation

A. Auxiliary Graph

To the best of our knowledge, a solvable mathematical formulation for the MLAS problem is still an open question. In this part, we fill this gap by modifying the original digraph $\overrightarrow{G}$ to a new digraph $\overrightarrow{G}'$. As mentioned in Section II, $\overrightarrow{G}$ is obtained from G by replacing a link with two oppositely directed edges. For $\overrightarrow{G}'$, all edges starting from *sink* b are removed: there is no need to send data from the *sink*. Hence, only incidental edges are necessary. An auxiliary node b' (with index N) is added; a directional edge $b \rightarrow b'$ $(e_{K'})$ is also added, as shown in Fig. 4. Let b' be the *sink* of $\overrightarrow{G}'$, and let K' be the time slots in which edge $b \rightarrow b'$ is activated of an aggregation scheduling of $\overrightarrow{G}'$.

The intuition of adding auxiliary node b' is that, usually, sink b has multiple links, and each link could become the last scheduled send link; this situation makes the mathematical denotation of the optimizing objective extremely hard. By adding a virtual

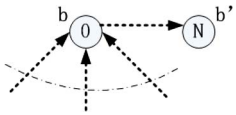


Fig. 4. New digraph of the example network. (a) With an auxiliary sink node.

TABLE I
SYMBOLS USED IN THIS PAPER

N	The number of nodes in $\overrightarrow{G}$; there are $N+1$ nodes in $\overrightarrow{G}'$.
i	One node in $\overrightarrow{G}'$.
M	The number of directional edges in $\overrightarrow{G}'$.
e	One directed edge in $\overrightarrow{G}'$.
$e_{K'}$	The last edge $b \rightarrow b'$ in $\overrightarrow{G}'$.
t	The time slot from 1 to N.
$S(i)$	The set of all edges with i as their sender in $\overrightarrow{G}'$.
$D(i)$	The set of all edges with i as their destination in $\overrightarrow{G}'$.
$W(e)$	The exclusive edges set of e in $\overrightarrow{G}'$, includes both edges that could have collision with e's destination and edges that could be affected by e's transmission.
$x_{e,t}$	The decision variable. $x_{e,t}=1$ if edge e is scheduled to transmit at slot t; $x_{e,t}=0$ otherwise.

sink b', the link $b \rightarrow b'$ is definitely the last scheduled active link: If $b \rightarrow b'$ is activated, then all data must be available in b. In that sense, the digraph $\overrightarrow{G}'$ is more convenient for the formulation of the optimizing objective function. A scheduling of $\overrightarrow{G}'$ can directly produce a scheduling of $\overrightarrow{G}$, with $K = K' - 1$.

B. Formal Formulation

Table I lists the symbols used in this paper.

Obviously, a naive solution for $\overrightarrow{G}'$ with $K' = N$ could be that all nodes, except b', send data packets (in one-by-one fashion) along an arbitrary inward tree. That is, K' should be less than N after the optimization. The objective function of MLAS becomes

$$\text{Min: } K' = 1 * x_{e_{K'},1} + 2 * x_{e_{K'},2} + \cdots + N * x_{e_{K'},N} \quad (1)$$

subject to the following constraints:

$$\begin{aligned}
&\text{(a)} \quad \sum_{e \in S(i)} \sum_{t=1}^{N} x_{e,t} = 1 \ \forall i \in \overrightarrow{G}' \setminus b' \\
&\text{(b)} \quad \sum_{t'=t+1}^{N} x_{e',t'} \leq 1 - \sum_{e \in S(i)} x_{e,t}, \ \forall i \in \overrightarrow{G}' \ \forall e' \in D(i) \ \forall t \\
&\text{(c)} \quad x_{e',t} \leq 1 - x_{e,t} \ \forall e \ \forall e' \in W(e) \ \forall t \\
&\text{(d)} \quad \sum_{e \in S(i)} x_{e,t} + \sum_{e' \in D(i)} x_{e',t} \leq 1 \ \forall i \in \overrightarrow{G}' \ \forall t. \qquad (2)
\end{aligned}$$

Constraint (a) guarantees only transmission per node (except for the auxiliary node b') throughout the whole aggregation. With $t' = t+1, \ldots, N$, constraint (b) ensures that, once a node transmits, it can no longer be the destination for any transmission (since its transmission implies the completion of the aggregation performed by that node and all its children). Constraint (c) captures the interference-free requirement. Constraint (d) ensures half-duplex operation that a sender cannot receive simultaneously when it has been assigned to transmit in a time slot.

C. Optimal Solution for the Example Network

One contribution of this paper is that we present the first solvable mathematical formulation of the problem. Thus, for small-scale networks (less than ten nodes), we can directly derive the optimal solution by existing optimization software, such as the Gnu linear programming kit (GLPK) tool [15].

