This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

Nearly Constant Approximation for Data Aggregation Scheduling in Wireless Sensor Networks

Scott C.-H. Huang*, Peng-Jun Wan*†, Chinh T. Vu‡, Yingshu Li‡ and Frances Yao*
*Computer Science Department, City University of Hong Kong
Kowloon, Hong Kong. Emails: {shuang,pwan,csfyao}@cityu.edu.hk
†Department of Computer Science, Illinois Institute of Technology
Chicago, IL 60616. Email: wan@cs.iit.edu
‡Department of Computer Science, Georgia State University
Atlanta, GA 30303. Emails: {yli,chinhvtr}@.cs.gsu.edu

***Abstract*— Data aggregation is a fundamental yet time-consuming task in wireless sensor networks. We focus on the latency part of data aggregation. Previously, the data aggregation algorithm of least latency [1] has a latency bound of $(\Delta - 1)R$, where Δ is the maximum degree and R is the network radius. Since both Δ and R could be of the same order of the network size, this algorithm can still have a rather high latency. In this paper, we designed an algorithm based on *maximal independent sets* which has an latency bound of $23R + \Delta - 18$. Here Δ contributes to an *additive* factor instead of a *multiplicative* one; thus our algorithm is *nearly constant approximation* and it has a significantly less latency bound than earlier algorithms especially when Δ is large.**

I. Introduction

A wireless sensor network consists of a self-configuring network of sensors equipped with RF transceivers and thus connected by wireless links. There is no infrastructure in such network and nodes are to organize themselves arbitrarily. These features make them suitable for a wide variety of applications like emergency medical situations, battlefield surveillance, traffic monitoring. In these applications, quite often we need to gather data from those sensors to a fixed sink and process them. If the data we gathered can be merged such as taking the maximum or minimum of them, we call this type of application *data aggregation*. Otherwise, we call it *data collection*.

In this paper, we focus on data aggregation and try to reduce its *latency* by constructing a good *schedule*. This is a challenging issue [2] [1], since singling out the interference issue and doing it at the MAC layer results in a latency too high to be practical. There is a must to do scheduling while dealing with it at the same time. Currently, the state-of-the-art scheduling algorithm has a latency bound of $(\Delta - 1)R$, where Δ is the maximum degree and R is the network radius. Since both Δ and R could be of the same order of the network size, this algorithm can still have a rather high latency. In this paper, we designed an algorithm based on *maximal independent sets* which has an latency bound of $23R + \Delta - 18$. Here Δ contributes to an *additive* factor instead of a *multiplicative* one; thus our algorithm is *nearly constant approximation* and it has a significantly less latency bound than earlier algorithms especially when Δ is large.

The rest of this paper is organized as follows. In §II we present related work. Problem formulation is presented in §III. We present our main data aggregation algorithm and analyze it in §IV. Conclusion and future work are presented in §VI.

This work was supported in part by the Research Grants Council of Hong Kong under Project Numbers CityU 1165/04E and CityU 122105 and by the National Science Foundation of the US under Grant Numbers 557904 and CCF-0545667.

II. Related Work

In wireless sensor networks, broadcast and data aggregation are the most fundamental and useful operations. Broadcast algorithms have been studied extensively in the literature since the 80's [3],[4],[5], [6],[7], [8],[9], [10],[11],[12], [13],[14]. Data aggregation, by comparison, is still relatively new but its importance cannot be overemphasized. Data aggregation, sometimes called *convergecast* is about a sensor network with a base station such that all (or some) nodes collect data and report to the base station via wireless communication. Annamalai *et al* [2] designed a heuristic algorithm for both broadcast and convergecast. The convergecast tree constructed in their algorithm can be used for broadcast as well. For convergecast alone, Upadhyayula *et al* [15] designed another heuristic algorithm aiming at reducing energy and latency. These two works mentioned above all used heuristic approaches and ran simulations to verify their results without doing theoretical analysis.

On the theoretical side, Kesselman and Kowalski [16] designed a randomized, distributed algorithm that has latency $O(\log n)$. In their model, they assume each node can vary its transmission range to reduce links. Chen *et al* [1] designed a $(\Delta - 1)$-approximation algorithm for data aggregation, where Δ is the maximum degree of the network graph. They also proved that the minimum data aggregation time problem is NP-hard.

