Computers and Electrical Engineering 58 (2017) 244–264

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Lifetime balanced data aggregation for the internet of things

Zi Li[a], Wensheng Zhang[a], Daji Qiao[a], Yang Peng[b,*]

[a] Iowa State University, Ames, Iowa 50011, USA
[b] University of Washington Bothell, Bothell, Washington 98011, USA

ARTICLE INFO

Article history:
Received 1 May 2016
Revised 19 September 2016
Accepted 20 September 2016
Available online 15 October 2016

Keywords:
Internet of things
Data aggregation
Lifetime-balancing
End-to-end delay requirement

ABSTRACT

This paper proposes LBA, a lifetime balanced data aggregation scheme for the Internet of Things (IoT) under an application-specified end-to-end delay requirement. In contrast to existing aggregation schemes, LBA aims to prolong the IoT network lifetime under network heterogeneity and dynamics, while ensuring the required data delivery delay. To achieve this goal in a distributed manner, LBA adaptively adjusts the aggregation delays of neighboring devices to balance the lifetime between them. As such balancing takes place in all neighborhoods, the minimal device lifetime in the network is increased gradually, thus prolonging the lifetime of the entire network. The effectiveness of LBA is demonstrated via extensive experiments on a testbed. Generally, when the network presents a higher degree of heterogeneity and dynamics, LBA's performance gain over a state-of-the-art non-adaptive data aggregation scheme becomes more significant, and the gap between LBA's performance and its theoretical upper bound gets smaller.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

In the Internet of Things (IoT), tiny IoT devices are usually powered by small batteries with limited energy supplies. When deploying these devices for long-term applications such as continuous environmental monitoring, it is critical to prolong the IoT network lifetime.

Data aggregation [1,2] has been widely recognized and applied as an effective technique to reduce the communication cost and extend the operational lifetime of devices. With data aggregation, a device is allowed to hold data received from neighboring devices or data generated by itself for a period of time (referred to as *aggregation delay*), aggregate them together, and send out the aggregated result. This way, redundancy is eliminated from raw data and the communication cost is reduced. In this paper, we present a unique data aggregation scheme, called LBA (Lifetime Balanced Aggregation), to prolong the IoT network lifetime while satisfying the application-specified end-to-end data delivery delay requirement.

1.1. Motivation and contributions

In a data aggregation scheme, the extent to which data volume can be suppressed depends highly on the aggregation delay. With a larger aggregation delay, more data may be suppressed and less energy may be consumed on data transmissions; hence, the communication efficiency may be increased. However, more delay would be added to the end-to-end data delivery. Therefore, there is a tradeoff between communication efficiency and end-to-end delay. As the first contribution of

* Corresponding author.
E-mail addresses: zili@iastate.edu (Z. Li), wzhang@iastate.edu (W. Zhang), daji@iastate.edu (D. Qiao), yangpeng@uw.edu (Y. Peng).

http://dx.doi.org/10.1016/j.compeleceng.2016.09.025
0045-7906/© 2016 Elsevier Ltd. All rights reserved.

Table 1
Comparison with selected existing schemes.

Schemes	Bounded Delay	Asynchronous	Distributed	Handle Network Dynamics
[12,13]	No	Yes	Yes	No
[14]	Yes	No	No	No
[15]	Yes	No	Yes	No
[16]	Yes	No	Yes	Yes
[17]	Yes	Yes	Yes	No
LBA	Yes	Yes	Yes	Yes

the paper, we study the generic (ρ, Θ) *end-to-end delay requirement*, which requires that each sensory data shall reach the sink within Θ time after its generation with a probability of at least ρ. Here, ρ and Θ are application-specified parameters. We conduct a comprehensive study on the relation between data traffic, aggregation delay, (ρ, Θ) end-to-end delay requirement, and device lifetime, which provides a theoretical foundation to the proposed LBA scheme.

In practice, devices may have different lifetime due to various reasons. For example, the following devices may have a shorter lifetime than their peers: (i) devices with a battery of poorer quality; (ii) bottleneck devices on a data collection tree; (iii) solar-powered devices [3,4] but deployed at shady locales; or (iv) wireless-chargeable devices [5–8] but deployed far away from the charger. Moreover, the IoT network conditions are heterogeneous and dynamic by nature. Data traffic, link quality, and network topology often vary over time. Thus, device lifetimes may change over time, and in general, it is difficult to identify or predict which devices have a shorter lifetime. As energy depletion in a single device may cause network disconnection or create coverage holes, which could render the entire network nonfunctional, many IoT network applications [9–11] have defined the *network lifetime* as the minimal device lifetime among all devices in the network, which is also the focus of our study in this paper. Based on the above observations, the second contribution of the paper is the design and implementation of the LBA scheme to prolong the IoT network lifetime under the end-to-end delay constraint.

It has the following features:

- LBA adjusts the aggregation delays of neighboring devices (along the source-to-sink path) together in a collaborative manner, so that the lifetime between them may be balanced without increasing the end-to-end delay. This way, the end-to-end delay requirement is satisfied while, as neighboring devices keep balancing their lifetime, the minimal device lifetime in the network is increased and thus the IoT network lifetime is extended.
- LBA is a distributed and scalable scheme as the adjustment of aggregation delays only occurs locally between neighbors.
- LBA is able to deal with network heterogeneity and dynamics in a timely manner, as the adjustment of aggregation delays is triggered as soon as the lifetime between neighboring devices becomes unbalanced due to network changes.
- LBA is able to adjust aggregation behaviors dynamically to the network changes such as route switching, packet loss, etc., and, therefore, is applicable in practical scenarios.
- LBA is a generic data aggregation scheme, which is applicable to heterogeneous networks that experience performance tradeoff between system lifetime and end-to-end delay. These networks include both IoT and traditional sensor networks.

We have implemented and evaluated LBA on a testbed of 32 TelosB nodes, and demonstrated its effectiveness with extensive experiment results under various network configurations.

1.2. Comparison with existing schemes

Many data aggregation schemes have been proposed in the past, which we will discuss in detail in Section 7. While most of these schemes focus on traffic suppression and communication efficiency, a few of them consider the tradeoff between communication efficiency and end-to-end delivery delay. In Table 1, we compare LBA with these schemes in terms of whether an end-to-end delay bound can be guaranteed, whether time synchronization is required between devices, whether a scheme is a distributed and localized solution, and whether a scheme is able to handle network heterogeneity and dynamics.

Among these schemes, [12] simply degrades the packet utility as the delivery delay increases while [13] conducts data aggregation on the shortest path tree to reduce the delivery delay; but, none of them is able to provide a delay bound guarantee on the end-to-end data delivery. Hariharan and Shro [14], Solis and Obraczka [15], Xiang et al. [16] minimize the network-wide energy consumption based, however, on an important assumption that time is synchronized between devices. Becchetti et al. [17] proposes an asynchronous and distributed data aggregation scheme. It allocates the end-to-end delay bound equally at each device along the source-to-sink path. Unfortunately, it cannot deal with network heterogeneity or dynamics, which are critical and practical factors that affect the network lifetime.

In comparison, the proposed LBA scheme is a distributed solution without requiring time synchronization between devices. It extends the network lifetime under the end-to-end delay constraint, even under various network dynamics such as packet losses, channel contention, and routing changes.

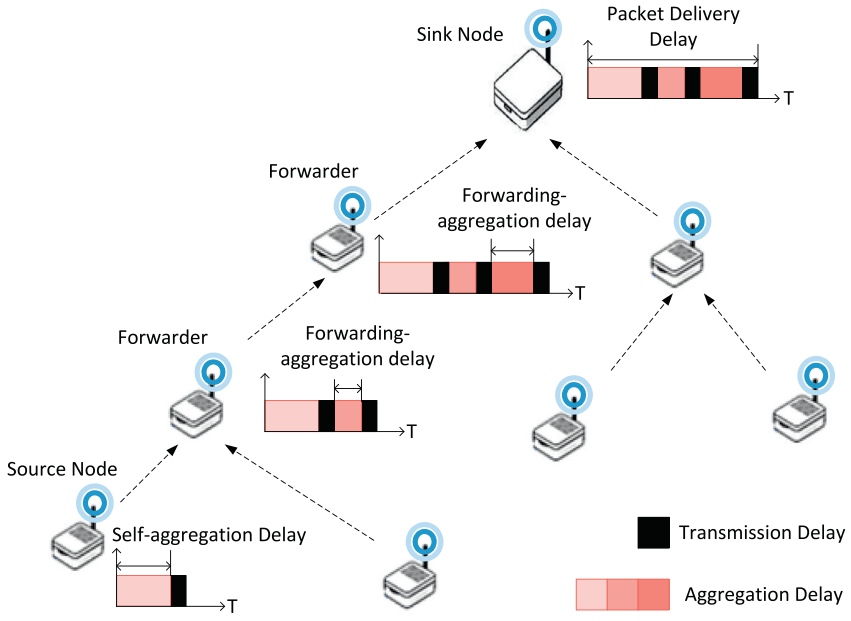


Fig. 1. System model.

1.3. Organization

The rest of the paper is organized as follows. Section 2 describes the system model and overviews the proposed LBA scheme. Sections 3–5 present the details of analysis, design, and implementation of LBA. Section 6 reports the experiment results. Section 7 reviews the related works and Section 8 concludes the paper.

2. System model and design overview

2.1. System model

We study the data aggregation problem in the Internet of Things, where each node[1] generates and reports sensory data periodically, and all the nodes form a data collection tree rooted at the sink. To build and maintain the data collection tree, a routing protocol such as the collection tree protocol (CTP) [18] may be used. Due to the dynamic nature of the channel conditions, the network topology may vary over time. In practice, however, the network topology may not change very frequently as link qualities usually are stable over a relatively long period of time [19]. An asynchronous duty cycle MAC protocol such as RI-MAC [20] is employed. Nodes may not be time-synchronized or energy-synchronized (i.e., their energy supplies and consumption rates may be different).

A typical data aggregation process is shown in Fig. 1. In a data collection tree, a source node may not send out a data packet immediately after it is generated. Instead, the node may wait for a certain period of time (called the *self-aggregation delay* (SAD)), and aggregate all the data packets generated during the period, thus reducing the total 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 (called the *forwarding-aggregation delay* (FAD)) in order to aggregate all the data packets received during the period. Therefore, the *overall aggregation delay* (OAD) involved in delivering a data packet from a source node to the sink is composed of an SAD (at the source node) and multiple FADs (at forwarding nodes).

Moreover, we assume the *total aggregation* model [21] in this work. 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 IoT network applications [17,22,23]. For example, in monitoring applications, users often are more interested in the maximum, minimum, average, or percentile statistics of sensory data, rather than the raw data themselves.

2.2. Problem statement

From the data aggregation process described above, we can see that a longer aggregation delay (SAD or FAD) may increase the end-to-end data delivery delay, but could allow more packets to be aggregated, which may result in reduced

[1] To simplify the presentation, we refer to IoT device as “node” in the rest of this paper.