We solve the mathematical formulation of the example graph by the GLPK tool [15]: It takes three days for the GLPK tool to find the optimal solution. The optimal scheduling, which takes only six time slots, is shown in Fig. 3(e). There is an apparent gap between the latency of the SPT/DST approach and the optimal scheduling. Based on these observations, we claim that *doing convergecast in an SPT is neither a necessary nor a sufficient condition*. A similar claim can be proposed for DST as well.

The optimal solutions of small networks give us insight of the MLAS scheduling. While for larger scale networks, it is still computationally infeasible, even if we have a solvable formulation, since MLAS is NP-complete. That is why we need polynomial-time approximation algorithms such as SPT/DST-based approaches or GGT, which is the main contribution of this paper.

V. GREEDY GROWING TREE

A. Basic Idea

Maximizing parallel transmissions is the key for an MLAS solution. Motivated by backward induction, we propose the GGT algorithm to schedule the aggregation process in a reverse order: The last aggregation step is scheduled first by selecting the last $sender$ to the *sink*, then the $\langle sender, receiver \rangle$ pairs of the next-to-last step, and so on, until all nodes are scheduled to send once.

The idea behind our proposed GGT algorithm is to construct larger and larger spanning trees rooted at the *sink*: the temporary spanning tree contains only the *sink* node at the beginning; in each round, all nonleaf nodes of the temporary spanning tree are the candidates of $receivers$; and all leaf nodes are the candidates of $senders$. Here, the $\langle sender, receiver \rangle$ set is selected in a manner such that the opportunity of parallel transmissions can be maximized; the new $senders$ are then added to generate an expanded temporary tree before entering the next iteration. It is noted that, in contrast to previous works, tree construction of GGT is directly guided by the scheduling algorithm step by step. For each temporary tree, all neighbored nodes are the candidates of $\langle sender, receiver \rangle$ pairs of the next step, to maximize the opportunity of parallel communications in each step. As a result, the problem is how to select the pairs in each round. Here, we present a *Principal* for the scheduling design:

The selection of $\langle sender, receiver \rangle$ pairs should consider how to maximize the opportunity of parallel transmission of both current and later rounds, and the choices that benefit later rounds should have priority.

TABLE II
SUBROUTINES USED

$Neighbor(i, V')$	The set of neighbors of a node i in a node set V' at G.
$NumNeighbor(i, V')$	The number of neighbors of a node i in a node set V' at G.
$ArticulationWeight(i)$	The weight of an articulation node i at G.
$PilotWeight(i)$	The weight of a pilot node i at G.
$CriticalWeight(i)$	The weight of a critical node i at G.
$SizeOf(V')$	the number of nodes in V'.

Due to the reverse scheduling nature of GGT, the selection of $senders$ in the *current* round would become nonleaf nodes of the temporary trees in all subsequent rounds. Merely focusing on maximizing the opportunity of parallelization of the *current* scheduling round is somehow shortsighted and very likely to impair scheduling in later rounds; the *Principal,* in fact, articulates that the chosen $senders$ should also serve as a "better" tree node for subsequent scheduling.[2]

Let r denote round index and T_r be the temporary tree. Table II lists the subroutines used in the algorithm. For candidate $senders$, we present three sorting rules to order them in a selection sequence.

1) *Priority Rule 1:* First, sort them based on the increasing order of $NumNeighbor(i, T_r)$.
2) *Priority Rule 2:* For nodes with the same order by *Priority Rule 1*, sort them based on the increasing order of $NumNeighbor(i, V \setminus T_r)$.
3) *Priority Rule 3:* For nodes with the same order by *Priority Rules 1* and *2*, sort them based on lexicographic order.

The intuition behind *Priority Rule 1* is that the less the number of neighbors of a node in T_r, the less the opportunity that the selected node's transmission would interference with subsequent candidates of $senders$, hence maximizing the opportunity of parallelization. The intuition behind *Priority Rule 2* is that, if a node with lower $NumNeighbor(i, V \setminus T_r)$ value is selected, then, in the next round, it would interface to a lower number of new leaves; hence, its opportunity of transmission interference as a $receiver$ is reduced. *Priority Rule 3* is used to finally break the ties.

B. Avoiding Local Optimization

An intuitive example is shown in Fig. 2(b), where, apparently, node 4 should be preferred, during the scheduling, from the global point of view: All nodes in its subtree cannot be scheduled until node 4 is scheduled. To prevent the GGT algorithm from falling into the pit of local optimization, we also identify special nodes in the topology that might have great impact over later rounds; those nodes should have priority during the scheduling process. Three types of such nodes are defined and listed by their priority.

