International Journal of Future Generation Communication and Networking
Vol. 9, No. 5 (2016), pp. 83-98
http://dx.doi.org/10.14257/ijfgcn.2016.9.5.09

On Latency-Efficient Transmission Scheduling for In-Network Data Aggregation in Duty-Cycled Wireless Sensor Networks

Sun Chengting and Wang Zhen

Lianyungang Technical College, Information Engineering College, Lianyungang Jiangsu, China, 222006
sunchengting01@163.com

Abstract

In-network data aggregation is a fundamental traffic pattern in many applications of wireless sensor networks (WSNs). Data aggregation scheduling aims to find a collision-free transmission schedule scheme for data aggregation while minimizing the total network latency. This paper focuses on the data aggregation scheduling problem in duty-cycled WSNs (dc-WSNs), in which low-duty-cycle techniques are employed for energy-consuming operations. Based on greedy strategy, we propose two latency-efficient data aggregation scheduling algorithms, namely GAS-PAS and GAS-SAS for dc-WSNs. We theoretically derive the latency upper bounds of the proposed algorithms, and the results demonstrate that both GAS-PAS and GAS-SAS achieve constant approximation to the optimal latency. We also conduct extensive simulations to show that the proposed scheduling algorithms can improve data aggregation latency in dc-WSNs under various network settings, comparing with state-of-the-art algorithms in the literature.

***Keywords**: Wireless sensor networks; data aggregation; scheduling; duty cycle; latency*

1. Introduction

In many applications of wireless sensor networks (WSN), the base station (the sink) needs to gather the monitoring results from the network periodically. Usually it is unessential for the sink to collect all the raw data packets produced by the sensor nodes. Instead, the sink may only concentrate on some statistical information about the monitoring results, such as the MAX (or MIN, AVG and so on) value among the numerous data packets [6]. At such circumstances, data aggregation is adopted, in which data packets produced by different sensor nodes are compressed or aggregated at the intermediate nodes along the path to the sink. Data aggregation decreases the amount of packets transmitted in the network and hence economizes lots of energy notably. However, data aggregation is a time-consuming operation mainly for two reasons. The first is that the intermediate nodes have to wait packets from some other nodes for aggregation. The second is that the interference among adjacent transmission links may be severe and hence lead to collisions and retransmissions. In order to minimize the total delay for the sink to obtain the aggregated monitoring result, data aggregation scheduling is extensively researched to supply a collision-free scheduling for concurrent transmission links under certain interference constraints [8]. This is the so called Minimum Latency Aggregation Scheduling (MLAS) problem, which is proved to be NP-hard in [4] and has been extensively investigated in [1, 3-4, 8-9]. However, previous works on this topic usually assume that the sensor nodes are awake all the time and able to receive or transmit packets whenever they need to, which is, however, impractical when taking into account that sensor nodes are limited by power and idle listening consumes as much energy as wireless communication [16].

To conserve energy, low duty cycle is adopted for WSN, in which sensor nodes not involved in communication turn off their radios and go into sleep mode completely. They then periodically wake up for a short time (denoted by active slot) to send packets or to check the channel status for potentially packets reception. Though duty cycling can save energy notably, it may result in additional latency due to the uncoordinated communication between the sender and the corresponding receiver, *i.e.*, the intended receiver may be in sleep mode when the sender tries to transmit data to it [17].

We consider data aggregation scheduling in duty-cycled WSN (dc-WSN) with the goal of minimizing the total latency. This problem is showed to be NP-hard as well. For two cases, *i.e.*, dc-WSN with predetermined active slot and dc-WSN with schedulable active slot (the definitions of which are presented in Section 4.2), we propose two efficient scheduling algorithms, namely GAS-PAS and GAS-SAS using greedy strategy. The proposed algorithms are based on a carefully constructed data aggregation tree, and aims to make maximal schedule in each time slot. Theoretical analyses are given to show that both GAS-PAS and GAS-SAS are constant approximation algorithms with latency bounds of $28R+2\Delta-29$ and $(16+\lceil 13/|P| \rceil) R+\Delta+\lceil(\Delta-1)/|P| \rceil-13-\lceil 12/|P|\rceil$ working periods, respectively. Here R, Δ and |P| are the network radius, the maximum node degree and the number of time slots during a working period, respectively. Extensive simulations are carried out to evaluate the performance of our algorithms with comparison to some state-of-the-art approaches in the literature. The results show that our algorithms outperform the others in terms of aggregation latency under various settings.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 formulates the aggregation scheduling problem. The detailed scheduling algorithms and the corresponding performance analysis are presented in Section 4. Simulations are given in Section 5. Finally, Section 6 concludes the paper.

2. Related Works

Data aggregation, defined as the global process of gathering and routing information through a multi-hop network, as well as processing data at intermediate nodes with the objective of reducing energy consumption [1], has been well studied in recent years. Most of the work on this topic is based on a data aggregation tree. Wu *et al*. [15] prove that constructing a maximum lifetime aggregation tree is NP-hard and propose a near-optimal polynomial algorithm. Tung-Wei Kuo *et al*. [11] prove that constructing an aggregation tree with minimum energy cost of data transmission is NP-hard and propose a O(1)-approximation algorithm for it. Li *et al*. [5] study the problem of data gathering with optional aggregation, aiming to maximizing the lifetime.

