LBA: Lifetime Balanced Data Aggregation in Low Duty Cycle Sensor Networks

Zi Li, Yang Peng, Daji Qiao, and Wensheng Zhang
Iowa State University, Ames, IA, USA
Email: {zili, yangpeng, daji, wzhang}@iastate.edu

***Abstract*—This paper proposes LBA, a lifetime balanced data aggregation scheme for asynchronous and duty cycle sensor networks under an application-specific requirement of end-to-end data delivery delay bound. In contrast to existing aggregation schemes that focus on reducing the energy consumption and extending the operational lifetime of each individual node, LBA has a unique design goal to balance the nodal lifetime and thus prolong the network lifetime more effectively. To achieve this goal in a distributed manner, LBA adaptively adjusts the aggregation holding time between neighboring nodes to balance their nodal lifetime; as such balancing take place in all neighborhoods, nodes in the entire network can gradually adjust their nodal lifetime towards the globally balanced status. Experimental studies on a sensor network testbed shows that LBA can achieve the design goal, yield longer network lifetime than other non-adaptive and nodal lifetime-unaware data aggregation schemes, and approach the theoretical upperbound performance, especially when nodes have highly different nodal lifetime.**

I. Introduction

A. Motivations

In a sensor network, sensor nodes are usually powered by small batteries with limited energy supplies. When applying the network for long-time applications such as continuous environmental monitoring, how to prolong the network lifetime is of critical importance.

Through eliminating inherent redundancy in raw sensory data, in-network data aggregation [1], [2] has been widely applied as an effective technique to reduce communication cost and extend the lifetime of a sensor network. With the data aggregation mechanism, a node should be allowed to hold data received or generated by itself for a while, aggregate the data in bulk, and send out only the aggregated results. The extend to which data volume can be suppressed highly depends on how long a node can hold data before sending them out. Generally, the longer can a node hold data, the more data can it suppress and hence the higher is the communication efficiency. However, holding data introduces extra data delivery delay. In many sensor network applications, the value of sensory data could be greatly depreciated or even become zero if the data is delivered to the sink with a delay longer than a certain application-specific delay bound. Therefore, the allowed holding time is constrained by the application-specific requirement on end-to-end data delivery delay.

It is important to make full use of the available but constrained holding time for aggregation to prolong network lifetime. This becomes especially demanding in the context of multi-hop sensor networks. Along each multi-hop source-to-sink path, the allowed holding time should be allocated to all nodes appropriately to ensure the required end-to-end data delivery delay is not violated along the path. Research [3] has been conducted to optimize the distribution of holding time according to the distribution of data traffic and the network topology. However, non-uniform nodal lifetime, which is a critical factor impacting network lifetime, has been neglected.

Due to various environment and system reasons, sensor nodes in the same network may have various nodal lifetime. For example, nodes with batteries of poorer quality, nodes that are bottlenecks on a collection tree, and solar-rechargeable nodes deployed to shady locales may have shorter lifetime than their peers. As the energy depletion in a sensor node may cause network disconnection or create coverage holes, which could render the entire sensor network nonfunctional, many sensor network applications [4]–[6] define the network lifetime as the minimal nodal lifetime among all sensor nodes in the network. Therefore, in order to prolong the network lifetime, it is critical to prolong the lifetime of the shortest-nodal-lifetime nodes.

Despite the need for a multi-layer holistic approach to balance nodal lifetime and thus prolong the network lifetime, it is necessary to design a data aggregation scheme that can take into account the non-uniform nodal lifetime among nodes, attempt to balance their nodal lifetime, and thus more effectively prolong the network lifetime.

B. Related Works

Many in-network data aggregation protocols [7]–[9] have been proposed, but the timeliness of the data delivery was not a concern. Although Ye *et al.* [10] formulated the energy-delay tradeoff problem as a semi-Markov decision process by depreciating the data revenue as aggregation holding time increases, no explicit end-to-end delivery delay bound was considered. To bound end-to-end delivery delay, many existing works [11]–[13] require time synchronization between neighboring nodes. Solis *et al.* [11] employed the concept of cascading timeout where a node's aggregation timeout happens right before its parent's to achieve high aggregation degree with small delay overhead. Xiang *et al.* [12] explored the joint data aggregation and timeliness of data delivery problem, and proposed a utility-based scheme called tPack to minimize the whole network communication cost. Assuming a synchronized time-slotted system, [13] formulated the energy-delay tradeoff problem as an integer optimization problem. Different from these works, our scheme does not require synchronization. Moreover, all of the afore-mentioned works aim to minimize the energy consumption, without considering the lifetime balance between nodes which is critical in improving the network lifetime.

[3] investigated the problem of energy efficient data delivery within a delay bound and proposed two distributed schemes to balance energy consumption among sensor nodes. However, the schemes work only in homogeneous networks (e.g., the battery quality, radio energy consumption rate and initial nodal energy are the same) whereas our scheme can deal with the heterogenous situations. There have also been works [14], [15] on optimizing aggregation tree structure. Our scheme is

978-1-4673-0775-8/12/$31.00 ©2012 IEEE

orthogonal to these works, because our scheme works under any aggregation tree structure.

C. Contributions

In this paper, we present LBA, a low cost, asynchronous, delay-constrained data aggregation scheme for duty cycle sensor networks. LBA aims to prolong the network lifetime. The key idea is to dynamically adjust the aggregation holding time between two neighboring nodes and hence to balance their nodal lifetime. As neighboring nodes keep balancing their nodal lifetime, the nodal lifetime of all nodes in the entire network can be balanced gradually and the network lifetime can be extended.

We have implemented and experimented LBA in a testbed of 32 TelosB sensor nodes. Experimental results show that LBA effectively achieves the design goal of balancing the nodal lifetime, and prolongs the sensor network lifetime under various network configurations, especially the heterogeneous ones. Through theoretical analysis, we also proposed a network lifetime upperbound. According to our evaluation results, LBA can approach the lifetime upperbound especially when the nodes have highly different nodal lifetime.

D. Organization

In the rest of the paper, Section II describes the system model and problem definition. Sections III, IV and V present the design overview, analytical study and detailed design. Section VI reports the testbed implementation and experimental results. Finally, Section VII concludes the paper.

II. System Model and Problem Statement

A. System Model

We study data aggregation in a sensor network, where each sensor node could periodically generate and report sensory data, and all the nodes form a collection tree rooted at the sink. The nodes may not be time-synchronized or energy-synchronized (i.e., their energy supplies and consumption rates may not be uniform). The nodes are duty-cycled for energy saving, which are typical when the network is deployed for long-time monitoring. To build and maintain the data collection tree, a routing protocol such as the collection tree protocol (CTP) [16] may be employed. Underlying the routing protocol, an asynchronous and duty cycle MAC protocol such as RI-MAC [17] may be adopted.