1) *Articulation node*: If G becomes disconnected when we remove node i from G, then i is called an articulation node; its weight can be calculated as how many nodes are isolated from b because of its removal. This name comes from an *articulation point* in graph theory; one example is node 4 in Fig. 2(b) with a weight value of 5.
2) *Pilot node*: if a node i is on the shortest path from sink b to an articulation node, then i is called a pilot node; its weight can be calculated as the summation of the articulation node weight and its hops to the articulation node.
3) *Critical node*: the distance from sink b to all nodes can be summed up; if the summation increases when we remove node i from G, then i is called a critical node; its weight can be calculated as the gap between the old summation and the new summation. Two examples are nodes 6 and 7 in Fig. 2(b), each with a weight value of 2.

[2]Obviously, "better" here means it might improve parallelization.

C. Algorithm

We present our approximation algorithm in this section. The pseudocode of the GGT algorithm is shown in Algorithm 1.

Algorithm 1 GGT

Require:: MLAS instance (G, b)
1:$r \leftarrow 1, T_1 \leftarrow b$;
2:**while** $T_r \neq V$ **do**
3: $Z_r =$ leaves of T_r;
4: $Z_r = SortZr(G, T_r, Z_r)$;
5: $S_r \leftarrow \emptyset, D_r \leftarrow \emptyset$;
6: **while** $Z_r \setminus S_r \neq \emptyset$ **do**
7: $z \leftarrow$ next node in $Z_r \setminus S_r$;
8: Sort nodes $d \in Neighbor(z, T_r)$ based on increasing value of $NumNeighbor(d, Z_r)$;
9: **for** $d \in Neighbor(z, T_r)$ **do**
10: **if** $InterferenceJudge(S_r, z, d)$ **then**
11: $Neighbor(z, T_r) \leftarrow Neighbor(z, T_r) \setminus d$;
12: Continue;
13: **else**
14: $S_r \leftarrow S_r \cup z, D_r \leftarrow D_r \cup d$;
15: Break;
16: **end if**
17: **end for**
18: $Z_r \leftarrow Z_r \setminus z$;
19: **end while**
20: $T_{r+1} \leftarrow T_r \cup S_r$;
21: $r \leftarrow r + 1$;
22:**end while**
23:Output $S_k = S_{r+1-k}, D_k = D_{r+1-k}$;

GGT initially sets T_1 to be b (line 1) and then conducts a number of iterations. Each iteration performs a schedule of a round. In the rth iteration, T_r is a temporary tree rooted at b spanning all nodes scheduled until round $r-1$. From the nodes in Z T_r, GGT picks the leaves of T_r as candidate $senders$ for round r. Z_r is initially set to be the set of leaves of T_r (line 3); all nodes in Z_r are then sorted into a priority list by Algorithm 3 (line 4). The sender set S_r and receiver set D_r are empty at first (line 5).

After sorting Z_r, we examine nodes from Z_r one by one in a first-fit way (line 7): the new candidate $sender$ is tested against those nodes already in S_r/D_r based on the interference-free condition. Note that the interference test is bidirectional: the candidate $sender$ should have a $receiver$, which is not already in D_r. If there are multiple such candidate $receiver$, the one with the least $NumNeighbor(i, Z_r)$ value is selected (line 8): intuitively, this parent is the least likely to have interference with later candidate $senders$. If d does not pass the interference-free test, it is removed from the candidate $receiver$ set, and the next candidate $receiver$ is tested (lines 10–12). Otherwise, if d passes the test, z/d is added into S_r/D_r, respectively (line 14).

When the chance of acting as a $sender$ for every leaf in Z_r has been examined (line 18), nodes in S_r/D_r form the set of the $sender/recevier$ sets in round r. Subsequently, GGT adds S_r by setting $T_{r+1} = T_r \cup S_r$ and enters the $(r+1)$th iteration (line 20–21).

The interference-free condition is guaranteed by Algorithm 2.

Algorithm 2 InterferenceJudge

Require: : (S_r, z, d)
1:**for** $i = 0$ to $SizeOf(S_r)$ **do**
2: $p = S_r[i].destination$;
3: **if** $e(p, z) \in E$ **then**
4: return TRUE;
5: **end if**
6: **if** $e(S_r[i], d) \in E$ **then**
7: return TRUE;
8: **end if**
9: return FALSE;
10:**end for**

First, *InterferenceJudge* tests if the candidate $sender$ z would interference with S_r's parents (lines 2–5); then, it tests if the candidate $receiver$ d would be interfered by any already scheduled $sender$ in this round (lines 6–8). If z and d passes the check, a (interference) FALSE is returned (line 9); otherwise, a TRUE is returned (lines 4 and 7). In Algorithm 3, articulation nodes (lines 6–12), pilot nodes (lines 13–16), and critical nodes (lines 17–20) in Z_r are listed first; normal nodes in Z_r are listed on the order of *Priority Rules 1, 2,* and *3* (lines 20–26).