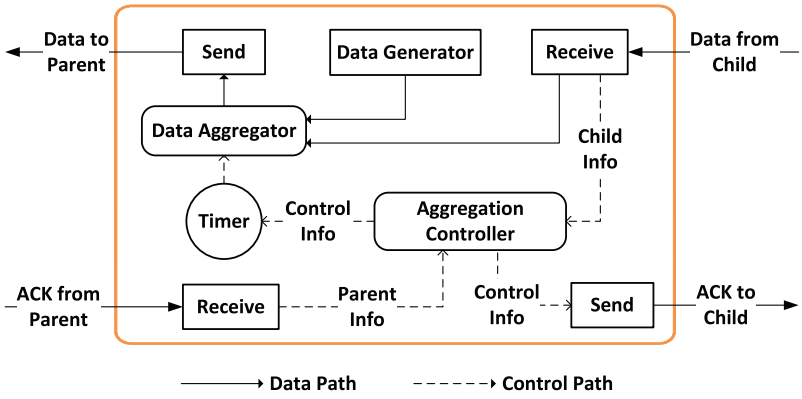


Fig. 2. Design overview of LBA.

nodal energy consumption on data transmissions and consequently a longer nodal lifetime. So, there is a tradeoff between delivery delay and nodal lifetime.

Ideally, all nodes in the network shall work together to determine the proper SAD and FAD values so that (i) the network lifetime, i.e., the minimal nodal lifetime among all nodes, is maximized; while (ii) the end-to-end delay requirement from source nodes to the sink is satisfied. Unfortunately, it is impractical to accomplish the above global optimization task in a distributed manner because it requires each node to know the network topology and the following information of every other node in the network: the residual nodal energy and the data generation rate. Acquiring this information could incur high communication overhead because of potentially large network scale and dynamic nature of the information. Moreover, it may also incur high computational overhead at each node as this problem has been proven to be NP-complete [17].

Instead, we study the following localized problem for each node i in the network:

Objective:

- $\max \min_{j \in \{i\} \cup C_i} \{L(j)\}$, where $L(j)$ is node j's nodal lifetime and its computation will be explained later in Section 3.3. C_i is the set of i's child nodes.

Subject to:

- *(ρ, Θ) end-to-end delay requirement:* With a probability of at least ρ, each sensory data shall reach the sink within Θ time after its generation, where ρ and Θ are application-specified parameters.

Output:

- i's aggregation parameters: SAD(i) and FAD(i).

The goal of this problem is to maximize the minimal nodal lifetime in i's neighborhood, which includes node i and its children node set C_i. As such procedure occurs in all neighborhoods, the minimal nodal lifetime in the entire network, i.e., the network lifetime, may be improved gradually. Note that the (ρ, Θ) end-to-end delay requirement is generic in the sense that other types of delay requirements may be obtained from it by assigning proper values to ρ and Θ. For example, by setting $\rho = 1$, the delay requirement becomes that all data shall reach the sink within a hard delay bound of Θ.

2.3. Design overview

We propose a scheme called LBA (Lifetime Balanced Aggregation) to address the problem defined above. In LBA, coordination only take place locally between parent-child nodes, which exchange information with each other and adjust their SAD and FAD parameters together in a collaborative manner. As shown in Fig. 2, when a parent node receives a data packet from its child node, it extracts the control information (e.g., the estimated nodal lifetime) embedded in the data packet and feeds them into the *Aggregation Controller* module, which decides how the parent node shall adjust its own SAD and FAD. The decision is then piggybacked into the ACK packet to the child node, based on which the child node adjusts its aggregation behavior accordingly. This way, the minimal nodal lifetime between parent-child nodes is increased while the end-to-end delay requirement is satisfied. Parent and child nodes' behaviors are elaborated in Section 4.

3. Analytical study

To provide a theoretical foundation to the proposed LBA scheme, we conduct a comprehensive study on the relation between the aggregation delays, the outgoing data rates, the nodal lifetime of each node, and the (ρ, Θ) end-to-end delay requirement. The result will be used in LBA to decide how parent-child nodes shall collaborate with each other to adjust

Table 2
List of notations used in this paper.

Notation	Meaning
SAD(i)	node i's self-aggregation delay
FAD(i)	i's forwarding-aggregation delay
CFAD(i)	cumulative forwarding-aggregation delay from i to sink
OAD(i)	overall aggregation delay from i to sink
$\lambda(i)$	i's self-data generation rate
$\mu(i)$	i's outgoing data rate
$\hat{\mu}(i)$	lower bound of i's outgoing data rate
$e(i)$	i's residual nodal energy
$L(i)$	i's estimated nodal lifetime
$h(i)$	i's hop count to sink
$d(j, i)$	one-hop communication delay from j to i
$d_{e2e}(i)$	cumulative communication delay from i to sink

the aggregation delays together (see Section 4 for details). Specifically, we first describe the nodal aggregation behavior in LBA and derive the outgoing data rate of each node. Then, we study the (ρ, Θ) end-to-end delay requirement in detail and present a sufficient condition to guarantee this requirement. Next, we present the formula for estimating the nodal lifetime. Finally, we derive an upper bound of the network lifetime achievable with LBA. Notations used in this paper are summarized in Table 2.

3.1. Nodal aggregation behavior in LBA

For each node i in the network, depending on the relation between FAD(i), SAD(i), $\sum_{j\in C_i}\mu(j)$ – the total incoming data rate from its children, and $\lambda(i)$ – its data generation rate, it behaves differently. In LBA, as FAD(i) is always smaller than SAD(i) (the proof of this statement is given in Section 3.2; refer to Inequality (12)), we have the following three types of nodal aggregation behaviors.

Type I: $\frac{1}{\sum_{j\in C_i}\mu(j)} \leqslant \text{FAD}(i) \leqslant \text{SAD}(i)$. This means that data packets arrive quickly in short intervals from the children. In this case, a timer is fired every FAD(i) interval on average (which follows an exponential distribution), upon which all data packets in the queue (including both data received from its children and self-generated data) are aggregated into a single packet (according to the *total aggregation* model discussed in Section 2) and forwarded to the parent. Therefore, the outgoing data rate in this case is $\mu(i) = \frac{1}{\text{FAD}(i)}$.

Type II: $\frac{1}{\sum_{j\in C_i}\mu(j)} > \text{FAD}(i)$ *and* $\frac{1}{\lambda(i)} \leqslant \text{SAD}(i)$. This means that data packets arrive less frequently from the children, but the node itself generates data packets quite often (more frequently than every SAD(i) time on average). In this case, the node treats data received from the children and self-generated data differently as follows:

- Whenever a data packet is received from a child node, the node immediately aggregates the packet with the self-generated data packets waiting in the queue (if any), and forwards to the parent. Hence, the forwarding-aggregation delay for the relayed data packets is zero.
- A timer is fired every SAD(i) interval on average (which follows an exponential distribution), upon which all self-generated data packets during the interval are aggregated and forwarded to the parent.

As a result, the outgoing data rate of node i in this case is $\mu(i) = \sum_{j\in C_i}\mu(j) + \frac{1}{\text{SAD}(i)}$.

Type III: $\frac{1}{\sum_{j\in C_i}\mu(j)} > \text{FAD}(i)$ *and* $\frac{1}{\lambda(i)} > \text{SAD}(i)$. This means that traffic is light with data packets slowly arriving from the children and generated by the node itself. In this case, every data packet is simply forwarded (without delay) to the parent upon its reception or generation. Therefore, the outgoing data rate is $\mu(i) = \sum_{j\in C_i}\mu(j) + \lambda(i)$.

To summarize, the outgoing data rate of node i can be derived as follows:

$$\mu(i) = \begin{cases} \dfrac{1}{\text{FAD}(i)} & : \dfrac{1}{\sum_{j\in C_i}\mu(j)} \leqslant \text{FAD}(i), \\ \displaystyle\sum_{j\in C_i}\mu(j) + \frac{1}{\text{SAD}(i)} & : \dfrac{1}{\sum_{j\in C_i}\mu(j)} > \text{FAD}(i) \\ & \quad \text{and } \dfrac{1}{\lambda(i)} \leqslant \text{SAD}(i), \\ \displaystyle\sum_{j\in C_i}\mu(j) + \lambda(i) & : \dfrac{1}{\sum_{j\in C_i}\mu(j)} > \text{FAD}(i) \\ & \quad \text{and } \dfrac{1}{\lambda(i)} > \text{SAD}(i). \end{cases} \tag{1}$$

Note that a node's incoming data rate is simply the sum of its children's outgoing data rates.

3.2. (ρ, Θ) end-to-end delay requirement

Theorem 3.1. *Consider the following path $n_0 \to n_1 \to \cdots \to n_k$, where n_0 is the source node and n_k is the sink. Let $d(n_i, n_{i+1})$ denote the one-hop communication delay from n_i to n_{i+1}, which includes all delay factors other than the aggregation delay, such as processing delay, transmission delay (and retransmissions if needed), etc. We claim that, as long as the following condition is satisfied:*

$$\mathrm{OAD}(n_0) \leqslant -\frac{\Theta - \sum_{i=0}^{k-1} d(n_i, n_{i+1})}{\ln\left(1-\rho^{1/k}\right)}, \tag{2}$$

where $\mathrm{OAD}(n_0) = \mathrm{SAD}(n_0) + \sum_{i=1}^{k-1} \mathrm{FAD}(n_i)$ is the overall aggregation delay along the path from n_0 to n_k, the (ρ, Θ) end-to-end delay requirement is satisfied for node n_0, meaning that with a probability of at least ρ, each data packet generated by n_0 would reach n_k within Θ time after its generation.

Proof. We start with the first hop: $n_0 \to n_1$. Let $D(n_0, n_1)$ denote the time for a data packet to reach n_1 from n_0. According to the analysis of the outgoing data rate in the previous section, as n_0 is the source node and does not have a child, it behaves according to either Type II or III upon generation of a data packet:

- Type II: It waits for $T(n_0)$ time before aggregating all buffered packets and forwarding to n_1, where $T(n_0)$ is a random variable that follows an exponential distribution with a mean of $\mathrm{SAD}(n_0)$, i.e., $D(n_0, n_1) = T(n_0) + d(n_0, n_1)$;
- Type III: It forwards the data packet immediately to n_1 without any delay, i.e., $D(n_0, n_1) = d(n_0, n_1)$.