0743-166X/07/$25.00 ©2007 IEEE

Practical issues of data aggregation, especially about the MAC layer, have also been studied in the literature. Huang and Zhang [17] studied packet loss and focused on reliability issues in data aggregation. Zhang *et al* [18] addressed the issue of bursty convergecast in real-time applications and focused on improving channel utilization and reducing retransmission-incurred channel contention. Krishnamachari *et al* [19] viewed data aggregation from another aspect. They considered the case where there is a subset of nodes whose data need to be sent to the base station and regard aggregating these data as a way to save energy. Intanagonwiwat *et al* , in a short paper [20], evaluated the impact greedy aggregation to increase the amount of path sharing and reduce energy consumption. Yu *et al* [21] also considered scheduling packet transmission and energy-latency tradeoff. Their goal was to reduce sensor nodes' energy dissipation subject to some latency constraints.

III. Problem Formulation

We consider a network consisting of n sensor nodes along with one base station. Each (sensor) node is equipped with an RF transceiver that can be used to send or receive data. We consider omni-directional antennae only, and a node's transmission/reception range is roughly a disk centered at that node. For simplicity, we further assume that all nodes have the same transmission range, and this type of network can be represented as a *disk graph* as follows. Let G be a graph representing a network of n nodes. An arc (or directed edge) exists from u to v if and only if v lies in u's transmission area (which is a disk). If all nodes have the same transmission range, then we can normalize their radius to 1 and use a *unit disk graph* to model it. In this case, an edge exists between u, v if and only if the distance between them is less than 1. For simplicity, we only consider unit disk graphs.

A node can either send or receive data at one time, and it can receive data correctly only if exactly one of its neighbors is transmitting at that moment. If two or more nodes are transmitting simultaneously and there is a node in their overlapped transmission area, then this node cannot receive the message clearly since both transmissions are interfering with each other. This type of situation is called *collision*. The main task of sensor nodes is to collect data and transmit back to the base station, and data can be 'aggregated' all the way to the base station. In other words, if a node has received one packet from its neighbor before its scheduled transmission time, then it can merge this packet with its own data packet and simply sends this merged packet later. This model is particularly useful in situations like making a query about the maximum temperature, in which two (or more) pieces of data can be merged by taking their maximum. Another situation where packets cannot be merged does exist, and it is called *data collection*. In this paper we only focus on *data aggregation* meaning that data can be merged all the way to the base station.

Given a unit disk graph $G = (V, E)$ along with a base station $b \in V$. Consider two subsets $A, B \subset V$ with $A \supset B$. We say that data are *aggregated* from A to B by one time slot if all nodes in $A - B$ transmit in one slot simultaneously then the data at all nodes of $A - B$ will be received collision-free by some nodes in B.

A data aggregation *schedule* can be thought of as a sequence of senders $\{S_1, S_2, \ldots\}$ (in which $S_i \subset V$, $\forall i$) satisfying the *data aggregation property*. This sequence represents that all nodes in S_1 transmit in the first time slot, followed by all nodes in S_2 transmitting in the second time slot and so on so forth. The data aggregation property simply means that after S_1 transmits data will be aggregated from V to $V - S_1$ and after S_2 transmits data will be further aggregated from $V - S_1$ to $V - S_1 \cup S_2$. If we continue this process, finally all data will be aggregated to one single node b, which is the base station. Without loss of generality we may assume each node only transmits once, since multiple transmission does not result in any advantage for the following reason. If, in a schedule, a node transmits more than once, then the data at this node will be aggregated many times to some other nodes. Since a single piece of this information alone is enough as long as it can eventually get to the base station collision-free, we can always modify the schedule by properly removing multiple transmissions while preserving the same latency (number of time slots that all data are aggregated to b). The property of single transmission is essentially equivalent to all S_i's being disjoint. The formal definition of the data aggregation property is as follows.

A data aggregation schedule is a sequence of senders $\{S_1, S_2, \ldots\}$ satisfying the following conditions.

1) $S_i \cap S_j = \emptyset$, $\forall i \neq j$
2) $\bigcup_{i=1}^{l} S_i = V - \{b\}$
3) Data are aggregated from $\overline{\bigcup_{i=1}^{k} S_i}$ to $\overline{\bigcup_{i=1}^{k+1} S_i}$ for all $k = 1, 2, , \ldots, l-2$ and finally data are aggregated from $\overline{S_1 \cup \cdots S_{l-1}}$ to b in time slot l.

