Joint Aggregation and MAC Design to Prolong Sensor Network Lifetime

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 JAM, a Joint Aggregation and MAC design, to improve the sensor network lifetime under the end-to-end delay constraint. The key idea is to adjust both network traffic (via data aggregation) and communication overhead (via duty-cycled MAC) in a holistic manner at each individual node as well as between neighbors. As a result, JAM extends the sensor network lifetime more efficiently and effectively than the state-of-the-art solutions while guaranteeing the desired delay bound and achieving a lower level of average nodal power consumption. JAM is a lightweight and distributed solution with limited control information exchanged between neighbors only, which makes it deployable in practical sensor networks. Extensive ns-2 simulation and TinyOS experiment results are used to demonstrate the effectiveness of JAM in prolonging the network lifetime.**

I. Introduction

A. Motivation

Extending the lifetime of sensor networks is critically important when the networks are deployed for long-term monitoring applications. Besides duty cycling, balancing nodal lifetime is another major approach for network lifetime extension, because the network lifetime is often defined as the minimal nodal lifetime among all nodes in the network [1]–[3]. Following this approach, a variety of duty-cycled MAC protocols [4]–[6] and data aggregation schemes [7], [8] have been proposed recently.

Duty-cycled sensor networks rely on MAC protocols to establish rendezvous between sender and receiver nodes. The incurred MAC-layer communication overhead is distributed between sender and receiver in different manners with different MAC protocols. To balance nodal lifetime, the authors of [4]–[6] proposed to adapt the distribution of communication overhead among neighbors according to their nodal lifetime, i.e., longer-lifetime nodes shall absorb more communication overhead than their shorter-lifetime neighbors.

Through eliminating inherent redundancy in raw sensory data, in-network data aggregation [9], [10] can effectively reduce network traffic. For the sake of aggregation, a node needs to hold data (received or self-generated) for a while. Clearly, a node can suppress more data traffic with a longer holding time. However, holding data introduces extra delay, and the value of sensory data could be greatly depreciated if the data is delivered to the sink with a delay longer than an application-specified delay bound. To balance nodal lifetime while maintaining a required end-to-end delay bound, the authors of [7], [8] proposed to adapt the distribution of holding time among nodes along the same route according to their nodal lifetime; i.e., shorter-lifetime nodes shall hold data longer to reduce its outgoing data traffic while longer-lifetime nodes shall hold data for a shorter time so that the end-to-end delay requirement is still guaranteed.

As the aforementioned MAC and data aggregation schemes improve the network lifetime from two distinct perspectives, it is naturally an attractive idea to explore the possibility of a joint design of MAC and data aggregation to further improve the network lifetime. This, however, is a challenging task due to the following reasons. Firstly, most of the MAC protocols were designed without explicitly considering the end-to-end delay requirement. As a result, they may yield uncontrollable end-to-end delay under certain practical scenarios; this is unacceptable for data aggregation applications that often require a stringent end-to-end delay bound. Secondly, even if there exists a MAC protocol that can provide a certain delay guarantee, when it works with a data aggregation scheme, it is non-trivial to decide how to divide the allowed end-to-end delay into two parts to serve as the delay constraints respectively for the MAC and data aggregation protocols, such that the maximum lifetime improvement can be accomplished. Therefore, it is important to develop an innovative approach to integrate MAC and data aggregation protocols together via a joint adjustment of their protocol parameters.

B. Related Work

Research has been conducted on designing duty-cycled MAC protocols and energy-efficient or lifetime-extending data aggregation schemes. However, there is no research on jointly adjusting data aggregation and MAC behaviors to extend the sensor network lifetime.

Numerous MAC protocols [4]–[6], [11] have been proposed to extend network lifetime through balancing nodal lifetime. Particularly, SEESAW [4] balances the energy consumption between sender and receiver through adapting the data retry interval at the sender side and the channel checking period at the receiver side. ZeroCal [6] targets at improving the fairness of energy utilization in duty-cycled sensor networks by dynamically tuning the nodal wakeup interval. GDSIC [5] decides the individual nodal wakeup interval through solving distributed convex optimization problems. Though the network lifetime can be prolonged by these schemes, they do not consider the end-to-end delay bound. pTunes [11] is a recently-proposed centralized solution, which formulates a multi-objective optimization problem, where prolonging network lifetime and guaranteeing the end-to-end delay can be solved together.

Most of existing data aggregation schemes [12]–[16] have the objective of minimizing the total network energy consumption instead of extending network lifetime, or do not consider the end-to-end delay requirement. The problem of balancing nodal lifetime under a delay constraint has been studied in [7], [8]. Particularly, Becchetti et al. [8] investigated the problem of energy-efficient data aggregation within a delay bound, and proposed two distributed schemes to balance energy consumption among sensor nodes. LBA [7] is a recently-proposed lifetime-balancing aggregation protocol. Through dynamically adjusting the aggregation holding time among neighbors to balance their nodal lifetime, LBA provides a low cost, asynchronous, and delay-constrained data aggregation scheme for duty-cycled sensor networks.

C. Contributions

In this paper, we propose a novel holistic approach, called JAM (Joint Aggregation and MAC), to jointly adjust MAC and data aggregation behaviors to extend the sensor network lifetime. The key idea of JAM is to coordinate the aggregation and MAC behaviors at each individual node as well as between neighbors, with the target of extending the minimal nodal lifetime in the neighborhood. As such coordination occurs in all neighborhoods, the network lifetime, i.e., the minimal nodal lifetime among all nodes in the network, may be improved. As JAM reduces both network traffic (via data aggregation) and communication overhead (via duty-cycled MAC), it prolongs the network lifetime more efficiently and effectively than previous works that only use one of the two techniques. The contributions of this work are summarized as follows.

- To the best of our knowledge, JAM is the first design on integrating and jointly configuring MAC and data aggregation protocols to extend the lifetime of duty-cycled sensor networks.
- JAM is a distributed and lightweight solution. It works through limited control information exchanged locally between neighbors.
- JAM has been implemented and evaluated on a sensor network testbed, and results show that it can achieve significant improvement on network lifetime compared to the state-of-the-art solutions.

D. Organization

The rest of the paper is organized as follows. Section II presents the system models and problem statement. Details of the JAM design and implementation are described in Section III. Section IV shows the performance evaluation results obtained from both ns-2 simulations and TinyOS testbed experiments. Finally, Section 5 concludes the paper.

II. Problem Description

A. System Models

We consider a sensor network deployed for monitoring applications where in-network data aggregation is allowed. Each sensor node generates and reports sensory data periodically, and all nodes form a data collection tree rooted at the sink. Protocols like CTP (Collection Tree Protocol) [17] could be used to build and maintain the data collection tree.

1) Aggregation Model: The *total aggregation* model [18] is adopted, which allows an arbitrary number of data packets generated and/or received at the same node to be suppressed into a single data packet. Such a model is useful in many sensor network applications, for example, when users are more interested in the maximum, minimum, average, or percentile statistics of sensory data, rather than the raw data themselves.

With this model, a source node may not send out a sensory 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 aggregates all data generated during the period to reduce the amount of data traffic to its parent node. Similarly, a forwarding node may not forward a data packet immediately after reception; it may wait for another period of time (called the *forwarding-aggregation delay* (FAD)) and aggregate all packets received during the period. Generally, the longer time a node waits for aggregation, the more data traffic can be suppressed; at the same time, data delivery latency is increased.