Hence, we have:

$$D(n_0, n_1) \leqslant T(n_0) + d(n_0, n_1). \tag{3}$$

Therefore, the probability that a data packet reaches n_1 within any given θ_0 time from its generation at n_0 is:

$$\begin{aligned} P[D(n_0, n_1) \leqslant \theta_0] &\geqslant P[T(n_0) + d(n_0, n_1) \leqslant \theta_0] \\ &= 1 - e^{-\frac{\theta_0 - d(n_0, n_1)}{\mathrm{SAD}(n_0)}}. \end{aligned} \tag{4}$$

Similarly, whenever an intermediate node n_i receives a data packet from its child, it either (i) waits for at most $T(n_i)$ time before it aggregates all buffered packets and forwards to n_{i+1} (Type I behavior), where $T(n_i)$ is a random variable that follows an exponential distribution with a mean of $\mathrm{FAD}(n_i)$; or (ii) forwards the data packet immediately to n_{i+1} without any delay (Type II or III behavior). Hence, we have:

$$D(n_i, n_{i+1}) \leqslant T(n_i) + d(n_i, n_{i+1}), \tag{5}$$

where $D(n_i, n_{i+1})$ is the time for a data packet to reach n_{i+1} from n_i. Subsequently, the probability that the data packet reaches n_{i+1} from n_i within any given θ_i time is:

$$\begin{aligned} P[D(n_i, n_{i+1}) \leqslant \theta_i] &\geqslant P[T(n_i) + d(n_i, n_{i+1}) \leqslant \theta_i] \\ &= 1 - e^{-\frac{\theta_i - d(n_i, n_{i+1})}{\mathrm{FAD}(n_i)}}, \end{aligned} \tag{6}$$

Let's consider a specific set of θ_i values:

$$\theta_i = \begin{cases} \dfrac{\left[\Theta - \sum_{i=0}^{k-1} d(n_i, n_{i+1})\right] \cdot \mathrm{SAD}(n_0)}{\mathrm{OAD}(n_0)} + d(n_0, n_1) \\ \qquad\qquad : i = 0; \\ \dfrac{\left[\Theta - \sum_{i=0}^{k-1} d(n_i, n_{i+1})\right] \cdot \mathrm{FAD}(n_i)}{\mathrm{OAD}(n_0)} + d(n_i, n_{i+1}) \\ \qquad\qquad : 0 < i < k. \end{cases} \tag{7}$$

It is clear that $\sum_{i=0}^{k-1} \theta_i = \Theta$. Therefore, the probability that a data packet generated by n_0 reaches the sink n_k (over k hops) within Θ time can be calculated as follows:

$$\begin{aligned} P\left[\sum_{i=0}^{k-1} D(n_i, n_{i+1}) \leqslant \Theta\right] &= P\left[\sum_{i=0}^{k-1} D(n_i, n_{i+1}) \leqslant \sum_{i=0}^{k-1} \theta_i\right] \\ &\geqslant \prod_{i=0}^{k-1} P[D(n_i, n_{i+1}) \leqslant \theta_i] \geqslant \left(1 - e^{-\frac{\theta_0 - d(n_0, n_1)}{\mathrm{SAD}(n_0)}}\right) \cdot \prod_{i=1}^{k-1} \left(1 - e^{-\frac{\theta_i - d(n_i, n_{i+1})}{\mathrm{FAD}(n_i)}}\right) \\ &= \left(1 - e^{-\frac{\Theta - \sum_{i=0}^{k-1} d(n_i, n_{i+1})}{\mathrm{OAD}(n_0)}}\right)^k. \end{aligned} \tag{8}$$

Plugging in Condition (2), we have

$$P\left[\sum_{i=0}^{k-1} D(n_i, n_{i+1}) \leqslant \Theta\right] \geqslant \rho, \tag{9}$$

which means that the (ρ, Θ) end-to-end delay requirement is satisfied for node n_0. □

In practice, as larger aggregation delays would allow more traffic to be aggregated to save more energy, LBA always requires equality in Condition (2):

$$\mathrm{OAD}(n_0) = -\frac{\Theta - \sum_{i=0}^{k-1} d(n_i, n_{i+1})}{\ln\left(1-\rho^{1/k}\right)}. \tag{10}$$

With this requirement, we can prove that $\mathrm{FAD}(n_i) \leq \mathrm{SAD}(n_i)$ for all intermediate nodes n_i between source node n_0 and sink n_k. This is because, as n_0 is farther away from sink than n_i, the maximum allowed overall aggregation delay for n_0 is smaller than that for n_i, as a data packet generated by n_0 has to traverse more hops to reach the sink (hence more communication delays) while the same end-to-end delay requirement has to be satisfied. In other words, we have:

$$\mathrm{OAD}(n_0) \leqslant \mathrm{OAD}(n_i). \tag{11}$$

This inequality also can be derived rigorously from Eq. (10). Consequently, we have:

$$\begin{aligned}
\mathrm{OAD}(n_0) \;&=\; \mathrm{SAD}(n_0) + \sum_{j=1}^{k-1} \mathrm{FAD}(n_j) \leqslant \mathrm{OAD}(n_i) \\
&\Rightarrow \sum_{j=i}^{k-1} \mathrm{FAD}(n_j) \leqslant \mathrm{OAD}(n_i) \\
&\Rightarrow \mathrm{FAD}(n_i) \leqslant \mathrm{OAD}(n_i) - \sum_{j=i+1}^{k-1} \mathrm{FAD}(n_j) = \mathrm{SAD}(n_i).
\end{aligned} \tag{12}$$

Similarly, we can prove that $\mathrm{SAD}(n_0) \leq \mathrm{SAD}(n_i)$ for all intermediate nodes n_i. This is because:

$$\begin{aligned}
\mathrm{OAD}(n_0) \;&\leqslant\; \mathrm{OAD}(n_i) \\
&\Rightarrow \mathrm{SAD}(n_0) + \sum_{j=1}^{k-1} \mathrm{FAD}(n_j) \leqslant \mathrm{SAD}(n_i) + \sum_{j=i+1}^{k-1} \mathrm{FAD}(n_j) \\
&\Rightarrow \mathrm{SAD}(n_0) \leqslant \mathrm{SAD}(n_i).
\end{aligned} \tag{13}$$

Inequality (13) will be used in Section 3.4 to analyze a lower bound of the outgoing data rate.

3.3. Nodal lifetime

In the proposed LBA scheme, we assume that all nodes run RI-MAC [20], a well-known asynchronous duty cycle MAC protocol. In fact, LBA can also work with other asynchronous duty cycle MAC protocols such as X-MAC [24] and A-MAC [25], by adjusting the nodal lifetime estimation accordingly.

For completeness, the sketch of RI-MAC is shown in Fig. 3. In RI-MAC, each node wakes up for ϕ time every T_r interval to check if there is any incoming packet intended for it. After turning on the radio, a node broadcasts a beacon to announce that it is ready to receive packets. On the other hand, if a node has a packet to send (e.g., node S in Fig. 3), it turns on its radio to wait for a beacon from the intended receiver (e.g., node R in Fig. 3). Upon receiving the 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 other senders that may have pending packets intended for R can start to communicate. If there are no more incoming/outgoing packets, R turns off its radio and goes to sleep.

With RI-MAC, we can estimate the nodal lifetime $L(i)$ of node i as follows:

$$L(i) = \frac{e(i)}{\left(\left(\frac{T_r}{2} + \tau\right) \cdot \mu(i) + \frac{\phi}{T_r} + \tau \cdot \sum_{j \in C_i} \mu(j)\right) \cdot P}. \tag{14}$$

Here, P is the energy consumption rate when a node's radio is on. A node could be both a sender and a receiver. As a sender, the node waits for $\frac{T_r}{2}$ time on average for its receiver to wake up and then consumes τ time for each transmission attempt. As a receiver, the node wakes up for ϕ time every T_r interval, and spends $\tau \cdot \sum_{j \in C_i} \mu(j)$ time to receive data packets. Hence, the denominator of Eq. (14) estimates the overall energy consumption rate at node i. Note that, in the above nodal lifetime estimation, we only consider the energy consumed by the communication component of a node, which contributes to the significant majority of the overall nodal energy consumption. Details about lifetime estimation under practical scenarios like time-varying channel conditions are discussed in Section 4.4.

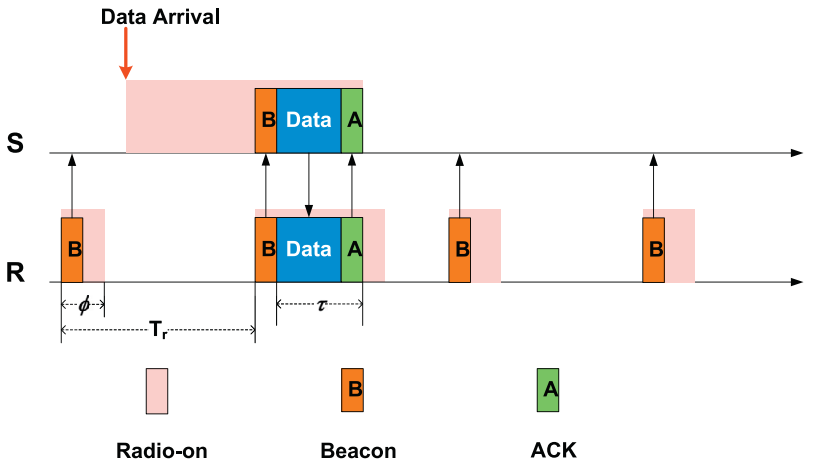


Fig. 3. The sketch of RI-MAC protocol. T_r is the receiver's beacon interval, ϕ is the receiver's channel checking period, and τ is the time needed by the sender and receiver to exchange a data packet and the corresponding ACK.

3.4. Performance upper bound

Given the nodal aggregation behaviors described in Section 3.1, and based on the analysis of the end-to-end delay requirement in Section 3.2 and the nodal lifetime in Section 3.3, we have the following theorem.

Theorem 3.2. *Let G_i be a subtree rooted at node i, and* FAD(i) *and* SAD(i) *be the forwarding-aggregation and self-aggregation delays of node i, respectively. Then, a lower bound of node i's outgoing data rate, denoted as $\hat{\mu}(i)$, can be calculated as:*

$$\hat{\mu}(i) = \begin{cases} \sum_{j\in G_i}\lambda(j) & : \mathrm{SAD}(i) < \dfrac{1}{\sum_{j\in G_i}\lambda(j)}; \\ \dfrac{1}{\mathrm{SAD}(i)} & : \mathrm{SAD}(i) \geqslant \dfrac{1}{\sum_{j\in G_i}\lambda(j)}. \end{cases} \tag{15}$$

Proof. There are two scenarios to be considered when calculating the lower bound $\hat{\mu}(i)$.