The number l is called the *data aggregation latency*. Now, the minimum data aggregation time problem can be formulated as follows. Given a graph $G = (V, E)$ and a base station $b \in V$, find a data aggregation schedule with minimum latency. This problem is proven to be NP-hard even for unit disk graphs [1]. So far, there is no good approximation algorithm for this problem, since the best known algorithm has ratio $\Delta - 1$ (where Δ is the maximum degree and it can be as large as the number of nodes in the network).

In the next section, we will present an approximation algorithm for this problem that has latency $23R + \Delta - 18$, where R is the network radius. Since the farthest node has to transmit back to the base station, data aggregation latency cannot be less than the network radius. If R is large, our algorithm has approximation ratio 27 from an asymptotic point of view.

IV. Our Data Aggregation Scheduling Algorithm

We present our approximation algorithm in this section. Our algorithm has a data aggregation latency $23R + \Delta - 18$ where R is the network radius and Δ is the maximum degree of the network. This result is significantly better than the currently

best known algorithm [1] whose latency is $(\Delta - 1)R$, since Δ becomes an *additive* term instead of multiplicative. If Δ is large, our algorithm achieves a significantly shorter latency. Scenarios with large Δ happens frequently in large-scale, dense networks in real life. Our algorithm has two phases.

(1) *Data Aggregation Tree Construction* (2) *Data Aggregation Scheduling*. Their details will be presented in next two subsections.

A. Data Aggregation Tree Construction

In this phase, we first construct a breadth first search tree for the network and divide all nodes into layers (where the 0-th layer is the base station and the 1st layer is its neighbors). We then form a maximal independent set $BLACK$ layer by layer (Algorithm 1) as follows. Starting from the 1st layer, we pick up a maximal independent set and mark these nodes black. We then move on to the 2nd layer and pick up a maximal independent set and mark these nodes black again. Note that the black nodes of the 2nd layer also need to be independent of those of the 1st layer. We repeat this process until all layers have been worked on. Those who are not marked black are marked white at last. The pseudocode of layered MIS construction is given in Algorithm 1.

Algorithm 1 Construct an MIS layer by layer

Divide all nodes into layers $L_1, L_2, \ldots, L_l$
$BLACK \leftarrow \emptyset$
for $i \leftarrow 1$ to l **do**
    Find an MIS $BLACK_i \subset L_i$ indep. of $BLACK$
    $BLACK \leftarrow BLACK \cup BLACK_i$
end for
return $BLACK$

To construct the data aggregation tree, we then pick some of the white nodes and color them blue to interconnect all black nodes as follows. To connect the 1st and 2nd layer, we look at the 2nd layer's black nodes. Each black node must have a parent on the 1st layer and this parent node must be white since black nodes are independent of each other. We add this white parent for each black node on the 2nd layer and also add an edge between them. These added white nodes are colored blue. Moreover, we know that this blue node must be dominated by a black node either on the 1st layer or the 0-th layer. We then add an edge between this blue node and its dominator. We repeat this process layer by layer and finally obtain the desired data aggregation tree in this manner. Note that, in this tree, each black node has a blue parent at the upper layer and each blue node has a black parent *at the same layer or the layer right above*. The pseudocode is given in Algorithm 2.

B. Data Aggregation Scheduling

The scheduling for data aggregation is quite straightforward. We use first-fit scheduling (Algorithm 3) throughout this part.

The first-fit procedure takes in an input of a sender set S, a receiver set R, the network graph G, and the data aggregation

Algorithm 2 Data aggregation tree construction

$T = (V_T, E_T)$, $V_T = V$, $E_T \leftarrow \emptyset$
▷ /* Connect black nodes layer by layer */
for $i \leftarrow 1$ to $l - 1$ **do**
    for all black nodes $v \in BLACK_{i+1}$ **do**
        Find its parent $p(v)$ in BFS tree
        Color $p(v)$ blue
        Find $p(v)$'s dominator $d_{p(v)}$ in $BLACK_i \cup BLACK_{i-1}$
        Add an edge between $p(v), v$ to E_T
        Add an edge between $d_{p(v)}, p(v)$ to E_T
    end for
end for
▷ /* Connect remaining white nodes */
for all remaining white nodes u **do**
    Find u's dominator d_u
    Add an edge between u, d_u to E_T
end for
return T

tree T. The senders are scheduled as follows. First we pick a node s_1 from S and look at whether it conflicts with any nodes or not. Obviously since it is the only node, it does not conflict with any nodes. We then add s_1 to a temporary set X and remove s_1 from S. Now we pick up another node s_2 from S and look at whether it conflicts with any node in X or not. We look at the parent of all nodes in X in T. Since there is only one node s_1 in X, we only look at the parent of s_1 in T, denoted by $p_T(s_1)$, and see whether s_2 is adjacent to $p_T(s_1)$ in G. If yes, we do nothing. If not, s_2 does not conflict with X and we add s_2 to X and remove s_2 from S. We repeat this process until we find the largest set X such that all other nodes in S will conflict at least one node in X. This X is the maximal possible set that we can schedule them to transmit for the same time slot, so we add X to the schedule. Now we initialize X to $\emptyset$ and repeat this process to find the maximal possible set to transmit simultaneously for the second time slot, and so on so forth until all elements in S are scheduled this way. The pseudocode for this is given in Algorithm 3.