Due to the broadcast nature and interference of wireless transmission, data aggregation often suffers from collisions. To avoid collisions, the MLAS problem is proposed and studied. MLAS means to assign each node with a transmitting slot and ensure that collisions do not occur, meanwhile minimizing the aggregation time. Chen *et al*. [4] prove that the MLAS problem is NP-hard and propose a scheduling algorithm with an upper bound of (Δ− 1) R time slots, where Δ is the maximum node degree and R is the network radius. Afterwards, Huang *et al*. [8] and Wan *et al*. [13] propose improved scheduling algorithms with upper bounds of 23R+Δ-18 and 15R+Δ-4, respectively. Yu *et al*. [3] propose the first distributed scheduling algorithm with an upper bound of 48R+6Δ+16, which is further improved by Li *et al*. [9] and Xu *et al*. [14]. The latest work on MLAS is done by Bagaa *et al*. [1], which differs from the previous work in adopting semi-structured and un-structured topology for data aggregation.

However, all the work assumes that the nodes are always awake and ignores the duty-cycling nature of WSN. In fact, to conserve energy, duty-cycling is widely used in WSN, where nodes work and sleep periodically. There has been much work on duty-cycling WSN. Buettner *et al*. [10] propose a short preamble MAC (X-MAC) for dc-WSN. To reduce the latency and enhance the reliability of data collection in dc-WSN, Cao *et al*. [16] propose a multi-pipeline scheduling approach which combines streamline technique with multi-path routing mechanism. While duty-cycling can save energy, it leads to uncoordinated communication between the sender and the receiver and hence causes excess challenge for data aggregation scheduling. In this paper, we focus on the aggregation scheduling problem in duty-cycling WSN, which is more challenging and not involved in the previous work, to the best of our knowledge.

3. Problem Definition

3.1. System Model

We model the sensor network as a connected graph $G(V,E)$. V is the set of vertexes indicating the sensor nodes (including the sink v_s) and E is the set of edges indicating the communication links between the nodes. Supposing u and v ($u,v \in V$) are two nodes, then edge (u,v) belongs to E if and only if u and v can communicate with each other directly, i.e. both u and v are in the transmission range of each other.

In the duty-cycling model, nodes wake and sleep periodically. We assume that the total lifetime of a node is divided into multiple working periods with the same length. A working period is further divided into $|P|$ time slots. A time slot is long enough for transmitting and receiving one packet. During a working period, a node is awake for one time slot (called active slot) and asleep for the rest. In dc-WSN, nodes can send data at any time slot but can only receive data when they are in the active slot. Thus the transmission between the sender u and the receiver v in dc-WSN is successful if and only if the following two rules are satisfied: (1) Transmission-Matching Rule: the transmitting slot of u should match with the active slot of v, (2) Collision-Free Rule: the transmission is collision-free, which means that v is not in the interference range of any other node which is transmitting synchronously with u.

Suppose that the nodes are homogenous with the same transmission radius of r. We consider the protocol interference model [7], in which the interference range of node u is a circle with centre to u. For simplicity, we assume the interference radius $r_I = r$, which means that the interference range is the same as the transmission range.

3.2. Problem Formulation

Let $A, B \subset V$ and $A \cap B = \varnothing$. We say data is aggregated from A to B at the time slot t during the working period w if all the nodes in A transmit data to some nodes in B synchronously and without suffering from any interference. We call A the Synchronous Sender Set (SSS). The goal of data aggregation is to aggregate all the data from $V \setminus v_s$ to v_s. To achieve it, we employ aggregation scheduling algorithm to find a sequence of Synchronous Sender Sets $\{S(1,1), S(1,2), ..., S(i,j), ..., S(w,t)\}$, where $S(i,j)$ is a SSS at the time slot j during the working period i. The sequence should satisfy the following conditions:

(1) $S(i_1, j_1) \cap S(i_2, j_2) = \varnothing, \forall (i_1, j_1) \neq (i_2, j_2)$

(2) $\bigcup_{(i,j)=(1,1)}^{(i,j)=(w,t)} S(i,j) = V \setminus \{v_s\}$

(3) Data is aggregated from $S(i,j)$ to $V - \bigcup_{(i,j)=(1,1)}^{(i,j)=(i,j)} S(i,j)$ for all (i,j) ranging from $(1,1)$ to (w,t) and the data is ultimately aggregated to v_s in w working periods.

Aggregation scheduling in dc-WSN is to find the sequence with the shortest time, i.e. to minimize w. Note that if $|P|=1$, the aggregation scheduling problem in dc-WSN turns out to be the original MLAS problem, which has been proven to be NP-hard in []. As the aggregation scheduling problem in dc-WSN is a general version of MLAS ($|P|\geq 1$), so it is also NP-hard, according to [2].

3.3. Related Preliminary

In this section, we introduce some graph theory based notations that will be used throughout the paper. The radius R of $G(V,E)$ with respect to v_s is defined as the maximum hop distance between u ($u \in V$) and v_s. The nodes that are i ($0 \leq i \leq R$) hops from v_s are referred to the i-th layer of G. Obviously, Layer 0 is v_s and Layer R is the nodes that are the farthest from v_s. The spanning tree of G can be denoted by $T(V_T, E_T)$, which is rooted at v_s. V_T equals V while E_T is a subset of E because in a tree nodes only need to connect with their parents and children. The depth d of T is defined as the maximum hop distance between u ($u \in V_T$) and v_s. The nodes with depth of i ($0 \leq i \leq d$) is referred to the i-th layer of T. Similarly, Layer 0 is v_s and Layer d is the nodes that are the farthest from v_s. In $G(V,E)$, a subset S ($S \subseteq V$) is an Independent Set (IS), if there is no edge between any pair of the nodes in V. An Independent Set S is a Maximal Independent Set (MIS) if there is no other IS that is a superset of S. In G, a subset S ($S \subseteq V$) is a Dominating Set (DS) if for each node u in V, it is either in S or is adjacent to some node in S. So a MIS is also a DS. A subset S ($S \subseteq V$) is a Connected Dominating Set (CDS) if S is a DS and S induces a connected sub-graph. If S is a DS, we call nodes in S dominators and nodes in $V \setminus S$ dominates.