Scenario I: $\mathrm{SAD}(i) < \frac{1}{\sum_{j\in G_i}\lambda(j)}$. We prove by contradiction. Assume that node i's outgoing data rate is less than $\sum_{j\in G_i}\lambda(j)$, i.e.,

$$\mu(i) < \sum_{j\in G_i}\lambda(j). \tag{16}$$

This means that some data must have been suppressed at some descendants of node i. Suppose node k is such a suppressing node and none of its own descendants suppresses any data. Therefore, we have:

$$\sum_{j\in G_k}\lambda(j) = \lambda(k) + \sum_{j\in C_k}\mu(j). \tag{17}$$

On the other hand, according to the discussions at the end of Section 3.2, we have:

$$\mathrm{FAD}(k) \leqslant \mathrm{SAD}(k) \leqslant \mathrm{SAD}(i). \tag{18}$$

Combining Eqs. (17) and (18) with the Scenario I condition that $\mathrm{SAD}(i) < \frac{1}{\sum_{j\in G_i}\lambda(j)}$, we have:

$$\mathrm{FAD}(k) \leqslant \mathrm{SAD}(i) < \frac{1}{\sum_{j\in G_i}\lambda(j)} \leqslant \frac{1}{\sum_{j\in G_k}\lambda(j)} \leqslant \frac{1}{\sum_{j\in C_k}\mu(j)}, \tag{19}$$

and

$$\mathrm{SAD}(k) \leqslant \mathrm{SAD}(i) < \frac{1}{\sum_{j\in G_i}\lambda(j)} \leqslant \frac{1}{\sum_{j\in G_k}\lambda(j)} \leqslant \frac{1}{\lambda(k)}. \tag{20}$$

According to the nodal aggregation behaviors described in Section 3.1, Eqs. (19) and (20) suggest that node k shall follow Type III behavior, which is to simply forward every data packet immediately without any aggregation upon its generation or reception. This contradicts with the assumption that k is a suppressing node. Therefore, we have

$$\mu(i) \geqslant \sum_{j\in G_i}\lambda(j). \tag{21}$$

Scenario II: $\mathrm{SAD}(i) \geqslant \frac{1}{\sum_{j\in G_i}\lambda_j}$. If none of the nodes on G_i performs any data suppression, we have $\mu(i) = \sum_{j\in G_i}\lambda_j \geqslant \frac{1}{\mathrm{SAD}(i)}$. Else, let $k \in G_i$ denote the node that performs data suppression but none of its ancestors on G_i suppresses any data; this implies that $\mu(i) \geq \mu(k)$. According to the analysis of the outgoing data rate in Eq. (1), when node k conducts data aggregation (Type I or II behavior), its outgoing data rate $\mu(k)$ is at least $\frac{1}{\mathrm{SAD}(k)}$, since $\mu(k) = \frac{1}{\mathrm{FAD}(k)} \geqslant \frac{1}{\mathrm{SAD}(k)}$ in Type I and $\mu(k) = \sum_{j\in C_k}\mu(j) + \frac{1}{\mathrm{SAD}(k)} \geqslant \frac{1}{\mathrm{SAD}(k)}$ in Type II. On the other hand, according to the discussions at the end of Section 3.2, we have:

$$\mathrm{FAD}(k) \leqslant \mathrm{SAD}(k) \leqslant \mathrm{SAD}(i). \tag{22}$$

Therefore, we have:

$$\mu(i) \geqslant \mu(k) \geqslant \frac{1}{\mathrm{SAD}(k)} \geqslant \frac{1}{\mathrm{SAD}(i)}. \tag{23}$$

□

Based on the above lower bound, we have developed an algorithm (formally presented in Algorithm 1) to compute an

Algorithm 1 Computation of performance upper bound.

Input: $\{e(i)\}$, $\{\lambda(i)\}$, ρ, Θ, G
Output: network lifetime upper bound L^*

low $\leftarrow \epsilon$; up $\leftarrow \infty$;
target $\leftarrow$ low; // target is the max achievable L^* value
while up − low $> \epsilon$ **do**
  FAD(sink) $\leftarrow 0$;
  reachable $\leftarrow$ false;
  for each node i in the pre-order traversal of G **do**
    $\ell^*(i) = \arg\min_{\ell(i)}\{\mathrm{OAD}(\ell(i))\}$ where $\ell(i)$ is a leaf node in the subtree rooted at i;
    calculate $\mathrm{OAD}(\ell^*(i))$ according to Theorem 3.1;
    for $\mathrm{FAD}(i) = 0 : \sigma : \mathrm{OAD}(\ell^*(i)) - \sum_{k\in s(i)}\mathrm{FAD}(k)$ **do**
      /* $s(i)$ is a path from i's parent to sink */
      calculate $\hat{\mu}(i)$ according to Theorem 3.2;
      if $\frac{e(i)}{\left(\left(\frac{T_T}{2}+\tau\right)\hat{\mu}(i)+\frac{\phi}{T_r}+\tau\sum_{j\in C_i}\mu(j)\right)^P} \geqslant$ target **then**
        reachable $\leftarrow$ true;
        break;
      end if
    end for
  end for
  if reachable = false **then**
    up $\leftarrow$ target;
    target $\leftarrow \frac{\text{low+up}}{2}$;
  else
    low $\leftarrow$ target;
    target $\leftarrow$ (up $= \infty$)?2 $*$ low : $\frac{\text{low+up}}{2}$;
  end if
end while
return low;

upper bound of the network lifetime when the proposed LBA scheme is used. The algorithm adopts the binary search approach to seek the maximum nodal lifetime that can be achieved via attempting all the valid distributions of the aggregation delay that do not violate the end-to-end delay requirement. In the computation, we have introduced the following relaxations:

- When computing node i's outgoing data rate using Eq. (1), its incoming data rate is assumed to be the sum of the lower bounds of its children nodes' outgoing data rates, which are given in Theorem 3.2. Hence, the computed outgoing data rate for node i is also a lower bound.
- When computing node i's nodal lifetime using Eq. (14), its incoming data rate is assumed to the sum of the lower bounds of its children nodes' outgoing data rates.

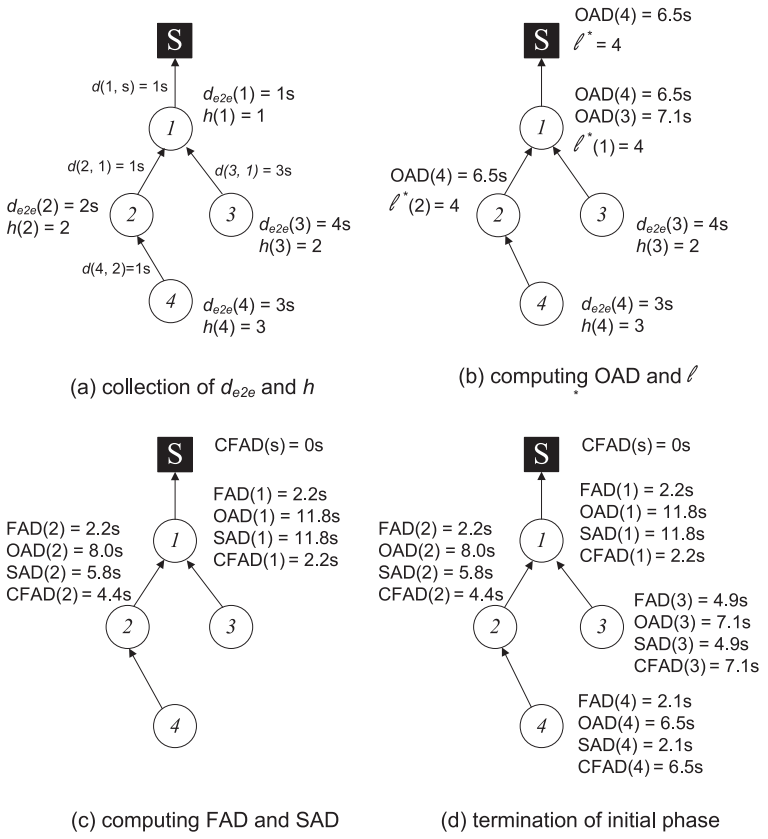


Fig. 4. An example of the initial phase of LBA with the ($\rho = 80\%, \Theta = 20\ s$) end-to-end delay requirement.

4. LBA design

In this section, we first describe the two operational phases of LBA: *initial phase* and *adaption phase*, for a given data collection tree with stable channel conditions. Then, we explain how to guarantee the end-to-end delay requirement when the collection tree topology or channel condition changes.

4.1. Initial phase

After the data collection tree has been established (e.g., with the CTP routing protocol [18]), each node needs to set its initial FAD and SAD values. For this purpose, each node measures the one-hop communication delay from itself to its parent node, and collects the following information: (i) the total communication delay (d_{e2e}) from the node to the sink; and (ii) the number of hops (h) from the node to the sink. Based on the collected information, each node computes its OAD (Overall Aggregation Delay) according to Eq. (10), which is the maximum allowed aggregation delay from the node to the sink. The information collection procedure can be easily integrated into routing protocols like CTP by embedding $d_{e2e}(j) = d(j,i) + d_{e2e}(i)$ and $h(j) = h(i) + 1$ into node j's routing beacons where i is j's parent node.

During the initial phase, LBA intends to evenly distribute the maximum allowed leaf-to-sink aggregation delay along the route. In the following, we use an example (as shown in Fig. 4) to explain how the initial phase works.

- Each leaf node reports its OAD and h values to the parent node.
- Based on the received information from its children, each node i identifies the leaf node $\ell^*(i)$ that has the minimum per-hop OAD value to the sink among all leaf nodes in the subtree rooted at i, i.e., $\ell^*(i) = \arg\min_{\ell(i)}\{\frac{\mathrm{OAD}(\ell(i))}{h(\ell(i))}\}$, and forwards $\mathrm{OAD}(\ell^*(i))$ and $h(\ell^*(i))$ to its own parent. In Fig. 4(b), node 2 only has one child (i.e., node 4) and hence it forwards OAD(4) and $h(4)$ to node 1, which calculates $\ell^*(1) = \arg\min_{\ell(1)}\{\frac{\mathrm{OAD}(4)}{h(4)}, \frac{\mathrm{OAD}(3)}{h(3)}\} = 4$, based on the information received from both children.
- The above procedure continues till the sink has identified the leaf node (denoted as ℓ^*) that has the globally minimum per-hop OAD value. In the example shown in Fig. 4, $\ell^* = 4$.

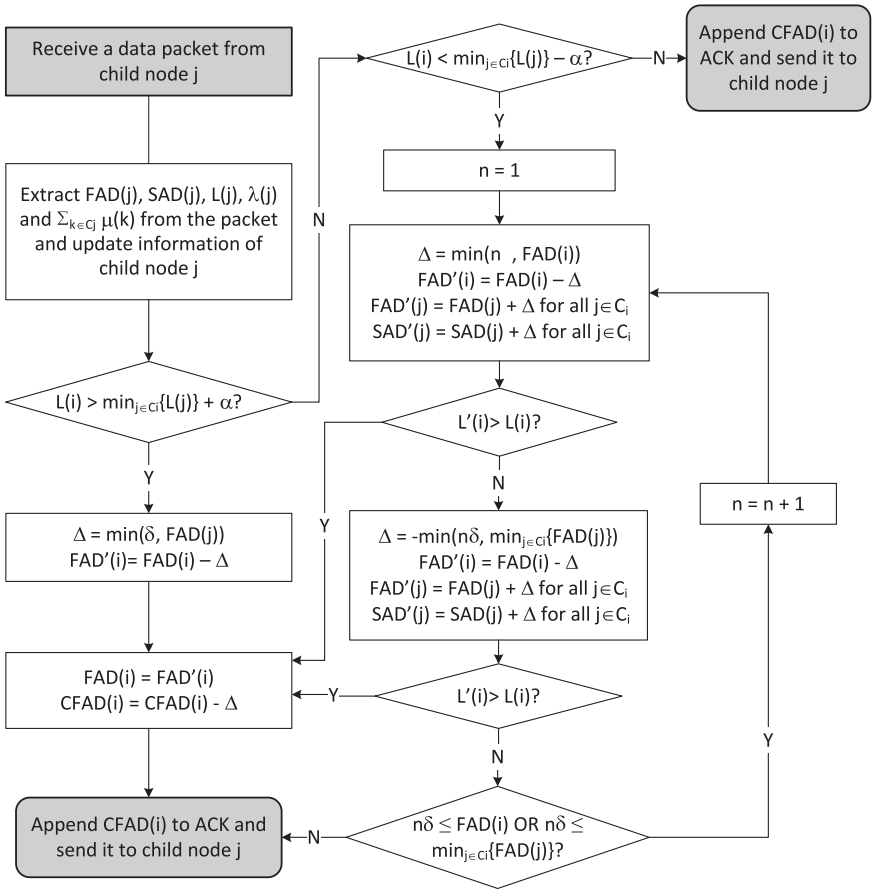


Fig. 5. Flowchart of adjusting aggregation delays when node i acts as a parent node.