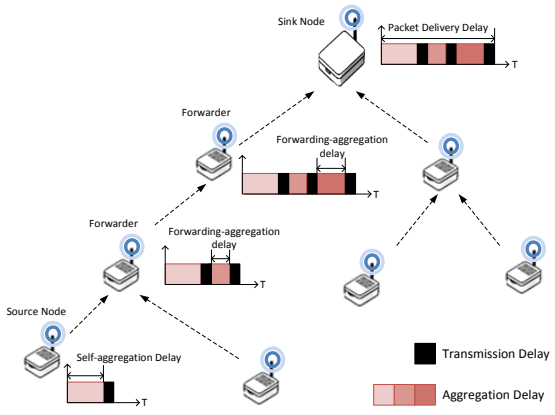


Fig. 1. System model.

A typical data aggregation process is shown in Figure 1. A source node may not send out sensory data packet immediately after it is generated; instead, the node may wait a certain period of time (*self-aggregation delay*), attempting to aggregate multiple data packets generated during the period and thus reduce the amount of data sent to its parent node. Similarly, a forwarding node may not forward a data packet immediately after the reception. It may wait for another period of time (*forwarding-aggregation delay*) in order to aggregate multiple data packets received during the period. Consequently, the delay involved in delivering a data packet from source to sink is composed of a self-aggregation delay and multiple forwarding-aggregation delays. In the rest of the paper, we denote the average data generation rate at each node i as λ_i, and the average forwarding-aggregation and self-aggregation delay at node i as w_i and w_i' respectively.

We assume the *total aggregation* model [18] in data aggregation; that is, an arbitrary number of data packets that are available at the aggregation time can be suppressed into a single data packet. Such model can be seen in many sensor network applications. For example, in monitoring applications, users often are more interested in the maximum, minimum or average values, or the percentile statistics, of sensory data, rather than the raw data themselves.

As we can see, the longer is the aggregation delay, the more data packets can be aggregated and thus the higher is the communication efficiency. However, monitoring applications often also require that data delivery delay be lower than a certain bound to assure timely awareness of the monitored environment. We assume the following generic delay requirement: *at least p percent of sensory data should be delivered to the sink within time D after the data has been generated, where D and p are application-specific parameters.*

B. Problem Statement

Our design objective is to dynamically determine the forwarding-aggregation and self-aggregation delays (i.e., w_i and w_i') for each node i to try to balance the lifetime of sensor nodes and hence prolong the network lifetime, under the condition that the data delivery delay requirement is satisfied. In this course, the differences between nodes, for example, different nodal energy levels, energy consumption rates and data generation rates, should be considered. Specifically, the problem can be formalized as follows:

Given: $T,\ \{\lambda_i\},\ \{e_i\}.$

Objective: $\max \min\{L_i\}$

Subject to: $\forall l, w_l' + \sum_{i \in s_l} w_i \leq D_p$, where l is leaf

Output: $\{w_i, w_i'\}.$

Here, T is the collection tree topology, e_i is the residual energy and L_i represents the nodal lifetime of node i. l is a leaf node and s_l denotes the path connecting the parent of l and the sink in T. D_p represents the maximal allowed delay, including both transmission and aggregation delay, along each source-sink path such that at least p percent of the source's sensory data can reach the sink within D time. The calculation of L_i and D_p is to be presented in Section IV.

III. Design Overview

Solving the above optimization problem in a centralized manner is impractical as it requires each node to know the residual energy levels, data generation rates of all other nodes, and the topology of the network. Acquiring these information could incur high communication overhead because of potentially large network scale and dynamic nature of the information. Therefore, we approach the problem in a distributed and localized

manner. Specifically, coordinations only take place between neighboring parent-child nodes, which exchange information with each other and coordinately adjust their aggregation delays.

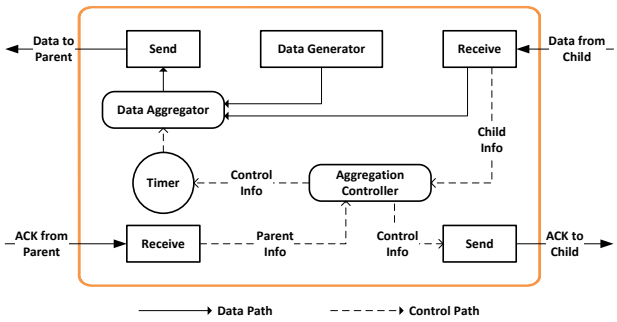


Fig. 2. Design overview.

As shown in Figure 2, the coordination is realized by piggy-backing some small pieces of control information at the end of the data and ACK packets exchanged between parent-child nodes. Specifically, when sending out a data packet to its parent, a node appends to the packet its estimated nodal lifetime and some other information related to the adjustment of aggregation delays. Upon receiving the packet, the parent node pushes the packet into a queue while feeding the appended information to the "Aggregation Controller" module. Based on the information, the parent node adjusts its own w_i, and appends some information to the ACK to instruct its children nodes on how to adjust their w_i values. When the ACK reaches a child node j, the node changes its w_j and w'_j accordingly. This way, as every parent-child pair keeps attempting to balance their nodal lifetime, the nodal lifetime of all nodes in the entire network is also adjusted gradually towards the balanced status.

IV. Analytical Study

To provide a theoretical foundation to direct and evaluate the performance of our design, we present extensive analysis in this section. Specifically, we first provide a formulation of nodal lifetime as a function of a node's data input/output rates. Then, the data I/O rates are formulated as functions of nodal data generation rates, self-aggregation and forwarding-aggregation delays. This is followed by the computation of the maximum aggregation and transmission delays that all the nodes on a source-sink path are allowed to introduce without violating a certain application-specific data delivery delay requirement. Based on the above analysis, we finally propose an algorithm to compute an upperbound for the network lifetime through intelligently determining the self-aggregation and forwarding-aggregation delays for each node.

A. Nodal Lifetime

As our design targets to be applicable in time-asynchronous duty cycle sensor networks, a time-asynchronous duty cycle MAC protocol is assumed. In the analysis and detailed design presented in this paper, we take RI-MAC [17], a well-known asynchronous and duty-cycled MAC protocol, as an example. Note that, our design can work with other MAC protocols, e.g., X-MAC [19], A-MAC [20], after replacing the nodal lifetime analysis module of RI-MAC with that of the alternative MAC protocol.

For self-containedness, the sketch of RI-MAC is illustrated in Figure 3. In this protocol, each node periodically wakes up for time ϕ every interval T_r to check if there is any incoming packet intended for the node. After turning on its radio, a node immediately broadcasts a beacon, announcing its readiness for

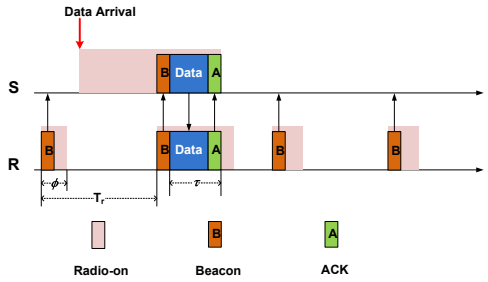


Fig. 3. The sketch of RI-MAC protocol. T_r is the receiver's beacon interval, ϕ is the receiver's channel checking period and τ is the time needed by the sender and receiver to exchange a data packet and the corresponding ACK.

