Bounding Communication Delay in Energy Harvesting Sensor Networks

Yu Gu and Tian He
{yugu,tianhe}@cs.umn.edu
Department of Computer Science and Engineering, University of Minnesota

***Abstract*—In energy-harvesting sensor networks, limited ambient energy from environment necessitates sensor nodes to operate at a low-duty-cycle, i.e., they communicate briefly and stay asleep most of time. Such low-duty-cycle operation leads to orders of magnitude longer communication delays in comparison with traditional always-active networks, imposing a new challenge in many time-sensitive sensor network applications (e.g., tracking and alert).**

In this paper, we introduce novel solutions for bounding sink-to-node communications in energy-harvesting sensor networks. We first present an optimal solution for the sink-to-one case and its distributed implementation. For the sink-to-many case, we theoretically prove its NP-Hardness and inapproximability property, followed by an efficient heuristic solution. We have evaluated our design with both extensive simulation and a TinyOS/Mote based implementation. Compared with an improved version of a state-of-the-art design, our delay maintenance design effectively provides E2E delay guarantees while consuming as much as 60% less energy.

I. Introduction

With the increasing application of cyber-physical systems, wireless sensor networks have emerged as a key technology for many long-term applications [1]–[4]. To support those long-term applications, energy-harvesting sensor networks, which extract energy from their surrounding environment, have become an increasing popular foundation for building those long-term applications. Recently, several pioneer works have built prototypes [5], [6] to verify and demonstrate the feasibility of powering sensors by exploiting ambient energy resources. Due to the varying environment conditions, one major characteristic of energy-harvesting networks is energy supply to individual nodes varies significantly over time [7], [8]. For nodes in energy-harvesting sensor networks, their activities therefore have to be adjusted continuously with varying energy supply. In addition, for the energy harvested from surrounding environment, usually it is not enough to continuously power sensor nodes in the network [8]–[10]. Consequently, nodes in those networks have to operate in low-duty-cycle, which means each node in the network has to activate briefly and stay in the dormant state most of the time. Effective data communication in such energy-dynamic and low-duty-cycle networks, therefore poses a new challenge over traditional energy-static sensor networks.

For many sensor network applications, sink node needs to actively communicate with other nodes in the network in order to perform operations such as disseminating commands to sensor nodes, configuring sensor setups, querying sensor states and so on [11]–[13]. For many of those operations, there is usually also a delay bound associated with them and require the messages sent out by the sink node to be received at destined receivers within a designated time bound [14]–[16]. However, although many existing delay guarantee solutions are able to effectively bound communication delays in traditional network settings, none of them consider the impact of energy-dynamics and low-duty-cycle operation, making them unsuitable for energy-harvesting networks [17]–[19].

To tackle both the communication challenge in energy-harvesting sensor networks and E2E delay requirement for sink-to-node communications, this work introduces a generic and efficient solution that provides E2E delay guarantee for sink-to-node communications in energy-harvesting sensor networks. Its main design objective is to *consume a minimum amount of energy while satisfying E2E delay bound specified by application requirements*. Specifically, the major intellectual contributions of this work are as follows:

- To our best knowledge, this is the first generic work to study distributed solutions for delay maintenance in energy-harvesting sensor networks.
- We introduce an energy-optimal solution for bounding E2E delay for sink-to-one communication. The whole solution is distributed and only requires neighbor information.
- For bounding E2E delays for sink-to-many communications, we prove the NP-Hardness and inapproximability property by deducting from planar 3-Satisfiability problem and propose an efficient heuristic solution to solve this problem.
- We have extensively evaluated our design with both simulation and a running test-bed to verify the effectiveness and energy-efficiency of our delay maintenance design in theory and practice.

The rest of this paper is organized as follows: Section II discusses the related work. Section III describes the need and challenges for bounding sink-to-node communication in energy-harvesting sensor networks. Section IV defines the network model and assumptions. Section V introduces our main design, followed by its evaluation in Section VI and Section VII. Section VIII concludes the paper.

II. Related Work

To overcome the lifetime limitation of battery-powered sensor networks, research on energy harvesting for sensor networks has been very active in recent years. Numerous running prototypes have been built to collect ambient energy from the environment to power the sensor nodes [5], [6]. However, for current research in energy-harvesting sensor networks, the focus is still mainly on the hardware design and power management to ensure continuous operation of sensors in the network [7]–[10]. There is few prior works investigate how changing node duty cycles affects communication in energy-harvesting networks. Most recently, ESC protocol introduces a transparent middleware for minimizing network wide delay with varying energy budget over time in energy-harvesting sensor networks [20].

For many sensor networks, timely data communication under low node duty-cycle is a key factor for them to interact with the physical world effectively. Numerous solutions have been designed to provide timely data communications in sensor networks. MMSPEED provides a multi-path and multi-speed routing protocol for probabilistic QoS guarantee in reliability and timeliness domains [18]. Pipelined Tone Wakeup (PTW) is introduced to achieve the balance between energy saving and E2E delay [19]. Lu et al. show how to minimize the communication delay given a duty-cycle budget for nodes in the network [21]. Gu et al. introduce a dynamic switch-based forwarding to minimize the impact of sleep latency in low duty-cycle networks [22]. Su et al. propose both on-demand and proactive routing algorithms for intermittently connected networks due to duty cycling [23]. Guo et al. introduce an opportunistic flooding for low-duty-cycle networks [24]. More recently, Gu et al. propose a centralized spatiotemporal delay control scheme for battery-powered low-duty-cycle networks [25].

Different from earlier works, which either focus on static batter-powered networks or passively minimizing delay under energy constraints, in this work we introduce the first proactive generic delay maintenance algorithm with minimum energy consumption in energy-harvesting networks.

III. Motivation

The motivation of this work is originated from our empirical experience of designing and deploying energy-harvesting sensor networks [8]. In those networks, the energy supply usually is very dynamic, which can vary significantly within a day and across days [7], [8]. In addition, for the energy harvested from surrounding environment, usually it is not enough to continuously power sensors in the network [8]–[10]. Consequently, nodes in those networks have to operate in low-duty-cycle, which means each node in the network has to activate briefly and stay in the dormant state most of the time. Effective data communication in such energy-dynamic and low-duty-cycle networks, therefore poses a new challenge over traditional energy-static sensor networks.

For many sensor network applications, besides passively receive data messages from other nodes in the network, a sink node usually needs to actively communicate with all other nodes in the network, such as sending commands and queries [13], [26]. For those commands and queries, a communication delay bound is usually associated with them [27]. However, for many power management protocols that are geared to minimize energy consumption while satisfying quality of service requirements (e.g., quality of sensing [28], quality of tracking [29], [30]), the node working schedules generated by them cannot guarantee that the messages sent out by the sink node can reach its destinations within a designated time frame. For example, Figure 1 shows after node working schedule assignment by two state-of-the-art sensing protocols [31], [32], the percentage of nodes in the network with E2E delays from the sink above the specified delay bound, where the network has an average of 2% duty cycle and 10 neighboring nodes per node. From Figure 1, we can see until the delay bound is fairly large (above 250 units of time for the optimal sleep scheduling [31] and above 350 units of time for the virtual patrol [32]), there are always some nodes can not receive the message from the sink within the specified delay bound. Consequently, to provide guarantee on the E2E delay from the sink node to all other nodes in the network, further actions need to be taken on top of those existing power management protocols.