Now we have all white nodes transmit to black nodes using the first-fit subroutine described above. Then, from the last layer S_l we aggregate data layer by layer as follows. The black nodes at layer l, denoted by $BLACK_l$, first transmit to their parent blue nodes $BLUE_{l-1}$, then $BLUE_{l-1}$ transmit to their parent black nodes $p_T(BLUE_{l-1})$ in T. Note that $p_T(BLUE_{l-1})$ must be a set of black nodes in $BLACK_{l-1} \cup BLACK_{l-2}$ because the black nodes were selected layer by layer. These transmissions are all scheduled using the first-fit subroutine to avoid collision and we repeat this process layer by layer. Data will therefore be aggregated layer by layer and going all the way to the base station. The pseudocode for this is given in Algorithm 3.

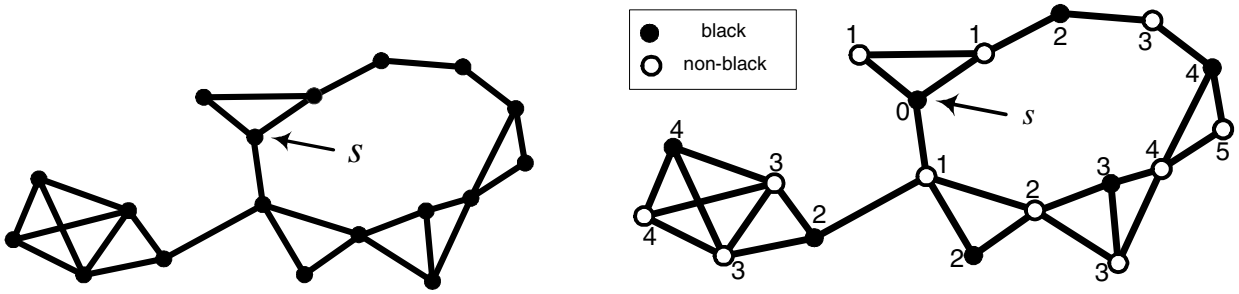


Fig. 1. (a) Topology of G (b) Black node selection (numbers represent hop-distance from s)

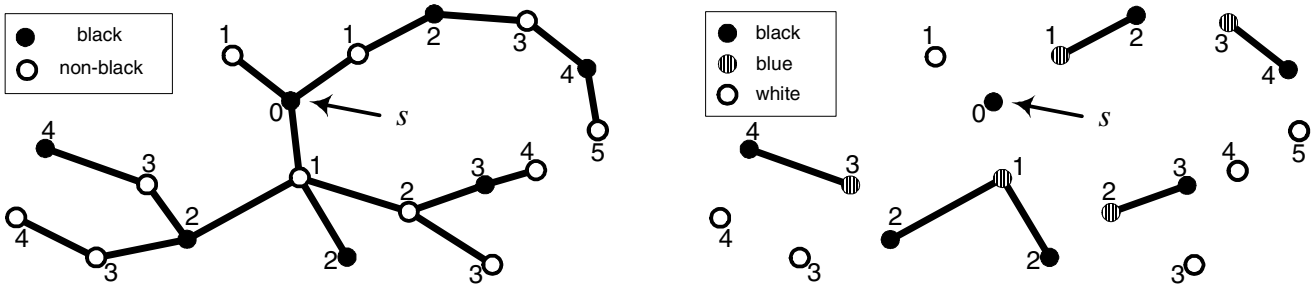


Fig. 2. (a) BFS tree of G (b) Blue node selection

Algorithm 3 First-fit scheduling

Input: Sender set S, Receiver set R, network $G = (V, E)$, data aggregation tree T
Output: Schedule $\mathcal{W}$, a collection of subsets in V

procedure FIRSTFIT(S, R, G, T)
  $X \leftarrow \emptyset$
  $\mathcal{W} \leftarrow \{\emptyset\}$
  while $S \neq \emptyset$ **do**
    for all $u \in S$ **do**
      if $\forall x \in X$, $\big(p_T(x), u\big) \notin E$ **then**
        Add u to X and remove u from S
      end if
    end for
    Add X to $\mathcal{W}$
  end while
  return $\mathcal{W}$
end procedure