4. Data Aggregation Scheduling Algorithms

Our aggregation scheduling algorithms consist of two phases. First, a data aggregation tree is constructed. Second, aggregation scheduling is performed based on the constructed tree. We adopt an existing approach or the first phase and the second phase is the key part of this paper.

4.2. Data Aggregation Tree Construction

We adopt an existing approach proposed by Wan *et al.* [12] to construct a data aggregation tree $T(V_T, E_T)$ from $G(V,E)$. In the approach, the authors select a CDS from V as the virtual backbone of an ad hoc network by the way of picking a MIS first and then a few connectors to interconnect the nodes in MIS. The main steps of the approach are summarized as follows.

(1) The nodes in G are marked WHITE and divided into layers according to the hop distances of the nodes with v_s initially. Then a BFS tree T_{BFS} is constructed and a MIS U is selected layer by layer (v_s is in U). The nodes in U are called dominators

and marked BLACK. The dominators in Layer i ($0 \le i \le R$) is denoted by U_i, where R is the radius of G.

(2) We then pick some nodes in $V \setminus U$ to interconnect the BLACK nodes in G. These selected nodes form a Connector Set W. From Layer 1 to Layer R, for node u ($u \in U$) its parent in the BFS tree of G, denoted by $p(u)$, is selected as a connector, added to W and marked GREY. We then pick a dominator $d_{p(u)}$ for $p(u)$ in the same or upper layer. So $p(u)$ is responsible for interconnecting the disjointed nodes u and $d_{p(u)}$. The corresponding edges are added to E_T.

(3) After each WHITE node is connected to one of its adjacent BLACK nodes randomly, the data aggregation tree based on a reduced CDS is produced. We can easily obtain some properties that lie in the constructed tree as follows.

a) Each WHITE node has a parent marked BLACK.

b) Each BLACK node except for the sink has a parent marked GREY.

c) Each GREY node has a parent marked BLACK.

4.2. Aggregation Scheduling Algorithms

The goal of data aggregation scheduling is to minimize the total delay through seeking a sequence of Synchronous Sender Sets (SSS). These sets should comply with the Transmission-Matching Rule and Collision-Free Rule stated in Section 3.1. In the Transmission-Matching Rule, the transmitting slots of the senders, which are the schedule objects, should match the active slots of the receivers. So the active slot of a node during a working period in dc-WSN is of great importance on the schedule strategy and hence impacts the total delay of data aggregation. Therefore, we consider the following two cases of dc-WSN.

- *Dc-WSN with predetermined active slot*: the active slot during a working period is predetermined for each node after the network is deployed, and our algorithm takes it as an input for scheduling the transmitting slots of the senders.
- *Dc-WSN with Schedulable Active Slot*: the active slot during a working period is not predetermined and can be scheduled jointly with the transmitting slots of the senders for achieving a combinational schedule strategy.

We propose two Greedy Aggregation Scheduling algorithms, namely GAS-PAS and GAS-SAS for dc-WSN with Predetermined Active Slot and dc-WSN with Schedulable Active Slot, respectively.

4.2.1. GAS-PAS

In GAS-PAS, we divide the global aggregation scheduling problem into a series of sub-problems. Each problem can be solved by calling a common *Greedy Scheduling Sub-Procedure* (GSSP) to generate a collision-free schedule sequence for an input sender set. We first introduce GSSP, the pseudo-code of which is given in Algorithm 1.

Algorithm 1 GSSP (*A, B, G, T*)

Input: a sender set A, a receiver set B, a graph G and a data aggregation tree T
Output: A schedule sequence of $S(w,t)$

```
1   $M \leftarrow \varnothing$, $A_1, A_2, ......, A_{|P|} \leftarrow \varnothing$
2   for each node $u$ in $A$ do
3     if the active slot of $p(u)$ is $i$ $(1 \le i \le |P|)$
4       add $u$ to $A_i$
5   for $i \leftarrow 1$ to $|P|$ do
6     $w \leftarrow 1, t \leftarrow i$
7     while $A_i \ne \varnothing$ do
8       randomly pick a node from $A_i$, add it to $M$ and remove it from $A_i$
9       for each node $v$ in $A_i$ do
10        if $\forall u \in M, p(u) \ne p(v) \&\&(p(u), v) \notin E$
                $\&\&(p(v), u) \notin E$ then
11          add $v$ to $M$, remove $v$ from $A_i$
12      $S(w,t) \leftarrow M$, $M \leftarrow \varnothing$, and output $S(w,t)$
13      $w \leftarrow w+1$
```

The main steps of GSSP are as follows. First we divide A into $|P|$ sub-sets, namely $A_1, A_2, ..., A_{|P|}$, where A_i is made up of nodes whose parents will be awake at the time slot i during each working period. For each non-null subset A_i, we initialize the working period w to 1 and the time slot t to i, and then repeat the following process. We randomly pick a node u in A_i, add it to a temporary set M and remove it from A_i. For each node in A_i which is not conflicting with any node in M, it will be added to M and removed from A_i. Therefore the nodes in M are not conflicting with each other and by assigning M to $S(w,t)$, we get a collision-free schedule for a sub-set of A_i. We continue the process until all nodes in A_i are scheduled, i.e. A_i becomes NULL. At last, A_i can be divided into a schedule sequence of $S(1,i), S(2,i), ..., S(w_i, i)$, where w_i denotes the number of working periods that is needed for scheduling all the nodes in A_i.

