An Energy-Efficient Aggregation Scheduling Algorithm with Minimal Latency in Wireless Sensor Networks

Liqun Shan, Jinkuan Wang, Wei Wei, Zhigang Liu

1,2 School of Information Science & Engineering Northeastern University
Shenyang, China, E-mail: snoweer_1011@163.com

2 Engineering Optimization & Smart Antenna Institute Northeastern University at Qinhuangdao
Qinhuangdao, China, E-mail: wjk@mail.neuq.edu.cn

3 Northeast Petroleum University Qinhuangdao, China

Abstract

In this paper, we address the problem of data aggregation scheduling problem to minimize the latency in wireless sensor networks (WSNs). An efficient distributed synchronous aggregation scheduling method is proposed to structure a collision-free schedule for data aggregation in WSNs. By using a connected dominating set (CDS) as an aggregation tree, we implement the synchronous aggregation scheduling. We prove that the latency of the aggregation schedule generated by our algorithm is at most $4R + \Delta$ time-slots where $R$ is the network radius and $\Delta$ is the maximum node degree in the communication graph of the original network. Analysis and simulation results show the validity and superiority of the algorithm.

Keywords: Wireless Sensor Networks, Connected dominating set, Aggregation scheduling

1. Introduction

Data aggregation is a primitive operation in wireless sensor network applications such as emergency medical situations, environmental monitoring and battlefield surveillance. In most applications, quite often we need to gather data from sensors to a fixed sink node that collects a packet from every other node and every intermediate node combines all received packets with its own packet into a single packet of fixed-size according to some aggregation function such as logical and/or, maximum, or minimum. This type of application is called data aggregation. We focus on generic environmental monitoring applications which collect sensing data continuously over a long period of time and aim at providing collision-free and shorter packet delay.

In data aggregation, many issues need to be resolved such as energy conservation, deployment strategies, routing in dynamic environment and latency [1]. Previous researches on in-network data aggregation seldom consider the collision problem but leave it to the MAC layer. Solving this problem in MAC layer results in latency too high to be practical. Recently, a few researchers begin to consider the collision problem for reducing data aggregation latency and try to construct a feasible schedule to eliminate collisions during data aggregation [2] [3] [4] [5] [6] [7].

The data aggregation scheduling problems have been extensively studied in recent years [10] [11] [12] [13]. The most related ones are as follows. Kesselman and Kowalski [2] proposed a randomized and distributed algorithm for aggregation in a n-node sensor network with an expected latency of $O(\log n)$. In their model, there are two assumptions. One is that each sensor node has the capability of detecting whether a collision occurs after transmitting data. Another one is that sensor nodes can adjust their transmission range without any limitation. These assumptions pose some challenging issues for hardware design and the latter assumption is almost impossible when the network scale is very large. Chen et al. [3] proposed an algorithm to generate a collision-free schedule with latency bound of $(\Delta + 1)R$. They also proved that the minimum data aggregation time problem is NP-hard. Huang et al. [4] proposed a scheduling algorithm with the latency bound of $23R + \Delta + 18$ time-slots, where $R$ is the network radius and $\Delta$ is maximum node degree. However the interference model used in [4] is a simple primary interference model: no node can send and receive simultaneously. Under Protocol Interference Model, Yu et al. [5] proposed a distributed scheduling algorithm generating collision-free schedules...
that has a latency bound of $24D + 6\Delta + 16$ time-slots, where $D$ is the network diameter. Xu et al. [6] proposed a scheduling algorithm generating collision-free schedules that has a latency bound of $16R + \Delta - 14$ time-slots. [6] choose the topology center of the UDG as the root that is not practical, because the root is common node with limited energy and prone to be failure. Wan et al. [7] proposed a scheduling algorithm generating collision-free schedules that has a latency bound of $15R + \Delta - 4$ time-slots.

Among these algorithms mentioned above, some work on centralized aggregation scheduling that are not practical, especially when the network topology often changes in a large sensor network, and other work on distributed aggregation scheduling that can need the requirement of the dynamic network properties. But data scheduling of these algorithms uses a bottom-up or up-bottom approach that don’t consider nodes in different dominate nodes execute the transmission at the same time and still have high latencies. Moreover, each dominant node costs at most $\Delta$ time slots in [6] [7] that is not accurate.