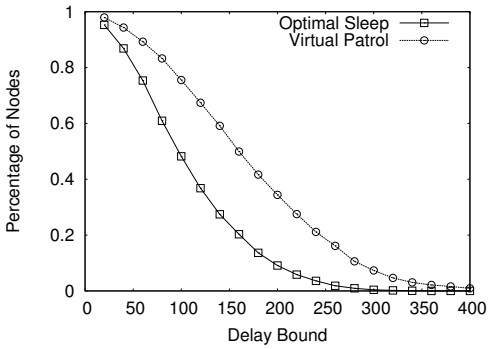


Fig. 1. Percentage of Nodes Beyond Delay Bound vs. Delay Bound

In summary, on observing the energy dynamics, low-duty-cycle operation of energy-harvesting sensor networks, and the lack of consideration for bounding communication delay in existing power management protocols, we aim at providing a generic and efficient delay bound maintenance design for those emerging networks.

IV. Preliminaries

This section defines the network model and assumptions related to our E2E delay maintenance design for energy-harvesting sensor networks.

A. Network Model

Suppose there is an energy-harvesting sensor network of N sensor nodes. At any point of time t, a node is either in an active or a dormant state. An active node is able to sense its surrounding environment, transmit a packet or receive a packet. A dormant node turns all its function modules off except a timer to wake itself up. All nodes in the network

have their own respective working schedules. Those working schedules are shared with neighboring nodes and are normally asynchronous in order to reduce information redundancy among neighboring nodes [31], [32]. For a dormant node, it will wake up if (i) it is scheduled to switch to the active state, or (ii) it has some packets to send to a neighboring node that is active at that time. In other words, *a node can wake-up and transmit a packet when its receiver is in the active state, but can only receive a packet when it is self in the active state*.

For a node, since its neighbors consistently transit between active and dormant states, the connectivity between a pair of nodes therefore becomes time-dependent. Formally, for a given time t, we can denote the network topology as $G(t) = (V, E(t))$, where V is the set of N nodes in the network and $E(t)$ is the set of directed edges at time t. A directed edge $e_{ij}(t)$ belongs to $E(t)$ if node i and node j are within each other's communication range, and node j is in the active state so that it is able to receive packets from node i.

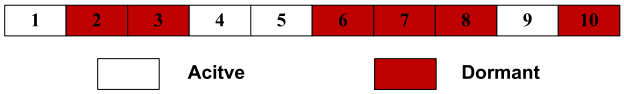


Fig. 2. Example of Working Schedule

The node working schedules, managed by various power management protocols, usually are periodic [23], [31]–[35]. Consequently, let the duration of a period be T, we can divide it into a sequence of time instances with length τ, where τ is the finest granularity of time duration in a given sensor network system. For a node, its working schedule therefore can be represented by a set of time instances that the node is in the active state within period time T. Formally, let Γ_i denote the working schedule of node i, we can have $\Gamma_i = \{t_1, t_2, t_3, ..., t_n\}$, where t_j is the time instance that node i is active and we call t_j an *active instance* for node j. For example, Figure 2 shows active/dormant activities of a node i with a period duration time 10τ and working schedule $\Gamma_i = \{1, 4, 5, 9\}$.

In always-active networks, a sender can immediately send its packets to a one-hop receiver once the sender has a packet ready to be sent. The one-hop packet delivery latency, which includes processing delay, transmission delay and propagation delay, normally in the order of milliseconds. However, for nodes in duty-cycled sensor networks with asynchronous working schedules, the sender may have to wait for its receiver to transit to active state before it can send a packet. We define *sleep latency* ($d_{ij}(t)$) as the time duration from the moment that the sender i has a packet ready to be sent at time t to the moment that the receiver j is in the active state. For example, let working schedule for two neighboring nodes i and j be $\Gamma_i = \{1\}$ and $\Gamma_j = \{5\}$, respectively. Assume node i has a packet ready to be sent to node j at time 1, then sleep latency is $d_{ij}(1) = 5 - 1 = 4$. Sleep latency is usually in the order of seconds, which is orders of magnitude longer than other delivery latencies. Therefore in this paper, we only consider sleep latency for measuring E2E delays.

B. Assumptions

Based on the network model, we make several assumptions as follows:

- In energy-harvesting networks, a node in the network shares its dynamic working schedules with all its immediate neighbors after joining the network, a process normally referred to as low-duty-cycle rendezvous. Currently, there are many effective solutions such as Disco [36] and ESC [20] for achieving such low-duty-cycle rendezvous and sharing dynamic node working schedules with neighbors.
- The network is locally synchronized so that a node knows when it can send a packet to its neighbors given their working schedules. Simple and low-cost local synchronization techniques [37] can achieve an accuracy of $2.24\mu s$ with the cost of exchange a few bytes of packets among neighboring nodes every 15 minutes. Since the duration of each active instance is typically above $10,000\mu s$ [22], [36], the accuracy of $2.24\mu s$ is far more than sufficient. In addition, since we do not require transmission starts at the beginning of an active instance, this further relaxes the requirement of accuracy for time synchronization.
- Packet can be successfully delivered from a sender to a receiver within an active instance. For a typical TinyOS packet with a total packet size of 47 bytes, given the duration of an active instance, say $20ms$, a MicaZ node with CC2420 radio chip can attempt at least 13 transmissions. In other words, although low-power wireless links are generally unreliable [38], as long as the link quality p between a sender and its receiver is above 30%, we can ensure that at least 99% of packets can be successfully transmitted within an active instance. Since many neighbor management protocols filter neighboring nodes with low link qualities [39], [40], the requirement of at least 30% delivery ratio is realistic and reasonable in practical sensor network settings.
- Data traffic/congestion is low. In low-duty-cycle networks, a packet is transmitted only when the receiver is in the active state, and nodes in such networks only activate very briefly and stay in the dormant state for the majority of time. Therefore, data rate in such networks is usually low and queueing delay is negligible. As shown in several pioneering low-duty-cycle network works, this assumption holds well in practice [21]–[23].
- Here we would emphasize that the operations of an energy-harvesting network do not depend on TDMA. Unlike TDMA networks, we do not require a node to start a transmission at the beginning of a time instance. As long as a node knows the wake-up time of its neighboring nodes, the node can deliver messages to its receivers. In terms of performance, CSMA is more favorable in energy-harvesting networks where network traffic is low. This is because a node in TDMA networks has to wait for its turn to transmit and becomes inefficient in such

scenarios. Although our design works in both TDMA and CSMA network, CSMA is a better choice. During our experiment on a physical testbed, we use default B-MAC [41] in TinyOS, which is a CSMA-based MAC protocol.

V. Main Design

The goal of our design is to bound E2E delays from the sink node to nodes in the network with minimal energy consumption. In this section, we first presents an overview of the design and then discuss each step of design in detail.