2) MAC Model: To conserve energy without time synchronization overhead, it is desired to employ an asynchronous and duty-cycled MAC protocol for long-term monitoring applications. The design principle of our proposed scheme does not require a particular MAC protocol. Instead, our design works compatibly with any asynchronous duty-cycled MAC protocol, as long as the protocol allows a node's duty cycle to be adjusted dynamically. To simplify the presentation, however, we assume each node runs an RI-MAC [19] like protocol as shown in Figure 1.

In RI-MAC, each node wakes up every Tr interval to interact with potential senders. Upon wakeup, it sends out a beacon and then checks the channel activity for ϕ time. If a data packet is received within ϕ, it replies with an ACK; otherwise, it goes back to sleep. On the other hand, if a node has a data packet to send, it remains awake and waits idly for the receiver's beacon to start data transmission. In the worst-case scenario, the sender has to stay awake for Tr time before rendezvous with the receiver, which incurs a transmission delay of Tr. Different from the original RI-MAC protocol that has a fixed Tr, we assume Tr is dynamically tunable. As can be observed, a larger Tr reduces the receiver's channel polling frequency

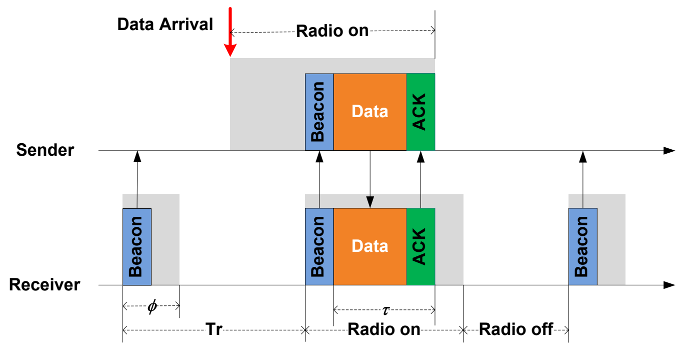


Fig. 1. An RI-MAC like protocol but with a tunable Tr parameter.

and hence its energy consumption; on the other hand, it increases the sender's energy consumption on idly waiting for the beacon, and meanwhile increases the transmission delay over the sender-to-receiver link.

3) Delay Model: Three types of delays are involved along a source-to-sink path: (i) one SAD at the source node, (ii) multiple FADs at the forwarding nodes, and (iii) multiple transmission delays over the links. With the aggregation and MAC models described above, to guarantee that the maximum end-to-end packet delivery delay from any source node i to the sink node is bounded by an application-specific parameter D, we need to ensure that[1]

$$\mathrm{SAD}(i) + \sum_{m \in \mathcal{S}_i} (\mathrm{Tr}(m) + \mathrm{FAD}(m)) \leqslant D, \tag{1}$$

where $\mathcal{S}_i$ is the path from node i's parent to sink. As larger aggregation delays would allow more traffic to be aggregated to save more energy, the delay bound shall be fully utilized to maximize network lifetime, meaning that the equality shall hold true in Inequality (1) in practical schemes.

B. Problem Statement

To effectively prolong the sensor network lifetime under the end-to-end packet delivery delay constraint, it is critical to have a holistic approach to adjust data aggregation and MAC behaviors of all sensor nodes. Ideally, all sensor nodes shall work together to maximize the minimum nodal lifetime in the entire network. Unfortunately, this global objective is impossible to accomplish in a realistic sensor network, as it requires each node to know the residual energy levels and data generation rates of all other nodes, and the topology of the network, which are highly dynamic and often unpredictable by nature. Instead, we study the following localized problem for each sensor node i in the network:

[1]In practice, data or ACK packets may get lost due to collision, interference, or deteriorated channel condition. As a result, the sensor node may need to retransmit multiple times before the data packet can be delivered successfully. This issue has been dealt with in JAM (as shown in Section III-F2) by replacing Tr with ETX·Tr in Equations (1) and (2), where ETX is the expected number of transmission attempts to deliver a data packet successfully over one hop. To simplify the presentation, we set ETX $= 1$ during the explanation of the JAM design in the next section, while the practical Equation (14) (given in Section III-F2) is used in the actual JAM implementation.

Objective:

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

Subject to:

- $\mathrm{SAD}(i),\ \mathrm{FAD}(i),\ \mathrm{Tr}(i) \geqslant 0$;
- *End-to-End Delay Requirement:*

$$\mathrm{SAD}(i) + \sum_{m \in \mathcal{S}_i} (\mathrm{Tr}(m) + \mathrm{FAD}(m)) = D. \tag{2}$$

Output:

- i's MAC protocol parameter: $\mathrm{Tr}(i)$;
- i's data aggregation parameters: $\mathrm{SAD}(i)$ and $\mathrm{FAD}(i)$.

The goal of this problem is to maximize the minimal nodal lifetime in i's neighborhood. As such procedure occurs in all neighborhoods, the minimal nodal lifetime in the entire network, i.e., the network lifetime, may be improved gradually.

III. The Proposed JAM Scheme

In this section, we propose a protocol called *JAM* to address the problem defined above. In JAM, coordination only occurs between a sensor node and its child nodes, through exchanging lightweight control information as well as adjusting their aggregation and MAC behaviors together in a collaborative manner. Figure 2 gives an overview of the JAM scheme. To ease the presentation, we use the topology shown in Figure 3 as an example to explain the design details of JAM.

JAM consists of four modules: Aggregator, OREM (Output Rate Estimation Module), LEM (Lifetime Estimation Module), and ICCM (Intra-node Cross-layer Collaboration Module). In

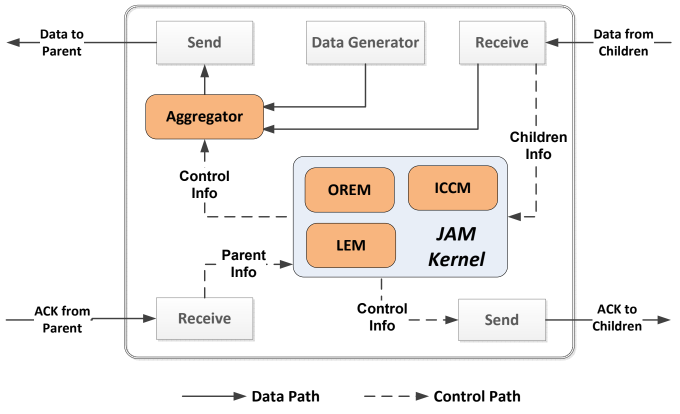


Fig. 2. JAM Overview.