▷ $p_T(x)$ = parent of x in T

Algorithm 4 Data aggregation scheduling

Construct an MIS layer by layer ▷ Algorithm 1
Construct the data aggregation tree T ▷ Algorithm 2
FIRSTFIT($WHITE, BLACK, G, T$)
for $i \leftarrow l - 1$ to 1 **do**
  FIRSTFIT$\big(BLACK_i, BLUE_{i-1}, G, T\big)$
  FIRSTFIT$\big(BLUE_{i-1}, p_T(BLUE_{i-1}), G, T\big)$
end for

C. An Example

Figure 1(a) shows the topology of G, and figure 1(b) shows the selection of layered MIS (i.e. black nodes) as described in algorithm 1. In the first step, the source s is selected in the MIS and colored black. In the second step, since the source is black, all nodes at layer 1 must all be white, otherwise it won't be independent of s. In the third step, we will select an independent set at layer 2, which must also be independent of the nodes at the previous layer, namely layer 1, although there is no black node at layer 1 and this restriction does not have any effect on selecting independent set at layer 2. In figure 1(b), we show the hop-distance of each node to s. Note that 3 black nodes were selected at layer 2, 1 black node was selected at layer 3, and 2 black nodes were selected at layer 4. We keep doing this until all layers have been worked on. The black node selection merely depends on the topology of G and it has nothing to do with the BFS tree. Not until we select blue nodes do we need to consider the BFS tree.

Figure 2(a) shows the BFS tree of G as well as the black nodes. Now, we try to add appropriate blue nodes to interconnect those black ones, as described in algorithm 2. Since the source does not have an upper layer and there are no black nodes at layer 1, we start from layer 2 directly. For each black node v at layer 2 (there are 3 of them), we find its parent $p(v)$ in the BFS tree, color it blue, and connect v to $p(V)$ as shown in figure 2(b). Here we see 2 nodes at layer 1 are colored blue and connected to their black children at layer 2. Those which are not colored blue remain white, and there is 1 such node. We also connect these 2 blue nodes and the white node to the source s since they are dominated by s. We keep working on layer 3, find 1 black node at this layer, and color

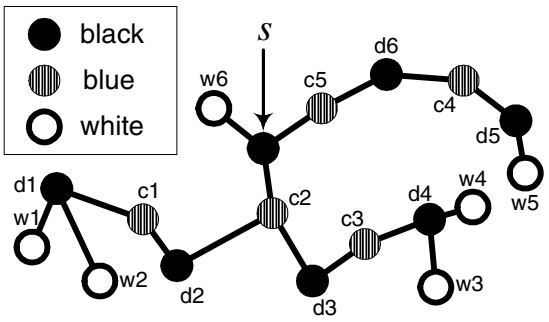


Fig. 3. Data aggregation tree

its parent blue. Note that this black node has to be independent of all black nodes at above layers (i.e. layers 0,1,2) as well. We then connect the blue node at layer 2 to its black dominator, as shown in Figure 3. We repeat this process for every layer until all nodes are connected and the data aggregation tree is fully built, as shown in figure3.

To schedule the data aggregation transmission, we follow algorithm 3. First we schedule the white nodes. We arbitrary pick a node, say w1, and schedule its transmission for the most available time slot, which is slot #1. We then pick w2 and find that we cannot schedule its transmission in slot #1 since w1 is already scheduled to transmit in this slot which will interfere w2's transmission. Then we move to w3 and find we can schedule it for #1 as well. However, when we look at w4 we find we cannot schedule it for #1 because of w3. We keep doing this until we examine all white nodes and find that w1,w3,w5,w6 can be scheduled concurrently for time slot #1. Now we schedule the remaining white nodes w2 and w4 using the same method, and we find that, similarly, they can be scheduled concurrently for #2. Now we schedule the transmission from $BLACK_4$ to $BLUE_3$. We find the there are only two nodes in $BLACK_4$, namely d1 and d5, and we use the first-fit algorithm to schedule them. We find that they can both be scheduled for #3. Then we schedule $BLUE_3$ to $p_T(BLUE_3)$ and schedule c1 and c4 for #4. Note that $p_T(BLUE_3)$ could be at layer 2 or 3. We use this method layer by layer until all nodes have been scheduled this way.