We now give detail description about collision, which is expressed mathematically in Line 10 of GSSP. For a scheduled node u in M, there are three classes of senders that would generate collisions if any node in them is scheduled simultaneously with u under the protocol interference model with $r_I = r$, and these nodes are called conflicting nodes of u. The first class is the senders whose parents are the same with u. The second class is the senders whose transmission ranges cover the parent (i.e. receiver) of u. The third class is the senders whose parents (i.e. receivers) are covered by u. We take Figure 1 for example. Nodes 1, 2, 4, 7 are senders. Nodes 2, 3, 5 are receivers and assumed to be awake at the same time slot during a working period. Supposing that sender 1 has been scheduled to transmit, collisions will be generated at the receivers 2, 2 and 5 respectively if the sender 3, 4 and 7 are scheduled simultaneously with sender 1. Thus senders 3, 4 and 7 are all conflicting nodes of sender 1 and should be scheduled later.

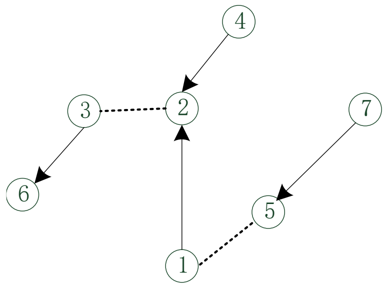

Figure 1. An Example of the Three Cases of Conflicting Nodes

For dc-WSN with predetermined active slot, our scheduling algorithm GAS-PAS is proposed, which schedules the WHITE nodes first and then deals with the BLACK and GREY nodes. The pseudo-code of GAS-PAS is given in Algorithm 2. GAS-PAS starts from scheduling the WHITE nodes by calling GSSP. After the WHITE nodes are scheduled, they are removed. The remainder tree T' will only contain BLACK and GREY nodes. We then schedule the remainder nodes in T layer by layer also by calling GSSP. Let the set $N(i)$ $(0 \le i \le d')$ represent the nodes in the *i*-th layer of T', here d' is the depth of T'. For the *i*-th round of scheduling, we treats $N(i)$ as the input sender set A and $N(i-1)$ as the input receiver set B. When GAS-PAS terminates, a collision-free schedule will be produced for each node in $V \setminus \{v_s\}$ and after the nodes in $V \setminus \{v_s\}$ have executed their schedules, all the data would be successfully aggregated to the sink v_s without suffering from any interference.

Algorithm 2 GAS-PAS (*G*, *T*)

```
Input: $G(V,E), T$.
1   GSSP($V \setminus (U \cup W), U \cup W, G, T$)
2   Remove the nodes in $V \setminus (U \cup W)$ from $T$
3   for $i \leftarrow d'$ to 1 do
4       GSSP($N(i), N(i-1), G, T'$)
```

Note: d' is the depth of the remainder tree T' where the WHITE nodes have been removed.

4.2.2. GAS-SAS

For dc-WSN with schedulable active slot, we propose GAS-SAS to generate a collision-free schedule with the minimum delay. GAS-SAS is similar with GAS-PAS in the form of scheduling layer by layer. The main difference lies in the sub-procedure GAS-SAS calls, which changes the sorely scheduling of the senders to jointly scheduling of both the senders and the receivers. Algorithm 3 shows the pseudo-code of the Modified Greedy Scheduling Sub-Procedure (MGSSP) called by GAS-SAS.

In MGSSP, we try to schedule as much nodes as possible that can transmit at the same slot ranging from 1 to $|P|$ during the same working period. We use two temporary sets in MGSSP, namely M1 and M2, which contain the senders that are scheduled *in one slot* during the same working period and the senders that are scheduled *in any time slot* during the same working period, respectively. To achieve it, we first randomly pick a node from A, add it to $M1$ and remove it from A. For each node in A which satisfies the following two conditions: (a) be collision–free with any node in $M1$, (b) has a different parent from any node in $M2$, will be added to $M1$ and removed from A. Condition (a) is the same with MSSP. Condition (b) lies in the fact that a receiver (*i.e.*, a parent) can only be active in one slot during a working period. As a result, if a node is scheduled in the current working period, then its brothers must be scheduled in the next working period or later.

By replacing the sub-procedure GSSP in GAS-PAS with MGSSP, we can easily obtain the pseudo-code of GAS-SAS, which is showed in Algorithm 4. As GAS-SAS is similar with GAS-SAS formally, you can refer to Section 4.2.1 for detailed description.

Algorithm 3 MGSSP (*A, B, G, T*)

```
Input: a sender set $A$, a receiver set $B$, a graph $G$ and a data aggregation tree $T$
Output: A schedule sequence of $S(w,t)$
1    $M1 \leftarrow \varnothing,\ M2 \leftarrow \varnothing,\ w \leftarrow 1, t \leftarrow 1$
2    randomly pick a node from $A$, add it to $M1$ and remove it from $A$
3    While $A \neq \varnothing$ do
4        for each node $v$ in $A$ do
5            if $\forall u \in M, p(u) \neq p(v) \,\&\&\, (p(u),v) \notin E$
                 $\&\&\, (p(v),u) \notin E$ then
6                if $\forall u \in M2, PT(v) \neq PT(u)$ then
7                    add $v$ to $M1$ and remove it from $A$
8        $S(w,t) \leftarrow M1$, output the schedule $S(w,t)$
9        $t \leftarrow t+1$
10       if $t \leq |P|$ then
11           $M2 \leftarrow M2 \bigcup M1,\ M1 \leftarrow \varnothing$
12       else then
13           $M1 \leftarrow \varnothing,\ M2 \leftarrow \varnothing,\ t \leftarrow 1, w \leftarrow w+1$
```

Algorithm 4 GAS-SAS (*G, T*)

```
Input: $G(V,E)$, $T$.
5    MGSSP($V \setminus (U \bigcup W)$, $U \bigcup W$, $G$, $T$)
6    Remove the nodes in $V \setminus (U \bigcup W)$ from $T$
7    for $i \leftarrow d'$ to 1 do
8        MGSSP($N(i)$, $N(i-1)$, $G$, $T'$)
Note: $d'$ is the depth of the remainder tree $T'$ where the WHITE nodes have been removed.
```

4.3. Algorithm Analysis

In this section we present detailed analyses about the latency bounds of GAS-PAS and GAS-SAS, which are of great importance on the efficiency of our algorithms.