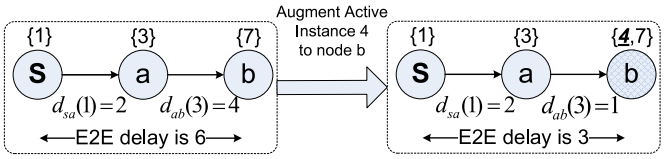


Fig. 3. Example of Active Instance Augmentation

A. Design Overview

For many power management protocols (e.g., sensing, tracking and communication), after they decide node working schedules in the network, usually they cannot provide E2E delay guarantees for sink-to-node communications [22], [31], [32], [42], [43]. Consequently, actions have to be taken in order for deployed sensor network to meet the communication delay bounds specified by applications. On the other hand, for the node working schedules that are determined by power management protocols, usually they are designed to meet other application requirements such as quality of sensing coverage [44]–[46], quality of target tracking [47], [48] and so on. In other words, the node behaviors of those original working schedules should be preserved. Otherwise we may affect the quality of service originally provided by those power management protocols. In order to *bound the communication delay and preserve the original node activities*, in this work we propose to augment additional active instances to nodes in the network. For example, in the linear network shown in Figure 3, the original E2E delay from sink s to node b is 6 units of time. After augmenting an active instance 4 to node b, the sleep latency between node a and node b reduces to 1 and the new E2E delay is just 3 units of time. In addition, for the original working schedule, node b activates at time 7. After augmenting the active instance 4 to node b, node b activates at both time 4 and 7. Consequently, we preserve the original activity pattern of node b.

In the following sections, we will introduce how we can augment a minimal number of active instances to the network such that the E2E delay from the sink to all nodes in the network is within the delay bound B. For the sake of clarity, we describe the scenario with a single sink node, that can be extended easily for scenarios with multiple sink nodes.

B. Finding Minimal Delays for Active Instance Augmentation

In this section, we introduce how to find the optimal path for augmenting h active instances in the network such that the E2E delay from the sink to a node j is minimized.

Before presenting the detailed algorithm of finding minimal delays for active instance augmentation, first we would like to introduce an important observation on how active instance should be augmented such that the optimality in terms of delay can be guaranteed. For the network model presented in Section IV, it exhibits FIFO property in packet delivery process. In a FIFO network, the packet arrival order at destination is the same as sending order at the source node. It is known that waiting in FIFO networks can never reduce E2E delay [49], an intermediate relay node therefore should forward the packet as soon as possible to ensure the minimal E2E delay. Consequently, when augmenting an active instance to a node, we should *reduce the sleep latency between a sender and a receiver to one* τ, the finest granularity of time duration in the system.

Specifically, to record the minimal E2E delay from the sink to a node j, we define the following metric:

D_j^h: The minimal delay a packet travels from the sink to node j along a path with at most h active instances augmented.

1) Initial State: For any node j that are one-hop away from the sink s, when $h = 0$, the initial delay from the sink to node j is simply the sleep latency between them. In addition, when $h = 1$, the delay from the sink to node j is simply 1 since node j is only one-hop away from the sink and the augmented active instance reduces this one-hop delay to 1. Consequently, we can have following initial state for any node j that is one-hop away from the sink:

$$D_j^h = \begin{cases} d_{sj}, & h = 0 \\ 1, & h = 1 \end{cases} \quad (1)$$

2) Recursive Solution: In this section, we discuss how we distributively compute the minimal D_j^h value based on its neighbors' corresponding delay values. For any node j, the minimal packet delivery delay from the sink to node j consists of two possibilities:

- The packet could be firstly transmitted from the sink to a certain intermediate node p at time t (possibly through multiple hops), and then go directly from node p to node j through one single edge without augmenting any additional active instance to node j.
- The packet could also firstly be delivered to the immediate node p at time t (possibly through multiple hops), and then augment an active instance to node j so as to reduce sleep latency between node p and node j to 1 if the path from the sink to node p has less than h augmented active instances.

Therefore, for the optimal E2E delay at node j through a neighboring node p after augmenting at most h active instances into the network, we can have following equation:

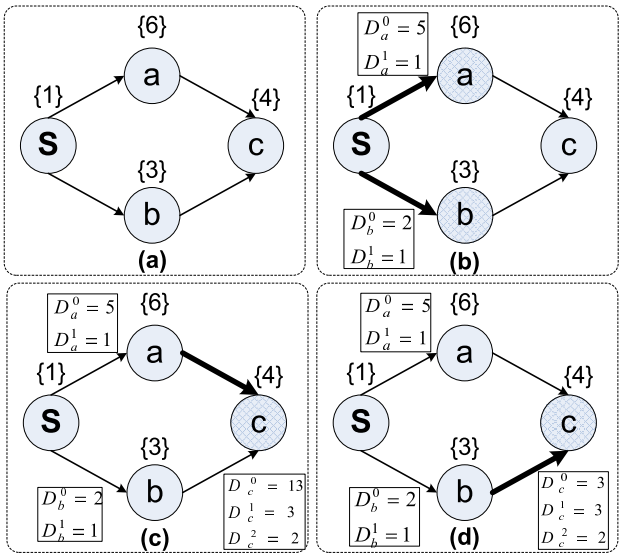


Fig. 4. Example of Distributed Delay Computation

$$D_j^h = Min \begin{cases} D_j^h, \\ D_p^{h-1} + 1, \quad h > 0 \\ D_p^h + d_{pj}(t) \end{cases} \tag{2}$$

3) Walkthrough Example: To further illustrate above delay computation process, we provide a walkthrough of the described algorithm on the network shown in Figure 4. In Figure 4, we assume the sink node s has a packet ready to be sent to all other nodes in the network at time 1 and the duration of a period is 10 units of time. For the initial state, node a and node b are one-hop away from the sink s, according to Equation 1, we can have following initial states at node a and node b (Figure 4b):

$$D_a^0 = 5 \quad , \quad D_a^1 = 1 \quad , \quad D_b^0 = 2 \quad , \quad D_b^1 = 1.$$

Then for node c, the minimal delay to reach it from sink s is either through node a or through node b. Assume node c first receives delay update from node a, then for D_c^0, which denotes the delay latency from sink s to node c without any active instance augmentation, its value is simply the sum of sleep latencies from sink s to node a and then from node a to node c. Consequently, we have the following equation for D_c^0 (Figure 4c):

$$D_c^0 = D_a^0 + d_{ac}(6) = 5 + 8 = 13.$$

For D_c^1, it denotes the minimal delay from sink s to node c with maximal one active instance augmentation along the path. For the path from sink s to node c through node a, we can augment this active instance at either node c or node a, according to Equation 2, we have:

$$\begin{aligned} D_c^1 &= Min \begin{cases} D_a^0 + 1 = 5 + 1 = 6, \\ D_a^1 + d_{ac}(2) = 1 + 2 = 3 \end{cases} \\ &= 3 \end{aligned}$$

For D_c^2, it means the minimal delay from sink s to node c with maximal two active instance augmentation along the path. Since node c is also two-hop away from sink s, we can augment one active instance at both node a and c and have:

$$D_c^2 = D_a^1 + 1 = 1 + 1 = 2.$$

Similarly, when node c receives delay update from node b, the delay at node c is updating again for h equals 0, 1 and 2 (Figure 4d).

$$\begin{aligned} D_c^0 &= Min \begin{cases} D_c^0 = 13, \\ D_b^0 + d_{bc}(3) = 2 + 1 = 3 \end{cases} \\ &= 3 \\ D_c^1 &= Min \begin{cases} D_c^1 = 3, \\ D_b^0 + 1 = 2 + 1 = 3, \\ D_b^1 + d_{bc}(2) = 1 + 2 = 3, \end{cases} \\ &= 3 \\ D_c^2 &= Min \begin{cases} D_c^2 = 2, \\ D_b^1 + 1 = 1 + 1 = 2, \end{cases} \\ &= 2 \end{aligned}$$

C. Maintaining Pairwise E2E Delay Bound

In previous sections, we describe how to find the minimal delay from the sink to a node with at most h augmented active instances. In this section, we discuss how to utilize this information to maintain the E2E delay bound B.