receiving packets. If a node has packet to send (e.g., node S in Figure 3) it turns on its radio if it is off, and keeps the radio on to wait for the beacon from the intended receiver (e.g., node R in Figure 3). Upon receiving the expected beacon, S sends out its data immediately, which will be acknowledged by R with an ACK after R has received the data. Then, S turns off its radio, while R and any other sender who has pending packet intended for R can start transmission. If there are no more incoming/outgoing packets, R turns off its radio and goes to sleep. Further, due to the lossy nature of the wireless channels, a sender may need several transmission attempts to successfully send out a packet. Let ETX_i denote the expected transmission attempts[1] at node i. Let m denote the maximum number of transmission trials before a sender gives up its transmission attempt until the next time it receives beacon from its intended receiver. Then, we can estimate the nodal lifetime L_i as follows:

$$\frac{e_i}{((T_r(\lceil \frac{ETX_i}{m} \rceil - \frac{1}{2}) + \tau(ETX_i \% m))\lambda'_i + \frac{\phi}{T_r} + \tau \sum_{j \in C_i} \lambda'_j)P}. \quad (1)$$

Here, P is the energy consumption rate when a node's radio is on, λ'_i denotes the data output rate of node i, and τ is the time to send or receive a data packet. $\sum_{j \in C_i} \lambda'_j$ denotes the data input rate at node i, where C_i is the set of i's children and λ'_j is the data output rate of node j. A node could be both a sender and a receiver. As a sender, the node waits $\frac{T_r}{2}$ on average for its receiver to wake up and then consumes time τ for each transmission attempt until the number of unsuccessful attempts reaches m, in which case it waits for another T_r and then repeats the transmission attempts. Hence, with average number of attempts ETX_i for each data packet, the overall energy consumption for the successful transmission of a data packet is $T_r(\lceil \frac{ETX_i}{m} \rceil - \frac{1}{2}) + \tau(ETX_i \% m)$. As a receiver, the node wakes up for time ϕ every interval T_r, and spends time $\tau \sum_{j \in C_i} \lambda'_j$ on receiving data packets. Hence, the denominator of Eq. (1) estimates the energy consumption rate at node i. Note that, we here only estimate the energy consumption for communication, which is usually the most significant part of nodal energy consumption. It can be easily extended to include the energy consumption for other reasons such as sensing and computation.

B. Nodal Data Input/Output Rates

To facilitate theoretical analysis and evaluation of our design, we assume sensory data packets are generated at node i with the intervals following an exponential distribution of mean $1/\lambda_i$, and therefore the flow of sensory data packets generated by a node follows a Poisson distribution of mean λ_i.

[1]Note that routing protocols such as CTP [16] in sensor networks have provide mechanisms for automatical measuring ETX_i for node i. We therefore do not describe how to measure ETX_i in this paper.

Our design also intentionally shapes the data packet flows after aggregations to follow the Poisson distribution. Particularly, if node i is a leaf node, it maintains a timer A'_i which fires every time interval W'_i, where W'_i is a random variable following the exponential distribution with mean w'_i. If node i is a non-leaf node, it maintains a timer A_i which fires every time interval W_i, where W_i is a random variable following the exponential distribution with mean w_i. Every time when the timer (A'_i or A_i) fires, data packets are aggregated into a single packet and then sent to its parent node; if there is no data packet to send when the timer fires, a dummy packet is sent to ensure that the output data flow follows Poisson distribution. As described below, our scheme adopts a simple enhancement to keep the overhead low by reducing the dummy packets as much as possible.

Based on the above assumption and mechanism, we next formulate the nodal average data output rate (denoted as λ'_i for each node i) as a function of its data input rate (i.e., $\sum_{j\in C_i}\lambda'_j$ where C_i is the set of children nodes of i), its own data generation rate (i.e., λ_i) and its own forwarding-aggregation and self-aggregation delays (i.e., w_i and w'_i). Note that a node's data input rate is the sum of the data output rates of its children; hence, we do not need a separate analysis for data input rate.

When a node is a pure source, its self-aggregation delay w'_i (i.e., the maximal time it can wait to aggregate its own sensory data without violating the delay requirement) can be computed as follows:

$$w'_i = D_p - \sum_{k\in s_i}(w_k + d_k) - d_i, \tag{2}$$

where d_i denotes the average transmission delay at node i. That is, the sum of aggregation and transmission delays of all nodes on the path from node i to the sink should not exceed D_p, the maximum aggregation and transmission delay that any source-sink path is allowed to introduce. The output rate of node i is simply $\frac{1}{w'_i}$ if one packet (aggregated data or dummy) is sent every time timer A'_i fires. However, when the average data generation interval $\frac{1}{\lambda_i}$ is greater than w'_i, a large number of dummy packets may be generated. To reduce the bandwidth waste under this scenario, we instead allow each data packet to be sent out immediately after its generation. With this simple enhancement, the number of dummy packets is reduced and the data output rate of node i, denoted as λ^s_i, becomes:

$$\lambda^s_i = \begin{cases} \lambda_i & : w'_i \le \frac{1}{\lambda_i}; \\ \frac{1}{w'_i} & : otherwise. \end{cases} \tag{3}$$

Generally, when node i is a pure forwarder or both forwarder and source, it can use timer A_i by default to regulate output data flow, and the output data rate would be $\frac{1}{w_i}$. However, when the data packets are received or generated by the node in an interval greater than w_i, many dummy packets will have to be sent. To save bandwidth, the following optimization can be applied:

- If packets from descendant nodes arrive at node i with an average interval greater than w_i, timer A_i is not needed. In this case, there are following three subcases:
  - If the arrival interval of packets from descendants is no greater than w'_i, then the packets can be sent in the following way. Whenever a packet from descendants arrives at i, it is aggregated with any un-sent packets generated by i, and the aggregation result is sent to the parent node.
  - If the arrival interval of descent packets is greater than w'_i and node i's self-generated packets arrive at an interval no smaller than w'_i, then any packet (no matter it is from descendants or self-generated) is forwarded to the parent node whenever it arrives at i.
  - Otherwise, timer A'_i is set to fire every interval W'_i where W'_i is a random variable following the exponential distribution of mean w'_i. Packet aggregation and transmission rules are as follows. Whenever a descendant packet arrives at i, it is aggregated with any un-sent self-generated packets, and the aggregation result is then sent to the parent node. Whenever timer A'_i fires, all packets held at node i are aggregated and the aggregation result is sent to the parent node; if there is no un-sent packet at i, a dummy packet is sent to the parent.
- Otherwise, timer A_i is used, and the default way for packet aggregation and sending is applied.

To summarize, node i's data output rate λ'_i can be calculated as follows:

$$\lambda'_i = \begin{cases} \sum_{j\in C_i}\lambda'_j & : w_i \le \frac{1}{\sum_{j\in C_i}\lambda'_j} \le w'_i \\ \lambda_i + \sum_{j\in C_i}\lambda'_j & : w'_i \le \frac{1}{\lambda_i}, w_i \le \frac{1}{\sum_{j\in C_i}\lambda'_j} \\ \frac{1}{w'_i} + \sum_{j\in C_i}\lambda'_j & : w'_i > \frac{1}{\lambda_i}, w'_i \le \frac{1}{\sum_{j\in C_i}\lambda'_j} \\ \frac{1}{w_i} & : otherwise. \end{cases} \tag{4}$$

With the above optimization, the data output flow from i still follows the Poisson distribution.