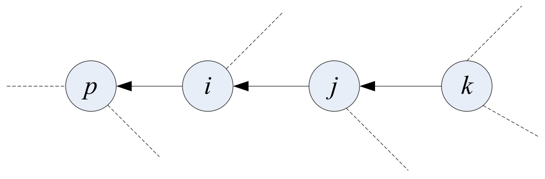


Fig. 3. Topology used to describe JAM details.

general, when node i receives a data packet from its child nodes or an ACK from its parent, it extracts the control information embedded in the packet and feeds them into JAM kernel. Here, the control information needed by JAM kernel includes five items from each child node j of i: $e(j)$, $\lambda(j)$, $\mathrm{SAD}(j)$, the delay introduced at node j: $D(j) = \mathrm{FAD}(j) + \mathrm{Tr}(j)$, and j's own input data rate $\sum_{k\in\mathcal{C}(j)} \mu(k)$ where $\mu(k)$ is the output data rate of a child node k of j; and two items from parent node p of i: $\mathrm{Tr}(p)$ and $\theta(p)$. We will explain the meaning of θ in Section III-E. With these information, JAM kernel decides how node i shall adjust its MAC behavior (i.e., Tr) and aggregation behavior (i.e., SAD and FAD) to improve the minimal nodal lifetime in its neighborhood. The decision is then piggybacked into the ACK packet to its child nodes. Upon being notified, each child node adjusts its MAC and aggregation behaviors.

A. Aggregator Module

The aggregator module controls the nodal aggregation behavior. Specifically, it works as follows at node i:

Case I: $\dfrac{1}{\sum_{j\in\mathcal{C}(i)} \mu(j)} \leqslant \mathit{FAD}(i) \leqslant \mathit{SAD}(i)$. In this case, node i has a data packet arrival rate no lower than $1/\mathrm{FAD}(i)$ and $1/\mathrm{SAD}(i)$. A timer is fired every $\mathrm{FAD}(i)$ interval. When it fires, all data packets received and self-generated since its last firing are aggregated into a single packet, and then forwarded to the parent node. This way, we ensure that packets received from i's child nodes are held for no longer than $\mathrm{FAD}(i)$ time. Also, since the self-generated data packets are aggregated and forwarded every $\mathrm{FAD}(i)$ time, which is always smaller than $\mathrm{SAD}(i)$ in JAM, their delay requirement is guaranteed as well. We will prove $\mathrm{FAD}(i) \leqslant \mathrm{SAD}(i)$ for any node i in Section III-B.

Case II: $\dfrac{1}{\sum_{j\in\mathcal{C}(i)} \mu(j)} > \mathit{FAD}(i)$ *and* $\dfrac{1}{\lambda(i)} \leqslant \mathit{SAD}(i)$. In this case, data packets from child nodes arrive at a rate lower than $1/\mathrm{FAD}(i)$ while the node itself generates data packets at a rate no lower than $1/\mathrm{SAD}(i)$. Data received from children and self-generated data are treated differently as follows:

- Whenever a data packet is received from a child node, the packet is aggregated immediately with all the self-generated data packets that have not yet been aggregated, and then forwarded to the parent node. Hence, the forwarding-aggregation delay is zero.
- A timer is fired every $\mathrm{SAD}(i)$ interval. When it fires, all the self-generated data packets that have not yet been aggregated are suppressed into a single packet and then forwarded to the parent node.

Case III: $\dfrac{1}{\sum_{j\in\mathcal{C}(i)} \mu(j)} > \mathit{FAD}(i)$ *and* $\dfrac{1}{\lambda(i)} > \mathit{SAD}(i)$. In this case, data packets from child nodes arrive at a rate lower than $1/\mathrm{FAD}(i)$ while the node itself generates data packets at a rate lower than $1/\mathrm{SAD}(i)$. As the data packet arrival/generation rates are low, every data packet is simply forwarded to the parent node immediately upon its reception or generation.

B. Output Rate Estimation Module (OREM)

This model is part of the JAM kernel (shown in Figure 4) and is used to estimate the nodal output data rate. Before explaining how it works in detail, we first prove $\mathrm{FAD}(i) \leqslant \mathrm{SAD}(i)$ for any node i. According to the end-to-end delay requirement in Equation (2), node i shall satisfy:

$$\mathrm{SAD}(i) + \sum_{m\in\mathcal{S}_i} (\mathrm{Tr}(m) + \mathrm{FAD}(m)) = D.$$

Similarly, for every child node j of i, we have:

$$\mathrm{SAD}(j) + (\mathrm{FAD}(i) + \mathrm{Tr}(i)) + \sum_{m\in\mathcal{S}_i} (\mathrm{Tr}(m) + \mathrm{FAD}(m)) = D.$$

Hence, it follows that:

$$\mathrm{FAD}(i) \leqslant D - \sum_{m\in\mathcal{S}_i} (\mathrm{Tr}(m) + \mathrm{FAD}(m)) = \mathrm{SAD}(i). \qquad (3)$$

With the four inputs: $\mathrm{FAD}(i)$, $\mathrm{SAD}(i)$, $\sum_{j\in\mathcal{C}(i)} \mu(j)$, and $\lambda(i)$, OREM estimates the output data rate of node i according to the three cases described in Section III-A as follows:

$$\mu(i) = \begin{cases} \frac{1}{\mathrm{FAD}(i)} & : \frac{1}{\sum_{j\in\mathcal{C}(i)} \mu(j)} \leqslant \mathrm{FAD}(i), \\ \sum_{j\in\mathcal{C}(i)} \mu(j) + \frac{1}{\mathrm{SAD}(i)} & : \frac{1}{\sum_{j\in\mathcal{C}(i)} \mu(j)} > \mathrm{FAD}(i) \\ & \quad \text{and } \frac{1}{\lambda(i)} \leqslant \mathrm{SAD}(i), \\ \sum_{j\in\mathcal{C}(i)} \mu(j) + \lambda(i) & : \frac{1}{\sum_{j\in\mathcal{C}(i)} \mu(j)} > \mathrm{FAD}(i) \\ & \quad \text{and } \frac{1}{\lambda(i)} > \mathrm{SAD}(i). \end{cases} \qquad (4)$$

Note that a node's input data rate is simply the sum of the output data rates from all of its children.

C. Lifetime Estimation Module (LEM)

The LEM module is another module in the JAM kernel and is used to estimate the nodal lifetime. Its input consists of $e(i)$, $\mathrm{Tr}(i)$, $\mathrm{Tr}(p)$ (i.e., Tr of node i's parent node p), $\mu(i)$, and $\sum_{j\in\mathcal{C}(i)} \mu(j)$. Its output is the estimated nodal lifetime $L(i)$, which is computed as follows:

$$L(i) = \frac{e(i)}{c(i)}, \qquad (5)$$

where $e(i)$ is the residual energy and $c(i)$ is the energy consumption rate:

$$c(i) = \left(\frac{\mathrm{Tr}(p)}{2} + \tau\right)\mu(i)P + \frac{\phi}{\mathrm{Tr}(i)}P + \tau \sum_{j\in C(i)} \mu(j)P. \qquad (6)$$

Here, P is the power consumption rate when a node's radio is on. To send a data packet, node i waits for $\frac{\mathrm{Tr}(p)}{2}$ time on average for its parent node p to wake up and then spends τ time for the transmission. Hence, it consumes $\left(\frac{\mathrm{Tr}(p)}{2} + \tau\right)\mu(i)P$ power on average for data transmissions. The second term in Equation (6) represents the average amount of power consumed to monitor channel for ϕ time every $\mathrm{Tr}(i)$ interval, while the third term is the average amount of power consumed for data receptions. As radio is the most energy-consuming component, we ignore other energy consumptions such as sensing and computation, which could be easily plugged into Equation (6).