In a network with N nodes, the longest simple path from the sink to any other node consists of at most $N-1$ edges. Consequently, D_j^{N-1} denotes the minimal E2E delay can be achieved from the sink to a node j by augmenting at most $N-1$ active instances along the path. In other words, this is the lower bound of E2E delay from the sink to node j. If there is no constraints on which node has extra energy for augmenting active instances, the minimal E2E delay at node j is simply $D_j^H = H$, where H is the hop count from the sink to node j. The convergence time for all D_j^H values at a node, similar to the distributed Bellman-Ford Algorithm, is $\mathcal{O}(DH^2)$, where D is the density of the network.

For a node j, after obtaining all its D_j^h values, where $h = 1, 2, ..., M$ and D_j^M is the smallest achievable E2E delay at node j, the E2E delay maintenance procedure then goes as follows:

1) Firstly, node j checks whether its original E2E delay falls below the bound B. Specifically, D_j^0 states the minimal delay from the sink to node j without any active instance augmentation. If $D_j^0 \leq B$, then there is

no action initiated at node j. Otherwise node j initiates active instance augmentation process below.

2) Node j finds the smallest value of h, such that $D_j^h \leq B$. In this way, we guarantee minimal energy consumption for bounding E2E delay from the sink to node j.
3) According to the smallest value of h, node j looks for the previous hop node p that results the minimal D_j^h value. If $D_j^h = D_p^{h-1} + 1$, then node j augments one active instance which reduces sleep latency between node p and node j to one. Otherwise there is no active instance augmentation at node j. This step continues along the path from node j to the sink until all h active instances have augmented.

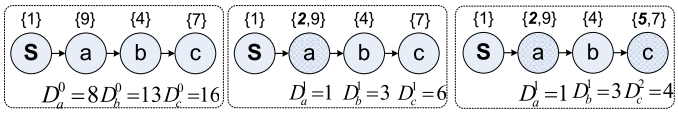


Fig. 5. Delays for a Linear Network

To further demonstrate E2E delay maintenance process, we provide a simple walkthrough example as shown in Figure 5. Figure 5 shows a 3-hop linear network and the corresponding D_j^h values for nodes in the network with varying maximum number of augmented active instances h.

Firstly, assume the E2E delay bound for sink node s to reach node c is $B = 16$, then at node c, it looks up its delay values and finds that $D_c^0 \leq B$. Consequently, node c confirms that without augmenting any active instance in the network, the E2E delay from the sink node to itself is below bound B and there is no further action necessary.

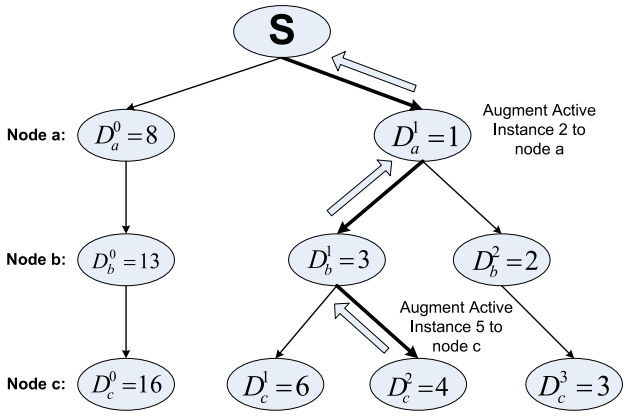


Fig. 6. Examples of Delay Maintenance

However, if an application requires tighter delay bound, such as $B \leq 4$, then we have to initiate the delay maintenance process. For $B \leq 4$, node c checks its delay values and finds that $D_c^0 > B$, $D_c^1 > B$ and $D_c^2 \leq B$. In other words, we affirm by augmenting two active instance to the network, we can achieve the application specified delay bound. The next question, therefore is how we distributively decide where to augment these active instances. For augmenting two active instance ($h = 2$), it is corresponding to the third scenario in Figure 5. In order to better illustrate active instance augmentation process, Figure 6 visualizes the minimal delay tree for augmenting at most three active instances to the linear network shown in Figure 5. In Figure 6, vertices are minimal delays at individual nodes by augmenting h active instances, where $h = 0, 1, 2, 3$. Edges in Figure 6 denote how minimal delay at a node by augmenting h active instances is obtained from its previous hop nodes. By tracing from the leaf node to the sink s of this delay tree, we can easily see where and how active instances should be augmented. Specifically for delay bound $B \leq 4$, by tracing up the minimal delay tree from leaf node D_c^2, then to intermediate nodes D_b^1 and D_a^1, we augment active instance 5 and 2 to node c and node a, respectively.

To implement this process distributively, it is sufficient for a node only knows minimal E2E delays to reach itself with h active instance augmentations, as well as how they are obtained from previous hop nodes. By comparing E2E delays between its previous hop node and itself, a node decides whether or not to augment an active instance to its working schedule. Then this node triggers its previous hop node to start this delay maintenance process. This whole process continues along the reverse communication path from the sink node to the destined receiver. Since the time complexity of delay maintenance at individual nodes is just $\mathcal{O}(1)$, for bounding E2E delay from the sink to a node, the total time complexity is $\mathcal{O}(H)$.

D. Considering Energy Constraints at Individual Nodes

In previous sections, we present the delay maintenance design as if there is no energy constraints and each node in the network can freely augment active instances to its working schedule. However, in energy-harvesting networks, the energy available to a node is strictly limit to the energy-harvesting rate and the capacity of energy storage device at the node. For example, for solar-powered sensor nodes, nodes under direct sunlight could harvest significant more energy than nodes that are placed in shade. The energy available to a node (i.e., the total number of active instances a node can afford), therefore could vary significantly at any given period of time. Many existing energy management protocols for energy-harvesting sensor networks have provided effective solutions on deciding appropriate duty-cycle (the number of active instances within a period) with in-situ energy supply [8]–[10], [50]. In this section, we discuss how we incorporate such constraints in maximal affordable duty cycles provided by lower layer power management protocols to our design.

To consider energy constraints at individual nodes in our design, we can simply extend our delay computation introduced in Section V-B. Specifically, we modify the recursive solution 2 to following equation:

$$D_j^h = Min \begin{cases} D_j^h, \\ D_p^{h-1} + 1, \quad h > 0, E_j > 0 \\ D_p^h + d_{pj}(t) \end{cases} \tag{3}$$

where E_j is the maximum number of active instance augmentations that node j can afford.