Then, the sink broadcasts a CFAD (Cumulative Forwarding Aggregation Delay) value to its children, where CFAD(i) is the sum of all forwarding aggregation delays from node i to the sink. Clearly, CFAD(sink) $= 0$ s. Upon receiving CFAD(i) from its parent node i, node j acts as follows:

- If node j is a non-leaf node, it updates $\mathrm{FAD}(j) = \frac{\mathrm{OAD}(\ell^*(j)) - \mathrm{CFAD}(i)}{h(\ell^*(j)) - h(j) + 1}$ with an objective of evenly distributing the remaining OAD along the branch from j to $\ell^*(j)$. To satisfy the end-to-end delay requirement for self-generated data, node j updates $\mathrm{SAD}(j) = \mathrm{OAD}(j) - \mathrm{CFAD}(i)$. Then, j sends $\mathrm{CFAD}(j) = \mathrm{CFAD}(i) + \mathrm{FAD}(j)$ to all of its children. As shown in Fig. 4(c), node 1 sets $\mathrm{FAD}(1) = 6.5/3 \approx 2.2$ s, $\mathrm{SAD}(1) = \mathrm{OAD}(1) - \mathrm{CFAD}(\mathrm{sink}) = 11.8$ s, and sends $\mathrm{CFAD}(1) = \mathrm{CFAD}(\mathrm{sink}) + \mathrm{FAD}(1) = 2.2$ s to nodes 2 and 3.
- If j is a leaf node, it simply sets $\mathrm{FAD}(j) = \mathrm{SAD}(j) = \mathrm{OAD}(j) - \mathrm{CFAD}(i)$. As shown in Fig. 4(d), leaf node 4 sets $\mathrm{FAD}(4) = \mathrm{SAD}(4) = \mathrm{OAD}(4) - \mathrm{CFAD}(2) = 2.1$ s. It is easy to verify that the end-to-end delay requirement is satisfied for all source nodes in Fig. 4(d) while the OAD value is evenly distributed along the branch from the sink to node 4 (i.e., the ℓ^* node).

4.2. Adaption phase

During the adaption phase, each parent node and its children adjust their SAD and FAD values together in a collaborative manner to balance the nodal lifetime between them. More specifically, a child node piggybacks the following information in the data packet to its parent: *its estimated nodal lifetime, SAD, FAD, outgoing data rate*, and *incoming data rate*. Based on these information, the parent node decides how to adjust its own SAD and FAD values, and then piggybacks the updated CFAD value in the ACK to its children. Upon receiving the ACK, the child node updates its SAD and FAD values accordingly to maintain the end-to-end delay bound.

Fig. 5 illustrates the behaviors of node i as a parent node. When receiving a data packet from its child node j, node i extracts $L(j)$, FAD(j), SAD(j), $\mu(j)$ and $\sum_{k \in C_j} \mu(k)$ from the packet. If the difference between $L(i)$ (node i's lifetime) and the minimum lifetime among node i's children is greater than a certain threshold α, the aggregation delays are adjusted according to the following two cases.

Case I: $L(i) > \min_{j\in C_i} L(j) + \alpha$. In this case, node i has a longer lifetime than the minimum lifetime among its children. From Eqs. (14) and (1), we know that the lifetime (L) of a child node is a decreasing function of its outgoing data rate (μ), which in turn is a non-increasing function of its SAD and FAD values. This means that the lifetime of a child node is a non-decreasing function of its SAD and FAD values. Therefore, to increase $\min_{j\in C_i} L(j)$, LBA adjusts the aggregation delays as follows:

- Let $\Delta = \min(\delta, \text{FAD}(i))$, where δ is a small value.
- For node i: $\text{FAD}(i) = \text{FAD}(i) - \Delta$; $\text{SAD}(i)$ remains unchanged; and $\text{CFAD}(i) = \text{CFAD}(i) - \Delta$.
- Node i piggybacks the updated $\text{CFAD}(i)$ value in the ACK to every child node j. Upon receiving the ACK, node j updates $\text{FAD}(j) = \text{CFAD}(j) - \text{CFAD}(i) = \text{FAD}(j) + \Delta$, $\text{SAD}(j) = \text{OAD}(j) - \text{CFAD}(i) = \text{SAD}(j) + \Delta$, and $\text{CFAD}(j) = \text{CFAD}(i) + \text{FAD}(j)$.

Case II: $L(i) < \min_{j\in C_i} L(j) - \alpha$. In this case, node i has a shorter lifetime than all of its children; hence, LBA will attempt to increase $L(i)$. According to Eq. (14), the lifetime of the parent node $L(i)$ is affected by both its outgoing data rate $\mu(i)$ and its incoming data rate $\sum_{j\in C_i} \mu(j)$. One possible adjustment strategy to increase $L(i)$ is to increase $\text{FAD}(i)$ to reduce $\mu(i)$. However, node i's children $j \in C_i$ would have to reduce their $\text{FAD}(j)$ values in order to satisfy the end-to-end delay requirement, which results in an increased incoming data rate $\sum_{j\in C_i} \mu(j)$ to node i. Therefore, it is unclear whether this strategy will indeed extend $L(i)$. Similarly, it is also unclear how $L(i)$ may be affected if $\text{FAD}(i)$ is decreased (which will increase $\mu(i)$) and consequently $\text{FAD}(j)$ for each child j is increased (which will increase $\sum_{j\in C_i} \mu(j)$). Due to such uncertainties, LBA allows node i to examine the effects of multiple adjustment strategies (to increase/decrease $\text{FAD}(i)$ and consequently decrease/increase $\text{FAD}(j)$ in steps of δ) on $L(i)$, until either an effective strategy has been found to increase $L(i)$ or all possible strategies have been attempted.

4.3. Handling of route and channel dynamics

In the presence of route and channel changes, to maintain the end-to-end delay requirement, it is critical to adjust the SAD and FAD values properly in a timely manner. In LBA, the maintenance procedure to handle route and channel dynamics is initiated by the leaf nodes. Consider a leaf node ℓ and let k denote its parent node. Depending on the relation between $\text{OAD}(\ell)$ and $\text{CFAD}(k)$, node ℓ behaves differently as follows.

If $\text{OAD}(\ell) \geq \text{CFAD}(k)$, node ℓ simply updates $\text{SAD}(\ell) = \text{OAD}(\ell) - \text{CFAD}(k)$ so the end-to-end delay requirement is still satisfied after the update. For example, as shown in Fig. 6(b), when the link condition over (3, 1) improves and the communication delay is reduced to 1 s, node 3 computes its new $\text{OAD}(3) = 8.0$ s. As $\text{OAD}(3) \geq \text{CFAD}(1)$, node 3 updates $\text{SAD}(3) = \text{OAD}(3) - \text{CFAD}(1) = 5.8$ s.

On the other hand, if $\text{OAD}(\ell) < \text{CFAD}(k)$, the overall aggregation delay allowed at ℓ is more stringent than the cumulative forwarding aggregation delay from its parent node to the sink. In this case, the end-to-end delay may be temporally violated. To address this issue, node ℓ sets $\text{FAD}(\ell) = \text{SAD}(\ell) = 0$ s and sends a request to its parent node k with $\text{OAD}(\ell)$ value embedded. Upon receipt of the request, node k compares $\text{OAD}(\ell)$ with the CFAD value received from its own parent, say node m. There are two possible situations:

- If $\text{OAD}(\ell) \geq \text{CFAD}(m)$, node k sets $\text{FAD}(k) = \text{OAD}(\ell) - \text{CFAD}(m)$ and leaves $\text{SAD}(k)$ unchanged. For example, as shown in Fig. 6(c), when node 3 switches its parent from node 1 to node 4, $\text{OAD}(3)$ becomes 5.5 s which is less than $\text{CFAD}(4) = 6.5$ s. Consequently, node 3 sets $\text{SAD}(3) = 0$ s and sends a request to node 4. As $\text{OAD}(3) \geq \text{CFAD}(2)$, node 4 sets $\text{FAD}(4) = \text{OAD}(3) - \text{CFAD}(2) = 1.1$ s.
- If $\text{OAD}(\ell) < \text{CFAD}(m)$, node k sets $\text{FAD}(k) = 0$ s and sends a request to its parent m with $\text{OAD}(\ell)$ embedded. This procedure continues till $\text{OAD}(\ell)$ can be accommodated by one of its ancestors. For example, as shown in Fig. 6(d), after $d(3, 4)$ changes from 1 s to 5 s, $\text{OAD}(3)$ becomes 4.1 s which is less than $\text{CFAD}(2) = 4.4$ s. As a result, node 4 sets $\text{FAD}(4) = 0$ s and sends a request to node 2. As $\text{OAD}(3) > \text{CFAD}(1)$, node 2 sets $\text{FAD}(2) = \text{OAD}(3) - \text{CFAD}(1) = 1.9$ s and the system stabilizes. Note that, as a parent node always informs its children of the most updated CFAD value in the ACK, each child node can update its SAD accordingly. In the above example, node 4 sets $\text{SAD}(4) = \text{OAD}(4) - \text{CFAD}(2) = 6.5 - 4.1 = 2.4$ s.

4.4. Handling of packet loss

When the channel condition deteriorates, data or ACK packets may get lost, and a node may need to retransmit for multiple times before the data packet can be delivered successfully. As a result, more energy may be consumed for transmission. This issue has been dealt with in LBA by extending the nodal lifetime estimation (i.e., Eq. (14) in Section 3.3) to include $\text{ETX}(i)$ – the expected number of transmission attempts to deliver a data packet successfully from node i to its parent node:

$$L(i) = \frac{e(i)}{\left(\left(T_r\left(\left\lceil \frac{\text{ETX}(i)}{m} \right\rceil - \frac{1}{2}\right) + \tau\left(\text{ETX}(i)\%m\right)\right)\mu(i) + \frac{\phi}{T_r} + \tau \sum_{j\in C_i} \mu(j)\right)P}. \tag{24}$$

Here, m is the maximum number of trials before a node gives up its transmission attempt, till the next time it receives a beacon from the intended receiver. Note that the measurement of ETX is readily available in many routing protocols such as CTP [18], and thus not an extra overhead.