The main contributions of this paper are as follows. We propose an efficient algorithm that will construct a data aggregation tree and a synchronous TDMA schedule for all links in the tree such that the latency of aggregating all data to the sink node is approximately minimized. An efficient distributed method is proposed to structure a collision-free schedule for data aggregation in WSNs. We prove that the latency of the aggregation schedule generated by our algorithm is at most $4\delta + 15R - 16$ time-slots where $R$ is the network radius and $\delta$ is the maximum node degree in the communication graph of the original network.

2. Model and problem definition

2.1 Network model

We consider a WSN consisting of $N$ nodes, given a graph $G = (V, E)$ representing the WSN and a base station (sink node) $s \in V$, to find a data aggregation scheduling with minimum latency. All the sensor nodes are homogeneous. Each sensor node is equipped with a RF transceiver that can be used to send or receive data. We assume that each sensor node has omni-directional antenna and the transmission coverage of any sensor node is a circle with unit radius centered at the sensor. For simplicity, we assume that all nodes other than the sink have the same transmission range (radius) $r$ such that both node $u$ and $v$ form a communication link whenever their Euclidean distance $\|u - v\| \leq r$. Let $N_1(i)$ denote the set of neighbors of node $i$, $N_1(i) = \{ j \mid d_j \leq r, j \in N \}$, and $N_2(i)$ denote the set of two hops’ neighbors of node $i$, $N_2(i) = \{ j \mid r \leq d_j \leq 2r, j \in N \}$, where $d_j$ is the Euclidean distance of node $i$ and node $j$. We normalize their transmission radius to one unit and use a unit disk graph (UDG). Transmission is deterministic and proceeds in synchronous time rounds controlled by a global clock. In each time round, any node cannot send and receive data simultaneously, i.e. any node either sends data or receives data in a round.

Built based on connected dominating set for the topology of WSNs, energy consumption of the nodes which are responsible for performing the aggregation and forwarding of network information is far greater than energy consumption of the nodes which can periodically go to sleep. Therefore, a reasonable construction of dominating set is essential for extending the network lifetime. From the perspective of a single node, the node with the more residual energy, more transmission distance and more aggregation gain as a aggregation node can effectively use the network of energy resources, which can greatly extend the reconstruction time of the network topology, and is conducive to maximum network life cycle; from the perspective of the entire network, the node with the more node degrees as a aggregation node can minimize the number of nodes in a working state, which help to reduce the total network energy consumption, and is easy to extend the network life cycle. Therefore, considering the impact of the aggregation gain, the remaining energy, node degrees and transmission distance for the aggregation node select, this paper presents a weighted combination model.

$$\omega = \sum_{i} \frac{E_i - E_{\text{e}}}{E_i} \times \frac{h}{H} \times D_i$$  

(1)
Where \( G_i \) denotes the aggregation gain at node \( i \), \( E_i \) and \( E_r \) present the initial energy and residual energy respectively, \( E_t \) denotes the communication threshold, \( h_i \) presents the hop numbers between the node \( i \) and sink node, \( H \) denotes the max hop numbers in WSNs, \( D_i \) is the node degree at node \( i \), and \( a, b, c, d \) present weight parameters, \( a > 0, b > 0, c > 0, d > 0 \).

At the stage of network initialization, the aggregation gain \( G_i \) is set to 1 for all nodes. After nodes collect and send some data, the average aggregation gain \( G_{avg} = \frac{\sum G_i}{|M|}, M \) is the set of dominated nodes. The node which has not been a dominated node uses \( G_{avg} \) as their aggregation gain, i.e. \( G_i = G_{avg} \).