Compare with the original recursive equation, the only difference for new Equation 3 is when node j tries to reduce sleep latency between node p and itself to 1 (case $D_p^{h-1}+1$ in Equation 3), it first checks whether or not it can afford augment one active instance to its working schedule.

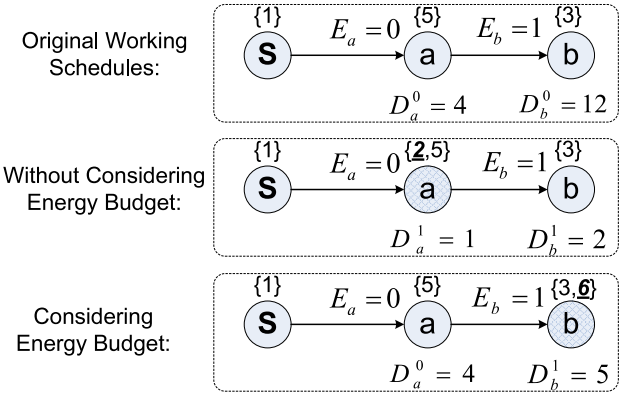


Fig. 7. Example of Considering Energy Budget

Figure 7 demonstrates the impact of considering energy budget for maintaining E2E delay bound. As seen from Figure 7, without considering the energy budget, we would choose node a to augment one active instance at the time 2 and have $D_a^1 = 1,\ \ D_b^1 = 2$. However, if node a cannot afford to augment any active instance ($E_a = 0$), then we can only augment one active instance at node b. Under such condition, the minimal E2E delay to node a and node b is $D_a^0 = 4$ and $D_b^1 = 5$, respectively. Clearly, the energy dynamics and variation in energy-harvesting sensor networks pose greater challenge to maintain E2E delay bound.

E. Maintaining Network-Wide E2E Delay Bound

In previous sections, we introduce the optimal solution for bounding E2E delay from the sink node to any given node in the network. However for many other applications, they may require E2E delays from the sink to all nodes in the network within a certain time bound. Under such scenarios, essentially, we need to bound E2E delays for one-to-many communications. However, we have proved that maintaining E2E delays for one-to-many communications with minimal energy consumption is NP-hard, which can be deducted from planar 3-Satisfiability. In addition, we also prove the problem of one-to-many communications is inapproximable. The detailed proof is omitted due to the space constraints.

Since maintaining network-wide E2E delay bound is NP-hard, in this section, we present a heuristic solution that effectively bounds E2E delays from the sink to all nodes in the network. The main idea is starting from the nodes with the largest original E2E delays (leaf nodes in the minimal delay tree, can be locally identified by checking whether there are neighbors with larger E2E delays), we follow the pairwise delay maintenance design in previous sections and augment the minimal number of active instances into the network to bound the E2E delay from the sink to those nodes. During above process, the new D_j^h values at any affected node j are also updated. The heuristic algorithm continues until all nodes in the network have their E2E delays from the sink fall below the delay bound B or there is no available active instance can be augmented to reduce E2E delays for nodes with delay larger than the delay bound B (with energy constraints in the network, it is possible that there are some nodes whose delay can not meet the delay bound B within a period of time). The intuition behind this heuristic algorithm is for those nodes with larger original E2E delays, their minimal delay paths usually include one or more nodes whose E2E delays from the sink are also beyond bound B. Consequently, by first bounding E2E delays for nodes with the largest delays, usually we can bound E2E delays for multiple nodes simultaneously and thus save both time and energy to achieve E2E delay bound network wide.

F. Tackling Energy Dynamics in Maintaining E2E Delay Bound

In previous sections, we present given a set of working schedules in the network, how we can bound E2E delays from the sink to all nodes in the network. However, as shown in many empirical measurement results [7], [8], energy supply to individual nodes varies significantly over time in energy-harvesting networks. In order to ensure continuous operation, node working schedules have to change dynamically with energy-harvesting rate [10], [20]. Consequently, such change in node working schedules may lead to violation of delay bound requirement at some nodes.

Luckily our distributed solution presented earlier is able to efficiently handle such energy dynamics. First of all, since our delay maintenance design is able to bound E2E delays from sink to all nodes in the network with minimal number of active instance augmentations, nodes are able to store the maximal amount of extra harvested energy to their energy storage devices such as rechargeable batteries or super capacitors [7], [8]. Such conserved energy at energy storage devices then can be utilized to support normal operation of sensor nodes during energy-insufficient time, and consequently reduces the impact of duty-cycle changes on E2E delays. Secondly, even a node has to reduce its duty cycle due to insufficient energy supply, unless this node removes the particular active instance that yields the minimal E2E delay from the sink to itself, there is no E2E delay change for this node and therefore there is no violation of E2E delay bound requirement. Finally, if a node removes the particular active instance that yields the minimal delay to reach itself, it will recompute its D_j^h values and disseminate new values to its neighboring nodes whose delay values are dependent on it. Such delay dissemination process continues until all affected nodes update their delay values. After updating delay values, only if some nodes notice their E2E delays are above the specified delay bound, then they begin to execute delay maintenance procedure discussed in earlier sections. Otherwise there is no further actions necessary

for maintaining the delay bound in the network. Since energy harvesting rate usually will not change within a very short period of time at individual nodes [7], [8], with the help of energy buffering at energy storage devices we can carefully manage the interval of duty-cycle changes at individual nodes to achieve fast convergence speed and low message overhead for maintaining delay bound in the network [51].

VI. Evaluation

In order to understand the performance of our delay maintenance design under various network settings, in this section we provide extensive simulation results against an improved version of streamlined wake-up scheme proposed by Cao et al. [31], which is a state-of-the-art solution for minimizing E2E delay in low-duty-cycle networks.

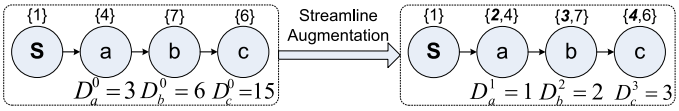


Fig. 8. Example of Streamline Augmentation

A. Baseline Setup

In original streamlined wake-up scheme, the authors first label nodes according to the shortest hop count from the sink. Then they begin to augment active instances and build a route from the sink to each node without any extra sleep latencies. For example, Figure 8 shows an example of streamline augmentation for a linear network. In Figure 8, streamline scheme augments active instance 2,3 and 4 to node a, node b and node c, respectively. Consequently, E2E delays from sink node s to all nodes in the network are minimized.

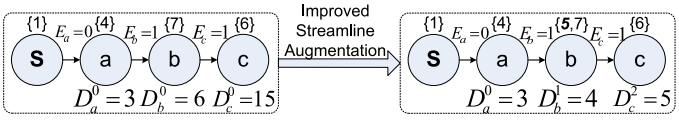


Fig. 9. Example of Improved Streamline Augmentation

In order for streamline scheme to bound E2E delays with minimal energy consumption while considering energy budget at individual nodes, here we introduce an improved version of streamline scheme as our baseline for performance comparison. Specifically, for bounding E2E delay of a node j, we allow streamlined wake-up scheme to find an optimal route such that the minimal number of active instance are augmented by removing extra sleep latencies from the sink to node j at energy-rich nodes until the delay bound has been met. Figure 9 shows an example of improved streamline wake-up scheme. In Figure 9, let us assume delay bound is 5 units of time. Since node a cannot afford to augment any active instance, the improved streamline wake-up scheme augments active instance 5 to node b (the second nearest node to sink s). In this way, the improved streamline wake-up scheme achieves the designated delay bound 5 for all nodes in an energy-constrained network with the minimal energy consumption. The performance difference of our delay bound maintenance design and the improved streamlined wake-up scheme is then purely due to the choice of nodes for active instance augmentations.

