Minimizing Delay and Maximizing Lifetime for Wireless Sensor Networks With Anycast

Joohwan Kim, *Student Member, IEEE*, Xiaojun Lin, *Member, IEEE*, Ness B. Shroff, *Fellow, IEEE*, and Prasun Sinha

***Abstract*—In this paper, we are interested in minimizing the delay and maximizing the lifetime of event-driven wireless sensor networks for which events occur infrequently. In such systems, most of the energy is consumed when the radios are on, waiting for a packet to arrive. Sleep–wake scheduling is an effective mechanism to prolong the lifetime of these energy-constrained wireless sensor networks. However, sleep–wake scheduling could result in substantial delays because a transmitting node needs to wait for its next-hop relay node to wake up. An interesting line of work attempts to reduce these delays by developing "anycast"-based packet forwarding schemes, where each node opportunistically forwards a packet to the first neighboring node that wakes up among multiple candidate nodes. In this paper, we first study how to optimize the anycast forwarding schemes for minimizing the expected packet-delivery delays from the sensor nodes to the sink. Based on this result, we then provide a solution to the joint control problem of how to optimally control the system parameters of the sleep–wake scheduling protocol and the anycast packet-forwarding protocol to maximize the network lifetime, subject to a constraint on the expected end-to-end packet-delivery delay. Our numerical results indicate that the proposed solution can outperform prior heuristic solutions in the literature, especially under practical scenarios where there are obstructions, e.g., a lake or a mountain, in the coverage area of the wireless sensor network.**

***Index Terms*—Anycast, delay, energy-efficiency, sensor network, sleep–wake scheduling.**

I. Introduction

RECENT advances in wireless sensor networks have resulted in a unique capability to remotely sense the environment. These systems are often deployed in remote or hard-to-reach areas. Hence, it is critical that such networks operate unattended for long durations. Therefore, extending network lifetime through the efficient use of energy has been a key issue in the development of wireless sensor networks. In this paper, we focus on event-driven asynchronous sensor networks with low data rates, where events occur rarely. This is an important class of sensor networks that has many applications such as environmental monitoring, intrusion detection, etc. In such systems, there are four main sources of energy consumption: energy required to keep the communication radios on; energy required for the transmission and reception of control packets; energy required to keep sensors on; and energy required for data transmission and reception. The fraction of total energy consumption for data transmission and reception is relatively small in these systems because events occur so rarely. The energy required to sense events is usually a constant and cannot be controlled. Hence, the energy expended to keep the communication system on (for listening to the medium and for control packets) is the dominant component of energy consumption, which can be controlled to extend the network lifetime. Thus, sleep–wake scheduling becomes an effective mechanism to prolong the lifetime of energy-constrained event-driven sensor networks. By putting nodes to sleep when there are no events, the energy consumption of the sensor nodes can be significantly reduced.

Various kinds of sleep–wake scheduling protocols have been proposed in the literature. Synchronized sleep–wake scheduling protocols have been proposed in [2]–[6]. In these protocols, sensor nodes periodically or aperiodically exchange synchronization information with neighboring nodes. However, such synchronization procedures could incur additional communication overhead and consume a considerable amount of energy. On-demand sleep–wake scheduling protocols have been proposed in [7] and [8], where nodes turn off most of their circuitry and always turn on a secondary low-powered receiver to listen to "wake-up" calls from neighboring nodes when there is a need for relaying packets. However, this on-demand sleep–wake scheduling can significantly increase the cost of sensor motes due to the additional receiver. In this paper, we are interested in asynchronous sleep–wake scheduling protocols such as those proposed in [9]–[11]. In these protocols, each node wakes up independently of neighboring nodes in order to save energy. However, due to the independence of the wake-up processes, additional delays are incurred at each node along the path to the sink because each node needs to wait for its next-hop node to wake up before it can transmit the packet. This delay could be unacceptable for delay-sensitive applications, such as fire detection or a tsunami alarm, which require the event reporting delay to be small.

Prior work in the literature has proposed the use of *anycast* packet-forwarding schemes (also called opportunistic forwarding schemes) to reduce this event reporting delay [12]–[20]. Under traditional packet-forwarding schemes, every node has one designated next-hop relaying node in the neighborhood, and it has to wait for the next-hop node to wake up

Manuscript received August 16, 2008; revised April 02, 2009; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Das. This work supported in part by the National Science Foundation through awards CNS-0626703, CNS-0721477, CNS-0721434, and CCF-0635202 and by ARO MURI awards W911NF-07-10376 (SA08-03) and W911NF-08-1-0238. An earlier version of this paper has appeared in the Proceedings of IEEE INFOCOM 2008 [1].

J. Kim and X. Lin are with School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: jhkim@purdue.edu; linx@purdue.edu).

N. B. Shroff is with Department of Electrical and Computer Engineering and Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: shroff@ece.osu.edu).