D. Intra-node Cross-layer Collaboration Module (ICCM)

According to Equation (2), SAD value at the source node can be computed once the FAD and Tr values of its ancestor nodes have been determined. As both MAC and aggregation behaviors affect nodal lifetime, their behaviors should be coordinated. Specifically, as each forwarding node i introduces a delay of $D(i) = \text{FAD}(i) + \text{Tr}(i)$, the objective of the ICCM module in the JAM kernel is to determine proper $\text{FAD}(i)$ and $\text{Tr}(i)$ values for a given $D(i)$, so that nodal lifetime $L(i)$ is maximized (denote as $L^*(i)$). To achieve this goal, there are two cases to consider:

Case I: FAD$(i) < \dfrac{1}{\sum_{j\in\mathcal{C}(i)}\mu(j)}$. Depending on the relation between $\text{SAD}(i)$ and $\lambda(i)$, $\mu(i)$ equates the second or third case of Equation (4). In these two cases, $\mu(i)$ is not affected by $\text{FAD}(i)$ and hence nodal lifetime is only affected by $\text{Tr}(i)$ according to Equations (5) and (6). As the energy consumption rate ($c(i)$) in Equation (6) is a decreasing function of $\text{Tr}(i)$, optimal nodal lifetime is achieved when $D(i)$ is fully allocated to $\text{Tr}(i)$ and hence $\text{FAD}(i) = 0$:

$$L^*\left(i \,\middle|\, \text{FAD}(i) < \frac{1}{\sum_{j\in\mathcal{C}(i)}\mu(j)}\right) = L(i \mid \text{FAD}(i) = 0). \quad (7)$$

Case II: FAD$(i) \geqslant \dfrac{1}{\sum_{j\in\mathcal{C}(i)}\mu(j)}$. In this case, the output data rate is $\mu(i) = \dfrac{1}{\text{FAD}(i)}$ according to Equation (4). In order to achieve optimal nodal lifetime in this case, we need to minimize the following term according to Equation (6):

$$\left(\frac{\text{Tr}(p)}{2} + \tau\right)\frac{1}{\text{FAD}(i)} + \frac{\phi}{\text{Tr}(i)}, \quad (8)$$

which we denote as $f(\text{FAD}(i))$. To ease the presentation, we use α to denote $\frac{\text{Tr}(p)}{2} + \tau$. Plugging in $\text{Tr}(i) = D(i) - \text{FAD}(i)$, we can rewrite $f(\text{FAD}(i))$ as:

$$f(\text{FAD}(i)) = \frac{\alpha}{\text{FAD}(i)} + \frac{\phi}{D(i) - \text{FAD}(i)}. \quad (9)$$

By solving $f(\text{FAD}(i))' = 0$, we have:

$$L^*\left(i \,\middle|\, \text{FAD}(i) \geqslant \frac{1}{\sum_{j\in\mathcal{C}(i)}\mu(j)}\right) = \begin{cases} L\left(i \,\middle|\, \text{FAD}(i) = \frac{D(i)}{1+\sqrt{\frac{\phi}{\alpha}}}\right) : \text{when } \frac{1}{\sum_{j\in\mathcal{C}(i)}\mu(j)} \leqslant \frac{D(i)}{1+\sqrt{\frac{\phi}{\alpha}}} \leqslant D(i), \\ \max\left\{L\left(i \mid \text{FAD}(i) = D(i)\right), L\left(i \,\middle|\, \text{FAD}(i) = \frac{1}{\sum_{j\in\mathcal{C}(i)}\mu(j)}\right)\right\} \\ \qquad : \text{otherwise.} \end{cases} \quad (10)$$

To summarize, for a given $D(i)$, $L^*(i)$ and the corresponding optimal $\text{FAD}(i)$ and $\text{Tr}(i)$ can be computed by:

$$L^*(i) = \max\{\text{Equation (7)}, \text{Equation (10)}\}, \quad (11)$$

$$\text{FAD}^*(i) = \arg_{\text{FAD}(i)}\max\{L^*(i)\}, \quad (12)$$

$$\text{Tr}^*(i) = D(i) - \text{FAD}^*(i). \quad (13)$$

E. JAM Kernel

As the core of the JAM scheme, the JAM kernel of node i is triggered to execute periodically every W time, or on demand whenever its parent node p changes $\text{Tr}(p)$ or $\text{FAD}(p)$ and consequently the delay $D(p)$. We use $\theta(p)$ to denote the change in $D(p)$. The JAM kernel is executed to decide:

- how the extra delay $\theta(p)$ introduced at parent node p shall be absorbed by node i (in the amount of $\theta(p) - \Delta$ via updating $D(i)$ to $D'(i) = D(i) - (\theta(p) - \Delta)$) and its child nodes (in the amount of Δ via updating $D(j)$ to $D'(j) = D(j) - \Delta$); and
- how node i shall split $D'(i)$ into $\text{Tr}'(i)$ and $\text{FAD}'(i)$,

so that the minimal nodal lifetime within node i's neighborhood can be increased. Detailed working procedure of the JAM kernel is illustrated in Figure 4 and explained below, while the effect of W will be discussed in Section III-F3.

The JAM kernel of node i takes the following inputs: $\text{Tr}(p)$ and $\theta(p)$ from parent node p, and $\lambda(j)$, $\text{SAD}(j)$, $D(j)$, and $\sum_{k\in C(j)}\mu(k)$ from each child node j. Based on these inputs, it checks the candidate values of Δ from an ordered sequence $\Omega = \langle 0, \delta, -\delta, 2\delta, -2\delta, \cdots\rangle$ one by one till the first feasible Δ value has been found to increase the minimal nodal lifetime within node i's neighborhood, or till all values in Ω have been exhausted. Here, δ is a system parameter. To ensure $D'(i) \geqslant 0$, the smallest element in Ω is set to $-\left\lfloor \frac{D(i)-\theta(p)}{\delta} \right\rfloor \delta$. Similarly, to ensure $D'(j) \geqslant 0$, the largest element in Ω is set to $\left\lfloor \frac{\min_{j\in C(i)} D(j)}{\delta} \right\rfloor \delta$. For each candidate value of Δ, the JAM kernel executes the following:

- Iteratively, $\text{FAD}'(i)$ takes a value from range $[0, D'(i)]$ with a small step ε. Here, ε is a system parameter. For each $\text{FAD}'(i)$ value, $\text{Tr}'(i) = D'(i) - \text{FAD}'(i)$.
- Based on $\text{FAD}'(i)$, $\text{Tr}'(i)$, and the inputs from each child node j, the ICCM module is called to compute the maximum lifetime that node j can achieve, denoted as $L^*(j)$, and the corresponding $\text{FAD}^*(j)$, according to Equations (11) and (12), respectively.
- $\text{FAD}^*(j)$ is fed into the OREM module to compute the output data rate $\mu^*(j)$ of node j. With $\mu^*(j)$ from all child nodes, the output data rate $\mu'(i)$ of node i is also computed using the OREM module.
- Then, the LEM module is called to estimate the nodal lifetime $L'(i)$. If $\min\{L'(i), L^*(j), \forall j \in \mathcal{C}_i\}$ is larger than the highest L^Δ that has been found so far, L^Δ is updated, and the corresponding $\text{FAD}'(i)$ and $\text{Tr}'(i)$ values are recorded as $\text{FAD}^\Delta(i)$ and $\text{Tr}^\Delta(i)$.
- When $\text{FAD}'(i)$ reaches the boundary condition, i.e., $\text{FAD}'(i) > D'(i)$, if the best achievable L^Δ improves the minimal nodal lifetime within node i's neighborhood, i.e., $L^\Delta > \min\{L(i), L(j), \forall j \in \mathcal{C}_i\}$, $\text{FAD}^\Delta(i)$ and $\text{Tr}^\Delta(i)$ are output, $\theta(i) = \Delta - \theta(p)$ is appended to ACK packets