Lemma 1 Let A and B be the input sender set and receiver set of GSSP, respectively. For each node u in B, let $\Delta(u)$ be the number of nodes adjacent to u in A. Let w_{m} be the maximum number of working periods that is needed for GSSP to schedule all the nodes in A, then we have

$$w_m \le 2\Delta_m - 1 \text{, where } \Delta_m \text{ is } \max_{u \in B} \Delta(u) .$$

PROOF. In GSSP, the input sender set A is divided into $|P|$ sub-sets and each sub-set is scheduled at the same time slot during different working periods by the same way. Let A_m be the sub-set that needs w_m working periods to be scheduled. To find w_m is equivalent to find the number of the collision-free sub-sets which A_i is further divided into. Let these sub-sets be $\{M_i \,|\, 1 \le i \le w_m\}$, which are all non-null and collision-free. Let $x_i \in M_i$, then any two nodes in the set $X = \{x_i \,|\, 1 \le i \le w_m\}$ are conflicting with each other. Let $PX = \{p(x_i) \,|\, x_i \in X\}$ be the set of parents of nodes in X. We construct a bipartite $G_B(X \bigcup PX, EX)$, where EX is the set of edges initialized to NULL. We declare that for node $x_i \in X$ and node $p(x_i) \in PX$, edge $(x_i, p(x_j))$ will be added to EX if and only if x_i and $p(x_i)$ are in the transmission range (interference range) of each other. For the conflicting nodes x_i and x_j (*i*≠*j*), there are three cases of collisions as stated in Section 4.2. For the second and third cases, according to the declaration, we know that either $(x_i, p(x_j))$ or $(x_j, p(x_i))$ can be added to EX. For the first case, i.e. $p(x_i)$ and $p(x_j)$ are the same, then both $(x_i, p(x_j))$ and $(x_j, p(x_i))$ can be added to EX according to the declaration above. In a word, we can conclude that if node x_i and x_j (*i*≠*j*) conflict, at least one edge can be added to EX. Note that edge $(x_i, p(x_i))$ ($1 \le i \le w_m$) can also be added to EX according the declaration. As there are totally $C_{w_m}^2$ pairs of conflicting nodes, so we conclude that the number of edge in EX is

$$|EX| \ge C_{w_m}^2 + w_m \tag{1}$$

On the other hand, for that the maximum number of nodes in X adjacent to one node in PX is Δ_m, *i.e.*, $p(x_i)$ has at most Δ_m adjacent links, so we have

$$|EX| \le \Delta_m * w_m \tag{2}$$

According to (1) and (2), we have

$$C_{w_m}^2 + w_m \le \Delta_m * w_m \tag{3}$$

Thus

$$w_m \le 2\Delta_m - 1 \tag{4}$$

This finishes the proof.

Lemma 2 Let A and B be the input sender set and receiver set of MGSSP, respectively. For each node u in B, let $\Delta'(u)$ be the number of nodes adjacent to u in A. Let w'_m be the maximum number of working periods that is needed for MGSSP to schedule all the nodes in A, then we have

$$w'_{\mathrm{m}} \le \Delta'_m + \left\lceil \frac{\Delta'_m - 1}{|P|} \right\rceil \text{, where } \Delta'_m \text{ is } \max_{u \in B} \Delta'(u) .$$

PROOF. From the aforementioned proof for GSSP, we can figure out that there are at most $2\Delta'_m - 1$ nodes (denoted by C) in A which are conflicting with each other and hence cannot be scheduled at the same time slot during the same period. We assume that there are at most k ($k \le \Delta'_m$) nodes (denoted by C_1) in C which have the same parents and hence must be scheduled using k working periods. For the remainder nodes in $C \backslash C_1$, they can be scheduled either during the k working periods or during other working periods. As MGSSP is base on greedy strategy (see Line 4-7) which seeks to find as much nodes that can be scheduled at the same slot as possible, so for nodes in $C \backslash C_1$, some of them (may not all) may be scheduled during the k periods with nodes in C_1. However, note that MGSSP randomly pick nodes for scheduling, so if all the nodes in $C \backslash C_1$ are picked before the nodes in C_1, then it needs at most another $\lceil (2\Delta'_m - k - 1)/|P| \rceil$ working periods to schedule these nodes. It should be clarified that some nodes in $C \backslash C_1$ with the same parent may not be scheduled in these $\lceil (2\Delta'_m - k - 1)/|P| \rceil$ periods for their common parent can only be awake at one slot during a working period. At such circumstances these nodes can all be scheduled together with the nodes in C_1, where k working periods is enough for them to be scheduled. So we can conclude that the maximum working periods w'_{m} needed for MGSSP is

$$w'_{\mathrm{m}} = k + \left\lceil \frac{2\Delta'_m - k - 1}{|P|} \right\rceil \tag{5}$$

For $k \le \Delta'_m, |P| \ge 1$, we have

$$w'_{\mathrm{m}} \le \Delta'_m + \left\lceil \frac{\Delta'_m - 1}{|P|} \right\rceil \tag{6}$$

This finishes the proof.