P. Sinha is with Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: prasun@cse.ohio-state.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2009.2032294

when it needs to forward a packet. In contrast, under anycast packet-forwarding schemes, each node has multiple next-hop relaying nodes in a candidate set (we call this set *the forwarding set*) and forwards the packet to the *first* node that wakes up in the forwarding set. It is easy to see that, compared to the basic scheme in [9]–[11], anycast clearly reduces the expected one-hop delay. For example, assuming that there are k nodes in the forwarding set, and that each node wakes up independently according to the Poisson process with the same rate, then anycast can result in a k-fold reduction in the expected one-hop delay.

However, anycast does not necessarily lead to the minimum expected end-to-end delay because a packet can still be relayed through a time-consuming routing path. Therefore, the first challenge for minimizing the expected end-to-end delay is to determine how each node should choose its anycast forwarding policy (e.g., the forwarding set) carefully. The work in [12]–[14] proposes heuristic anycast protocols that exploit the geographical distance to the sink node. The work in [15] and [16] considers MAC-layer anycast protocols that work with the separate routing protocols in the network layer. However, these solutions are heuristic in nature and do not directly minimize the expected end-to-end delay. The algorithms in [17]–[20] use the hop-count information (i.e., the number of hops for each node to reach the sink) to minimize some state-dependent cost (delay) metric along the possible routing paths. However, these algorithms do not directly apply to asynchronous sleep–wake scheduling, where each node does not know the wake-up schedule of neighboring nodes when it has a packet to forward. (In Section V, we will introduce another hop-count-based algorithm inspired by the idea in [19] and [20].)

The second challenge stems from the fact that good performance cannot be obtained by studying the anycast forwarding policy in isolation. Rather, it should be jointly controlled with the parameters of sleep–wake scheduling (e.g., the wake-up rate of each node). Note that the latter will directly impact both network lifetime and the packet-delivery delay. Hence, to optimally tradeoff network lifetime and delay, both the wake-up rates and the anycast packet-forwarding policy should be jointly controlled. However, such interactions have not been systematically studied in the literature [12]–[20].

In this paper, we address these challenges. We first investigate the *delay-minimization problem*: given the wake-up rates of the sensor nodes, how to optimally choose the anycast forwarding policy to minimize the expected end-to-end delay from all sensor nodes to the sink. We develop a low-complexity and distributed solution to this problem. We then formulate the *lifetime maximization problem*: given a constraint on the expected end-to-end delay, how to maximize the network lifetime by jointly controlling the wake-up rates and the anycast packet-forwarding policy. We show how to use the solution to the delay-minimization problem to construct an optimal solution to the lifetime-maximization problem for a specific definition of network lifetime.

Before we present the details of our problem formulation and the solution, we make a note regarding when the anycast protocols and the above optimization algorithms are applied. We can view the lifetime of an event-driven sensor network as consisting of two phases: *the configuration phase* and *the operation phase*. When nodes are deployed, the configuration phase begins, during which nodes optimize the control parameters of the anycast forwarding policy and their wake-up rates. It is during this phase that the optimization algorithms discussed in the last paragraph will be executed. In this phase, sensor nodes do not even need to follow asynchronous sleep–wake patterns. After the configuration phase, the operation phase follows. In the operation phase, each node alternates between two subphases, i.e., *the sleeping subphase* and *the event-reporting subphase*. In the sleeping subphase, each node simply follows the sleep–wake pattern determined in the configuration phase, waiting for events to occur. Note that since we are interested in asynchronous sleep–wake scheduling protocols, the sensor nodes do not exchange synchronization messages in this sleeping subphase. Finally, when an event occurs, the information needs to be passed on to the sink as soon as possible, which becomes the event-reporting subphase. It is in this event reporting subphase when the anycast forwarding protocol is actually applied, using the control parameters chosen during the configuration phase. Note that the configuration phase only needs to be executed once because we assume that the fraction of energy consumed due to the transmission of data is negligible. However, if this is not the case, the transmission energy will play a bigger role in reducing the residual energy at each node in the network. In this case, as long as the fraction of energy consumed due to data transmission is still small (but not negligible), the practical approach would be for the sink to initiate a new configuration phase after a long time has passed.

The rest of this paper is organized as follows. In Section II, we describe the system model and introduce the delay-minimization problem and the lifetime-maximization problem that we intend to solve. In Section III, we develop a distributed algorithm that solves the delay-minimization problem. In Section IV, we solve the lifetime-maximization problem using the preceding results. In Section V, we provide simulation results that illustrate the performance of our proposed algorithm compared to other heuristic algorithms in the literature.