Considering that the non-dominated nodes impact the aggregation gain of the dominated nodes. We present the concept of “correlation degree”. Nodes with high correlation degree share a dominant node that will help to improve the aggregate gain of the dominated node. We define the correlation degree \( \gamma(i, j) \) between nodes \( i \) and \( j \) as

\[
\gamma(i, j) = e^{-\sigma d_{ij}}
\]  

(2)

Where \( d_{ij} \) is the distance between nodes \( i \) and \( j \), \( \sigma > 0 \) is the correlation parameter. The correlation degree between nodes will be reduced as the distance increases. After the dominated node performs the aggregation, it transmits the correlation degree as the standard that a non-dominated node chooses the next dominated node.

2.2 Problem Definition

Let \( X, Y \subset N \) and \( X \cap Y = \emptyset \). If all the nodes in \( X \) transmit data simultaneously in one time-slot and all data are received by some nodes at the same time in \( Y \) with collision-free, we say data are aggregated from \( X \) to \( Y \) in one time-slot. A data aggregation scheduling can be thought of as a sequence of senders \( \{X_1, X_2, \ldots, X_l\} \) (in which \( X_i \subset N, \forall i \)) satisfying the data aggregation property. This sequence represents that all nodes in \( X_1 \) transmit in the first time slot, followed by all nodes in \( X_2 \) transmitting in the second time slot and so on. All data will be aggregated to one single node \( s \), which is the sink node. Then a data aggregation scheduling with latency \( l \) can be a sequence of sender sets \( X_1, X_2, \ldots, X_l \) satisfying the following conditions:

1. \( X_i \cap X_j = \emptyset, \forall i \neq j \);
2. \( \bigcup_{i=1}^{l} X_i = N \);
3. Data are aggregated from \( X_i \) to \( N \setminus \bigcup_{j<i} X_j \) at time-slot \( k \), for all \( k = 1, 2, \ldots, l \) and all the data are aggregated to the sink node \( s \) in \( l \) time-slots.

The number \( l \) is called data aggregation latency. This paper focuses on constructing a distributed aggregation scheduling algorithm to find a schedule \( X_1, X_2, \ldots, X_l \) in a distributed way such that \( l \) is minimized. This problem is proved to be NP-hard in [3]. This paper proposes an approximate distributed algorithm with at most latency \( 4\delta + 15R - 16 \), where \( R \) is the network radius and \( \delta \) is the maximum node degree.

3. Aggregation Scheduling Algorithm

Our distributed aggregation scheduling algorithm, named DAS, consists of two phases. One is to construct a distributed aggregation tree. Another one is to perform the distributed aggregation scheduling.

3.1 Distributed aggregation tree construction
The algorithm is based on a connected dominating set (CDS) to construct the aggregation tree. We first construct a maximal independent set \( I \), and then construct a connected set \( B \). \( I \cup B \) is a connected dominating set.

To reduce overall energy consumption of the network, the average weight of the dominated nodes must be maximized. Therefore, to find a selection dominated node scheme that can make dominated nodes to maximize the average weight, reduce network traffic and maximize network life cycle. Meanwhile, the dominated node in order to reduce coverage overlap and to ensure network connectivity, we require the following control node selection program constraints: (1) any two dominated nodes are not adjacent; (2) each dominant node has at least another dominated node within its two hops. Maximize the average weight of dominated node can be attributed to the following nonlinear integer programming problem:

\[
\text{Max} \sum_{i \in I} s_i \omega_i / \sum_{i \in I} s_i
\]

Subject to
\[
s_i + s_j \leq 1 \quad \forall i \in V, \forall j \in N_i(i)
\]
\[
s_i + s_j \geq 2 \quad \forall i \in V, \forall j \in N_2(i)
\]
\[
s_i, s_j \in \{0, 1\}
\]

Where \( s_i \) denotes status indicator, \( s_i = 1 \) presents dominated node and \( s_i = 0 \) presents non-dominated node.