Algorithm 3 SortZr

Require: : (G, T_r, Z_r)
1:$Z_{\rm tmp} \leftarrow \emptyset$;
2:$ArticulationList \leftarrow$ Articulation Nodes of Z_r, sort based on $ArticulationWeight(i)$;
3:$PilotList \leftarrow$ Critical Nodes of Z_r, sort based on $PilotWeight(i)$;
4:$CriticalList \leftarrow$ Critical Nodes of Z_r, sort based on $CriticalWeight(i)$;
5:$NormalList \leftarrow$ Normal Nodes of Z_r;
6:**for** $j = 0$ to $SizeOf(ArticulationList)$ **do**
7: $i = ArticulationList[j]$;
8: $LowerBound = log2((SizeOf(V \setminus T_r) - ArticulationWeight(i))/SizeOf(T_r))$;
9: **if** $ArticulationWeight(i) > LowerBound$ **then**
10: Add i to the tail of $Z_{\rm tmp}$;
11: **end if**
12:**end for**
13:**for** $j = 0$ to $SizeOf(PilotList)$ **do**
14: $i = PilotList[j]$;
15: Add i to tail of $Z_{\rm tmp}$;
16:**end for**
17:**for** $j = 0$ to $SizeOf(CriticalList)$ **do**
18: $i = CriticalList[j]$;
19: Add i to tail of $Z_{\rm tmp}$;
20:**end for**
21:*Priority Rule 1:* Sort $NormalList$ based on $NumNeighbor(i, T_r)$;
22:*Priority Rule 2:* There always exist nodes that have the same number of neighbors in T_r, again we sort them according to $NumNeighbor(i, V \setminus T_r)$;
23:**for** $j = 0$ to $SizeOf(NormalList)$ **do**
24: $i = NormalList[j]$;
25: Add i to tail of $Z_{\rm tmp}$;
26:**end for**
27:Output $Z_r = Z_{\rm tmp}$;

D. GGT for the Example Network

Fig. 2(a) shows the topology of G. Fig. 5(a) shows the initial state of GGT, as described in Algorithm 1. There is no special node in the topology. Node 0 is the sole member of the T_1 tree. Then, nodes 1, 2, and 3 are leaves. In the first round [see Fig. 5(b)], nodes 1 and 3 have priority over node 2 based on *Priority Rule 2*; based on *Priority Rule 3*, the sequence is 1–3–2. Node 1 is first added into S_1; both nodes 3 and node 2 would interfere with node 1 because they share the same parent node 0. The scheduling result of round 1 is $S_r = 1$ and $D_r = 0$.

In the second round [see Fig. 5(c)], $T_2 = 0$, 1, and nodes 2, 3, 4, and 5 become leaves. Nodes 3, 4, and 5 have priority over 2 due to the *Priority Rule 1*. Sequence $3 > 4 > 5$ is determined by *Priority Rule 2*. Hence, the examine sequence is {3,4,5,2}. After scheduling, we get $S_2 = 3$, 4 and $D_2 = 0$, 1. The process continues in rounds 3–7 [correspondingly, Fig. 5(c)–(g)]. After seven rounds, all nodes are added into the growing tree. Finally, we reverse the sequence, and the final scheduling is shown in Fig. 5(h). An interesting finding is that our result use only seven slots, which is very close to the optimal scheduling [shown in Fig. 3(e)] for this network topology.

VI. Evaluation

A. Experiment Setup

We implemented SDA [3], WIRES [16], First-Fit [4], SAS [5], and IAS [9] algorithms to compare their performances with GGT. Note that we redress the interference judge procedure of First-Fit for fair comparison. We randomly deploy N sensors into a square region with edge length L; the density