C. Lowerbound of a Subtree's Data Output Rate

Based on the above analysis of nodal data output rates, we next present a lowerbound of a subtree's data output rate in the following Lemma 4.1. Note that the result is also to be used in computing the performance upperbound.

Lemma 4.1: Let T_i denote a subtree rooted at i, and w_i and w'_i be the forwarding-aggregation and self-aggregation delays of node i respectively. Then a lowerbound of the data output rate of T_i, denoted as $\hat{\lambda}'_i$ is as follows.

$$\hat{\lambda}'_i = \begin{cases} \sum_{j\in T_i}\lambda_j & : w'_i \le \frac{1}{\sum_{j\in T_i}\lambda_j}; \\ \frac{1}{w'_i} & : otherwise. \end{cases} \tag{5}$$

Proof: (sketch) There are totally two cases as in the following.

Case I: $w'_i \le \frac{1}{\sum_{j\in T_i}\lambda_j}$. For this case, we prove by contradiction that the data output rate of T_i cannot be less than $\sum_{j\in T_i}\lambda_j$. Suppose the output rate is lower than the value, then some data should have been suppressed at some node. Suppose node $k \in T_i$ is such a suppressing node but none of its descendants can suppress data. The overall data generation rate at the subtree rooted at T_k is $\sum_{j\in T_k}\lambda_j$, which is no greater than $\sum_{j\in T_i}\lambda_j$; i.e.,

$$\frac{1}{\sum_{j\in T_i}\lambda_j} \le \frac{1}{\sum_{j\in T_k}\lambda_j}. \tag{6}$$

Also due to $w_k \le w'_i$, $w'_k \le w'_i$ and $w'_i \le \frac{1}{\sum_{j\in T_i}\lambda_j}$, it holds that

$$w_k \le max\{w_k, w'_k\} \le \frac{1}{\sum_{j\in T_k}\lambda_j} \le \frac{1}{\sum_{j\in C_k}\lambda'_j}. \tag{7}$$

Further, it is obvious that $\sum_{j\in T_k}\lambda_j \ge \lambda_k$; together with Eq. (7), it holds that

$$w'_k \le max\{w_k, w'_k\} \le \frac{1}{\sum_{j\in T_k}\lambda_j} \le \frac{1}{\lambda_k}. \tag{8}$$

According to Equations (4), (7) and (8), it follows that $\lambda'_k = \lambda_k + \sum_{j\in C_k}\lambda'_j$, where, as no descendant of k suppresses data, $\sum_{j\in C_k}\lambda'_j = \sum_{j\in C_k}\sum_{l\in T_j}\lambda_l$. Therefore, $\lambda'_k = \sum_{j\in T_k}\lambda_j$; i.e., the number of data packets does not decrease, which is a contradiction.

Case II: $w_i' > \frac{1}{\sum_{j\in T_i}\lambda_j}$. In this case, the data output rate of T_i is $\frac{1}{w_i'}$ when w_i is set to w_i'; that is, totally $A = \sum_{j\in T_i}\lambda_j - \frac{1}{w_i'}$ amount of data is suppressed. We can show that, in other settings, the amount of suppressed data cannot exceed A. Specifically, suppose nodes $j_1, j_2, \cdots, j_n$ in T_i can suppress data. Then the amount of suppressed data is at most $A' = \sum_{j\in T_i}\lambda_j - \sum_{k=1}^{n}\frac{1}{w_{j_k}}$, according to Eq. (10). Also note that, for each $k = 1, \cdots, n$, $w_{j_k} \le w'_{j_k} \le w_i'$; hence, $\sum_{k=1}^{n}\frac{1}{w'_{j_k}} \ge \frac{1}{w_i'}$. Therefore, $A' \le A$. ■

D. Maximum Total Aggregation and Transmission Delay Allowed for Each Source-Sink Path

Suppose the application-specific data delivery requirement is that at least p percent of data should be delivered to the sink within time D after being generated. The maximum total aggregation and transmission delay that all the nodes on a source-sink path are allowed to introduce is estimated as follows. Let

$$i_0 \to i_1 \to \cdots \to i_n \to \text{sink} \tag{9}$$

be a path from leaf node i_0 to the sink, and d_j be the transmission delay at node i_j. Then, the maximum total aggregation and transmission delay allowed for all nodes on the path is

$$D_p = -\frac{D - \sum_{j=0}^{n} d_j}{\ln(1 - \sqrt[n]{p})}. \tag{10}$$

The rationale is explained by the following Lemma 4.2 and its proof.

Lemma 4.2: If the maximum delay allowed for a source-sink path defined in Eq. (9) is as Eq. (10) and the transmission delay at node i_j is d_j, then at least p percent of the data can arrive at the sink within time D after being generated.

Proof: For any $j = 1, \cdots, n$, let us define

$$D_j = (D - \sum_{k=0}^{n} d_k)\frac{w_{i_j}}{w'_{i_0} + \sum_{k=1}^{n} w_{i_k}} + d_j. \tag{11}$$

As the forwarding-aggregation delay interval W_{i_j} at node i_j follows the exponential distribution of mean w_{i_j}, the percentage of packets experiencing delay no more than D_j at node i_j is

$$1 - e^{-\frac{D_j - d_j}{w_{i_j}}} = 1 - e^{-\frac{D - \sum_{k=0}^{n} d_k}{w'_{i_0} + \sum_{k=1}^{n} w_{i_k}}}. \tag{12}$$

Let

$$D_0 = (D - \sum_{k=0}^{n} d_k)\frac{w'_{i_0}}{w'_{i_0} + \sum_{k=1}^{n} w_{i_k}} + d_0. \tag{13}$$

Similarly, we can get the percentage of packets generated by node i_0 that experiences delay no more than D_0 at node i_0 is

$$1 - e^{-\frac{D_0 - d_0}{w'_{i_0}}} = 1 - e^{-\frac{D - \sum_{k=0}^{n} d_k}{w'_{i_0} + \sum_{k=1}^{n} w_{i_k}}}. \tag{14}$$

Hence, the percentage of packets generated by source i_0 and experiencing delay no more than $\sum_{j=0}^{n} D_j = D$ during their trips to the sink is

$$(1 - e^{-\frac{D_0 - d_0}{w_{i_0}}})\prod_{j=1}^{n}(1 - e^{-\frac{D_j - d_j}{w_{i_j}}}) = (1 - e^{-\frac{D - \sum_{j=0}^{n} d_j}{w'_{i_0} + \sum_{j=1}^{n} w_{i_j}}})^n, \tag{15}$$

according to Eq. (11) and (13).