The nonlinear integer programming problem (3) has \( O(2^N) \) time complexity. We present a low complexity method of the CDS structure. The main idea of this algorithm is: before the dominated nodes selection is performed in each round, the sink node first broadcasts the average aggregate gain across the network. Each dominated node sends the last round correlation degree to each its non-dominated node. At the beginning of selection, each node is colored white, and broadcasts their own \( \omega_i \) value to their neighbors. Each node compares weight value with its neighbor nodes, if it has the maximum weight value, it is selected as the dominant node and broadcasts dominated messages, otherwise the node is still white. If the white neighbor node receives the dominated message, it is colored gray. If the gray node receives the message that all neighbor nodes whose weigh value are larger than its weigh value are out of competition, it is selected as the dominated node and broadcasts control messages. If a gray node receives multiple control messages, it will select the node with the largest correlation degree as its dominated node. We repeat this process until all nodes are not white. The pseudocode of MIS construction is given in Algorithm 1.

**Algorithm 1 Construct a max independent set (MIS)**

Initially, each node colors itself WHITE;

\[
V_{\text{white}} \leftarrow V;
\]

\[
V_{\text{gray}} \leftarrow \emptyset;
\]

\[
I \leftarrow \emptyset;
\]

While \( V_{\text{white}} \neq \emptyset \) do

\{ 
\[
i \leftarrow \arg \max_{i \in V_{\text{white}}} \{w(i)\};
\]

\[
i \text{ colors itself black};
\]

\[
I \leftarrow I \cup \{i\};
\]

The nodes in \( N_i(i) \) color themselves GREY

\[
V_{\text{gray}} \leftarrow V_{\text{gray}} \cup N_i(i);
\]

\[
V_{\text{white}} \leftarrow V_{\text{white}} - V_{\text{gray}} - \{i\};
\]

\}

In the algorithm 1, all black nodes form a MIS set by a distributed approach. Black nodes broadcast their control message. If a gray node receives more than one control message, it will select the node with the largest correlation degree as its dominated node.
Non-dominated nodes periodically collect environmental data, and transmit these data to the dominated node. The dominated nodes perform data aggregation and sent aggregated information to the sink node by multi-hop way. In order to build a data forwarding optimized network structure, the connected nodes should satisfy the following three conditions: minimize the number of connected nodes; maximize the average weight of connected nodes; ensure two adjacent dominated nodes there is a connection node. The pseudocode of CDS construction is given in Algorithm 2.

Algorithm 2 Construct a CDS to form an aggregation tree level by level

\[
B \leftarrow \emptyset;
\]

For each \( i \in I \) do

If \( k_i > 1 \) then

For all \( j \in I \) do

\[ j \leftarrow \arg\max_{mN_j \cap U} \{w(j)\}; \]

End for

\[ k \leftarrow \arg\max_{k \in N_i \setminus \{N_j\}} \{w(k)\}; \]

k colors itself blue;

End if

\[ B \leftarrow B \cup \{k\}; \]

End for

When the network runs into the stable phase, \( G_i \) and \( \gamma \) are changing. After a period of time, the entire network run into the next round of work cycle, recalculate \( G_i \) and \( \gamma \), select dominated nodes and connected nodes, and construct a new CDS.

**Theorem 1:** The message complexity is \( \mathcal{O}(N) \).

**Proof.** The sink node broadcasts the average aggregate gain \( G_{avg} \) across the network. Each node forwards \( G_{avg} \) once, and total \( N \) messages with \( G_{avg} \) are forwarded. The dominated node \( i \) sends the correlation degree \( \gamma \) to the non-dominated nodes that select node \( i \) as their dominated node. Assume the number of the dominated nodes is \( D \) at the last round, and there are \( N-D \) correlation message. At the beginning of selection, each node broadcasts their own \( \theta \) value to their neighbors, and there are \( N \) message. Assume the number of the dominated nodes is \( K \) at the new round, and the dominated nodes broadcast \( K \) message. The non-dominated nodes broadcast \( N-K \) message and send \( N-K \) message. Therefore, the cost is \( N+(N-D)+N+K+2(N-K)=4N-D-K \) to construct the CDS. The message complexity is \( \mathcal{O}(N) \).

According to the algorithm 1 and 2, we construct the aggregation tree just like Fig. 1. All the black nodes that are dominators form an MIS. Each blue node connects two black nodes and is lower-level black node’s parent node. The grey nodes are leaves and dominatees. All nodes form a CDS. For aggregation, the CDS is an aggregation tree.