II. System Model

We consider a wireless sensor network with N nodes. Let $\mathcal{N}$ denote the set of all nodes in the network. Each sensor node is in charge of both detecting events and relaying packets. If a node detects an event, the node packs the event information into a packet and delivers the packet to a sink s via multihop relaying. We assume that every node has at least one such multihop path to the sink. We also assume that there is a single sink. However, the analysis can be generalized to the case with multiple sinks (see [21, Subsection III-D]).

We assume that the sensor network employs *asynchronous* sleep–wake scheduling to improve energy efficiency, and nodes choose the next-hop node and forward the packet to the chosen node using the following basic sleep–wake scheduling protocol. This basic protocol generalizes typical asynchronous sleep–wake scheduling protocols in [9]–[11] to account for anycast. For ease of exposition, in this basic protocol, we assume that there is a single source that sends out event-reporting packets to the sink. This is the most likely operating mode because when nodes wake up asynchronously and with low duty-cycles, the chance of multiple sources generating event-reporting packets simultaneously is small. Furthermore, this basic protocol ignores the detailed effects of collision.

(We can extend this basic protocol to account for the case of collisions by multiple senders (including hidden terminals) or by multiple receivers. The detailed protocol is provided in Section V of our online technical report [21].) The sensor nodes sleep for most of the time and occasionally wake up for a short period of time t_{active}. When a node i has a packet for node j to relay, it will send a beacon signal and an ID signal (carrying the sender information) for time periods t_B and t_C, respectively, and then hear the medium for time period t_A. If the node does not hear any acknowledgment signal from neighboring nodes, it repeats this signaling procedure. When a neighboring node j wakes up and senses the beacon signal, it keeps awake, waiting for the following ID signal to recognize the sender. When node j wakes up in the middle of an ID signal, it keeps awake, waiting for the next ID signal. If node j successfully recognizes the sender, and it is a next-hop node of node i, it then communicates with node i to receive the packet. Node j can then use a similar procedure to wake up its own next-hop node. If a node wakes up and does not sense a beacon signal or ID signal, it will then go back to sleep. In this paper, we assume that the time instants that a node j wakes up follow a Poisson random process with rate λ_j. We also assume that the wake-up processes of different nodes are independent. The independence assumption is suitable for the scenario in which the nodes do not synchronize their wake-up times, which is easier to implement than the schemes that require global synchronization [3]–[5]. The advantage of Poisson sleep–wake scheduling is that, due to its memoryless property, sensor nodes are able to use a time-invariant optimal policy to maximize the network lifetime (see the discussion at the end of Section III-B). While the analysis in this paper focuses on the case when the wake-up times follow a Poisson process, we expect that the methodology in the paper can also be extended to the case with non-Poisson wake-up processes, with more technically involved analysis.

A well-known problem of using sleep–wake scheduling in sensor networks is the additional delay incurred in transmitting a packet from source to sink because each node along the transmission path has to wait for its next-hop node to wake up. To reduce this delay, we use an anycast forwarding scheme as described in Fig. 1. Let $\mathcal{C}_i$ denote the set of nodes in the transmission range of node i. Suppose that node i has a packet, and it needs to pick up a node in its transmission range $\mathcal{C}_i$ to relay the packet. Each node i maintains a list of nodes that node i intends to use as a forwarder. We call the set of such nodes the *forwarding set*, which is denoted by $\mathcal{F}_i$ for node i. In addition, each node j is also assumed to maintain a list of nodes i that use node j as a forwarder (i.e., $j \in \mathcal{F}_i$). As shown in Fig. 1, node i starts sending a beacon signal and an ID signal successively. All nodes in $\mathcal{C}_i$ can hear these signals, regardless of whom these signals are intended for. A node j that wakes up during the beacon signal or the ID signal will check if it is in the forwarding set of node i. If it is, node j sends one acknowledgment after the ID signal ends. After each ID signal, node i checks whether there is any acknowledgment from the nodes in $\mathcal{F}_i$. If no acknowledgment is detected, node i repeats the beacon-ID-signaling and acknowledgment-detection processes until it hears one. On the other hand, if there is an acknowledgment, it may take additional time for node i to identify which node acknowledges the beacon-ID signals, especially when there are multiple

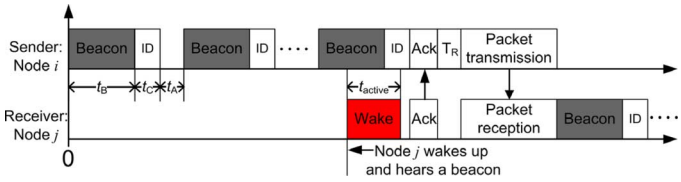


Fig. 1. System model.

nodes that wake up at the same time. Let t_{R} denote the resolution period, during which time node i identifies which nodes have sent acknowledgments. If there are multiple awake nodes, node i chooses one node among them that will forward the packet. After the resolution period, the chosen node receives the packet from node i during the packet transmission period t_{P}, and then starts the beacon-ID-signaling and acknowledgment-detection processes to find the next forwarder. Since nodes consume energy when awake, t_{active} should be as small as possible. However, if t_{active} is too small, a node that wakes up right after an ID signal could return to sleep before the following beacon signal. In order to avoid this case, we set $t_{\text{active}} = t_A + \epsilon_{\text{detect}}$, where ϵ_{detect} is a small amount of time required for a node to detect signal in the wireless medium. In the rest of the paper, we assume that ϵ_{detect} is negligible compared to t_A.