Furthermore,

$$w'_{i_0} + \sum_{j=1}^{n} w_{i_j} \le D_p = -\frac{D - \sum_{j=0}^{n} d_j}{\ln(1 - \sqrt[n]{p})}. \tag{16}$$

Combining Eq. (16) into Eq. (15), we have

$$(1 - e^{-\frac{D - \sum_{j=0}^{n} d_j}{w'_{i_0} + \sum_{j=1}^{n} w_{i_j}}})^n > (1 - e^{\ln(1 - \sqrt[n]{p})})^n = p. \tag{17}$$

■

E. Performance Upperbound

In this section, we develop an algorithm (formally presented in Algorithm 1) to compute the performance upperbound, based on the nodal lifetime model built in Sections IV-A and IV-B, the output data rate lowerbound analyzed in Section IV-C, and the maximum allowed end-to-end delivery latency computed in Section IV-D. The algorithm adopts the binary search approach to seek the maximum nodal lifetime that can be achieved by every node through attempting all legal ways of aggregation delay distribution which does not violate the delay requirement. In the computation, we have introduced several relaxations:

- Perfect channel condition has been assumed for each link; i.e., for each node i, ETX_i is always set to 1.
- When computing the output data rate for a node i using Eq. (4), its data input rate is assumed to be the sum of the lowerbounds of its children subtrees' output rates, where the result presented in Lemma 4.1 is applied. Hence, the computed output data rate for node i is also a lowerbound.
- When computing the nodal lifetime of a node i using Eq. (1), its data input rate is also assumed to the sum of the lowerbounds of its children subtrees' output rates.

Algorithm 1 Computation of the Performance Upperbound

Input: $\{e_i\}$, $\{\lambda_i\}$, D_p, T
Output: network lifetime upperbound $\hat{L}$

$low \leftarrow \epsilon, up \leftarrow \infty$
/* low/up is the lower/upper bound of the network lifetime */
$target \leftarrow low$
/* target is the maximum achievable network lifetime */
calculate D_p according to Lemma 4.2
while $up - low > \epsilon$ **do**
  $w_{sink} \leftarrow 0, w'_{sink} \leftarrow D_p$
  $reacheable \leftarrow false$
  for each node i in the pre-order traversal of T **do**
    for w_i from 0 to $D_p - \sum_{k\in s_i} w_k$ with step σ **do**
      calculate $\hat{\lambda}_i'$ according to Eq. (4) and (5) with w_i
      if $\frac{e_i}{((\frac{T_r}{2}+\tau)\hat{\lambda}_i' + \frac{\phi}{2T_r} + \tau\sum_{j\in C_i}\lambda_j')P} \ge target$ **then**
        $reacheable \leftarrow true$
        break
  if $reacheable = false$ **then**
    $up \leftarrow target$
    $target \leftarrow \frac{low+up}{2}$
  else
    $low \leftarrow target$
    $target \leftarrow (up = \infty)?2 * low : \frac{low+up}{2}$
return low

V. Detailed Design

The scheme operates in two phases: initial phase and adaption phase.

A. Initial Phase

After the collection tree has been built (i.e., the routing process has completed), each node i needs to compute the initial values of w_i and w_i'. For this purpose, our scheme requires each on-tree node i to know (i) the maximum allowed aggregation and transmission delay D_p, (ii) the average transmission delay d_i for transmitting a data packet from itself to its parent, and (iii) how many hops (H_i) it is away from its furthest descendant. Knowing these information but unaware of either the data rates or energy levels of individual nodes, our scheme intends to distribute the maximum allowed aggregation delay (i.e., D_p minus transmission delays) in a fair manner. Specifically, the initial value w_i can be determined using the following protocol:

- Denote $\bar{D}_i$ the maximal allowed delay for the subtree rooted at node i. Hence, the sink has $\bar{D}_{sink} = D_p$ and sends it to its children.

- Upon receiving $\bar{D}_i$ from parent node i, node j acts as follows:
  - If node j is non-leaf (i.e., $H_j > 0$), it sets $w_j = \frac{\bar{D}_i}{H_j+1} - d_j$ and $w'_j = \bar{D}_i - d_j$. Then, it sets $\bar{D}_j = \bar{D}_i - \frac{\bar{D}_i}{H_j+1}$ and sends $\bar{D}_j$ to its children.
  - If j is a leaf, it sets $w_j = w'_j = \bar{D}_i - \frac{T_r}{2}$ as it does not forward any data and sets $w'_j = \bar{D}_i - \frac{T_r}{2}$.

The three pieces of information needed by each node can be obtained in the following ways.

- The sink can compute and broadcast to all on-tree nodes D_p after the tree has been built. Note that, the sink can get the average per-hop transmission delay and the length of the longest branch on the tree through querying the nodes on the tree. Using these information, together with the application-specified parameters D and p, the sink can compute D_p by using Eq. (10).
- For the routing and route maintenance purposes, it is typical that neighboring nodes are required to exchange beacons periodically to know the expected number of transmission needed for successful delivery of a packet over a link (i.e., ETX). With the knowledge of ETX, it is easy to estimate per-transmission delay. Particularly, if the underlying MAC protocol is RI-MAC with period T_r and the ETX of the link from node i to its parent is denoted as ETX_i, then the expected per-packet transmission delay for i is $T_r(\lceil \frac{ETX_i}{m} \rceil - \frac{1}{2}) + \tau(ETX_i \% m)$ as in Eq. (1), where T_r and m are parameters in RI-MAC.
- To help a non-leaf node know how many hops it is away from its furthest leaf node, a simple method is as follows. Each data packet sent from a leaf node is piggy-backed with a field H which is initiated to 0. As the packet is forwarded hop-by-hop upwards, the value in the field is incremented. By observing the H values of forwarded packets for a certain period of time, each non-leaf node i can obtain its H_i.

B. Adaption Phase

In this phase, each node interacts with its parent unless its parent is the sink; it also interacts with its children nodes unless it is a leaf node. In both types of interaction, the node and its parent or children need to adjust their forwarding-aggregation and self-aggregation delays, attempting to gradually balance their lifetime. The behaviors of a node in such interactions are different depending on whether it acts as a parent (i.e., interacting with its children) or a child (i.e., interaction with it parent). As a basic difference, a child node j needs to report to its parent its current lifetime (L_j), forwarding-aggregation delay (w_j), self-aggregation delay (w'_j), data output rate (λ'_j) and data input rate ($\sum_{k \in C_j} \lambda'_k$). To save communication overhead, these information can be piggy-backed to data packets sent to its parent. As a parent node, on the hand, needs to make decision on how to adjust the forwarding-aggregation and self-aggregation delays of its own and its children, and notify the decision to its children. More specifically, the parent and child behaviors are described as follows.

1) Behaviors of a Parent Node: Figure 4 illustrates the behaviors of node i as a parent node. When receiving a data packet from its child node j, parent node i extracts L_j, w_j, w'_j, λ'_j and $\sum_{k \in C_j} \lambda'_k$. Every a certain time interval, the parent node is triggered by a timer to start checking the lifetime difference between itself and its children, which may be followed by adjustments of the aggregation delays when necessary. Particularly, only if the lifetime difference between node i and the its shortest-lifetime child is greater than a certain threshold α, the adjustments are performed. The timer interval and the threshold α are both adjustable system parameter that should be appropriately set, not too large or too small, in order to not miss the opportunity of lifetime balance or introduce unnecessary trashes.