**Figure 1.** Data aggregation tree

**Figure 2.** Latency with different number of nodes
3.2. Distributed parallel aggregation scheduling

Next, the network performs aggregation scheduling according to the constructed aggregation tree. Let $B$ denote all black nodes set, $b_j \in B$ where $1 \leq j \leq |B|$. Let $W$ denote all grey nodes, $w_i \in W$. Set the coordinate of $b_j$ is $(x_{b_j}, y_{b_j})$ and the coordinate of $w_i$ is $(x_{w_i}, y_{w_i})$, $1 \leq i \leq |W|$. We construct different virtual disks centered at every black nodes of radius one, and divide every disk into four equal portions: $area_{a_1}$, $area_{a_2}$, $area_{a_3}$ and $area_{a_4}$ along the horizontal and vertical directions. The $area_{a_1}$ is formed by the nodes in the range $x_{b_j} \leq x_{w_i}$ and $y_{b_j} > y_{w_i}$ in $b_j$’s children nodes. The $area_{a_2}$ is formed by the nodes in the range $x_{b_j} > x_{w_i}$ and $y_{b_j} \leq y_{w_i}$ in $b_j$’s children nodes. The $area_{a_3}$ is formed by the nodes in the range $x_{b_j} \geq x_{w_i}$ and $y_{b_j} > y_{w_i}$ in $b_j$’s children nodes. The $area_{a_4}$ is formed by the nodes in the range $x_{b_j} < x_{w_i}$ and $y_{b_j} \leq y_{w_i}$ in $b_j$’s children nodes. The communications process in $area_{a_i}$ are executed at the same time in synchronous time-slots, where $\forall j, i \in [B]$. Nodes of four fields are scheduled in clockwise order, that is to say, every node in $area_{a_i}$ finishes transmitting the information, next $b_j$ receives the information of every node in $area_{a_2}$, and so on.

We will determine the number of time slot in four equal portions in the following first-fit manner. Assume that $D_u$ and $D_v$ are both neighboring dominant nodes in $B$. $D_u$ denotes the set of nodes in the $area_{a_1}$ of $D_u$. The set of nodes with respect to the $area_{a_1}$ of $D_v$ are denoted by $D_v$. Consider ID increasing order $<v_i, v_{i+1}, ..., v_{i+4}>$ of $D_u$ and $<v_j, v_{j+1}, ..., v_{j+4}>$ of $D_v$. Initially, $U = \{v_i\}$ and $V = \emptyset$. For $x=2$ up to $|D_u|$, for $y=1$ up to $|D_v|$, if $v_{i+y} \notin V$ and $v_{j+y}$ is first not adjacent to $v_i$, add $v_{j+y}$ to $V$. Because the matching nodes in $U$ and $V$ are pairwise conflict-free, $|U|$ is equal to $|V|$ and they use $|U|$ time slot to complete the communication. The nodes in $D_u/V$ and $D_v/U$ are conflicting and need separate time slot. According to the algorithm, we can achieve the number of time slot is equal to $\delta_{b_j} = |D_u/V| + |D_v/U| + |V|$.

$b_j$ gets the number $\delta_{b_j, 1}$, $\delta_{b_j, 2}$, $\delta_{b_j, 3}$ and $\delta_{b_j, 4}$ of nodes in different areas. All nodes in $B$ broadcasts $\delta_{b_j, 1}$, $\delta_{b_j, 2}$, $\delta_{b_j, 3}$ and $\delta_{b_j, 4}$ messages in WSNs and get the max values of $\delta_{b_j, 1}$, $\delta_{b_j, 2}$, $\delta_{b_j, 3}$ and $\delta_{b_j, 4}$ that separately denote $\delta_1$, $\delta_2$, $\delta_3$ and $\delta_4$. The nodes in the $area_{a_1}$ are assigned corresponding time slot $t$ deferred to the conflict-free in $\delta_i$ and are sleep in the rest of the time. For example, the first time slot is assigned in Figure 3. Every dominator has the same scheduling period $\delta_1 + \delta_2 + \delta_3 + \delta_4$. The max value of the scheduling period is $4\delta - 1$. The dominatees in all nodes of $B$ periodically send data to $b_j$ if and only if they are in the corresponding time slot and are in sleep mode in the rest of the time. In $\delta_1 - \delta_{b_j, 1}$, $\delta_2 - \delta_{b_j, 2}$, $\delta_3 - \delta_{b_j, 3}$ and $\delta_4 - \delta_{b_j, 4}$ interval, the $b_j$ node is in sleep state.