D. Analysis

Here we show that it takes $23R + \Delta - 18$ to aggregate data to the base station. We look at the abovementioned 3 parts separately.

(1) *WHITE to BLACK*: This part only has to be done once for the whole schedule and it has nothing to do with layers. We claim that it takes at most $\Delta - 1$ time slots to finish this part. This can be proved in a straightforward manner. Note that each black node has at most Δ neighbors. Since black nodes are mutually independent, black nodes cannot be neighbors of each other. Thus, these Δ (or less) neighbors must be either white or blue. However, each black node must have one blue parent, so there must be at least one blue nodes among these Δ neighbors. As a result, a black node can have at most $\Delta - 1$ white neighbors and the first-fit scheduling is guaranteed to finish within $\Delta - 1$ time slots.

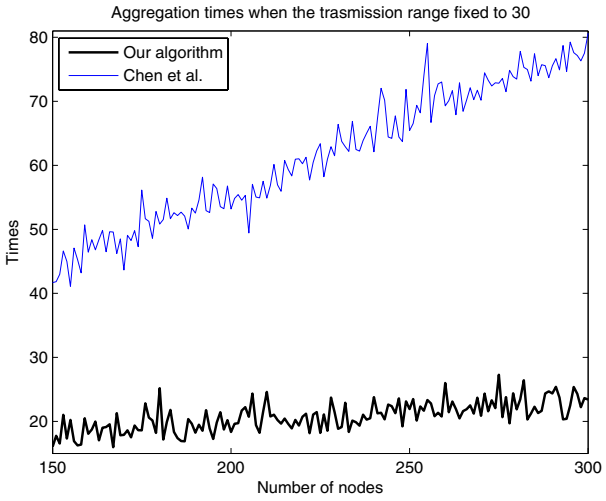
(2) *BLACK to BLUE*: This part and the next one have to be done repeatedly. When data are aggregated to the previous layer this part has to repeat. We claim that it takes at most 4 time slots to finish the transmission. We pick up a black node. This black node has to transmit to its blue parent. Now we look at this blue parent node and observe that it has at most 5 black neighbors. This follows from the fact that there are at most 5 independent points in a unit disk. Among this 5 or less black neighbors, one of them must be this blue node's parent. Therefore, there can be at most 4 black node competing to transmit with respect to this blue node and it takes at most 4 time slots to finish the transmission.

(3) *BLUE to BLACK*: This part has to be done layer by layer as described above. We claim that it takes at most 19 time slots to finish the transmission except on the second layer's transmission to the first, which needs 20 time slots. Similarly, picking up a blue node, we need to estimate how many other blue nodes are competing to transmit to the same black parent. Fix this black parent node u and consider the unit disc D_1 centered at it, and we count how many blue nodes lie inside this disk. Note that each blue node must have at least a black child because blue nodes are selected to interconnect black nodes. For this reason the number of blue nodes lying inside this unit disk cannot exceed the number of black nodes lying inside D_2, the disk of radius 2 centered at u. According to Lemma 1.1 in the appendix, there can be at most 21 independent points in a disk of radius 2. In other words, there can be at most 21 black nodes in a disk or radius 2 centered at every black node, and each black node u can have at most 20 2-hop black neighbors (since we can exclude u itself). Moreover, we are going to show that we can subtract one more black node except for the second layer. The reason is that among these black points in D_2 one of them must be node u's parent's parent (which must also be black and lie inside D_2). Therefore, for all other layers, 19 time slots are enough.

Now we estimate the data aggregation latency. WHITE-to-BLACK needs $\Delta - 1$ time slots. From layer R to layer 2, BLACK-to-BLUE and BLUE-to-BLACK together needs $(19 + 4) \cdot (R - 2)$ time slot. Transmission from layer 2 to layer 1 needs 20+4 time slots and transmission from layer 1 to the base station needs 5 time slots (since there are at most 5 independent points in a unit disk). Altogether, it takes at most $\Delta - 1 + 23(R - 2) + 24 + 5 = 23R + \Delta - 18$ time slots, as desired.