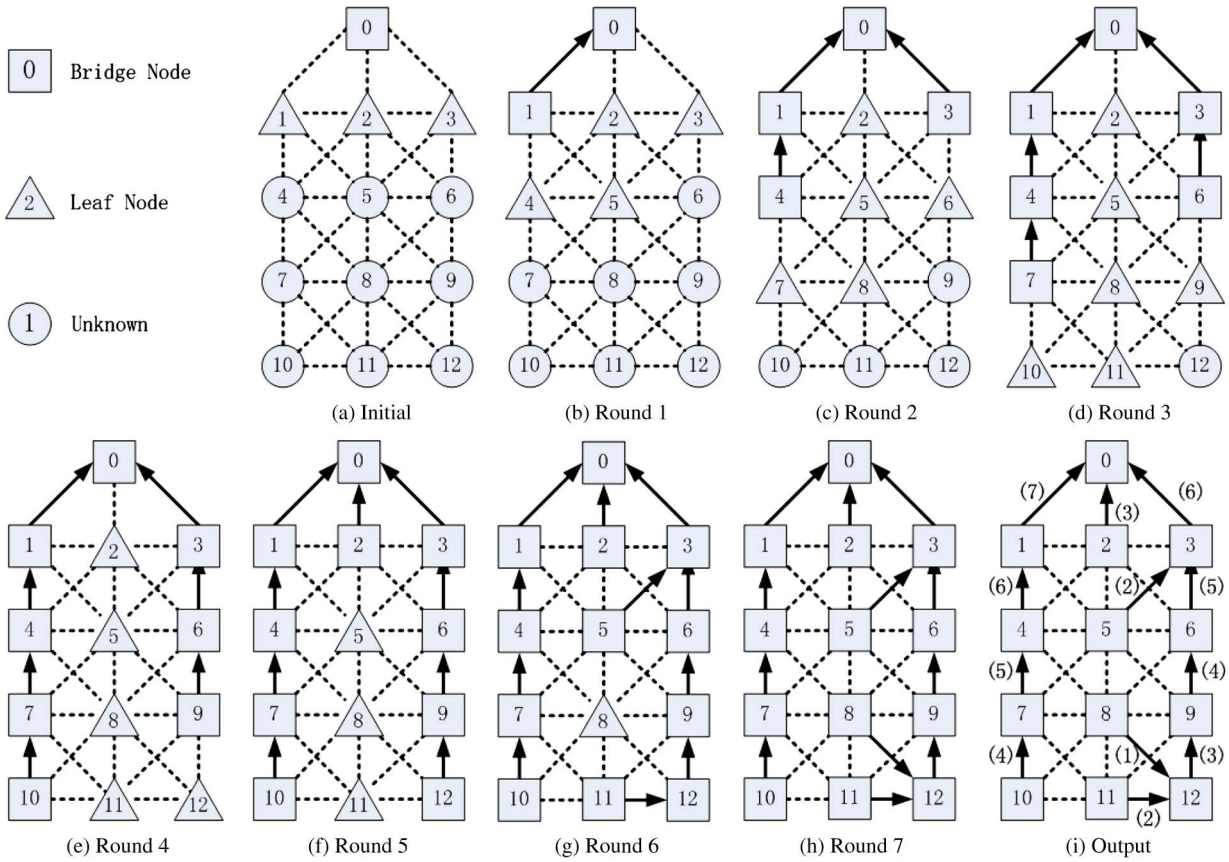


Fig. 5. (a) Initial tree. (b)–(h) Scheduling of each step. (i) Final output.

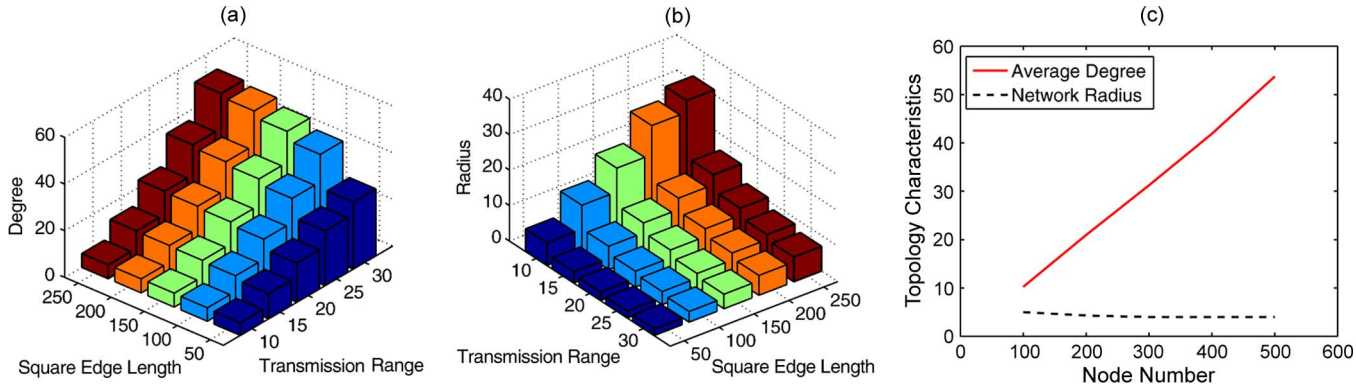


Fig. 6. Sink at the center increases λ from 10 to 30, with $L = 50$, 100, 150, 200, and 250. (a) Average node degrees. (b) Average network radius. (c) Increase density with $\lambda = 20$ and $L = 100$.

of nodes is determined by $O(N/L^2)$. All sensors have the same transmission range λ. For a randomly generated topology, its characteristics can be denoted by average node degree Φ and network radius R. These values reflect the topology characteristics.