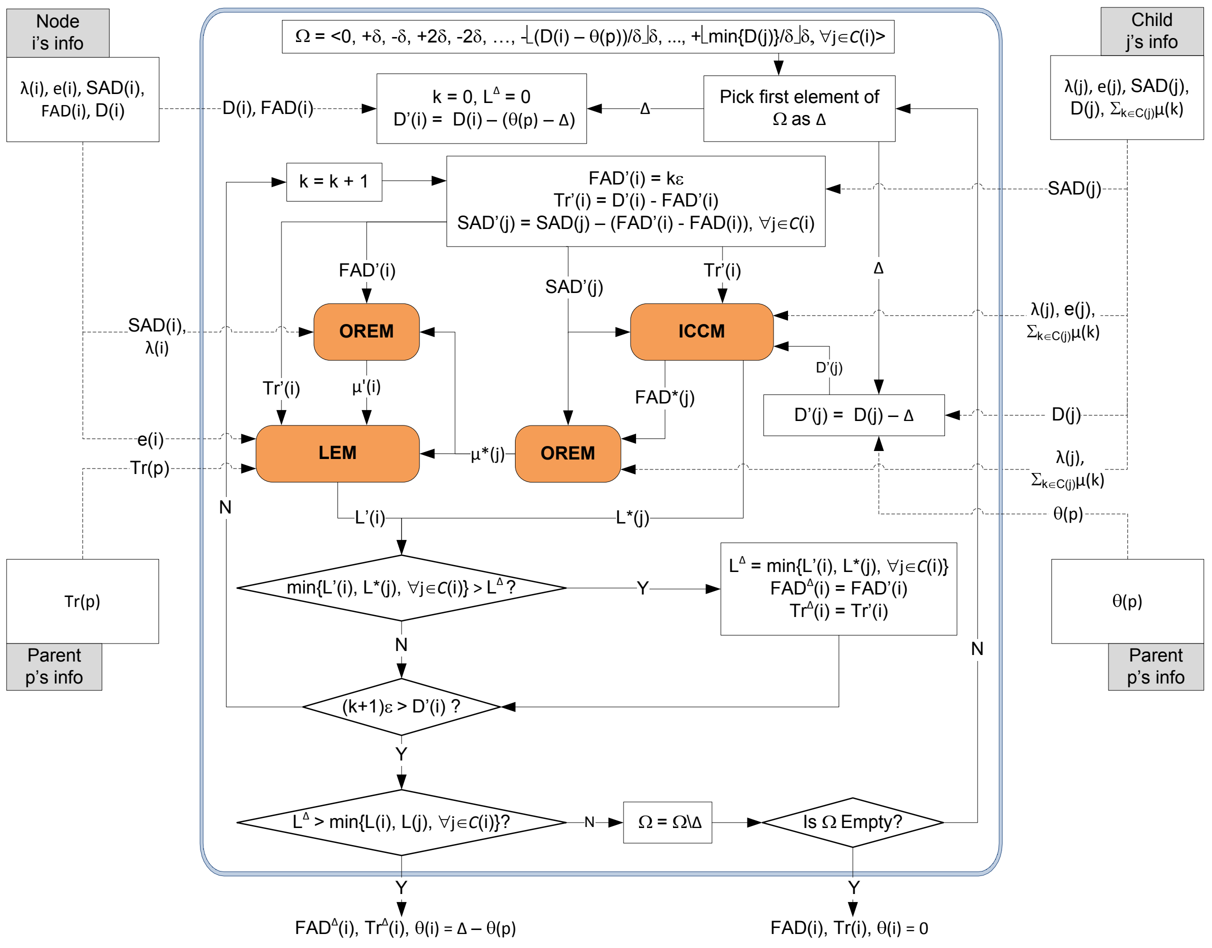


Fig. 4. JAM kernel of node i. With parent, children, and self information, node i decides a proper Δ value so that $L^{\Delta} > \max\min\{L(i), L(j), \forall j \in \mathcal{C}_i\}$. The figure only shows the interactions between ICCM, OREM, and LEM modules for child node j of i. In the actual JAM implementation, these interactions are executed for every child node of i.

to all child nodes, and search completes; otherwise, the next candidate Δ value will be tested.

Note that, if all the candidate values of Δ in Ω have been checked but none of them improves the minimal nodal lifetime within node i's neighborhood, FAD(i) and Tr(i) remain unchanged and $\theta(i) = 0$.

In our implementation of the JAM scheme, we set $\delta = 0.5$ s and $\varepsilon = 0.1$ s. This means that, when the average per-hop delay is 10 s, there is a total of 40 candidate values for Δ, and 100 candidate values for FAD$'$. Therefore, in the worst-case scenario, the JAM kernel needs to iterate 4000 times to complete the execution, which is acceptable in practice. To further reduce the complexity, we adopt a two-step heuristic as follows. We first use a larger ε value (i.e., 1 s) to conduct the initial search; when a feasible FAD$'$ has been found, we use a smaller ε value (i.e., 0.1 s) to refine the search around it. With this simple heuristic, the search space for FAD$'$ is reduced to 30 and the total number of iterations in the worst-case scenario is reduced to 1200 in the above example.

F. Other Design Considerations

1) JAM Initialization: After the data collection tree has been established (i.e., the routing table of each node becomes stable), each node i needs to decide its initial FAD(i) and Tr(i) values. In JAM, all nodes start with a default Tr value and the rest of the end-to-end delay bound is evenly distributed to nodes along the source-to-sink path. For this purpose, each node i periodically measures the most updated (i) accumulative delays from its parent node to the sink ($D(i \to s)$), i.e., the second term in the end-to-end delay requirement in Equation (2), and (ii) hop count to its farthest descendant ($h(i)$).

We assume that the sink's radio is always on and its wakeup interval is Tr$(s) = 0$ s. As the sink is the end of data delivering paths, FAD$(s) = 0$ s. $D(s \to s) = 0$ s is then broadcast to all of its children. Upon receiving $D(i \to s)$, each child node j acts as follows:

- If node j is not a leaf node, it sets $D(j) = \frac{D-D(i\to s)}{h(j)+1}$, FAD$(j) = D(j) - $ Tr(j), SAD$(j) = D - D(i \to s)$, and sends $D(j \to s) = D(i \to s) + D(j)$ to all of its children. In the example shown in Figure 5, node 1 sets

$D(1) = 15/3 = 5$ s, $\text{FAD}(1) = 4$ s, $\text{SAD}(1) = 15$ s, and sends $D(1 \to s) = 5$ s to nodes 2 and 3.

- If node j is a leaf node, it simply sets $\text{SAD}(j) = D - D(i \to s)$. In the example shown in Figure 5, leaf node 4 sets $\text{SAD}(4) = 15 - 10 = 5$ s.