A. Anycast Forwarding and Sleep–Wake Scheduling Policies

In this model, there are three control variables that affect the network lifetime and the end-to-end delay experienced by a packet: wake-up rates, forwarding sets, and priority.

1) Wake-Up Rates: The sleep–wake schedule is determined by the wake-up rate λ_j of the Poisson process with which each node j wakes up. If λ_j increases, the expected one-hop delay will decrease, and so will the end-to-end delay of any routing paths that pass through node j. However, a larger wake-up rate leads to higher energy consumption and reduced network lifetime.

In the rest of the paper, it is more convenient to work with the notion of awake probability, which is a function of λ_j. Suppose that node i sends the first beacon signal at time 0, as in Fig. 1. If no nodes in $\mathcal{F}_i$ have heard the first $m-1$ beacon and ID signals, then node i transmits the mth beacon and ID signal in the time-interval $[(t_B+t_C+t_A)(m-1), (t_B+t_C+t_A)(m-1)+t_B+t_C]$. For a neighboring node j to hear the mth signals and to recognize the sender, it should wake up during $[(t_B+t_C+t_A)(m-1)-t_A-t_C, (t_B+t_C+t_A)m-t_A-t_C]$. Therefore, provided that node i is sending the mth signal, the probability that node $j \in \mathcal{C}_i$ wakes up and hears this signal is

$$p_j = 1 - e^{-\lambda_j(t_B+t_C+t_A)}. \tag{1}$$

We call p_j the *awake probability* of node j.

Remarks: Due to the memoryless property of a Poisson process, p_j is the same for each beacon-ID signaling iteration, m.[1]

Note that there is a one-to-one mapping between the awake probability p_j and the wake-up frequency λ_j. Hence, the awake

[1]To hear the first ID signal, the neighboring node j should wake-up during $[-t_A, t_B]$, which results in a smaller awake probability $p_j = 1-e^{-\lambda_j(t_B+t_A)}$ than (1). For simplicity of analysis, we can set the duration of the first beacon signal to $t_B + t_C$ so that the awake probability is consistent at all beacon-ID signals.

probability is also closely related to both delay and energy consumption. Let $\vec{p} = (p_i, i \in \mathcal{N})$ represent the global awake probability vector.

2) Forwarding Sets: The forwarding set $\mathcal{F}_i$ is the set of candidate nodes chosen to forward a packet at node i. In principle, the forwarding set should contain nodes that can quickly deliver the packet to the sink. However, since the end-to-end delay depends on the forwarding set of all nodes along the possible routing paths, *the optimal choices of forwarding sets of these nodes are correlated*. We use a matrix $\mathbf{A}$ to represent the forwarding set of all nodes collectively, as follows:

$$\mathbf{A} \triangleq [a_{ij}, i = 1, \ldots, N, j = 1, \ldots, N]$$

where $a_{ij} = 1$ if j is in node i's forwarding set, and $a_{ij} = 0$ otherwise. We call this matrix $\mathbf{A}$ the *forwarding matrix*. Reciprocally, we define $\mathcal{F}_i(\mathbf{A})$ as the forwarding set of node i under forwarding matrix $\mathbf{A}$, i.e., $\mathcal{F}_i(\mathbf{A}) = \{j \in \mathcal{C}_i | a_{ij} = 1\}$. We let $\mathcal{A}$ denote the set of all possible forwarding matrices.

With anycast, a forwarding matrix determines the paths that packets can potentially traverse. Let $g(\mathbf{A})$ be the directed graph $G(V, E(\mathbf{A}))$ with the set of vertices $V = \mathcal{N}$ and the set of edges $E(\mathbf{A}) = \{(i, j) | j \in \mathcal{F}_i(\mathbf{A})\}$. If there is a path in $g(\mathbf{A})$ that leads from node i to node j, we say that node i is *connected to* node j under the forwarding matrix $\mathbf{A}$. Otherwise, we call it *disconnected from* node j. An acyclic path is the path that does not traverse any node more than once. If $g(A)$ has any cyclic path, we call it a cyclic graph; otherwise, we call it an acyclic graph.