All comparisons are fairly conducted on the same graph, and on each graph, the data are aggregated from the same set of nodes to the same *sink*. For each set of parameter configuration, we perform comparisons with ten random graphs and present the averaged result. For the MLAS problem, a well-known lower bound of the optimal solution is $\max\{\log_2(N), R\}$, where N is the number of sensors, and R is the radius of the network [9]. However, this bound is too loose. We use the SPT lower bound value provided in Malhotra's work [16] as an approximation of the overall lower bound and compare it with our simulation results.

B. Fixed Node Density

In this set of experiments, we keep node density fixed as $N/L^2 = 0.02$. For $L = 50$, 100, 150, 200, and 250, we have $N = 50$, 200, 450, 800, and 1250, respectively. The *sink* is located at the center. By varying λ for the same graph, the average degree Φ increases [as shown in Fig. 6(a)], and the network radius R decreases [as shown in Fig. 6(b)]. For the same λ value, degree Φ maintains stable with the increase of network

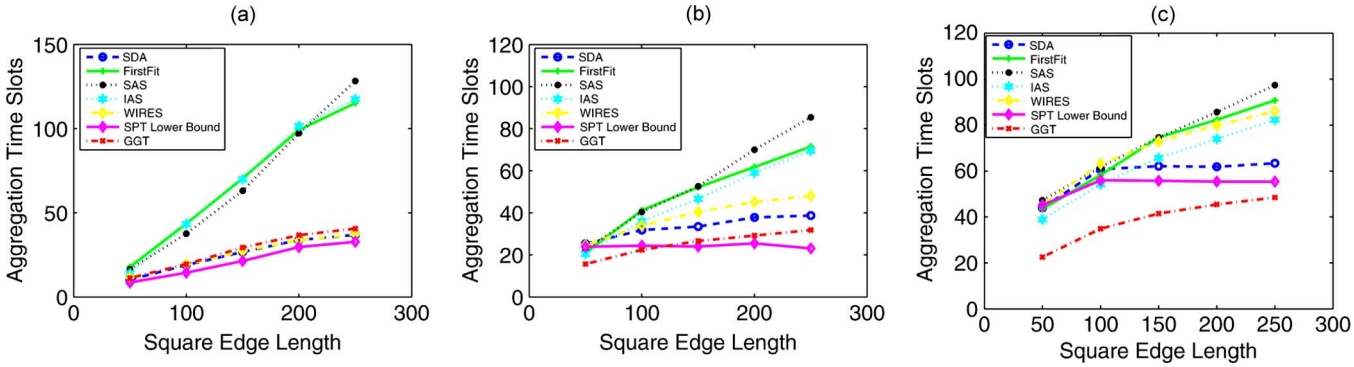


Fig. 7. With the sink at the center, the performance comparison with fixed node density and (a) $\lambda = 10$ ($\Phi \approx 5$), (b) $\lambda = 20$ ($\Phi \approx 20$), and (c) $\lambda = 30$ ($\Phi \approx 40$).

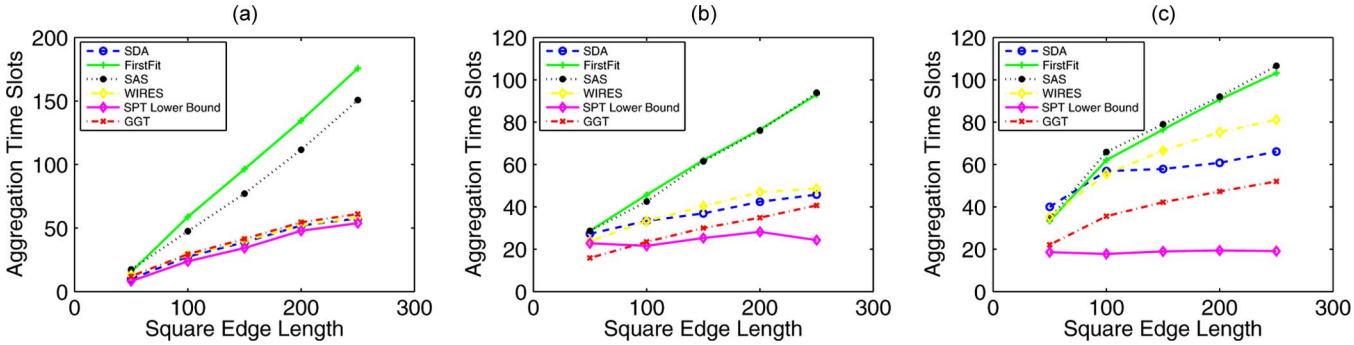


Fig. 8. With the sink at the corner, the performance comparison with fixed node density and (a) $\lambda = 10$ ($\Phi \approx 5$), (b) $\lambda = 20$ ($\Phi \approx 20$), and (c) $\lambda = 30$ ($\Phi \approx 40$).