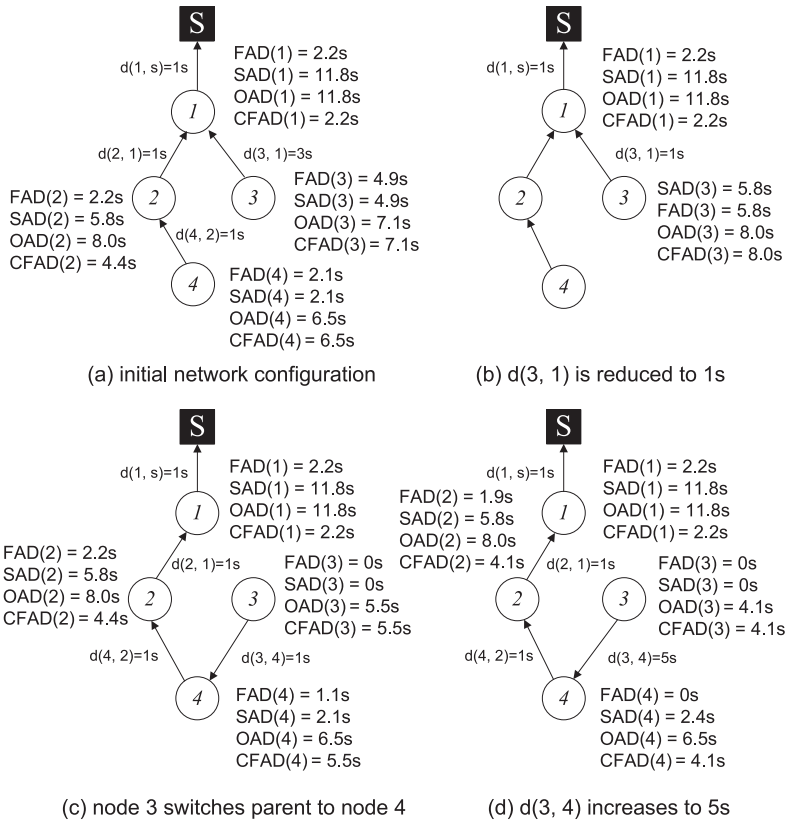


Fig. 6. Examples of maintaining the ($\rho = 80\%, \Theta = 20\ s$) end-to-end delay requirement in the presence of channel and route dynamics.

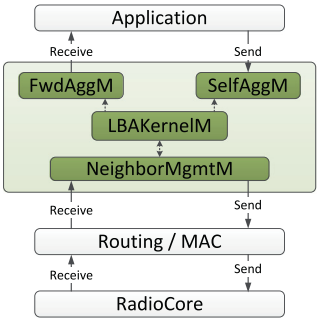


Fig. 7. Implementation of LBA in TinyOS.

5. Implementation

5.1. Software component

We have implemented LBA 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 on average. As shown in Fig. 7, this component sits between the application and routing layers to aggregate self-generated data from the application layer and sensory data received from the routing layer, and then forwards the aggregated result to the routing component when the aggregation timer is fired. More specifically, when data packets from a child node arrive at the routing layer, the piggybacked control information is extracted, passed to, and processed at the *LBAKernelM* module, which implements the main logic of LBA. The *NeighborMgmtM* module manages neighbors' information while the *FwdAggM* and *SelfAggM* modules maintain the aggregation timers. After deciding new FAD and SAD values, the *LBAKernelM* module notifies the *FwdAggM* and *SelfAggM* modules to adjust the aggregation timers accordingly; it also notifies the routing layer to piggyback the control information in the ACKs to child nodes.

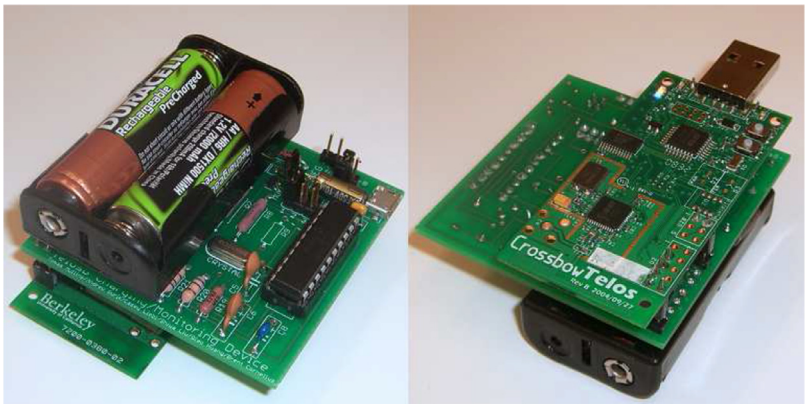


Fig. 8. TelosB power meter kit used in LBA. The operation power consumption of the kit is $2.4\,\mu W$ which is small compared to regular TelosB power consumption.

In our experiments, we choose CTP [18] as the routing layer protocol to build and maintain the data collection tree, and RI-MAC [20] as the MAC layer protocol.

5.2. Hardware component

Among all the control information piggybacked in the data packets, nodal lifetime is an important one. In order to estimate the nodal residual energy and the nodal lifetime, we have designed and fabricated a TelosB power meter kit as shown in Fig. 8 for this purpose. This kit measures the nodal energy consumption rate, based on which we can calculate the total energy consumed so far and estimate the nodal residual energy and the nodal lifetime.

During the experiments, we notice that it may take weeks to completely drain fully-charged batteries of a TelosB node. In order to complete all the experiments within a reasonable amount of time while demonstrating the features and performances of evaluated protocols, we study how fast a TelosB node consumes a designated small amount of energy, and evaluate its nodal lifetime as the time period for this designated amount of energy to be consumed. This also allows us to start the experiments with nodes at different initial energy levels, which simplifies and speeds up the evaluation process significantly.

Another benefit of this experiment methodology is that the effect of battery leakage is eliminated from the evaluation results. However, in a real IoT network where we need to estimate how long a node can operate with a fully-charged battery, we have to deal with the battery leakage issue. One possible solution is to integrate battery leakage models [26,27] into nodal lifetime estimation to improve the accuracy of the results.

6. Experiment results

6.1. Experiment setup

We evaluate the performance of the proposed LBA scheme using experiments, and compare it in terms of *network lifetime* and *end-to-end delay* with:

- UPPER (theoretical upper bound): Obtained using Algorithm 1;
- SL (spread latency scheme): The delay-bounded, time-asynchronous, distributed data aggregation scheme proposed by Becchetti et al. [17]. To the best of our knowledge, this is the scheme most related to LBA, though it does not deal with network heterogeneity or dynamics.

We set up a testbed network of 32 TelosB nodes, which form a tree topology (shown in Fig. 9) where node 0 is the sink. The RI-MAC parameters are set to: $T_r = 1$ s, $\phi = 40$ ms, and $\tau = 40$ ms. The LBA parameters are set to: $\alpha = 10$ min and $\delta = 0.1$ s. The data generation rate at source nodes follows a uniform distribution between 0.8 s and 1.2 s. The end-to-end delay requirement Θ varies from 20 to 140 s, and $\rho = 80\%$. The full energy level of a node is set to 200 J which can support a TelosB node running for about 45 min at 100% radio duty cycle. Each point in the result figures is averaged over 5 experiment runs.

In each experiment, all nodes operate according to the evaluated aggregation scheme. That is, each node waits for a certain amount of aggregation delay, merges all the packets received or generated during the waiting time into a single packet, and then forwards it to the parent node.

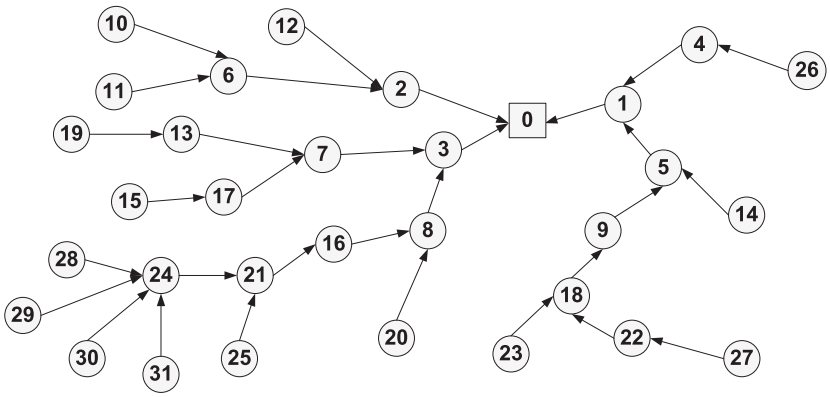
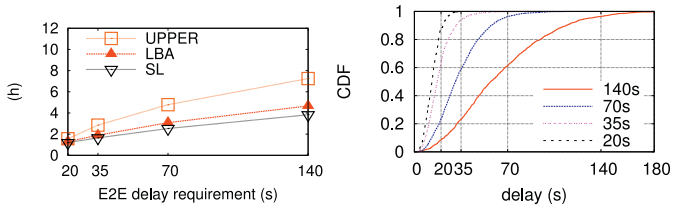
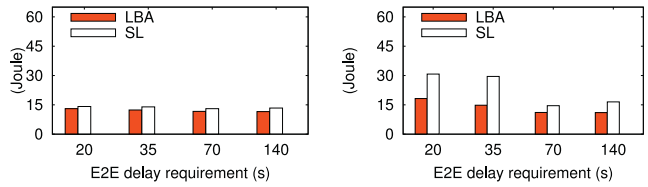


Fig. 9. Network topology used in the experiments.



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



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

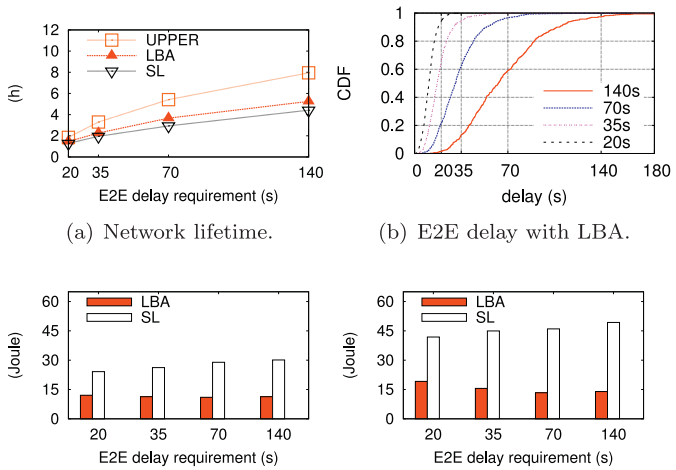
Fig. 10. Performance comparison when all nodes generate data packets. [Note: (c) and (d) plot the standard deviation and average of nodal residual energy when the first node dies].