3) Priority: Let b_{ij} denote the priority of node j from the viewpoint of node i. Then, we define the priority assignment of node i as $\vec{b}_i = (b_{i1}, b_{i2}, \cdots, b_{iN})$, where each node $j \in \mathcal{C}_i$ is assigned a unique number b_{ij} from $1, \cdots, |\mathcal{C}_i|$, and $b_{ij} = 0$ for nodes $j \notin \mathcal{C}_i$. When multiple nodes send an acknowledgment after the same ID signal, the source node i will pick the highest-priority node among them as a next-hop node. Although only the nodes in a forwarding set need priorities, we assign priorities to all nodes to make the priority assignment an independent control variable from forwarding matrix $\mathbf{A}$. Clearly, the priority assignments of nodes will also affect the expected delay. In order to represent the global priority decision, we next define a priority matrix $\mathbf{B}$ as follows:

$$\mathbf{B} \triangleq [b_{ij}, i = 1, \ldots, N, j = 1, \ldots, N].$$

We let $\mathcal{B}$ denote the set of all possible priority matrices.

Among the three control variables, we call the combination of the forwarding and priority matrices $(\mathbf{A}, \mathbf{B})$ *the anycast packet-forwarding policy* (or simply an anycast policy) because these variables determine how each node chooses its next-hop node. We also call the awake probability vector *the sleep–wake scheduling policy* because this variable affects when each node wakes up.

Remarks: Throughout this paper, we assume that the values of t_B, t_C, and t_A in Fig. 1 are given. In practice, these values should be chosen as small as possible (subject to practical constraints) in order for the sending node to have more frequent decision-making opportunities.

B. Anycast Objectives and Performance Metrics

In this subsection, we define the performance objectives of the anycast policy and the sleep–wake scheduling policy that we intend to optimize. We remind the readers that, although the sleep–wake patterns and the anycast forwarding policy are applied in the operation phase of the network, their control parameters are optimized in the configuration phase.

1) End-to-End Delay: We define the end-to-end delay as the delay from the time when an event occurs to the time when the first packet due to this event is received at the sink. We motivate this performance objective as follows: For applications where each event only generates one packet, the above definition clearly captures the delay of reporting the event information. For those applications where each event may generate multiple packets, we argue that the event reporting delay is still dominated by the delay of the first packet. This is the case because once the first packet goes through, the sensor nodes along the path can stay awake for a while. Hence, subsequent packets do not need to incur the wake-up delay at each hop, and thus the end-to-end delay for the subsequent packets is much smaller than that of the first packet.

When there is only one source generating event-reporting packets, the end-to-end delay of the first packet can be determined as a function of the anycast policy $(\mathbf{A}, \mathbf{B})$ and the sleep–wake scheduling policy $\vec{p}$. One may argue that it may be desirable to design protocols that can potentially reduce the end-to-end delay by adjusting the anycast policy dynamically after the event occurs, e.g., according to traffic density. However, this dynamic adjustment is not possible for the first packet because when the first packet is being forwarded, the sensor nodes have not woken up yet. Hence, to forward the first packet to the sink, the sensor nodes must use some preconfigured policies determined in the configuration phase (we remind the readers about the discussion of different phases at the end of the introductory section). After the first packet is delivered to the sink, the sensor nodes along the path to the sink have woken up. Thereafter, they are able to adapt their control policies dynamically, e.g., according to the traffic density. In this paper, since we are mostly interested in reducing the delay of the first packet, these dynamic adaptive policies are outside the scope of our paper. In other words, we mainly focus on the optimization of the anycast policy and the sleep–wake scheduling policy at the initial configuration phase.

Based on the preceding discussion, we define the end-to-end delay as the delay incurred by the first packet. Given $\mathbf{A}$, $\mathbf{B}$, and $\vec{p}$, the stochastic process with which the first packet traverses the network from the source node to the sink is completely specified and can be described by a Markov process with an absorbing state that corresponds to the state that a packet reaches the sink. We define $D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ as the expected end-to-end delay for a packet from node i to reach sink s when the awake probability vector $\vec{p}$ and anycast policy $(\mathbf{A}, \mathbf{B})$ are given. Since sink s is the destination of all packets, the delay of packets from sink s is regarded as zero, i.e., $D_s(\vec{p}, \mathbf{A}, \mathbf{B}) = 0$, regardless of $\vec{p}$, $\mathbf{A}$, and $\mathbf{B}$. If node i is disconnected from sink s under the forwarding matrix $\mathbf{A}$, packets from the node cannot reach sink s. In this case, the end-to-end delay from such a node i is regarded as infinite, i.e., $D_i(\vec{p}, \mathbf{A}, \mathbf{B}) = \infty$. From now on, we call "the expected end-to-end delay from node i to sink s" simply "the delay from node i."