Lemma 3 After removing the WHITE nodes from the data aggregation tree T, let d' be the depth of the remainder tree T', then $d' \le 2(R-1)$, where R is network radius of G.

PROOF. As the sink in G is BLACK, so all its neighbours are GREY. Thus the number of layers in which BLACK nodes exist does not exceed *R* (Layer 0 is included). For every two adjacent layers of BLACK nodes, there exists at most one layer of GREY nodes that interconnect the disjointed BLACK nodes. So the number of layers where GREY nodes exist does not exceed *R*-1. Hence in T', the total number of layers does not exceed 2*R*-1, implying that the depth does not exceed 2*R*-2, so we have $d' \le 2(R-1)$ .It is easy to find that in T', the nodes in the even layer are all BLACK and the nodes in the odd layer are all GREY, and the total number of layers is odd. This finishes the proof.

Lemma 4 [8] A GREY node is adjacent to at most 5 BLACK nodes in *G*.

Lemma 5 [8] Suppose that *u*, *v* and *w* are BLACK nodes in *G*, and both *v* and *w* are within two-hops of *u*. Let v' and w' be the corresponding GREY nodes which interconnect *v* and w with *u*, respectively. Then either v' is adjacent to *w* or w' is adjacent to *v*, if $vuw \le 2\arcsin\frac{1}{4}$.

Lemma 6 After removing the WHITE nodes from the data aggregation tree T, let d' be the depth of the remainder tree T'. For each node *u* in layer *i* $(0 \le i \le d'-1)$ of T', let $\Delta_{i+1}(u)$ be number of nodes adjacent to *u* in layer $i+1$, then we have the following conclusions.

$$\Delta_i(u) <= \begin{cases} 4, & i \text{ is odd} \\ 11, & i \text{ is even and } i > 0 \\ 12, & i = 0 \end{cases}$$

PROOF. We prove Lemma 6 based on Lemm4 and Lemm5. According to Lemma 4, a GREY node is adjacent to at most 5 BLACK nodes. Note that a GREY node has a BLACK parent in its upper layer, so it is adjacent to at most 4 BLACK nodes in its lower layer. As the nodes in the odd layer are GREY and the nodes in the even layer are BLACK, so for the odd layer *i*, node *u* (GREY) is adjacent to at most 4 nodes (BLACK) in layer *i*+1, *i.e.*, $\Delta_i(u)$ <= 4.

According to Lemma 5, we can figure out that a BLACK node is adjacent to at most 12 GREY nodes in T'. We prove this by contradiction. Assume that a BALCK node *u* is adjacent to *n* (*n* >= 13) GREY nodes. For each GREY node, it is not removed because that there exists at least one BLACK node which can only use it to connect with its dominator. So there are at least *n* BLACK nodes that can only use each of the *n* GREY nodes to connect with the corresponding dominators. With *drawer principle*, there exist two BLACK nodes *v* and *w* within two-hops of u that satisfy $vuw \le \frac{2\pi}{n} < 2\arcsin\frac{1}{4}, (n \ge 13)$. According Lemma 5, either *v* is adjacent to *w*' or *w*' is adjacent *v*, where *v*' and *w*' are the corresponding GREY nodes that *v* and *w* can only use to connect, respectively. This produces contradiction and hence *n* < 13, indicating that at most 12 GREY nodes is adjacent to a common BLACK node after removing the redundant ones. As the nodes in the odd layer are BLACK and the nodes in the even layer are GREY, so for the odd layer *i*, node *u* (BLACK) is adjacent to at most 12 nodes (GREY) in layer *i*+1, i.e. $\Delta_i(u)$ <= 12. Note that for the even layer *i* when *i* >0 , node *u* must have a GREY parent in layer *i*-1, so *u* can be adjacent to at most 11 GREY nodes in layer *i*+1, i.e. $\Delta_i(u)$ <= 11. This finishes the proof.

Theorem 1 The latency upper bound of GAS-PAS is $28R + 2\Delta - 29$ working periods, where *R* and Δ are the network radius and the maximum node degree of G, respectively.

PROOF. First the WHITE nodes in G are scheduled by GSSP, which regards all the WHITE nodes as the input sender set A and BLACK nodes in G as the input receiver set B. For that a BLACK node has a parent marked GREY, so a BLACK node in B is adjacent to at most $\Delta - 1$ WHITE nodes in A. Thus according to Lemma 1, it takes at most $2\Delta - 3$ to schedule all the WHITE nodes.

Then the BLACK nodes and GREY nodes are scheduled layer by layer. When scheduling the nodes in layer *i* of the data aggregation tree T, GSSP regards all the nodes in Layer *i* as the sender set A and all the nodes in Layer *i*-1 as the receiver set B. Based on Lemma 6 and Lemma 1, we can easily calculate that when scheduling the even layer (BLACK nodes), it takes at most 7 working periods and when scheduling the odd layer (WHITE nodes), it takes at most 21 working periods. The layer 1 is a special case, which takes at most 23 working periods. Let d' be the depth of T', according to Lemma 1 and Lemma 3, the maximum number of working periods needed for scheduling all the nodes except for the sink in T' is

$$\begin{aligned} & 2\Delta - 3 + \sum_{i=1}^{i=d} w_m(i) \\ & \le (21+7) * \frac{d'-2}{2} + 23 + 7 + (2\Delta - 3) \\ & \le 28 * (R-2) + 30 + (2\Delta - 3) \\ & = 28R + 2\Delta - 29 \end{aligned}$$

Therefore, in total it takes at most $28R+2\Delta-29$ working periods to schedule all the nodes in *G* by GAS-PAS.