6.2. Uniform initial nodal energy

Figs. 10 and 11 plot the experiment results when all nodes start with the full energy level. As shown in Fig. 10(a), when all nodes are sources, LBA yields a better performance than SL, but the performance improvement is not significant. This is due to the uniform initial nodal energy among all nodes. As shown in Fig. 10(c) and(d), SL results in a small deviation in residual nodal energy among nodes, which indicates that SL already performs well by assigning the aggregation delays fairly and statically to all nodes. Therefore, the additional benefit brought by LBA via dynamic assignment of aggregation delays is limited. Finally, Fig. 10(b) shows that the end-to-end delivery delay requirement is always satisfied when LBA is applied. We have similar observations from Fig. 11, which shows the results when only leaf nodes are sources.

6.3. Non-uniform initial nodal energy

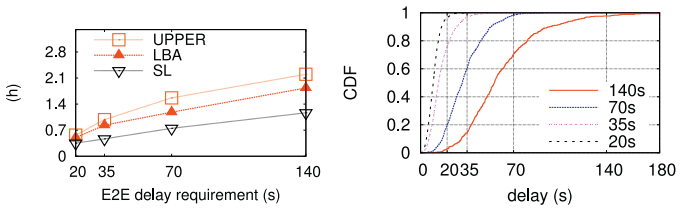
We also run experiments with nodes starting with different initial nodal energy levels. Specifically, we select x (< 32) nodes to start the experiments with y percentage of the full energy level where $y < 100$; other nodes start with full energy. All nodes are sources. Fig. 12 plots the results when $x = 2$ and $y = 25$, and it can be seen that LBA yields a significantly longer network lifetime than SL. Note that, with LBA and SL, nodes start with the same initial distribution of aggregation delays. However, as the distribution of aggregation delays remains unchanged with SL after initial assignment, if there is a significant difference between nodal energy levels, the nodes with lower energy levels may die much earlier than the nodes with higher energy levels, which may lead to a shortened network lifetime. In comparison, with heterogeneous initial nodal energy, LBA adjusts the distribution of aggregation delays dynamically so that the nodes with lower energy levels may be



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

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

Fig. 11. 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].



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

Fig. 12. Performance comparison under non-uniform initial nodal energy. Nodes 8 and 9 start with 25% of the full energy level while others start with full energy.

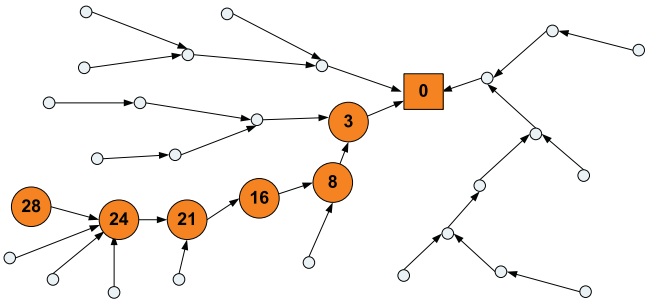
assigned more aggregation delays and hence transmit less frequently and consume less energy, which may lead to a more balanced nodal lifetime among nodes and yield a prolonged network lifetime.

To illustrate how LBA prolongs the lifetime of low energy (and hence short lifetime) nodes in the network, Fig. 13(b) and (c) plot the changing traces of nodal energy levels and aggregation delays of nodes along the path from node 28 to the sink (i.e., node 0), as shown in Fig. 13(a). We can see that when node 8 starts with 25% of the full energy level, its own aggregation delay keeps increasing while its parent's (node 3) and child's (node 16) aggregation delays keep decreasing. When node 16's aggregation delay reaches zero, it gains aggregation delay from its own child (node 21) to compensate node 8. In other words, node 8 can get help not only from its direct parent or child nodes, but also from other nodes in the network indirectly. As a result, node8/s energy drops more slowly than other nodes on the path, and the network lifetime is extended.

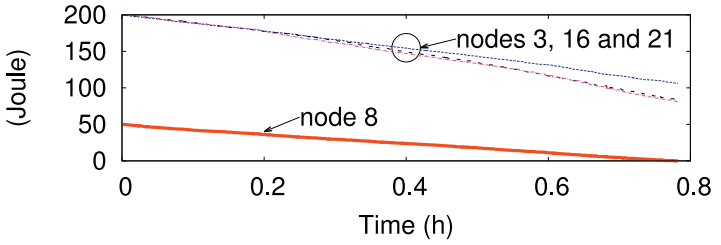
6.3.1. Impact of the initial nodal energy variation

To further evaluate the performance of LBA under network heterogeneity, we conduct two sets of experiments to evaluate the impact of the initial nodal energy variation on the network lifetime. In the first set, we vary the number of low-initial-energy nodes (denoted as x) from 0 to 8, while fixing the initial energy level of these nodes to $y = 25\%$ of the full energy level. As shown in Fig. 14(a), when $x > 0$, the performance of SL does not change much with different x values, because the network lifetime achieved by SL is limited as long as there exists at least one low-initial-energy node. On the other hand, the performance gain of LBA over SL is less significant as x increases. This is because, with more low-initial-energy nodes in the network, each of them obtains less amount of aggregation delay compensation and hence less lifetime improvement.

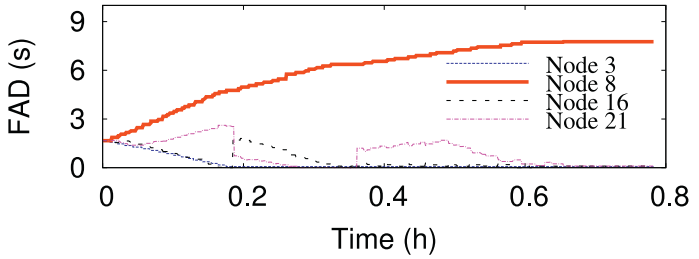
In the second set of experiments, we fix $x = 2$ while varying the initial energy level of these two nodes from $y = 100$ to 25% of the full energy level. As shown in Fig. 14(b), as y decreases, the performance of SL drops much faster than that of LBA. This is because the network lifetime achieved by SL is bounded by the shortest nodal lifetime among all nodes in the network. In comparison, with LBA, the nodal lifetime of low-initial-energy nodes can be increased through re-distribution of the aggregation delays. Though lower initial energy levels demand more re-distribution efforts and thus shorten the network lifetime, this effect is absorbed by all nodes along the same branch towards the sink; hence, the decreasing rate of the network lifetime is low with LBA.



(a) The studied route: from node 28 to the sink.

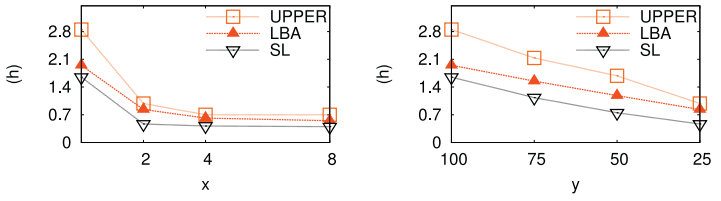


(b) Nodal residual energy.



(c) Nodal aggregation delay.

Fig. 13. Traces of nodal residual energy and aggregation delays of nodes 3, 8, 16, and 21. Node 8 starts with 25% of the full energy level while others start with full energy.



(a) $x = 0, 2, 4, 8$; $y = 25$. (b) $x = 2$; $y = 100, 75, 50, 25$.

Fig. 14. Impact of the initial nodal energy variation on the network lifetime. x is the number of low-initial-energy nodes. y denotes the initial energy level (as percentage of the full energy) of these x nodes.

6.4. Performance under network dynamics

In addition to the experiments conducted with heterogeneous initial nodal energy, we evaluate the performance of LBA under a more realistic situation of network dynamics where the network topology and the data generation intervals vary over time. Specifically, in this experiment, the routing tree is dynamically updated using the CTP protocol rather than fixed as shown in Fig. 9. All nodes are sources. Each node varies its data generation interval randomly in a specified range every 100 packets.

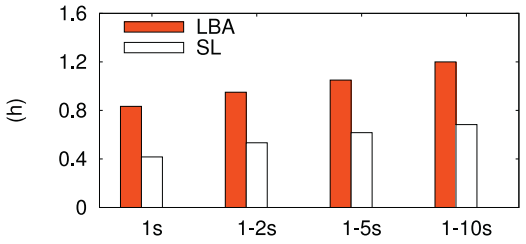
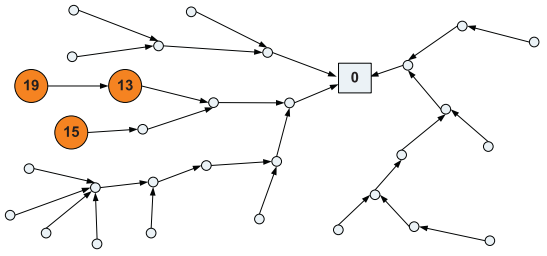
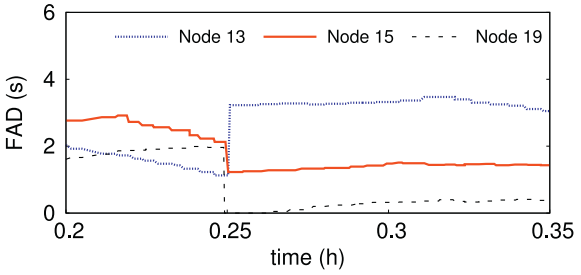


Fig. 15. Performance comparison under various data generation intervals. Nodes 8 and 9 start with 25% of the full energy level while others start with full energy.



(a) The studied route change scenario.



(b) Trace of aggregation delays.

Fig. 16. Trace of aggregation delays after node 19 switches its parent from node 13 to 15 at the time instance of about 0.25 h.

Fig. 15 compares the network lifetime as the data generation interval varies in different ranges, where the "1-Xs" label along the X-axis means that the range is between 1 and X s. As we can see, LBA outperforms SL significantly, which indicates that LBA can adapt to the updates of routing activities and the changes in data generation intervals.

To illustrate how LBA handles the network dynamics, Fig. 16(b) plots the changing trace of aggregation delays of nodes 13, 15, and 19. Originally, node 13 is node 19's parent and node 15 is a leaf node. At the time instance of about 0.25 h, due to routing updates, node 19 switches its parent from node 13 to node 15. Since the existing aggregation delay settings along the path from node 15 to the sink has been coordinated to achieve the end-to-end delay bound, after node 19 joins the path, the end-to-end delay from node 19 to the sink is over the delay requirement. Based on the LBA design described in Section 4.3, node 19 has to decrease its aggregation delay immediately after the route change. However, even after node 19 has dropped its aggregation delay to zero, the end-to-end delay requirement still cannot be satisfied. As a result, node 15 which is the new parent of node 19, also needs to reduce its aggregation delay, and then the end-to-end delay requirement can be satisfied. On the other hand, after node 19 changes its parent, node 13 becomes a leaf node and hence can increase its aggregation delay to reduce the outgoing data traffic, thus conserving energy.