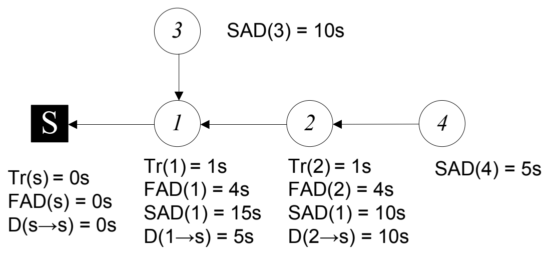


Fig. 5. JAM initialization example where the default Tr is 1 s and D is 15 s.

2) Handling of Packet Loss: In practice, data or ACK packets may get lost due to collision, interference, or deteriorated channel condition, and the sensor node may need to retransmit multiple times before the data packet can be delivered successfully. This issue has been dealt with in JAM by replacing Tr with ETX · Tr in the end-to-end delay requirement (i.e., Equation (2) in Section II-B), where ETX is the expected number of transmission attempts to deliver a data packet successfully over one hop:

$$\text{SAD}(i) + \sum_{m \in \mathcal{S}_i} (\text{ETX}(m^c, m)\text{Tr}(m) + \text{FAD}(m)) = D. \quad (14)$$

Here, m^c is the child node of m along the path from i to the sink. Similarly, Equations (6) and (10) have also been updated to include the ETX information. Note that measurement of ETX is readily available in many routing protocols such as CTP [17], and thus not an extra overhead.

3) Adaptive Adjustment Interval W: The JAM kernel is triggered to execute periodically every W interval, which poses a tradeoff. A small interval, i.e., frequent adjustment, makes JAM more responsive to the changes in the network and allows it to approach a balanced nodal lifetime distribution in the network sooner, but uses more computational resources. A large interval, on the other hand, uses less computational resources but may let nodes stay in suboptimal states for a longer time.

We adopt an adaptive approach to adjust W dynamically (between $W_{\min}$ and $W_{\max}$) to the network condition. Specifically, if the current state is already the best that can be found, i.e., no Δ in Ω improves the minimal nodal lifetime within the neighborhood as described in Section III-E, W is doubled till reaching $W_{\max}$. Otherwise, W is reset to $W_{\min}$. This way, when the nodal lifetime distribution is heterogeneous or the network configuration changes, JAM allows the network to reconverge quickly to the balanced nodal lifetime distribution; otherwise, it decreases the adjustment frequency exponentially to save the control overhead in the long term. In the JAM implementation, we set $W_{\min}$ and $W_{\max}$ conservatively to 1 minute and 16 minutes, respectively.

G. JAM Implementation

1) Software Component: We have implemented JAM in TinyOS 2.1.0. As shown in Figure 6, the shaded parts illustrate the core components of JAM in the software architecture: (i) the aggregation component that sits between application and routing layers, and (ii) the MAC component that is designed based on RI-MAC [19]. CTP [17] is adopted as the routing layer protocol to set up the data collection tree.

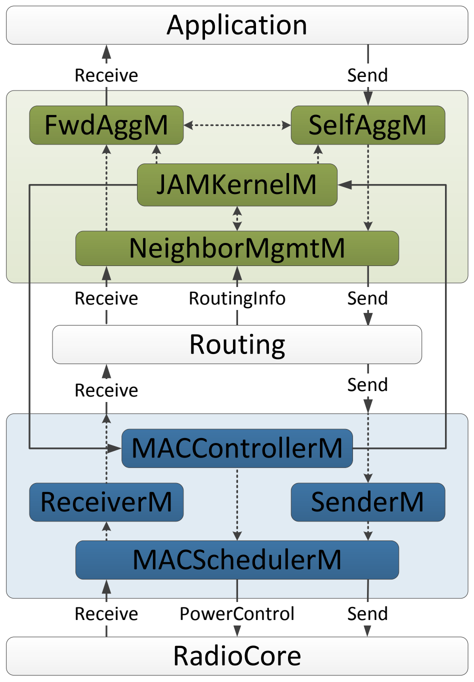


Fig. 6. Implementation of JAM in TinyOS.

When data packets from a child node j arrives at node i's MAC layer, the piggybacked control information will be extracted, passed to, and processed at the *JAMKernelM* module, which implements the JAM kernel. The *NeighborMgmtM* module manages all senders' information while the *FwdAggM* and *SelfAggM* modules maintain the aggregation timers. After deciding the new $\text{Tr}(i)$ and $\text{FAD}(i)$ values, the JAMKernelM module notifies the *MACControllerM* module to adopt the latest wakeup interval, and the FwdAggM and SelfAggM modules to adjust the aggregation timers. JAMKernelM also informs MACControllerM of $\theta(i)$ and piggybacks it in the ACKs to child nodes.

2) Hardware Component: Among all the control information piggybacked in data packets, nodal residual energy is an important piece. As TelosB motes do not provide an interface to measure nodal residual energy, we have designed and fabricated a TelosB power meter kit as shown in Figure 7. This kit measures the nodal power consumption rate, based on which we can calculate the total energy consumed so far. The nodal residual energy is then the difference between the battery energy capacity [20] and the consumed energy.

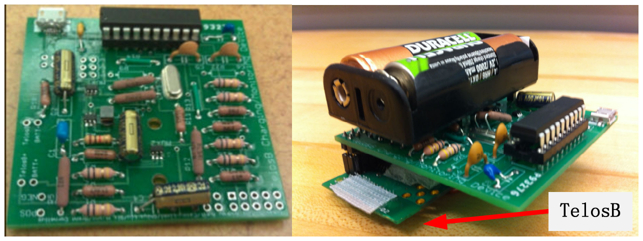


Fig. 7. TelosB power meter kit used in JAM.

IV. Performance Evaluation

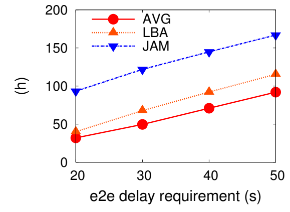
NS-2 based simulations and TinyOS testbed experiments have been conducted to evaluate the JAM performance in terms of *network lifetime*, *average nodal power consumption*, and *end-to-end delivery delay*. We compare JAM with a naive scheme called AVG which simply sets FAD and Tr values according to Section III-F1 without runtime adjustment, and LBA [7] which is a state-of-the-art lifetime-balancing aggregation protocol under delay constraint.

A. NS-2 Simulations

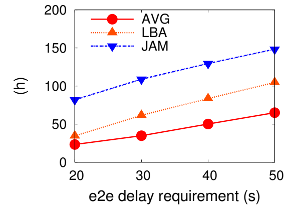
In the simulation, nodes are randomly deployed in a 500 m $\times$ 500 m area and the sink is located at the center of the area. The evaluation results are averaged over 30 different random topologies. We vary the end-to-end delay requirement, initial nodal energy distribution, and the network density in the simulation. The default initial nodal energy is 4500 Joules. The maximal communication range is 70 meters and the power consumption is 69 mW when radio is on. All nodes are sources and the data generation rate is a random value between 0.1 and 1 packet per second. The packet size is 128 bytes. In both simulations and testbed experiments, $\delta = 0.5$ s, $\varepsilon = 0.1$ s, $W_{\min} = 1$ minute, and $W_{\max} = 16$ minutes.