Theorem 2 The upper bound of the latency of GAS-SAS is $(16+\left\lceil\frac{13}{|P|}\right\rceil)R+\Delta+\left\lceil\frac{\Delta-1}{|P|}\right\rceil-13-\left\lceil\frac{12}{|P|}\right\rceil$ working periods, where R and Δ are the network radius and the maximum node degree of *G*, respectively.

PROOF. The proof of Theorem 2 is similar with Theorem 1 except that for Theorem 2, we employ Lemma 2 to bind the latency of MGSSP in GAS-SAS. Similarly, the maximum number of working periods needed for scheduling all the nodes except for the sink in *G* by GAS-SAS is

$$\Delta+\left\lceil\frac{\Delta-1}{|P|}\right\rceil+\sum_{i=1}^{d}w_m(i)$$

$$\leq\Delta+\left\lceil\frac{\Delta-1}{|P|}\right\rceil+(4+\left\lceil\frac{3}{|P|}\right\rceil+11+\left\lceil\frac{10}{|P|}\right\rceil)*(R-2)+4+\left\lceil\frac{3}{|P|}\right\rceil+12+\left\lceil\frac{11}{|P|}\right\rceil$$

$$\leq(16+\left\lceil\frac{13}{|P|}\right\rceil)R+\Delta+\left\lceil\frac{\Delta-1}{|P|}\right\rceil-13-\left\lceil\frac{12}{|P|}\right\rceil$$

This finishes the proof.

Form Theorem 1 and Theorem 2, we can conclude that the worst-case latency of GAS-PAS and GAS-SAS are both bounded by $O(R+\Delta)$ periods, which indicates that the proposed algorithms achieve constant approximation to the optimal latency [8].

5. Simulation Results

In this section, we conduct extensive simulations to evaluate the latency performance of GAS-PAS and GAS-SAS. The sensor nodes are deployed randomly and uniformly in a region of 200m*200m and the sink is deployed in the centre of the region. All nodes have the same transmission range and interference range. As our algorithms are designed for duty-cycling WSN, we first simulate the aggregation time in terms of the duty cycle, which is defined as the ratio between the active time and the whole working period, *i.e.*, 1/|*P*|. We fix the number of nodes in the region as 100 and the transmission range as 30m. The values of duty cycle are 5%, 6.67%, 10%, 12.5%, 20%, 25%, 33.3% and 50% according to |*P*|=20, 15, 10, 8, 5, 4, 3 and 2, respectively. From Figure 2, we can see that when the duty cycle increases, the number of working periods also increases. This lies in the fact more nodes can be scheduled when the length of a working period is long while few nodes can be scheduled when the length is short. However, when the duty cycle decreases, the total aggregation time slots may increase as showed in Figure 3. From both Figure 2 and Figure 3, we can see that given the same duty cycle, GAS-SAS needs fewer working periods than GAS-PAS because GAS-SAS tries to schedule as much nodes as possible during one working period.

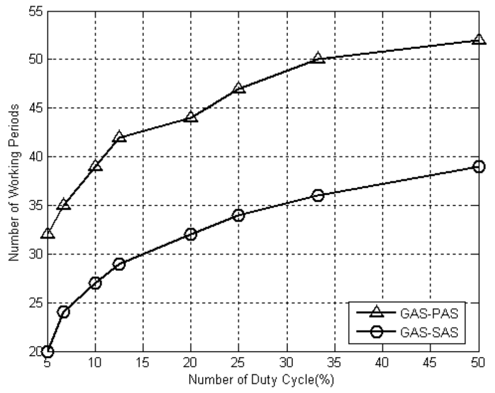

Figure 2. Number of Working Periods with Different Duty Cycle Value

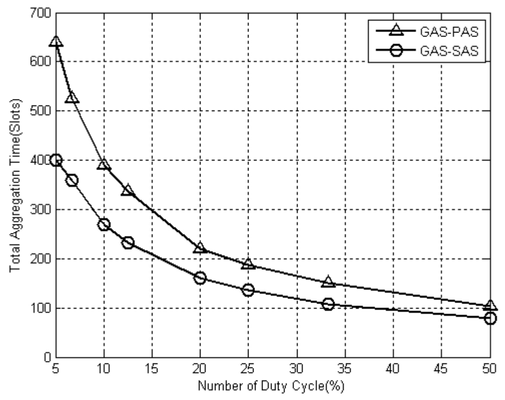

Figure 3. Total Aggregation Time Latency with Different Duty Cycle Value

We then compare GAS-PAS and GAS-SAS with the state-of-the-art algorithms for MLAS problem. The algorithms compared with are the best centralized algorithm EPAS in **[]** and the best distributed algorithm CluDDAS in []. As these algorithms are both designed for non-duty-cycling WSN, we slightly modify them for dc-WSN. If node *u* sends data to node *v* at the time slot *t* according to the schedule result of EPAS and CluDDAS, then *u* will send data to node *v* at the active slot of *v* during the working period *t* in the modified algorithms, namely M-EPAS and M-CluDDAS for dc-WSN. We fix the duty cycle as 20%. Figure 4 shows the total number of working periods in terms of the number of nodes in the region ranging from 100 to 800. Here the transmission range is fixed as 30m. Figure 5 shows the total number of working periods in terms of the transmission range varing from 15m to 50m, where the number of nodes is fixed as 400. We can see that the increasement of either the number of nodes or the transmission range will cause the nuber of working periods for all the algorithms increase because at such circumstances the interference between the nodes will be more severe. We also can see that both GAS-PAS and GAS-SAS outperform the others with less woring periods in Figure 4 and 5. This is because that conflicting nodes in non-duty-cycling WSN can transmit in the same working period in duty-cycling WSN, as long as they transmit in different slots during a working period.