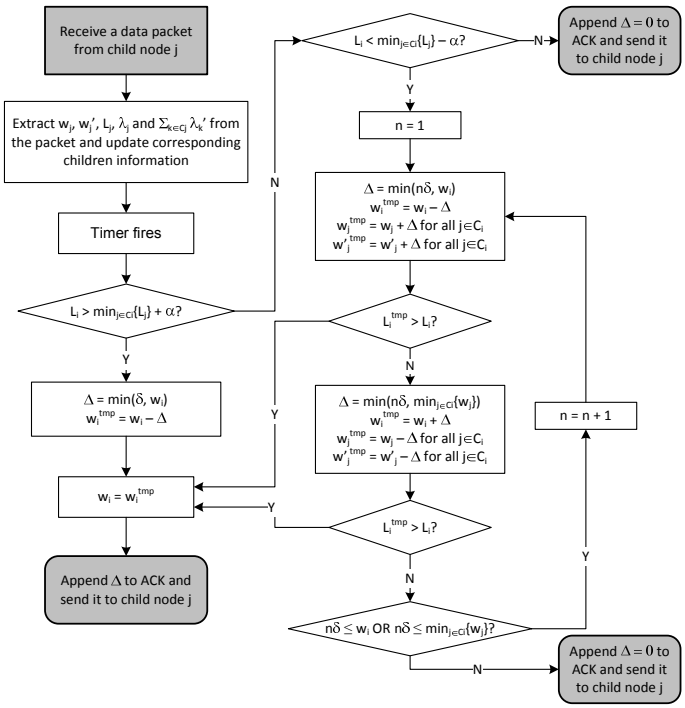


Fig. 4. Flowchart of adjusting w_i when node i acts as a parent node.

When it is necessary to adjust aggregation delays, there are two cases as follows.

- *Case I*: Node i has longer lifetime than its shortest-lifetime child (i.e., $L_i > \min_{j \in C_i} L_j + \alpha$). In this case, the scheme will attempt to increase the lifetime of the shortest-lifetime child. Letting $\Delta = \min(\delta, w_i)$, where δ is a small value, the following adjustments are applied.
  - For node i: $w_i = w_i - \Delta$ and w'_i remains unchanged.
  - For every child node j of node i: $w_j = w_j + \Delta$ and $w'_j = w'_j + \Delta$.

  The adjustment strategy is based on the following observations. From Equation(4), we can see that λ'_j is a non-increasing function of w_j and w'_j. Particularly, λ'_j remains unchanged if $w'_j \le \frac{1}{\lambda_j}$ and $w_j \le \frac{1}{\sum_{k \in C_j} \lambda'_k}$; it increases otherwise. If λ'_j decreases, as node j's input rate does not change, node j's lifetime will be increased according to Equation (1). If λ'_j does not change, w_j and w'_j will be further increased in the follow-up adjustment rounds. Hence, at least one of the conditions of $w'_j \le \frac{1}{\lambda_j}$ and $w_j \le \frac{1}{\sum_{k \in C_j} \lambda'_k}$ will be eventually broken, and then node j's lifetime can be extended.
- *Case II*: Node i has shorter lifetime than each child (i.e., $L_i < \min_{j \in C_i} L_j - \alpha$). In this case, the scheme will attempt to increase the lifetime of node i.

  According to Equation (1), node i's lifetime is affected by two factors: its data output rate λ'_i and data input rate $\sum_{j \in C_i} \lambda'_j$. To extend lifetime, node i may increase its w_i to reduce its data output rate; on the other hand, its children $j \in C_i$ will have to reduce their w_j values to satisfy the delay requirement. resulting in an increased input rate $\sum_{j \in C_i} \lambda'_j$. The combination of the above two effects may

or may not lead to lifetime increase. Similarly, decreasing w_i and increasing w_j for each child i will decrease its data input rate but may increase its data output rate, which may or may not lead to lifetime increase.

Due to the above uncertainty, the scheme will not immediately increase (or decrease) its w_i or ask its children to decrease (or increase) their aggregation delays as in the previous case. Instead, it tries the possible increase/decrease strategies, estimate their effects on its lifetime locally, until an effective strategy has been found or all possible strategies have been tried. If an effective strategy is found, i.e., a Δ (which could be positive or negation) is found such that adding Δ to w_i and subtracting it from w_j of every child j will result in a lifetime increase in node i, the strategy will be applied on node i and Δ will be sent to the children nodes.

2) Behaviors of a Child Node: As shown in Figure 5, a child node simply updates its w_i and w_i' according to its parent node's instruction.

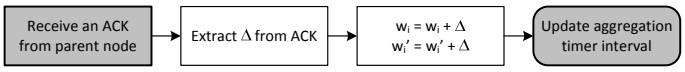


Fig. 5. Flowchart of adjusting w_i when node i acts as a child node.

VI. Implementation and Evaluation

A. Implementation

Our proposed LBA scheme has been implemented in TinyOS 2.1.0 as a middleware component, which takes 248 bytes of RAM and 9180 bytes of ROM for a tree network where each non-leaf node has 10 children nodes on average. This component sits between the application and routing layers. If it is disabled, data from the application layer is sent down to routing layer transparently; otherwise, self-generated data from application layer and received sensory data from routing layer are aggregated inside it, and the aggregated data will be sent to the routing component after the aggregation timer fires. CTP [16] is adopted as the routing layer protocol to build the data collection tree and RI-MAC [17] is used as the MAC layer protocol. RI-MAC is a receiver-initiated protocol for low duty cycled sensor networks, its energy consumption model matches with our lifetime estimation model analyzed in Section IV-A.

In the implementation, there are three types of existing messages required by different components and we take advantage of the availability of these messages to enable the aggregation information exchange with the minimum overhead. In each data message, the aggregation information including nodal lifetime, self-data generation interval, aggregation interval, incoming and outgoing intervals are piggybacked, and a parent node can collect these information of its child node when data communication happens between them. In each ACK message (software ACK used in RI-MAC), the updated aggregation interval value is added to notify the child node. In a receiver's beacon (required by RI-MAC), the parent node's lifetime is added.

B. Testbed Experiment Results

1) Setup: We evaluate the performance of the proposed adaptive assignment scheme (denoted as LBA in figures) in terms of network lifetime and end-to-end data delivery delay. The results are compared to the theoretical upperbound which is computed using Algorithm 1 (denoted as UPPER in figures) and the following two aggregation schemes:

- PTL (pushing aggregation delays to leaves): aggregation delays are only assigned to leaf nodes; the aggregation delays do not change after assignment. The delay assignment is implemented as follows. After tree has been built, the sink broadcasts to all nodes on the tree the maximum allowed aggregation and transmission delay D_p; then, each leaf node i takes $D_p - d_i$ (recalling d_i is its average transmission delay) as its aggregation delay.
- AVG (average assignment of aggregation delays): aggregation delays are assigned to nodes as in the *initial phase* of our proposed scheme; but the aggregation delays do not change after assignment.