Our first objective is to solve the following *delay-minimization problem*:

$$\min_{\mathbf{A},\mathbf{B}} \quad D_i(\vec{p},\mathbf{A},\mathbf{B}). \tag{2}$$

This problem is to find the optimal anycast forwarding policy $(\mathbf{A},\mathbf{B})$ that can minimize the delay from node i for given asynchronous sleep–wake scheduling policy (i.e., given wake-up rates $\vec{p}$). In Section III, we develop an algorithm that completely solves this problem for all nodes i, i.e., our solution minimizes the delays from all nodes simultaneously.

2) Network Lifetime: We now introduce the second performance metric, the network lifetime, and the corresponding lifetime-maximization problem (subject to delay constraints). Let Q_i denote the energy available to node i. We assume that node i consumes μ_i units of energy each time it wakes up. We define the expected lifetime of node i as $Q_i/\mu_i\lambda_i$. Note that implicitly in this definition of lifetime we have chosen not to account for the energy consumption by data transmission. As mentioned in the introductory section, this is a reasonable approximation for event-driven sensor networks in which events occur very rarely because the energy consumption of the sensor nodes is dominated by the energy consumed during the sleep–wake scheduling. For example, the IEEE 802.15.4-based low-powered sensor modules on IRIS sensor nodes consume 19.2 mW of power while awake (i.e., in an active mode) and 40.8 mW when transmitting at 250 kbps [22]. Assume that nodes stay awake only 1% of the time, and each node has to deliver 50 MB of information per year on average. Then, in a year, the total amount of energy consumed by a sensor node to just wake up is about $6054.9\ \mathrm{W\cdot s}$ ($365\ \mathrm{days/year} \times 24\ \mathrm{h/day} \times 60\ \mathrm{min/h} \times 60\ \mathrm{s/min} \times 0.01 \times 19.2\ \mathrm{mW}$).[2] In contrast, the energy consumption due to packet transmission for a year is about $66.8\ \mathrm{W\cdot s}$ ($50\ \mathrm{MB} \times 1024\ \mathrm{kB/MB} \times 8\ \mathrm{bits/byte} / 250\ \mathrm{kbps} \times 40.8\ \mathrm{mW}$), which is only 1% of the energy consumption due to waking up.

By introducing the power consumption ratio $e_i = \mu_i/Q_i$, we can express the lifetime of node i as

$$T_i(\vec{p}) = \frac{1}{e_i\lambda_i} = \frac{t_B + t_C + t_A}{e_i \ln \frac{1}{(1-p_i)}}. \tag{3}$$

Here, we have used the definition of the awake probability $p_i = 1 - e^{-\lambda_j(t_B+t_C+t_A)}$ from (1).

We define *network lifetime* as the shortest lifetime of all nodes. In other words, the network lifetime for a given awake probability vector $\vec{p}$ is given by

$$T(\vec{p}) = \min_{i\in\mathcal{N}} T_i(\vec{p}). \tag{4}$$

Based on the above performance metrics, we present the *lifetime-maximization problem* (which is the second problem we intend to solve in this paper) as follows:

$$\begin{aligned} (\mathbf{P}) \quad & \max_{\vec{p},\mathbf{A},\mathbf{B}} \quad T(\vec{p}) \\ & \text{subject to} \quad D_i(\vec{p},\mathbf{A},\mathbf{B}) \le \xi^*, \quad \forall i \in \mathcal{N} \\ & \qquad\qquad \vec{p} \in (0,1]^N, \quad \mathbf{A} \in \mathcal{A}, \quad \mathbf{B} \in \mathcal{B} \end{aligned}$$

where ξ^* is the maximum allowable delay. The objective of the above problem is to choose the anycast and sleep–wake scheduling policies $(\vec{p},\mathbf{A},\mathbf{B})$ that maximize the network lifetime and also guarantee that the expected delay from each node to sink s is no larger than the maximum allowable delay ξ^*.

Remarks: The lifetime definition in (4) is especially useful in dealing with the most stringent requirement for network lifetime such that all nodes must be alive to carry out the functionality of the sensor network [23]–[25]. In Section IV, we solve the lifetime-maximization problem $(\mathbf{P})$ with the lifetime definition in (4), using the solution of the delay-minimization problem as a component. Specifically, for any given $\vec{p}$, it would be desirable to use an anycast policy $(\mathbf{A},\mathbf{B})$ that minimizes the delay. Hence, the solution to the delay-minimization problem will likely be an important component for solving the lifetime-maximization problem, which is indeed the case in the solution that we provide in Section IV. There are application scenarios where alternate definitions of network lifetime could be more suitable, e.g., when the sensor network can be viewed as operational even if a certain percentage of nodes are dead. We believe that a similar methodology can also be used for other lifetime definitions, which we leave for future work.

III. Minimization of End-to-End Delays

In this section, we consider how each node should choose its anycast policy $(\mathbf{A},\mathbf{B})$ to minimize the delay $D_i(\vec{p},\mathbf{A},\mathbf{B})$, *when the awake probabilities $\vec{p}$ are given*. Then, in Section IV, we relax the fixed awake-probability assumption to solve Problem $(\mathbf{P})$.