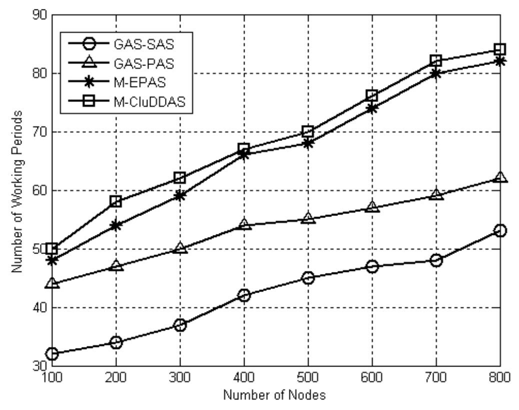

Figure 4. Number of Working Periods with Different Number of Nodes

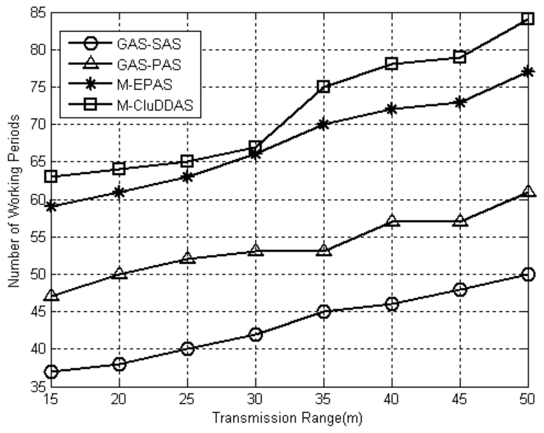

Figure 5. Number of Working Periods with Different Transmission Range

6. Conclusion

In this paper, we study the data aggregation scheduling problem in duty-cycling wireless sensor networks, aiming at minimizing the total latency. Based on a reduced Connected Dominating Set, we propose two heuristic algorithms GAS-PAS and GAS-SAS using greedy strategy for two cases of dc-WSN. Through theoretical analyses, we show that both GAS-PAS and GAS-SAS are nearly constant approximation. Simulations are conducted to show that both GAS-PAS and GAS-SAS have a good performance of latency comparing with state-of-the-art algorithms.

References

[1] Bagaa and Miloud, "Semi-structured and unstructured data aggregation scheduling in wireless sensor networks", INFOCOM, 2012 Proceedings IEEE, IEEE, **(2012)**.
[2] Clark, N. Brent, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs", Discrete Mathematics, vol. 86, no. 1, **(1990)**, pp. 165-177.
[3] Yu, Bo, J. Li, and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks", INFOCOM 2009, IEEE, IEEE, **(2009)**.
[4] Chen, Xujin, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks", Mobile Ad-hoc and Sensor Networks, Springer Berlin Heidelberg, **(2005)**, pp. 133-142.
[5] Li, Deying, Q. Zhu and W. Chen, "Efficient algorithm for maximum lifetime many-to-one data aggregation in wireless sensor networks", International Journal of Sensor Networks, vol. 9, no. 2, **(2011)**, pp. 61-68.
[6] Fasolo and Elena, "In-network aggregation techniques for wireless sensor networks: a survey", Wireless Communications, IEEE, vol. 14, no. 2, **(2007)**, pp. 70-87.

[7] Gupta, Piyush and P. R. Kumar, "The capacity of wireless networks", Information Theory, IEEE Transactions, vol. 46, no. 2, **(2000)**, pp. 388-404.
[8] Huang and C. H. Scott, "Nearly constant approximation for data aggregation scheduling in wireless sensor networks", INFOCOM 2007, 26th IEEE International Conference on Computer Communications, IEEE, **(2007)**.
[9] Li, Yingshu, L. Guo and S. K. Prasad, "An energy-efficient distributed algorithm for minimum-latency aggregation scheduling in wireless sensor networks", Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference, IEEE, **(2010)**.
[10] Buettner and Michael, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks", Proceedings of the 4th international conference on Embedded networked sensor systems, ACM, **(2006)**.
[11] Kuo, T. Wei and M. J. Tsai, "On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: NP-completeness and approximation algorithms", INFOCOM, 2012 Proceedings IEEE, **(2012)**.
[12] Wan, P. Jun, K. M. Alzoubi and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks", INFOCOM 2002, Proceedings of the 21st annual joint conference of the IEEE computer and communications societies, IEEE, vol. 3, **(2002)**.
[13] Wan and P. Jun, "Minimum-latency aggregation scheduling in multihop wireless networks", Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing, ACM, **(2009)**.
[14] Xu and X. Hua, "A delay-efficient algorithm for data aggregation in multihop wireless sensor networks", Parallel and Distributed Systems, IEEE transactions, vol. 22, no. 1, **(2011)**, pp. 163-175.
[15] Wu and Yan, "Constructing maximum-lifetime data gathering forests in sensor networks", IEEE/ACM Transactions on Networking (TON), vol. 18, no. 5, **(2010)**, pp. 1571-1584.
[16] Cao, Yongle, S. Guo and T. He, "Robust multi-pipeline scheduling in low-duty-cycle wireless sensor networks", INFOCOM, 2012 Proceedings IEEE, **(2012)**.
[17] Jiao and Xianlong, "Minimum latency broadcast scheduling in duty-cycled multihop wireless networks", Parallel and Distributed Systems, IEEE Transactions, vol. 23, no. 1, **(2012)**, pp. 110-117.

Authors

Sun Chengting, he is born in 1969, and currently serves as an Associate Professor in Information Engineering College, Lianyungang Technical College. His research interest is computer application technology.

Wang Zhen, he is born in 1981, and is now a Lecture in Information Engineering College, Lianyungang Technical College. His main research direction is computer software technology.