We set up a testbed network of 32 TelosB motes, forming a tree topology shown in Figure 6, where Node 0 is the sink (i.e., root). RI-MAC parameter T_r and the average data packet generation interval at source nodes are both set to 1 second, RI-MAC parameters ϕ and τ are 40 milliseconds. The end-to-end delivery requirement D changes from 20s to 140s, and $p = 80\%$. At the beginning of each experiment, the initial nodal energy level is uniform or non-uniform. When the initial energy level is uniform, each sensor node's battery is in the full energy level; when it is non-uniform, some nodes start with 1/4, 1/2 or 3/4 of the full energy level, while others still start with the full energy level. To save experiment time, the full energy level is set to 200 Joules which can support a mote running for 45 minutes at 100% radio duty cycle.

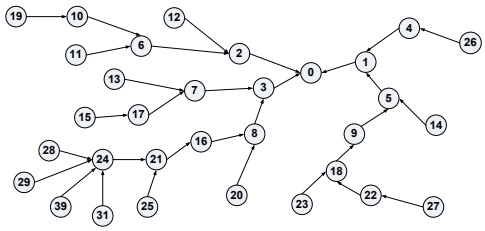


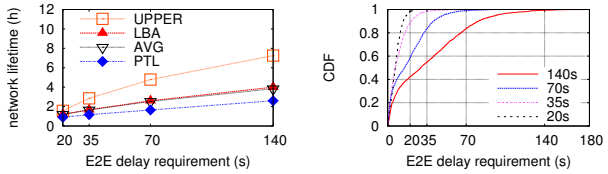
Fig. 6. Network topology in testbed experiments.

2) Performance Under Uniform Initial Energy Distribution: Figures 7 and 8 show the performance evaluation results when the initial energy levels of nodes are uniform.

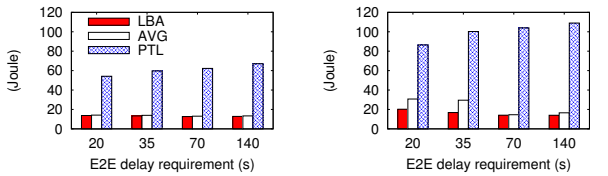
a) When all nodes are data sources: As shown in Figure 7(a), the network lifetime achieved by LBA and AVG is significantly higher than that of PTL. This is not surprising because PTL does not allow nodes to aggregate data they forward while LBA and AVG allow nodes to aggregate data generated or forwarded by them. Due to the uniform distribution of nodal energy level, LBA and AVG do not have significant different performance. Indeed, as shown in Figures 7(c) and 7(d), both schemes result in small deviation in nodal lifetime among all nodes, which indicates that the strategy of fairly and statically assigning aggregation delay adopted by AVG can already achieve good performance and therefore LBA does not bring much extra benefit. Figure 7(b) shows the CDF of end-to-end delay when LBA is applied. As can be seen, the end-to-end delay requirements can be met.

b) When only leaf nodes are data sources: As shown in Figure 8(a), the network lifetime achieved by LBA and AVG is still significantly higher than that of PTL. The results show that self-aggregation does not fully exploit the aggregation opportunities. Data packets resulted from self-aggregations can be further aggregated at joint points of multiple branches on the tree, and such opportunities can be seized by LBA and AVG. Also due to the uniform distribution of initial nodal energy level, the performance superiority of LBA over AVG is still small. However, as we can see from Figure 8(c) and 8(d), LBA

can balance nodal energy levels more effectively than AVG, which explains the longer lifetime achieved by LBA.

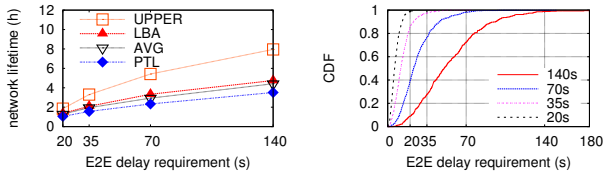


(a) Network lifetime. (b) E2E delay with LBA.

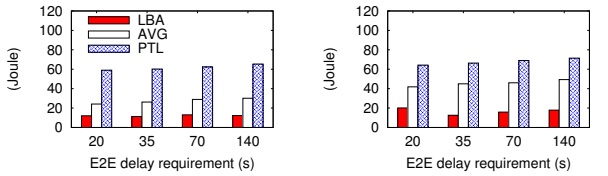


(c) Residual energy standard deviation. (d) Residual energy average.

Fig. 7. Performance comparison when all nodes generate data packets. [Note: (c) and (d) plot the standard deviation and average of nodal residual energy when the first node dies.]



(a) Network lifetime. (b) E2E delay with LBA.



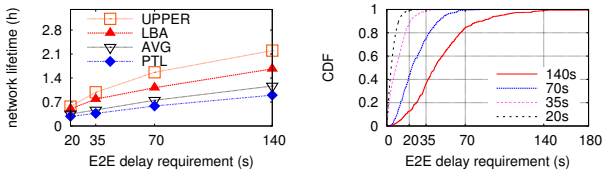
(c) Residual energy standard deviation. (d) Residual energy average.

Fig. 8. Performance comparison when only leaf nodes generate data packets. [Note: (c) and (d) plot the standard deviation and average of nodal residual energy when the first node dies.]

3) Performance Under Non-uniform Initial Energy Distribution: The network lifetime is affected by the degree of deviation in nodal lifetime, which is in turn affected by the degree of deviation in nodal residual energy level. Hence, we also evaluate the performance of aggregation schemes when the initial nodal energy levels are different among nodes. Particularly, in the experiments, we randomly chose a certain percentage of nodes to start working at a lower energy level than others who are with full initial energy level.

a) Performance comparison: Figure 9 demonstrates that LBA can achieve larger network lifetime improvement over both AVG and PTL in this scenario.

As the performance superiority of AVG over PTL has been obvious from previous experiments, we here focus on explaining the difference between LBA and AVG. With LBA and AVG, nodes start with the same distribution of aggregation delays, and the distribution of delays is unaware of the difference in nodal residual energy level or nodal lifetime. With AVG, the distribution of delays does not change after the initial distribution. Hence, if significantly differences of nodal residual energy (and hence lifetime) are present among the nodes, nodes with the lowest nodal lifetime will die much earlier than those with longer nodal lifetime, leading to shortened network lifetime. On the other hand, LBA can dynamically adapt the distribution of aggregation delays among nodes such that nodes with lower nodal lifetime can gain more aggregation delays from those of higher nodal lifetime, leading to balanced distribution of nodal lifetime among nodes and hence prolonged network lifetime.



(a) Network lifetime. (b) E2E delay with LBA.

Fig. 9. Performance comparison under non-uniform initial nodal energy. [Two nodes start with a low energy level equal to 1/4 of the full energy level while others start with the full energy level.]

b) A working trace: To better understand how LBA prolongs the lifetime of low energy (and hence short lifetime) nodes in the network, Figure 10 shows the changing trace of aggregation delays and nodal energy of all nodes on the path from node 28 to the sink. We can find that when node 8 starts working with only 1/4 of the full energy, its own aggregation delay keeps increasing while its ancestor's (node 3) and successors' (node 16) keep dropping accordingly. When node 16's aggregation delay reaches 0, it gains aggregation delay from its subtree (nodes 21, 24 and 28) to compensate node 8. In other words, node 8 can get help not only from its direct parent or child directly, but also from other nodes in the network indirectly. As a result, node 8's energy drops slowly compared to all other nodes on the path, and the network lifetime is significantly extended.