V. Simulation Results

For the simulation part, we randomly deploy sensors into a fixed region of size $100m \times 100m$. All sensors have the same transmission range. We compare our algorithm with Chen *et al* [1]. All comparisons are fairly conducted on the same graph, and on each graph data are aggregated from the same set of nodes to the same base station. Throughout this simulation, we always use number 0 as the root node and all leaves of the



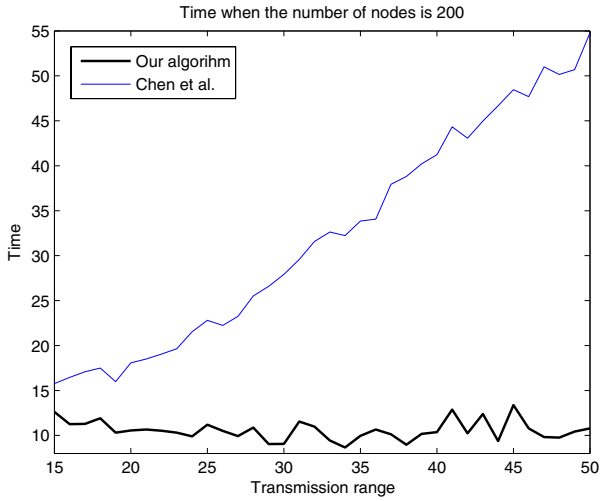


Fig. 4. (a) (b)

BFS tree rooted at node 0 as the data-storing nodes, i.e., nodes that we aggregate data from. Figures 4(a) and (b) represent the fluctuation in those graphs. For each configuration (same number of nodes, same transmission range), we do comparison with 100 random graphs.

In Figure 4(a), the transmission range of each sensor is fixed to $30m$. We measure the number of time slots needed to aggregate data from the leaves to the roots when the number of nodes varies from 150 to 300. In [1], the data aggregation latency (vertical axis: time) is basically proportional to the number of nodes, while in our algorithm it is not influenced much by the number of nodes. The pattern of those curves matched with our theoretically estimated latency bound. The bigger the number of nodes, the better the improvement of our algorithm in comparison with Chen's. Our algorithm's latencies are from 2.02 to 3.87 times smaller than that in [1].

In Figure 4(b), we do it with the graph of 200 nodes but the transmission range varies from $15m$ to $50m$. Similar to Figure 4(a), the curve representing time generated by Chen's algorithm is nearly linear, while the one generate by algorithm is so close to a horizontal line. The pattern of those curves illustrates the theoretical analysis in § IV-D.

VI. Conclusion and Future Work

In this paper, we investigated the data aggregation problem and considered its latency. We used the techniques of maximal independent sets and designed an algorithm of latency $23R + \Delta - 18$. For future work, we describe our short-term, medium-term, and long-term goals as follows.

Our short-term goal is applying our techniques to directional antennae. We believe that most techniques we developed here can be applied to the case of directional antennae immediately by simply re-investigating the geometrical property of it. Also, we will consider more general models than UDGs. For example, distinguishing the transmission range from the interference range and using two disks or other geometrical objects to represent them. Also, we'll investigate the data aggregation scheduling in Signal-to-Noise model.

Our medium-term goal is to find a better lower bound for optimal solution. In analyzing the approximation ratio, we actually used R as the lower bound. This is a very loose estimation and according to our test cases it is significantly larger than R, in most cases at least the clique number ω. However, we did find a counterexample such that the optimal latency is $O(\log \omega)$ and we simply could not find anything better so far. In the future work, we think it is likely to find a tighter lower bound for optimal latency, and the analysis part of this algorithm may be improved this way.

Our long-term goal is data collection scheduling. We studied data aggregation first because it has a simple model. In data collection, since data cannot be merged, nodes may need to transmit multiple times. This will essentially make our scheduling impractical. However, we believe that the MIS or some similar techniques may still be applied.

Appendix

We state the following theorem from [22] and we'll use it to prove Lemma 1.1 later.

Theorem 1.1 (Wegner Theorem): The area of the convex hull of any $n \geq 2$ non-overlapping unit-radius circular disks is at least $2\sqrt{3}(n-1) + \left(2-\sqrt{3}\right)\left\lceil\sqrt{12n-3}-3\right\rceil + \pi$.

Lemma 1.1: There can be at most 21 black nodes within any disk of radius two.

(*Proof.*) Fix a disk D_2 centered at a point u. Let S denote the set of black nodes in D_2. Since black nodes are mutually independent. If, for each node in S, we consider a disk of radius 1/2 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 2.5 centered at u. By applying Wegner Theorem with proper scaling, we have

$$2\sqrt{3}\left(|S|-1\right) + \left(2-\sqrt{3}\right)\left\lceil\sqrt{12\,|S|-3}-3\right\rceil + \pi < 25\pi.$$

Straightforward calculation shows that the maximum integer to make the above expression hold is $|S| = 21$. □

REFERENCES

[1] X. Chen, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," in *1st Int'l Conference on Mobile Ad-hoc and Sensor Networks–MSN'05*, pp. 133–142, 2005.