B. Simulation Setup

In the simulation, up to 300 nodes are randomly deployed in a 150m×150m square field. The communication range is set to be 25m. Except where otherwise specified, the default number of nodes in the network, node duty cycle and delay bound is 200, 2% and 150 units of time, respectively. Each data points in simulation figures is obtained by averaging 100 runs with different random seeds, node deployment and node working schedules.

C. System Performance Over Time

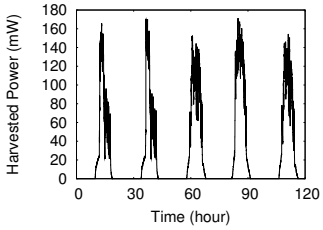


Fig. 10. Sample Node Energy Harvesting Rate

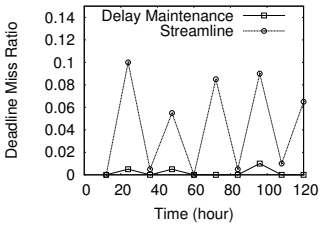


Fig. 11. Percentage of Nodes Beyond Delay Bound vs. Time

In an energy-harvesting network, energy-harvesting rate varies significantly both within a day and across days [7]. To study how well our delay maintenance design works under varying energy-harvesting rate, Figure 11 shows the percentage of nodes in the network whose E2E delays from the sink are beyond the bound for a period of 120 hours. The energy-harvesting rates used here are obtained from our empirical measurement at running prototypes and their corresponding energy-harvesting model [8]. Figure 10 shows sample node energy harvesting rate over 120 hours. Clearly from Figure 11, our delay maintenance design keeps E2E delays from the sink to nearly all nodes in the network below the specified bound, regardless of the level of energy supplies over time. For example, in 120-hour duration, the maximal percentage of nodes beyond delay bound is merely 1%, while the average over 120 hours is 0.2%. In contrast, the streamline design is rather sensitive to the change of energy supplies. When energy supply drops, the percentage of nodes beyond delay bound increases significantly. For example, the percentage of nodes beyond delay bound can be as high as 10% when energy-harvesting rate is low. For the period of 120 hours, the average percentage of nodes beyond delay bound is 4.15%. The main reason that delay maintenance design has much better performance than the streamline design is because delay maintenance design augments significant smaller number of active instances than that of the streamline to achieve the delay bound, therefore allows nodes in the network to store more energy during energy-rich time and use those buffered energy to help maintaining the delay bound when node energy supply

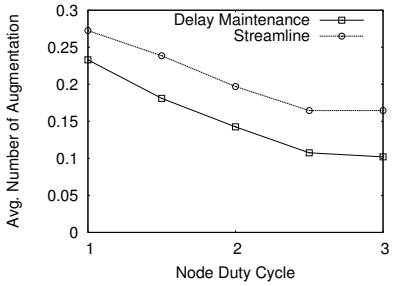


Fig. 12. Avg. Number of Augmentation vs. Node Duty-Cycle

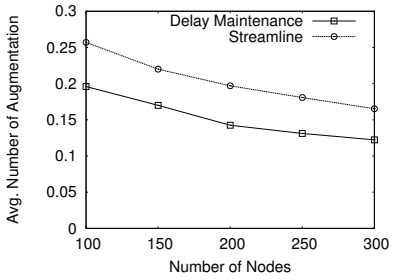


Fig. 13. Avg. Number of Augmentation vs. Number of Nodes

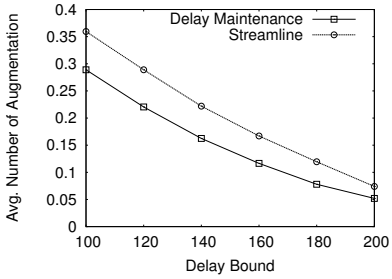


Fig. 14. Avg. Number of Augmentation vs. Delay Bound

falls in short. In the following sections, we will reveal reasons for better performance of delay maintenance design in more detail.

D. Comparison of Energy Consumptions

In this section, we aim at investigating energy consumption (the number of augmented active instances) for delay maintenance and the streamline to achieve delay bound under various network configurations.

In Figure 12, it shows the average number of active instances augmented to each node in the network for achieving the delay bound for all nodes in the network under various node duty cycles. Clearly from Figure 12, the average number of augmented active instances for delay maintenance is smaller than that of the streamline under all duty cycles. For example, at duty cycle of 3%, the average number of augmented active instance to each node in the network for delay maintenance and the streamline is 0.1 and 0.16, respectively. Consequently, at 3% duty cycle, delay maintenance saves about 60% energy of that of the streamline. Moreover, as trend in Figure 12 shows, the larger duty cycle is, the bigger gap for energy consumption between delay maintenance and the streamline.

Figure 13 shows the impact of node density on the average number of augmented active instances in the network. As node density increases, the average number of augmented active instances for both delay maintenance and the streamline decreases. However, for various node densities, we can see delay maintenance still outperforms the streamline design at all node densities. At all node densities, delay maintenance consumes above 30% less energy than that of the streamline design.

To study the impact of delay bound, Figure 14 shows the average number of augmented active instances under various delay bounds. As delay bound increases, the average number of augmented active instances for both designs decreases dramatically. This is because with looser bound on the E2E delay from the sink to all nodes in the work, there are fewer nodes whose delays are still beyond the bound and consequently result in smaller number of active instances augmented to the network. Similar to the results for duty cycles and node densities, delay maintenance augments less number of active instances than that of the streamline at all delay bounds. In average, delay maintenance augments about 40% less active instances than that of the streamline design.

In summary, in terms of energy consumption for bounding E2E delays, delay maintenance outperforms the streamline design significantly under different node duty cycles, different node densities, as well as different delay bounds.

VII. System Implementation

In addition to large-scale simulations, we implemented our delay maintenance design and the improved streamline wake-up scheme described in Section VI on the TinyOS/Mote platform with 11 MicaZ motes to further validate our design in practice.

A. Experiment Setup

We deploy 11 MicaZ nodes along a straight line. The transmission power at MicaZ motes is tuned down so that nodes form a 10-hop linear network. After deployment, each node starts to generate working schedule with the specified duty-cycle, which is controllable by a stand-alone base station node and corresponding GUI interface. Then, for all nodes in the network, they start to broadcast their existence and working schedules. Followed by neighbor discovery, sink node initiates delay computation process and nodes in the network begin to execute delay maintenance process. In this experiment, we use FTSP [37] to provide time synchronization among neighboring nodes and set the unit time τ as 20ms.