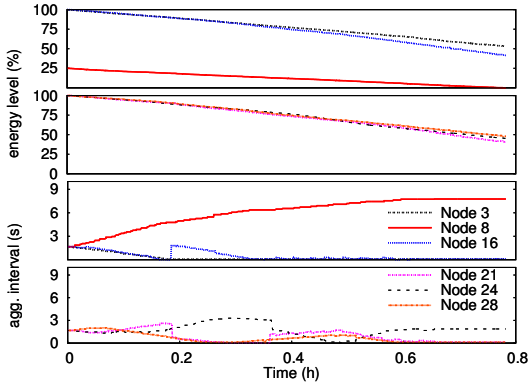


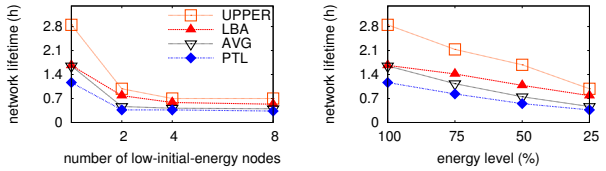
Fig. 10. Trace of aggregation delays and nodal energy [Node 8 starts with a low energy level equals to 1/4 of the full energy level while others start with the full energy level. $D = 35$ s, $p = 80\%$.]

c) Impact of the degree of deviation in nodal energy levels: To evaluate the impact of the different degrees of deviation in nodal energy levels, we conduct two sets of experiments.

In the first set, we fix the initial energy level of each low-initial-energy node to 1/4 of the full energy while varying the number of such low-initial-energy nodes in the 32-node network same as the above. As shown in Figure 11(a), as the number

of low-initial-energy nodes increases, the performance of AVG and PTL does not change much because the lifetime they can achieve can be affected by even a single such low-initial-energy node. The performance of LBA, however, decreases as the number of low-initial-energy nodes increases. This is because, with limited amount of total aggregation delays, less compensation can be obtained by each low-initial-energy node as the number of such nodes increases.

In the second set of experiments, we fix the number of low-initial-energy nodes while varying the initial energy level from 75% to 25% of the full energy. As can be observed from Figure 11(b), as the energy level decreases, the performance of AVG and PTL drops much faster than that of LBA. This is because the network lifetime achieved by AVG and PTL is bounded by the shortest-nodal-lifetime node; but with LBA, the nodal lifetime of such nodes can be increased through re-distribution of aggregation delays. Though the decrease of initial energy level demands more re-distribution efforts and thus can decrease the network lifetime, the decrease is shared among the nodes and thus the decreasing speed is slow.



(a) Network lifetime under different number of low-initial-energy nodes.

(b) Network lifetime under different low-initial-energy levels with two fixed low-initial-energy nodes.

Fig. 11. Impact of the degree of deviation in nodal energy levels. [$D = 35$ s, $p = 80\%$].

4) Performance Under Spatial and Temporal Various Data Generation Rates: We evaluate the performance under a more realistic situation where the data generation rates vary during the network operational time. Specifically, in this experiment, each node changes its data generation interval randomly in a specified range after generating every 100 packets. Figure 12 shows the lifetime achieved by three compared schemes as the data generation interval varies in different ranges, where a "1-Xs" label on the X-axis means the range is from 1 to X seconds. As we can see, the performance of LBA is always significantly higher than that of AVG and PTL when the data generation interval range changes. This indicates that LBA can work adaptively to the changes in data generation rates.

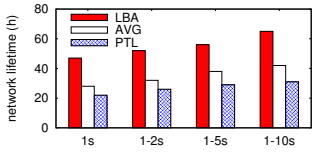


Fig. 12. Performance comparison under various data generation rates. All nodes are data sources and the end-to-end delay requirement D is 35s with $p = 80\%$. Two nodes in the network start working with 1/4 of the full energy while others start with full energy.

VII. Conclusions and Future Work

This paper introduced a lifetime balanced data aggregation scheme LBA for asynchronous duty cycle sensor networks. Through adaptively adjusting the aggregation holding time between neighboring nodes, LBA can effectively improve the nodal lifetime of nodes of lower energy supplies and/or higher energy consumption and thus prolong the network lifetime, which has been verified by extensive experimental evaluations based on a sensor network testbed. Some issues are left open for future research. For example, when the topology of data collection tree is highly dynamic, how to deliver high performance with low overhead is a practical and interesting problem that needs further investigation. Besides, how to combine LBA with energy-aware routing and MAC layer protocols and do the cross-layer optimization for network lifetime prolonging is another direction of future work.

Acknowledgement

This work is supported partly by the NSF under Grant CNS-0831874 and ECCS-1128312.

References

[1] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: An acquisitional query processing system for sensor networks," *ACM Trans. Database Syst*, 2005.

[2] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran, "Dfuse: a framework for distributed data fusion," in *SenSys*, 2003.

[3] L. Becchetti, A. Marchetti-Spaccamela, A. Vitaletti, P. Korteweg, M. Skutella, and L. Stougie, "Latency-constrained aggregation in sensor networks," *ACM Trans. Algorithms*, 2009.

[4] W. Wang, V. Srinivasan, and K. C. Chua, "Using mobile relays to prolong the lifetime of wireless sensor networks," in *MobiCom*, 2005.

[5] J. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *INFOCOM*, 2000.

[6] ——, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Trans. Netw.*, 2004.

[7] K. Fan, S. Liu, and P. Sinha, "Structure-free data aggregation in sensor networks," in *INFOCOM*, 2006.

[8] ——, "Scalable data aggregation for dynamic events in sensor networks," in *SenSys*, 2006.

[9] T. He, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "AIDA: Adaptive application independent data aggregation in wireless sensor networks," in *ACM Trans. Embed. Comput. Syst.*, 2004.

[10] Z. Ye, A. Abouzeid, and J. Ai, "Optimal policies for distributed data aggregation in wireless sensor networks," in *INFOCOM*, 2007.

[11] I. Solis and K. Obraczka, "The impact of timing in data aggregation for sensor networks," in *ICC*, 2004.

[12] Q. Xiang, J. Xu, X. Liu, H. Zhang, and L. Rittle, "When in-network processing meets time: Complexity and effects of joint optimization in wireless sensor networks," in *RTSS*, 2009.

[13] S. Hariharan and N. Shroff, "Maximizing aggregated revenue in sensor networks under deadline constraints," in *CDC*, 2009.

[14] Y. Wu, S. Fahmy, and N. B. Shroff, "On the construction of a maximum-lifetime data gathering tree in sensor networks: Np-completeness and approximation algorithm," in *INFOCOM*, 2008.

[15] C. Hua and T.-S. P. Yum, "Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 892 –903, 2008.

[16] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *SenSys*, 2009.

[17] Y. Sun, O. Gurewitz and D. Johnson, "RI-MAC: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks," in *SenSys*, 2008.

[18] S. F. Madden, M. J. Hellerstein, and W. J. M. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," in *OPERATING SYSTEMS REVIEW*, 2002, VOL 36.

[19] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble mac protocol for duty-cycled wireless sensor networks," in *SenSys*, 2006.

[20] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *SenSys*, 2010.