The delay-minimization problem is an instance of the stochastic shortest path (SSP) problem [26, ch. 2], where the sensor node that holds the packet corresponds to the "state," and the delay corresponds to the "cost" to be minimized. The sink then corresponds to the terminal state, where the cost (delay) is not incurred anymore. Let $i_0, i_1, i_2, \cdots$ be the sequence of nodes that successively relay the packet from the source node i_0 to sink node s. Note that the sequence is random because at each hop, the first node in the forwarding set that wakes up will be chosen as a next-hop node. If the packet reaches sink s after K hops, we have $i_h = s$ for $h \ge K$. Let $d_j(\vec{p},\mathbf{A},\mathbf{B})$ be the expected one-hop delay at node j under the anycast policy $(\mathbf{A},\mathbf{B})$, that is, the expected delay from the time the packet reaches node j to the time it is forwarded to the next-hop node. Then, the end-to-end delay $D_i(\vec{p},\mathbf{A},\mathbf{B})$ from node i can be expressed as $D_i(\vec{p},\mathbf{A},\mathbf{B}) = E[\sum_{k=0}^{\infty} d_{i_k}(\vec{p},\mathbf{A},\mathbf{B})]$.

In this section, we solve the delay minimization problem using a Dynamic Programming approach. Our key contribution is to exploit the inherent structure of the problem to greatly reduce the complexity of the solution. We start with the relationship between the delays of neighboring nodes.

A. Local Delay Relationship

We first derive a recursive relationship for the delay, $D_i(\vec{p},\mathbf{A},\mathbf{B})$. When node i has a packet, the probability $P_{j,h}$ that the neighboring node j becomes a forwarder right after the hth beacon-ID signals is equal to the probability that no nodes in $\mathcal{F}_i(\mathbf{A})$ have woken up for the past $h-1$ beacon-ID-signaling

[2]Note that this amount of energy is within the range of capacity of two typical 1500 mAh AA batteries ($1500\ \mathrm{mA\cdot h} \times 2 \times 1.5\ \mathrm{V} = 4.5\ \mathrm{W\cdot h} = 16200\ \mathrm{W\cdot s}$).

iterations, and that node j wakes up at the hth beacon-ID signals while all nodes with a higher priority than node j remain sleeping at the hth iteration, i.e.,

$$P_{j,h} = \left(\prod_{k \in \mathcal{F}_i(\mathbf{A})} (1 - p_k) \right)^{h-1} p_j \prod_{k \in \mathcal{F}_i(\mathbf{A}): b_{ij} < b_{ik}} (1 - p_k).$$

Conditioned on this event, the expected delay from node i to sink s is given by $(t_B + t_C + t_A)h + t_{\mathrm{R}} + t_{\mathrm{P}} + D_j(\vec{p}, \mathbf{A}, \mathbf{B})$, where the sum of the first three terms is the one-hop delay, and the last term is the expected delay from the next-hop node j to the sink (see Fig. 1). For ease of notation, we define the iteration period $t_I \triangleq t_B + t_C + t_A$ and the data transmission period $t_{\mathrm{D}} \triangleq t_{\mathrm{R}} + t_{\mathrm{P}}$. We can then calculate the probability $q_{i,j}(\vec{p}, \mathbf{A}, \mathbf{B})$ that the packet at node i will be forwarded to node j as follows:

$$q_{i,j}(\vec{p}, \mathbf{A}, \mathbf{B}) \triangleq \sum_{h=1}^{\infty} P_{j,h} = \frac{p_j \prod_{k \in \mathcal{F}_i(\mathbf{A}): b_{ij} < b_{ik}} (1 - p_k)}{1 - \prod_{j \in \mathcal{F}_i(\mathbf{A})} (1 - p_j)}. \tag{5}$$

Similarly, we can also calculate the expected one-hop delay $d_i(\vec{p}, \mathbf{A}, \mathbf{B})$ at node i as follows:

$$\begin{aligned} d_i(\vec{p}, \mathbf{A}, \mathbf{B}) &\triangleq \sum_{h=1}^{\infty} \sum_{j \in \mathcal{F}_i(\mathbf{A})} [(t_I h + t_{\mathrm{D}}) P_{j,h}] \\ &= t_{\mathrm{D}} + \frac{t_I}{1 - \prod_{j \in \mathcal{F}_i(\mathbf{A})} (1 - p_j)}. \end{aligned} \tag{6}$$