B. System Performance Comparison

First of all, we are interested to see whether we are able to successfully meet the specified delay bound in the network. Consequently, for both delay maintenance and streamline scheme, sink node sends 100 packets to the node that is 10-hop away, which represents the largest E2E delay in the network. Figure 15 shows measured E2E delays for both delay maintenance and streamline scheme under varying delay bounds. From Figure 15, we can see E2E delays for both schemes are under the specified delay bound. However, to achieve the same delay bound, two schemes pay very different costs. In Figure 16, we show the number of active instance augmentations in the network to achieve the specified delay bound by different schemes. Clearly, for all delay bounds, our delay maintenance design augments smaller number of active instances, which represents better energy efficiency. Especially when delay bound is relatively loose ($\geq 7s$), delay maintenance only augments half of active instances than that of the streamline design.

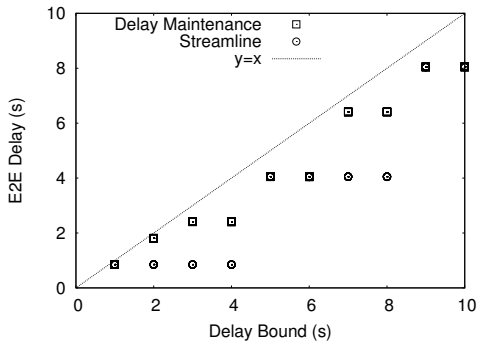


Fig. 15. E2E Delay vs. Delay Bound

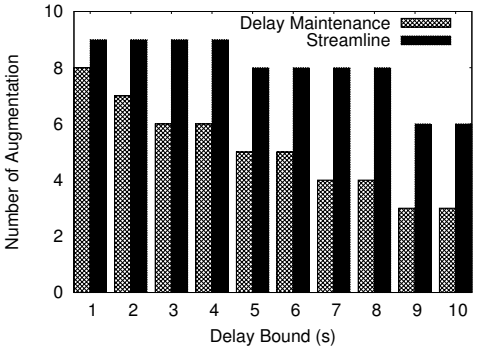


Fig. 16. Number of Augmentation vs. Delay Bound

C. System Insights

To further reveal more system insights of performance difference between delay maintenance and streamline, in Figure 17 we plot E2E delay reduction from the sink to the node which is 10-hop away under varying number of augmented active instances. From Figure 17, we can see benefited from the ability to augment optimal active instances, delay maintenance has much larger E2E delay reduction than that of streamline at all augmented active instances. For example, by augmenting 5 active instances, delay maintenance design reduces E2E delay to the furthest away node by 16.27s while streamline only reduces 7.93s, which is above two times difference.

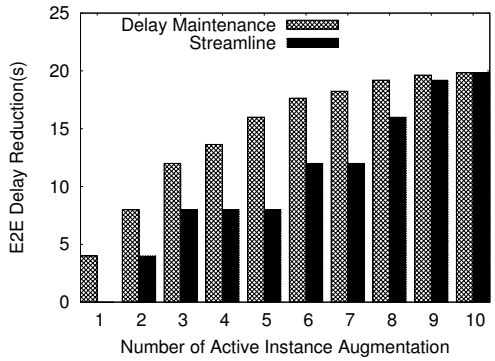
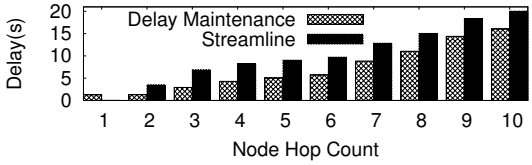
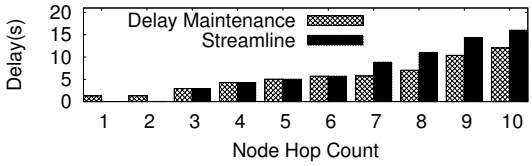


Fig. 17. E2E Delay vs. Number of Augmentation

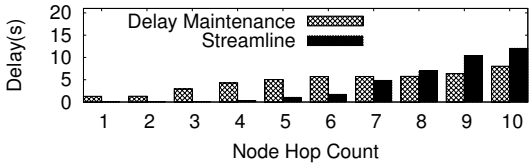
In Figure 18, we show snapshots of network wide E2E delays after augmenting up to 5 active instances to the network. From these five figures, we can see our delay maintenance design has more balanced E2E delays among nodes in the network while streamline has a much skewed distribution. Consequently, with the same number of active instance augmentations, our delay maintenance design is able to provide smaller delay bound. For example, after augmenting 5 active instances (Figure 18(e)), the largest delay in the network for delay maintenance design and streamline is 4.04s and 12.17s, respectively.



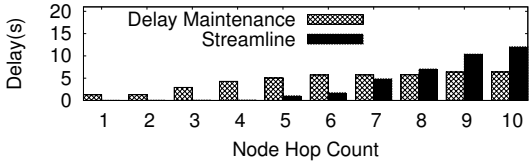
(a) After Augmenting 1 Active Instance



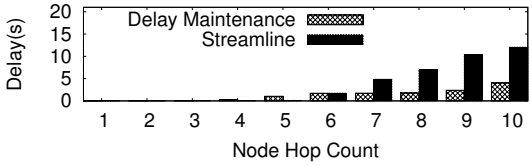
(b) After Augmenting 2 Active Instance



(c) After Augmenting 3 Active Instance



(d) After Augmenting 4 Active Instance



(e) After Augmenting 5 Active Instance

Fig. 18. Snapshots of Network Delays

VIII. Conclusion

In this paper, we present a distributed delay maintenance design for energy-harvesting sensor networks. We first introduce an optimal distributed algorithm for finding the minimal delay from the sink to a node by augmenting at most h active instances. In addition, we show how nodes in the network can cooperatively bound pairwise E2E delay with minimal energy consumption. For bounding E2E delays from the sink node to all nodes in the network with minimal energy consumption, we prove its NP-Hardness and propose an efficient heuristic solution. Through extensive simulation and test-bed experiments, we demonstrate our delay maintenance design is able to

effectively provide E2E delay guarantees in energy-harvesting networks and significantly outperforms an improved version of a state-of-the-art design in terms of energy consumption.

ACKNOWLEDGEMENT

This work was supported in part by NSF grants CNS-0917097, CNS-0845994, and CNS-0626609. We also received partial support from InterDigital and Microsoft Research.

REFERENCES

[1] L. Mo, Y. He, Y. Liu, J. Zhao, S.-J. Tang, X.-Y. Li, and G. Dai, "Canopy closure estimates with greenorbs: sustainable sensing in the forest," in *SenSys '09*, 2009.

[2] N. Ramanathan, T. Schoellhammer, E. Kohler, K. Whitehouse, T. Harmon, and D. Estrin, "Suelo: human-assisted sensing for exploratory soil monitoring studies," in *SenSys '09*, 2009.

[3] W. Hu, N. Bulusu, C. T. Chou, S. Jha, A. Taylor, and V. N. Tran, "Design and evaluation of a hybrid sensor network for cane toad monitoring," *ACM Trans. Sen. Netw.*, vol. 5, no. 1, 2009.

[4] M. Huntwork, A. Goradia, N. Xi, C. Haffner, C. Klochko, and M. Mutka, "Pervasive surveillance using a cooperative mobile sensor network," in *ICRA 2006*, 2006.

[5] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, "Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments," in *IPSN'06*, 2006.

[6] M. Rahimi, H. Shah, G. Sukhatme, J. Heidemann, and D. Estrin, "Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network," in *ICRA'03*, 2003.