Let $C$ be the set of connectors. The scheduling in $C \cup D$ is as the following. Observe that every dominator, except the root of the data aggregation tree, connects to at least one dominator in the upper level within 2-hops. We now determine the max number of time slots that a dominator node will use to connect to all dominators within 2 hops. Consider any dominator $u$, let $D_u(u)$ be the set of dominators within 2-hops of $u$ in the original communication network $G$. Based on a technique lemma implied from lemmas proved in [6], $u$ requires at most 12 connectors to connect $D_u(u)$. We use $C(u)$ to denote the set of connectors used to connect all dominators in $D_u(u)$. According to Wegner Theorem, there can be at most 5 dominant nodes in a unit disc centered at any node $u$ in $C(u)$. Each connector only receives the data from lower-level dominant nodes, so $u$ can receive the data from all neighboring dominators $D_u(u)$ at most 15 time slots. All dominators within 2 hops execute parallel scheduling by connectors.

**THEOREM 2.** By using distributed parallel aggregation scheduling, the sink can receive all the aggregated data in at most $4\delta + 15R - 16$ time-slots.
PROOF. Every dominator can aggregate its neighboring dominatees’ data in at most \(4\delta - 1\) time-slots within corresponding time-slot by parallel manner. Then connectors ensure that every dominator’s data can be aggregated upwards the root finally. Every dominator \(u\) including the root of data aggregation tree can collect aggregated data from all dominators in \(D_\delta(u)\) within at most 15 time-slots. Since there is no dominator in level \(R\), after at most 15\((R-1)\) time-slots, every dominator’s data can be aggregated to the root. Therefore within \(4\delta + 15R - 16\) time-slots, all the data can be aggregated to the sink node.

3.3 Scheduling adjusting of topology change

When a new node \(u\) joins in the network, it sends a HELLO message. All the nodes in \(u\)’s transmission range receive the HELLO message. For any black node \(v\) receiving the HELLO message, it sends back an ACK message including its node color. If \(u\) collects the ACK messages from its neighbors, it picks any one, say \(q\), as its parent and sends a CHILD message to \(q\). Node \(u\) becomes a leaf and colors itself grey. Otherwise \(u\) has no black neighbors, and \(u\) makes itself a black node. Since the network is connected, \(u\) must have at least a blue neighbor. \(u\) can randomly pick one as its parent from its neighbors and sends a CHILD message. Upon receiving a CHILD message, a node records \(u\) as its child and checks if it is a blue node. If not, it turns itself to a blue node.

If the dominate node \(v\) has not received node \(u\)’s response for a predefined time duration, \(v\) believes that \(u\) fails. Assume \(u\) fails. If \(u\) is a leaf node, its parent \(v\) removes \(u\) from its child list. If \(u\) is a blue node, its parent \(v\) and its black children will do something respectively. In this case, \(v\) deletes \(u\) from its child list and for any black child \(s\) of \(u\). We implement the following strategy to find a parent for \(s\). First, \(s\) finds a blue or gray neighbor \(q\), which is not its child, as its parent and sends a CHILD message to node \(q\). If \(s\) cannot find one, \(s\) turns its color to grey and finds its parent. Its grey children also need to find black parents for them. The grey nodes can be seen as the new joining nodes during the finding of their parents. \(s\)’s blue children find their parents using the strategy proposed as follows. If the connected node \(v\) has not received node \(u\)’s response for predefined time duration, \(u\) is a black node and fails. \(v\) deletes \(u\) from its child list and checks if it has any other black children. If not, \(v\) turns itself to a grey node. For any grey child of \(u\), again we take it as a new node joining in. For any blue child \(s\) of \(u\), \(s\) finds its parent in the following way. First, \(s\) checks if there are black neighbors who are not its children. If so, \(s\) randomly chooses one from them as its black parent and sends a CHILD message to it. Otherwise, \(s\) randomly chooses one of its children \(q\), which must be a black node, as its parent and sends a CHILD message to \(q\). As the process is recursive, any node who loses its parent can find a new one as long as the network is connected.