7. Related work

7.1. Data aggregation to suppress traffic

Numerous in-network data aggregation schemes [22,23,28] have been proposed in the past. Fan et al. [22] proposed two methods to improve the data aggregation ratio: Data-Aware Anycast (DAA) for spatial convergence and Randomized Waiting (RW) for temporal convergence. To address the scalability issue present in the above work, Fan et al. further proposed a semi-structured approach called ToD [23] that uses a structure-less technique on an implicitly constructed packet forwarding structure. He et al. [28] isolated aggregation decisions into a module that resides between the network and data link layers

to adaptively perform application-independent data aggregation. However, the timeliness of data delivery was not a technical concern of the afore-mentioned works.

7.2. Data aggregation for delay minimization

There have also been research efforts on designing and optimizing aggregation schedules to minimize data delivery delay. Wan et al. [29] proposed a set of rules to construct the minimum-latency schedule for data aggregation, subject to the interference constraint in synchronous time slotted networks. Yu et al. [30] studied how to build a collision-free schedule to minimize the data aggregation delay, and the work has been extended to multiple-sink scenarios in [31]. Bagaa et al. [32] studied the problem of minimizing data delivery delay in structureless and semi-structure data aggregation contexts. Assuming a time-slotted system and node-exclusive interference model, Joo et al. [33] proved a lower bound of the minimum sum of delay of all data packets. Nevertheless, all of these works targeted at minimizing the delay without considering the network lifetime.

7.3. Lifetime-delay tradeoff in data aggregation

Ye et al. [12] formulated the energy-delay tradeoff problem in data aggregation as a semi-Markov decision process by depreciating the data revenue as the aggregation holding time increases. Luo et al. [13] developed an algorithm to construct a max-lifetime shortest-path aggregation tree from the set of short spanning trees to strike a balance between nodal lifetime and data delivery delay. However, achieving an end-to-end data delivery delay bound was not a design objective in the above works.

To bound the end-to-end data delivery delay, many existing works [14–16] require time synchronization between neighboring nodes. Solis et al. [15] employed the concept of cascading timeout where a node's aggregation timer is fired right before its parent's to achieve a high aggregation degree with small delay overhead. Xiang et al. [16] 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. Hariharan and Shroff [14] formulated the energy-delay tradeoff problem as an integer optimization problem. Different from these works, our scheme does not require time synchronization. Moreover, all of the afore-mentioned works aimed to minimize the network-wide energy consumption, without considering the lifetime balance between nodes which is critical in improving the network lifetime. Although Zhang et al. [34] explored how to satisfy the end-to-end delay constraint in an asynchronous, contention-based sensor network, the design objective was to maximize the amount of information collected within the delay bound, rather than improving the network lifetime.

Becchetti et al. [17] investigated the problem of energy-efficient data delivery within a delay bound and proposed two distributed schemes to balance the energy consumption between nodes. However, the proposed schemes only work in homogeneous and static networks (e.g., the battery quality, radio energy consumption rate, and initial nodal energy are the same for all nodes) whereas our scheme can deal with heterogeneous and dynamic situations.

7.4. Aggregation tree construction and optimization

There have also been works [35–38] on optimizing the aggregation tree structure to either minimize the total network energy consumption or maximize the network lifetime. Our scheme is orthogonal to these works, because our scheme can work with any aggregation tree structure, even in the presence of channel contention, routing updates, or other network dynamics.

8. Conclusions

This paper introduced a lifetime balanced data aggregation scheme, called LBA, for asynchronous duty cycle IoT networks. Through adaptively adjusting the aggregation delays of neighboring devices in a distributed manner, LBA effectively improves the lifetime of IoT devices with lower energy supplies and/or higher energy consumption and thus prolongs the IoT network lifetime. This has been verified by extensive experimental evaluations on a testbed. One interesting problem that demands further investigation is how to improve LBA to work with multiple sinks. Besides, how to combine LBA with energy-aware routing or MAC layer protocols to further prolong the IoT network lifetime is another direction of future work.

Acknowledgement

A preliminary version of this work appeared in the proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM 2012) [39]. The work was supported partly by the NSF under grants CNS-0831874 and ECCS-1128312.

References

[1] Madden SR, Franklin MJ, Hellerstein JM, Hong W. TinyDB: an acquisitional query processing system for sensor networks. ACM Trans Database Syst 2005;30(1):122–73.

[2] Kumar R, Wolenetz M, Agarwalla B, Shin J, Hutto P, Paul A, et al. DFuse: a framework for distributed data fusion SenSys; 2003.
[3] Kansal A, Potter D, Srivastava MB. Performance aware tasking for environmentally powered sensor networks. SIGMETRICS Perform Eval Rev 2004;32(1):223–34.
[4] Dutta P, Hui J, Jeong J, Kim S, Sharp C, Taneja J, et al. Trio: enabling sustainable and scalable outdoor wireless sensor network deployments IPSN; 2006.
[5] Tong B, Li Z, Wang G, Zhang W. How wireless power charging technology affects sensor network deployment and routing. ICDCS; 2010.
[6] Li Z, Peng Y, Zhang W, Qiao D. Study of joint routing and wireless charging strategies in sensor networks. WASA; 2010.
[7] Peng Y, Li Z, Zhang W, Qiao D. Prolonging sensor network lifetime through wireless charging. RTSS; 2010.
[8] Li Z, Peng Y, Zhang W, Qiao D. J-RoC: a joint routing and charging scheme to prolong sensor network lifetime. ICNP; 2011.
[9] Wang W, Srinivasan V, Chua KC. Using mobile relays to prolong the lifetime of wireless sensor networks. MobiCom; 2005.
[10] Chang J, Tassiulas L. Energy conserving routing in wireless Ad-hoc networks. INFOCOM; 2000.
[11] Chang J, Tassiulas L. Maximum lifetime routing in wireless sensor networks. IEEE/ACM Trans Netw 2004;12(4):609–19.
[12] Ye Z, Abouzeid A, Ai J. Optimal policies for distributed data aggregation in wireless sensor networks. INFOCOM; 2007.
[13] Luo D, Zhu X, Wu X, Chen G. Maximizing lifetime for the shortest path aggregation tree in wireless sensor networks. INFOCOM; 2011.
[14] Hariharan S, Shroff N. Maximizing aggregated revenue in sensor networks under deadline constraints. CDC; 2009.
[15] Solis I, Obraczka K. The impact of timing in data aggregation for sensor networks. ICC; 2004.
[16] Xiang Q, Xu J, Liu X, Zhang H, Rittle L. When in-network processing meets time: complexity and effects of joint optimization in wireless sensor networks. RTSS; 2009.
[17] Becchetti L, Marchetti-Spaccamela A, Vitaletti A, Korteweg P, Skutella M, Stougie L. Latency-constrained aggregation in sensor networks. ACM Trans Algorithms 2009;6(1):1–20.
[18] Gnawali O, Fonseca R, Jamieson K, Moss D, Levis P. Collection tree protocol. SenSys; 2009.
[19] Zhao J, Govindan R. Understanding packet delivery performance in dense wireless sensor networks. SenSys; 2003.
[20] Sun Y, Gurewitz O, Johnson D. RI-MAC: areceiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. SenSys; 2008.
[21] Madden SF, Hellerstein MJ, Hong WJM. TAG: atiny AGgregation service for Ad-Hoc sensor networks. In: Operating Systems Review, 36; 2002. p. 131–46.
[22] Fan K, Liu S, Sinha P. Structure-free data aggregation in sensor networks. INFOCOM; 2006a.
[23] Fan K, Liu S, Sinha P. Scalable data aggregation for dynamic events in sensor networks. SenSys; 2006b.
[24] Buettner M, Yee GV, Anderson E, Han R. X-MAC: ashort preamble MAC protocol for duty-cycled wireless sensor networks. SenSys; 2006.
[25] Dutta P, Dawson-Haggerty S, Chen Y, Liang C-J M, Terzis A. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. SenSys; 2010.
[26] Online Link. http://www.mpoweruk.com/performance.htm.
[27] Wu B, White RE. Self-discharge model of a nickel-Hydrogen cell. J Electrochem Soc 2000;147:902–9.
[28] He T, Blum BM, Stankovic JA, Abdelzaher T. AIDA: adaptive application independent data aggregation in wireless sensor networks. ACM Trans Embed Comput Syst 2004;3:426–57.
[29] Wan P-J, Huang SC-H, Wang L, Wan Z, Jia X. Minimum-latency aggregation scheduling in multihop wireless networks. MobiHoc; 2009.
[30] Yu B, Li J, Li Y. Distributed data aggregation scheduling in wireless sensor networks. INFOCOM; 2009.
[31] Yu B, Li J. Minimum-time aggregation scheduling in multi-sink sensor networks. SECON; 2011.
[32] Bagaa M, Derhab A, Lasla N, Ouadjaout A, Badache N. Semi-structured and unstructured data aggregation scheduling in wireless sensor networks. INFOCOM; 2012.
[33] Joo C, Choi J-G, Shroff N. Delay performance of scheduling with data aggregation in wireless sensor networks. INFOCOM; 2010.
[34] Zhang J, Jia X, Xing G. Real-time data aggregation in contention-based wireless sensor networks. ACM Trans Sen Netw 2010;7(1):1–25.
[35] Wu Y, Fahmy S, Shroff NB. On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm. INFOCOM; 2008.
[36] Hua C, Yum T-SP. Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks. IEEE/ACM Trans Netw 2008;16(4):892–903.
[37] Xue Y, Cui Y, Nahrstedt K. Maximizing lifetime for data aggregation in wireless sensor networks. Mob Netw Appl 2005;10(6):853–64.
[38] Kuo T-W, Tsai M-J. On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: Np-completeness and approximation algorithms. INFOCOM; 2012.
[39] Li Z, Peng Y, Qiao D, Zhang W. LBA: lifetime balanced data aggregation in low duty cycle sensor networks. INFOCOM; 2012.

Zi Li received his Ph.D. degree in Computer Science from Iowa State University in 2013. He is currently a Staff Software Engineer in LinkedIn, Mountain View, California, USA. His research interests include Social Networks, Wireless Sensor Networks, and Mobile Computing.

Wensheng Zhang received his Ph.D. degree in Computer Science and Engineering from the Pennsylvania State University. He joined the Department of Computer Science at Iowa State University in 2005, where he is now an Associate Professor. His research interests include distributed systems, networks, and security.

Daji Qiao received the Ph.D. degree in Electrical Engineering: Systems from the University of Michigan, Ann Arbor, in 2004. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, Iowa State University, Ames, Iowa, USA. His research interests include Sensor Networks and Internet of Things, Wireless Networking and Mobile Computing, and Cyber Security.

Yang Peng received his Ph.D. degree in Computer Science from Iowa State University in 2014. He is currently an Assistant Professor in the Division of Computing and Software Systems, University of Washington Bothell, Bothell, Washington, USA. His research interests include Internet of Things, Wireless Sensor Networks, and Cyber-Physical Systems.