[as shown in Fig. 6(a)], and the network radius R increases proportionally to network size [as shown in Fig. 6(b)].

Fig. 7 shows the number of time slots needed when the number of nodes N varies from 50 to 1250 (i.e., L from 50 to 250), with the λ value of 10, 20, and 30. Generally speaking, the average latencies of two DST approaches are similar: Although their theoretical upper bounds are better, their realistic performances are much inferior to the two SPT approaches and GGT.

In low-degree scenarios [see Fig. 7(a)], there is almost negligible performance gap between SPT and GGT; their performance are close to the SPT lower bound. After analyzing the results, we found that the reason is that randomly generated low-degree topologies have many $Articulation$ and $Critical$ nodes; SPT is also suitable for such topologies.

While in high-degree scenarios [e.g., $\lambda = 20/30$ with $\Phi \approx 20/40$ in Fig. 7(b) and (c)], the average latency of GGT is much lower than that of the others. The reason is that there are overwhelming contentions among nodes; the opportunity of parallel transmission is greatly reduced due to the layered nature of the predefined trees. As a comparison, GGT achieves nearly half of latency, compared with its competitors. We can also see that the SPT lower bound is even higher than the result of GGT in Fig. 7(c). The reason is that, in such a high-degree scenario, sink b degree dominates the lower bound value.

To prove the generality of the proposed GGT approach to a wider range of topology, the *sink* is moved to one corner of the region to emulate a skewed node distribution. We will not consider IAS in the sets of experiments since it always chooses a node located at the center of the WSN as the aggregation root. The experiments are repeated, and results are shown in Fig. 8. Again, the performance of GGT is much better than others in high-degree networks. In this experiment, the SPT lower bound is much lower than that in the previous experiment [e.g., Fig. 8(c)]: Since the sink is moved to a corner, its number of direct neighbors is greatly reduced; the consequence is that the sink degree no longer dominates the lower bound.

C. Variable Node Density

In this experiment, we keep λ and L fixed at 20 and 100, respectively, while varying N to change the node density. Again, the *sink* is located at the center. In Fig. 9, we present the number of time slots needed when the number of nodes N varies from 100 to 500. As shown in Fig. 6(c), the network radius R remains stable, and Φ increases proportionally to N.

Similar to previous results, the latency curve using GGT is significantly lower than that using others. As shown in Fig. 9, the more nodes in a fixed field, the better the performance gain of GGT over its competitors. The results show that the transmission parallelization is greatly improved by our approach. Mostly, the GGT results are close to the values of the SPT lower bound. This situation implies that GGT is effective and might be close to the overall optimum.

VII. Conclusion

MLAS seeks to minimize the number of scheduled time slots for data aggregation task in multihop wireless sensor networks.

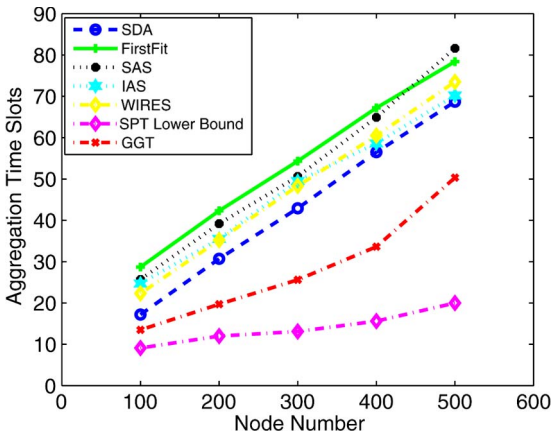


Fig. 9. Performance comparison with variable node density.

Existing aggregation schedules often utilize the two-phase approach: a tree construction phase followed by a scheduling phase. We argue the problem of the existing schedules: the performance of these algorithms could vary greatly, depending on the tree initially constructed; the opportunity of parallel transmission is greatly reduced due to the layered nature of the predefined tree.

In this paper, we have presented the first solvable mathematical formulation of the MLAS problem. The insight provided is that an optimal scheduling can be neither shortest path nor dominating set based. Inspired by backward induction theory, we proposed the GGT algorithm. In contrast with previous works, we gradually construct larger and larger spanning trees rooted at the $sink$; the tree construction is directly guided by the scheduling algorithm. The strictly defined priority rules maximize the opportunity of parallelization in each single round; as a result, the average aggregation time can be significantly reduced for high-degree networks.