[7] A. Kansal, D. Potter, and M. B. Srivastava, "Performance Aware Tasking for Environmentally Powered Sensor Networks," in *SIGMETRICS '04*, 2004.

[8] T. Zhu, Z. Zhong, Y. Gu, T. He, and Z.-L. Zhang, "Leakage-Aware Energy Synchronization for Wireless Sensor Networks," in *MobiSys '09*, 2009.

[9] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power Management in Energy Harvesting Sensor Networks," *TECS*, vol. 6, no. 4, 2007.

[10] C. Vigorito, D. Ganesan, and A. Bartoeeee, "Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks," in *SECON'07*, 2007.

[11] D. Lymberopoulos, A. Bamis, and A. Savvides, "A methodology for extracting temporal properties from sensor network data streams," in *MobiSys '09*, 2009.

[12] K. Romer, "Discovery of frequent distributed event patterns in sensor networks," in *EWSN'08*, 2008.

[13] 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.*, vol. 30, no. 1, 2005.

[14] C.-T. Huang, T.-H. Lin, L.-J. Chen, and P. Huang, "Xd: A cross-layer designed data collection mechanism for mission-critical wsns," in *MobiUS'09*, 2009.

[15] R. Mangharam, A. Rowe, R. Rajkumar, and R. Suzuki, "Voice over sensor networks," in *RTSS '06*, 2006.

[16] M.-Y. Nam, C.-G. Lee, K. Kim, and M. Caccamo, "Time-parameterized sensing task model for real-time tracking," in *RTSS'05*, 2005.

[17] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks," in *INFOCOM*, 2004.

[18] E. Felemban, C. Lee, E. Ekici, R. Boder, and S. Vural, "Probabilistic qos guarantee in reliablility and timeliness domains in wireless senosr networks," in *INFOCOM'05*, 2005.

[19] X. Yang and N. H. Vaidya, "A wakeup scheme for sensor networks: Achieving balance between energy saving and end-to-end delay," in *RTAS'04*, 2004.

[20] Y. Gu, T. Zhu, and T. He, "Esc: Energy synchronized communication in sustainable sensor networks," in *ICNP '09*, 2009.

[21] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay Efficient Sleep Scheduling in WIreless Sensor Networks," in *INFOCOM'05*, 2005.

[22] Y. Gu and T. He, "Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links," in *SenSys '07*, 2007.

[23] L. Su, C. Liu, H. Song, and G. Cao, "Routing in intermittently connected sensor networks," in *ICNP*, 2008.

[24] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links," in *MobiCom '09*, 2009.

[25] Y. Gu, , T. He, M. Lin, and J. Xu, "Spatiotemporal Delay Control for Low-Duty-Cycle Sensor Networks," in *RTSS'09)*, 2009.

[26] P. Sikka, P. Corke, P. Valencia, C. Crossman, D. Swain, and G. Bishop-Hurley, "Wireless adhoc sensor and actuator networks on the farm," in *IPSN '06*, 2006.

[27] Y. Li, C. Ai, C. T. Vu, Y. Pan, and R. Beyah, "Delay bounded and energy efficient composite event monitoring in heterogeneous wireless sensor networks," *IEEE TPDS*, 2009.

[28] H. M. Ammari and J. Giudici, "On the connected k-coverage problem in heterogeneous sensor nets: The curse of randomness and heterogeneity," in *ICDCS '09*, 2009.

[29] J. Hong, J. Cao, Y. Zeng, S. Lu, D. Chen, and Z. Li, "A location-free prediction-based sleep scheduling protocol for object tracking in sensor networks," in *ICNP '09*, 2009.

[30] K. Shah and M. Kumar, "Distributed independent reinforcement learning (dirl) approach to resource management in wireless sensor networks," in *MASS 2007*, 2007.

[31] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards Optimal Sleep Scheduling in Sensor Networks for Rare Event Detection ," in *IPSN'05*, 2005.

[32] C. Gui and P. Mohapatra, "Virtual Patrol: A New Power Conservation Design for Surveillance Using Sensor Networks," in *IPSN'05*, 2005.

[33] F. Wang and J. Liu, "Duty-cycle-aware broadcast in wireless sensor networks," in *INFOCOM'09*, 2009.

[34] J. Ma, W. Lou, Y. Wu, X.-Y. Li, and G. Chen, "Energy efficient tdma sleep scheduling in wireless sensor networks," in *INFOCOM '09*, 2009.

[35] S.-J. Tang, J. Yuan, X.-Y. Li, G. Chen, Y. Liu, and J. Zhao, "Raspberry: A stable reader activation scheduling protocol in multi-reader rfid systems," in *ICNP '09*, 2009.

[36] P. Dutta and D. Culler, "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications," in *SenSys '08*, 2008.

[37] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The Flooding Time Synchronization Protocol," in *SenSys'04*, 2004.

[38] M. Zuniga and B. Krishnamachari, "Analyzing the Transitional Region in Low Power Wireless Links," in *IEEE SECON'04*, 2004.

[39] M. Zamalloa, K. Seada, B. Krishnamachari, and A. Helmy, "Efficient geographic routing over lossy links in wireless sensor networks," *ACM TOSN*, vol. 4, no. 3, 2008.

[40] G. Zhou, T. He, and J. A. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks," in *MobiSys'04*, 2004.

[41] J. Polastre and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *SenSys'04*, 2004.

[42] J. Zhu and X. Wang, "Peer: a progressive energy efficient routing protocol for wireless ad hoc networks," in *INFOCOM 2005*, 2005.

[43] H. Jiang and S. Jin, "Scalable and robust aggregation techniques for extracting statistical information in sensor networks," in *ICDCS '06*, 2006.

[44] G. Wang, G. Cao, P. Bermn, and T. L. Porta, "Bidding protocols for sensor deployment," *IEEE Transaction on Mobile Computing*, vol. 6, no. 5, 2007.

[45] D. Wang, B. Xie, and D. P. Agrawal, "Coverage and lifetime optimization of wireless sensor networks with gaussian distribution," *IEEE Trans. Mob. Comput.*, vol. 7, no. 12, 2008.

[46] W. Shu, X. Liu, Z. Gu, and S. Gopalakrishnan, "Optimal sampling rate assignment with dynamic route selection for real-time wireless sensor networks," in *RTSS '08*, 2008.

[47] S. Subramaniam, V. Kalogeraki, and T. Palpanas, "Distributed real-time detection and tracking of homogeneous regions in sensor networks," in *RTSS'06*, 2006.

[48] L. Lazos, R. Poovendran, and J. A. Ritcey, "Probabilistic detection of mobile targets in heterogeneous sensor networks," in *IPSN '07*, 2007.

[49] R. K. Ahuja, J. B. Orlin, S. Pallottino, and M. G. Scutell, "Dynamic shortest paths minimizing travel times and costs," University of Pisa, Tech. Rep., 2001.

[50] A. Lachenmann, P. J. Marrón, D. Minder, and K. Rothermel, "Meeting lifetime goals with energy levels," in *SenSys '07*, 2007.

[51] Y. Huang and S. Bhatti, "Fast-converging distance vector routing for wireless mesh networks," in *ICDCS'08*, 2008.