The network topology is maintained in the form of aggregation tree. After the structure of the aggregation tree has changed, each black node who finds the number of its children nodes change need confirm new time slot locally and make a new schedule using EEDAS. In most cases the above process does not need much time and communication since it is done locally.

4. Simulation Results

We randomly deploy sensors into a region of 200m\(\times\)200m. We compare the performance of our algorithm EEDAS with the algorithm DAS for aggregation scheduling. We use matlab to implement EEDAS and DAS algorithm. In Fig. 2, the transmission radius is 25m. The figure shows the number of the time slots needed to aggregate data from the leaves to the sink by running the two algorithms while the number of nodes increases. Fig. 3 compares the number of slots to aggregate data using the two algorithms when the transmission radius varies. Here the number of nodes is fixed to 1600. It can be seen from the two figures that EEDAS outperforms DAS algorithm with much lower latencies. From the figures, EEDAS's latencies are 1.52 to 3.06 times smaller than DAS's latencies, and with the increment of the number of nodes and the transmission radius, the improvement of the latency will be larger.
Next we compare the running time of EEDAS and DAS in figure 4 when the number of nodes and the transmission radius vary. The running time refers to the time required to generate the aggregation scheduling. We set $1/20$ of a second for each time slot when running two algorithms, which is greater than the time required to transmit one packet. We can see that the running time of EEDAS is 2 to 8s on average. For instance, for 2000 nodes with transmission radius of 30m, the running time of DAS is 14.6s and the running time of EEDAS is only 3.2s. The reason is that many nodes can schedule simultaneously in EEDAS, and the occasion where only one node can schedule at each time maybe happen in DAS. As the number of the nodes or the transmission radius increases, the average size of the nodes' competitor sets also increases, thus each node has to compete with more competitors that costs more time.

We study the performance of EEDAS and DAS while confronting node joining and node failures. The increasing rate of latency refers to the ratio of the difference of the new and old latency to the old latency. Fig. 5 shows the increased latency after nodes join in the network with different joining rate. We find that in different network scales, the trends of latency increments are similar while the node joining rate increases but not much. For example, the increasing rate of latency is from 0.082 to 0.108 with different number of nodes when the node joining rate is 0.3. We find that the percentage of the latency increment is only a little larger than that of joining nodes. This result shows that the adaptive EEDAS scales well with the size of the network. The time latency increments are studied with different failure rate in Fig. 6. In most cases, we find that the latency difference is between -0.3 and 0.3 in EEDAS, while between -1 and 1 in DAS, which means the time latency keeps almost invariant with different node failure rate.

5. Conclusion

In this paper, we make a study of the data aggregation problem and its latency. We use the techniques of MIS and CDS to construct the aggregation tree. We propose a distributed aggregation scheduling algorithm. The latency of the aggregation schedule generated by our algorithm is at most $4\delta+15R-16$ time slots. Analysis and simulation results show that the EEDAS algorithm has smaller latency and better performance.
An Energy-Efficient Aggregation Scheduling Algorithm with Minimal Latency in Wireless Sensor Networks
Liqun Shan, Jinkuan Wang, Wei Wei, Zhigang Liu

This research has been supported by the National Natural Science Foundation of China under Grant No. 60874108, by the Fundamental Research Funds for the Central Universities under Grant No.N090423002, by Directive Plan of Science Research from the Bureau of Education of Hubei Province, China, under Grant No.Z2009105, and by the Natural Science Foundation of Hubei Province under Grant F2011501021.

References