REFERENCES

[1] H. Jiang, S. Jin, and C. Wang, "Parameter-based data aggregation for statistical information extraction in wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 8, pp. 3992–4001, Oct. 2010.

[2] V. Annamalai, S. K. S. Gupta, and L. Schwiebert, "On tree-based convergecasting in wireless sensor networks," in *Proc. IEEE WCNC*, 2003, pp. 1942–1947.

[3] X. Chen, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," in *Proc. MSN*, vol. 3794, *Lecture Notes Comput. Sci.*, 2005, pp. 133–142.

[4] S. C.-H. Huang, P. J. Wan, C. T. Vu, Y. Li, and F. Yao, "Nearly constant approximation for data aggregation scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2007, pp. 366–372.

[5] P.-J. Wan, S. C.-H. Huang, L. Wang, Z. Wan, and X. Jia, "Minimum-latency aggregation scheduling in multihop wireless networks," in *Proc. ACM MOBIHOC*, 2009, pp. 185–194.

[6] R. J. Aumann, *Backward Induction and Common Knowledge of Rationality—Games and Economic Behavior*. Amsterdam, The Netherlands: Elsevier, 1995.

[7] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[8] B. Yu, J. Li, and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2009, pp. 2159–2167.

[9] X. Xu, X. Li, X. Mao, S. Tang, and S. Wang, "A delay-efficient algorithm for data aggregation in multihop wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 1, pp. 163–175, Jan. 2011.

[10] J. Zhu and X. Hu, "Improved algorithm for minimum data aggregation time problem in wireless sensor networks," *Int J. Syst. Sci.*, vol. 21, no. 4, pp. 626–636, Dec. 2008.

[11] H. Li, Q. Sheng Hua, C. Wu, and F. C. M. Lau, "Minimum-latency aggregation scheduling in wireless sensor networks under physical interference model," in *Proc. ACM MSWiM*, 2010, pp. 360–367.

[12] L. Jia, G. Noubir, R. Rajaraman, and R. Sundaram, "GIST: Group-independent spanning tree for data aggregation in dense sensor networks," in *Proc. DCOSS*, 2006, pp. 282–304.

[13] C. Joo, J.-G. Choi, and N. B. Shroff, "Delay performance of scheduling with data aggregation in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[14] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum lifetime data gathering and aggregation in wireless sensor networks," in *Proc. ICN*, 2002, pp. 685–696.

[15] *Gnu Linear Programming Kit*. [Online]. Available: http://www.gnu.org/software/glpk/

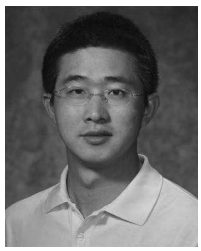
[16] B. Malhotra, I. Nikolaidis, and M. A. Nascimento, "Aggregation convergecast scheduling in wireless sensor networks," *Wireless Netw.*, vol. 17, no. 2, pp. 319–335, Feb. 2011.

Chen Tian (M'10) received the B.S., M.S., and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2000, 2003, and 2008, respectively.

He joined the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, as a Lecturer. His research interests include distributed networks, wireless networks, and network architecture.

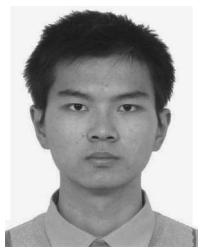
Hongbo Jiang (M'08) received the B.S. and M.S. degrees from Huazhong University of Science and Technology, Wuhan, China, and the Ph.D. degree from Case Western Reserve University, Cleveland, OH, in 2008.

He then joined the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, as an Associate Professor. His research interests include computer networking, particularly algorithms and architectures for high-performance networks and wireless networks.



Chonggang Wang (SM'09) received the Ph.D. degree in computer science from Beijing University of Posts and Telecommunications, Beijing, China.

He has conducted research with NEC Laboratories America, AT&T Labs Research, the University of Arkansas, and Hong Kong University of Science and Technology. He is currently with InterDigital Communications, King of Prussia, PA. His research interests include future Internet, machine-to-machine communications, and cognitive and wireless networks.



Zuodong Wu received the B.S. degree in 2010 from Huazhong University of Science and Technology, Wuhan, China, where he is currently working toward the M.S. degree with the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics.

His research interest is network protocols.

Jinhua Chen received the M.S. degree from Huazhong University of Science and Technology, Wuhan, China, in 2010.

He is currently with the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology. His research interest is network protocols.

Wenyu Liu (M'06) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1986 and the M.S. and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 1991 and 2001, respectively.

He is currently a Professor and Associate Chairman with the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology. His current research interests include computer graphics, multimedia information processing, and computer vision.