1) Performance under Different End-to-End Delay Requirements: Figure 8 compares the performances of all the evaluated schemes with the end-to-end delay requirement varying between 20 s and 50 s. Left column of Figure 8 shows evaluation results when all nodes start with 4500 J energy while right column shows results when the initial nodal energy is a random value between $4500 * (1 \pm 40\%)$ J.

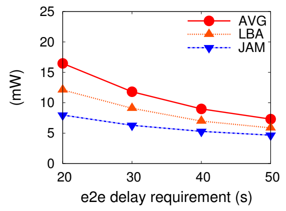
Different from LBA which does not improve the network lifetime much when the initial nodal energy distribution is homogeneous as shown in Figure 8(a) (similar finding has also been observed in [7]), JAM consistently improves the network lifetime in both homogeneous and heterogeneous initial energy settings under all end-to-end delay requirements. Specifically, when all nodes start with the same amount of energy, compared with LBA, JAM improves the network lifetime by 158% and 59% when the end-to-end delay requirement is 20 s and 50 s, respectively. When nodes start with different energy, the improvement ratio is 165% and 50% when the requirement is 20 s and 50 s, respectively.



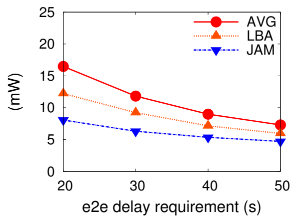
(a) Network lifetime.



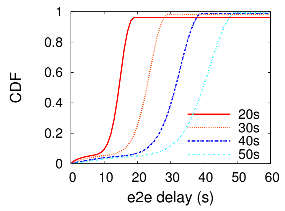
(b) Network lifetime.



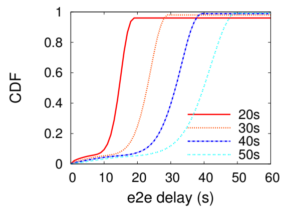
(c) Average nodal power consumption.



(d) Average nodal power consumption.



(e) End-to-end delay in JAM.



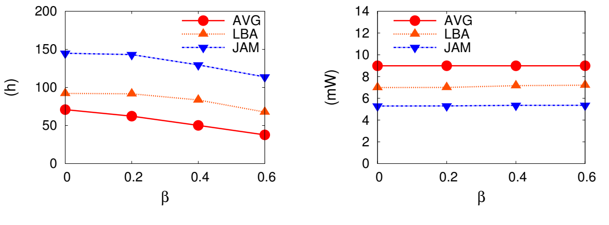
(f) End-to-end delay in JAM.

Fig. 8. Performance comparison under different end-to-end delay requirements. Number of nodes in the network is 60, and initial Tr is 1.5 s. Figures on the left column are results when initial nodal energy is 4500 J while figures on the right are results when initial nodal energy is a random value between $4500 * (1 \pm 40\%)$ J.

JAM yields more significant network lifetime improvement when the end-to-end delay requirement is more stringent because smaller delay bound means more limited FAD values for each node along source-to-sink paths, which may result in insufficient data suppression and hence heavier network traffic. As MAC behaviors in AVG and LBA are not jointly adjusted with aggregation behaviors, the communication overhead could be expensive. Hence, the heavy traffic could soon deplete the nodal energy and constrain the network lifetime. In comparison, JAM yields a much lower average nodal energy consumption due to its joint MAC and aggregation design, as shown in Figures 8(c) and 8(d).

Figures 8(e) and 8(f) plot the CDF (Cumulative Distribution Function) of the end-to-end delay for the JAM scheme. Results show that all the end-to-end delay requirements are well-satisfied. Similar to RI-MAC, JAM drops a data packet after a certain number (4 in our implementation) of failed transmission attempts. Therefore, when the end-to-end delay requirement is relatively small, i.e., 20 s, the heavy traffic deteriorates channel contentions and about 4% of data packets are dropped, while the data delivery ratio approaches 100% when the end-to-end delay requirement increases to 50 s. Due to space limitation, we omit the CDF results of the end-to-end delay for other evaluation scenarios, where JAM exhibits a similar performance as the ones shown in Figures 8(e) and 8(f).

2) Performance under Different Initial Energy: We now evaluate the impact of the initial nodal energy heterogeneity level (denoted as β). It is defined as follows. With a heterogeneity level of β, the initial nodal energy is a random value between $4500 * (1 \pm \beta)$ Joules. Results are shown in Figure 9.



(a) Network lifetime. (b) Average nodal power consumption.

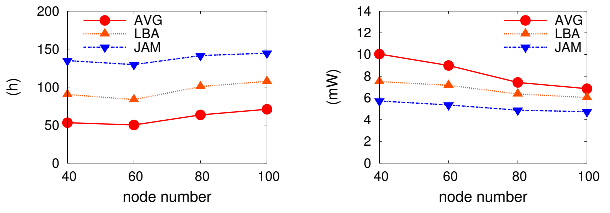
Fig. 9. Performance comparison under different initial nodal energy heterogeneity levels. Number of nodes in the network is 60, end-to-end delay requirement is 40 s, and initial Tr is 1.5 s.

With AVG, as Tr and FAD are not adjusted at runtime, network lifetime is bounded by the initial minimal-lifetime node. As the initial energy distribution becomes more and more heterogeneous (i.e., β increases), network lifetime achieved by AVG drops quickly. On the other hand, both LBA and JAM redistribute the end-to-end delay so that energy bottleneck nodes could be saved by other high-energy nodes along the same route and hence yield a significantly longer network lifetime. As a larger β demands more re-distribution efforts, the network lifetime drops as well with LBA and JAM. However, as all the nodes along the route absorb this effect collaboratively, the network lifetime drops less quickly. For example, network lifetime drops 21% with JAM when β increases from 0% to 60%, in comparison to 46% with AVG. Finally, thanks to the joint MAC and aggregation design, JAM yields a consistently 70% longer network lifetime than LBA under different β values.

3) Performance under Different Densities: As shown in Figure 10, JAM always yields a significantly longer network lifetime than other schemes, regardless of the network density. It is interesting to see (in Figure 10(b)) that the average nodal power consumption decreases as the network density increases. This is because, with a higher network density and consequently a higher node degree on average, a source node may reach the sink via a shorter path. This means less nodes are involved in the path and the overall energy consumption is thus reduced.

B. Testbed Experiments

We set up a testbed network of 32 TelosB motes, forming a fixed tree topology shown in Figure 11. All nodes are sources, and the data generation interval is uniformly distributed between 0.8 and 1.2 s. The default Tr value is 1.5 s, and the end-to-end delay requirement varies between 20 s and 40 s.



(a) Network lifetime. (b) Average nodal power consumption.

Fig. 10. Performance comparison under different network densities. End-to-end delay requirement is 40 s, initial Tr is 1.5 s, and initial nodal energy level is a random value between $4500 * (1 \pm 40\%)$ J.