[2] V. Annamalai, S. K. S. Gupta, and L. Schwiebert, "On tree-based convergecasting in wireless sensor networks," in *IEEE Wireless Communications and Networking–WCNC'03*, vol. 3, pp. 1942–1947, 2003.

[3] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg, "A lower bound for radio broadcast," *Journal of Computer and System Sciences*, vol. 43, no. 2, pp. 290–298, 1991.

[4] R. Bar-Yehuda, O. Goldreich, and A. Itai, "On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization," *Journal of Computer and System Sciences*, vol. 45, no. 1, pp. 104–126, 1992.

[5] D. Bruschi and M. Del Pinto, "Lower bounds for the broadcast problem in mobile radio networks," *Distributed Computing*, vol. 10, no. 3, pp. 129–135, 1997.

[6] I. Chlamtac and S. Kutten, "On broadcasting in radio networks–problem analysis and protocol design," *IEEETransactions on Communications*, vol. 33, pp. 1240–1246, 1985.

[7] I. Chlamtac and O. Weinstein, "The wave expansion approach to broadcasting in multihop radio networks," *IEEETransactions on Communications*, vol. 39, pp. 426–433, 1991.

[8] M. Elkin and G. Kortsarz, "Logarithmic inapproximability of the radio broadcast problem," *Journal of Algorithms*, vol. 52, pp. 8–25, 2004.

[9] M. Elkin and G. Kortsarz, "Polylogarithmic additive inapproximability of the radio broadcast problem," in *7th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems–APPROX'04*, 2004.

[10] M. Elkin and G. Kortsarz, "An improved algorithm for radio networks," 2005. An earlier version appeared in *SODA'05*.

[11] I. Gaber and Y. Mansour, "Centralized broadcast in multihop radio networks," *Journal of Algorithms*, vol. 46, no. 1, pp. 1–20, 2003.

[12] R. Gandhi, S. Parthasarathy, and A. Mishra, "Minimizing broadcast latency and redundancy in ad hoc networks," in *ACM MobiHoc'03*, pp. 222–232, 2003.

[13] D. R. Kowalski and A. Pelc, "Centralized deterministic broadcasting in undirected multi-hop radio networks," in *7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems–APPROX-RANDOM'04*, pp. 171–182, 2004.

[14] E. Kushilevitz and Y. Mansour, "An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks," *SIAM Journal on Computing*, vol. 27, pp. 702–712, 1998.

[15] S. Upadhyayula, V. Annamalai, and S. K. S. Gupta, "A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks," in *IEEE Global Telecommunications Conference–GLOBECOM'03*, vol. 6, pp. 3525–3530, 2003.

[16] A. Kesselman and D. Kowalski, "Fast distributed algorithm for convergecast in ad hoc geometric radio networks," in *2nd Annual Conference on Wireless On-demand Network Systems and Services–WONS'05*, pp. 119–124, 2005.

[17] Q. Huang and Y. Zhang, "Radial coordination for convergecast in wireless sensor networks," in *29th Annual IEEE International Conference on Local Computer Networks–LCN'04*, (Washington, DC, USA), pp. 542–549, IEEE Computer Society, 2004.

[18] H. Zhang, A. Arora, Y.-R. Choi, and M. G. Gouda, "Reliable bursty convergecast in wireless sensor networks," in *6th ACM int'l Symposium on Mobile Ad Hoc Networking and Computing–MobiHoc'05*, (New York, NY, USA), pp. 266–276, ACM Press, 2005.

[19] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *22nd Int'l Conference on Distributed Computing Systems–ICDCSW'02*, (Washington, DC, USA), pp. 575–578, IEEE Computer Society, 2002.

[20] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *22 nd Int'l Conference on Distributed Computing Systems–ICDCS'02*, (Washington, DC, USA), p. 457, IEEE Computer Society, 2002.

[21] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Energy-latency tradeoffs for data gathering in wireless sensor networks," in *IEEE INFOCOM'04*, (Hong Kong), Mar. 2004.

[22] G. Wegner, "Über endliche kreispackungen in der ebene," *Studia Scientiarium Mathematicarium Hungarica*, vol. 21, pp. 1–28, 1986.