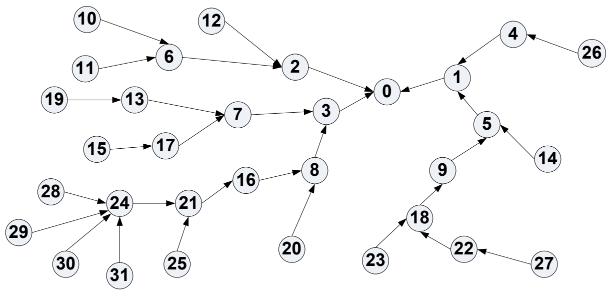
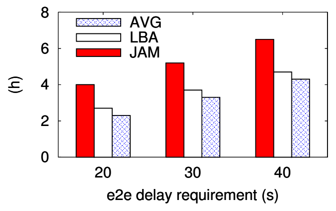


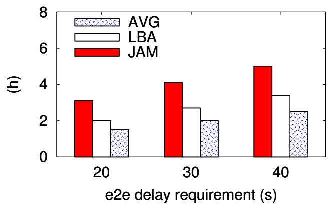
Fig. 11. Network topology in testbed experiments.

In order to complete the experiments within a reasonable amount of time, *we study how fast a node consumes a small designated amount of energy, and evaluate its nodal lifetime as the time period during which the designated amount of energy is consumed.* We run two sets of experiments. In the first set, the initial available energy distribution is uniform and all nodes have 450 Joules; results are shown in the left column of Figure 12. In the second set, the initial available energy at an individual node is a random value between 250 and 450 Joules; results are shown in the right column of Figure 12. Overall, experiment results confirm our observations in the simulation study.

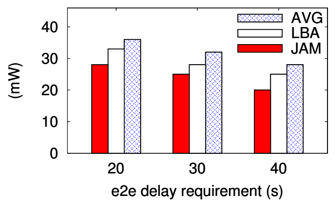
To illustrate how JAM adaptively tunes MAC and aggregation operational parameters to balance nodal lifetime within the neighborhood, we plot in Figure 13 the changing traces of the operational parameters of the nodes along the path $24 \rightarrow 21 \rightarrow 16$. We observe that during time period [0, 0.2] h, as shown in Figure 13(a), node 21 has a lower nodal lifetime than nodes 16 and 24. To improve node 21's lifetime, (i) node 16, as node 21's parent, decreases its Tr to save node 21's energy cost on idly waiting during transmissions; (ii) node 21 increases its FAD to reduce the amount of outgoing traffic; and (iii) node 24 decreases its Tr to ensure the delay requirement is satisfied. As a result, their nodal lifetimes are balanced gradually. During time period [0.2, 0.5] h, as all three nodes have reached a similar level of nodal lifetime, their operational parameters are relatively stabilized.



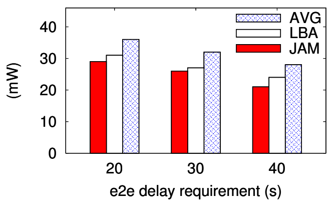
(a) Network lifetime.



(b) Network lifetime.

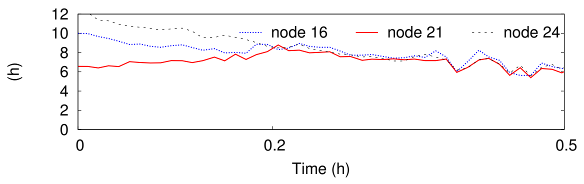


(c) Average nodal power consumption.

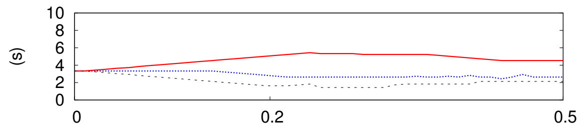


(d) Average nodal power consumption.

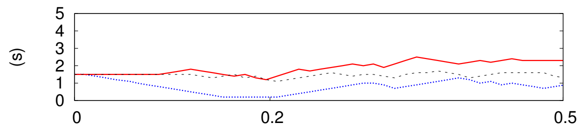
Fig. 12. Experiment results under different end-to-end delay requirements. Figures on the left column are results when initial available energy is 450 J while figures on the right are results when initial available energy is a random value between 250 and 450 J.



(a) Nodal lifetime.



(b) Nodal FAD values.



(c) Nodal Tr values.

Fig. 13. Changing traces of nodal lifetime, FAD, and Tr values along the path 24 $\rightarrow$ 21 $\rightarrow$ 16.

V. Conclusions and Future Work

In this paper, we present a new holistic approach called JAM to prolong the sensor network lifetime. Different from the existing works which adapt either MAC or aggregation behavior alone, JAM integrates the advantages of both approaches and therefore can extend the network lifetime more effectively. In addition, JAM can also meet the end-to-end delay requirement specified by the applications.

JAM was designed to work with static data collection trees. In the future, we will improve the JAM design by taking into account the routing strategy. As routing behavior also affects the network traffic distribution, a joint MAC, routing, and aggregation design may prolong the network lifetime even further.

Acknowledgment

This work is supported partly by the NSF under grants CNS-0831874 and EECS-1128312, and by the ONR under grant N00014-09-1-0748.

References

[1] W. Wang, V. Srinivasan, and K. C. Chua, "Using Mobile Relays to Prolong the Lifetime of Wireless Sensor Networks," in *MobiCom*, 2005.
[2] J. Chang and L. Tassiulas, "Energy Conserving Routing in Wireless Ad-hoc Networks," in *INFOCOM*, 2000.
[3] ——, "Maximum Lifetime Routing in Wireless Sensor Networks," *IEEE/ACM Trans. Netw.*, 2004.
[4] R. Braynard, A. Silberstein, and C. Ellis, "Extending Network Lifetime Using an Automatically Tuned Energy-Aware MAC Protocol," in *EWSN*, 2006.
[5] Z. Li, M. Li, and Y. Liu, "Towards Energy-Fairness in Asynchronous Duty-Cycling Sensor Networks," in *INFOCOM*, 2012.
[6] A. Meier, M. Woehrle, M. Zimmerling, and L. Thiele, "ZeroCal: Automatic MAC Protocol Calibration," in *DCOSS*, 2010.
[7] Z. Li, Y. Peng, D. Qiao, and W. Zhang, "LBA: Lifetime Balanced Data Aggregation in Low Duty Cycle Sensor Networks," in *INFOCOM*, 2012.
[8] L. Becchetti, A. Marchetti-Spaccamela, A. Vitaletti, P. Korteweg, M. Skutella, and L. Stougie, "Latency-Constrained Aggregation in Sensor Networks," *ACM Trans. Algorithms*, 2009.
[9] 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.
[10] 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.
[11] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele, "pTunes: Runtime Parameter Adaptation for Low-Power MAC Protocols," in *IPSN*, 2012.
[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] Z. Ye, A. Abouzeid, and J. Ai, "Optimal Policies for Distributed Data Aggregation in Wireless Sensor Networks," in *INFOCOM*, 2007.
[14] J. Zhang, X. Jia, and G. Xing, "Real-time Data Aggregation in Contention-based Wireless Sensor Networks," *ACM Trans. Sen. Netw.*, 2010.
[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] L. Xiang, J. Luo, and A. V. Vasilakos, "Compressed Data Aggregation for Energy Efficient Wireless Sensor Networks," in *SECON*, 2011.
[17] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *SenSys*, 2009.
[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] 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.
[20] Online Link, http://www.allaboutbatteries.com/Energy-tables.html.