Using the above notations, we can express the expected delay $D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ of node i for given awake probability vector $\vec{p}$, forwarding matrix $\mathbf{A}$, and priority matrix $\mathbf{B}$ as follows:[3]

$$\begin{aligned} &D_i(\vec{p}, \mathbf{A}, \mathbf{B}) \\ &= \sum_{h=1}^{\infty} \sum_{j \in \mathcal{F}_i(\mathbf{A})} [(t_I h + t_{\mathrm{D}} + D_j(\vec{p}, \mathbf{A}, \mathbf{B})) P_{j,h}] \qquad (7) \\ &= d_i(\vec{p}, \mathbf{A}, \mathbf{B}) + \sum_{j \in \mathcal{F}_i(\mathbf{A})} q_{i,j}(\vec{p}, \mathbf{A}, \mathbf{B}) D_j(\vec{p}, \mathbf{A}, \mathbf{B}). \qquad (8) \end{aligned}$$

We call (8) *the local delay relationship*, which must hold for all nodes i except the sink s. Recall that $D_s(\vec{p}, \mathbf{A}, \mathbf{B}) = 0$ regardless of the delay of the neighboring nodes. Note that from (5) and (6), the anycast policies of other nodes do not affect the one-hop delay $d_i(\vec{p}, \mathbf{A}, \mathbf{B})$ and the probability $q_{i,j}(\vec{p}, \mathbf{A}, \mathbf{B})$ of node i. Hence, we can rewrite the local delay relationship using the following delay function f that is only affected by the anycast policy of node i: For given delay values $\vec{\pi}_i = (D_j, j \in \mathcal{C}_i)$, forwarding set $\mathcal{F}_i$, and priority assignment $\vec{b}_i$, let

$$f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i) \triangleq t_{\mathrm{D}} + \frac{t_I + \sum_{j \in \mathcal{F}_i} p_j \prod_{k \in \mathcal{F}_i: b_{ij} < b_{ik}} (1 - p_k) D_j}{1 - \prod_{j \in \mathcal{F}_i} (1 - p_j)}. \tag{9}$$

[3]This equation and the analysis that follows ignore the delay for resolving the collision when multiple nodes send an acknowledgment after the same beacon-ID signal. Such an assumption leads to tractable analysis and is reasonable when the beacon-ID interval t_I is short and hence the chances that multiple nodes wake up at the same beacon-ID signal are low. Readers can refer to [21, Sec. V] for how to deal with collisions.

We call the function $f(\cdot, \cdot, \cdot)$ *the local delay function*. With the local delay function, we can express the local delay relationship (8) as $D_i(\vec{p}, \mathbf{A}, \mathbf{B}) = f(\vec{\pi}_i, \mathcal{F}_i(\mathbf{A}), \vec{b}_i)$, where $\vec{\pi}_i = (D_j(\vec{p}, \mathbf{A}, \mathbf{B}), j \in \mathcal{C}_i)$.

Let $D_i^*(\vec{p})$ be the minimal expected delay from node i to the sink for given awake probabilities $\vec{p}$, i.e., $D_i^*(\vec{p}) = \min_{\mathbf{A}, \mathbf{B}} D_i(\vec{p}, \mathbf{A}, \mathbf{B})$. Then, we can find the optimal anycast policy that achieves $D_i^*(\vec{p})$ for all nodes using the value-iteration algorithm [26, Section 1.3] as follows:[4]

Value-Iteration Algorithm

Initially, every node i sets its delay value $D_i^{(0)} = \infty$, and the sink sets $D_s^{(0)} = 0$. At each iteration $k = 1, 2, \cdots$, each node i updates its delay value $D_i^{(h)}$ by solving

$$D_i^{(h)} = \min_{\mathcal{F}_i, \vec{b}_i} f\left(\vec{\pi}_i^{(h-1)}, \mathcal{F}_i, \vec{b}_i\right) \tag{10}$$

where $\vec{\pi}_i^{(h-1)} = (D_j^{(h-1)}, j \in \mathcal{C}_i)$. Let $(\mathcal{F}_i^{(h)}, \vec{b}_i^{(h)})$ be the corresponding solution of (10).

As we will show later, the value-iteration algorithm will converge to the minimum delay values $D_i^*(\vec{p})$, and we can obtain the stationary optimal anycast policy. For this value iteration method to work, we will need an efficient methodology to solve (10). Note that since there are $2^{|\mathcal{C}_i|}$ possible choices of $\mathcal{F}_i$, where $|C_i|$ is the number of neighboring nodes of node i, an exhaustive search to solve (10) will have an exponential computational complexity. In the next subsection, we will develop a procedure with only linear complexity.

B. The Optimal Forwarding Set and Priority Assignment

In this subsection, we provide an efficient algorithm for the value iteration. For ease of exposition, let D_j denote the delay value of node j at the $(h-1)$-th iteration, and let $\vec{\pi}_i = (D_j, j \in \mathcal{C}_i)$. Our goal is to find the anycast policy $(\mathcal{F}_i, \vec{b}_i)$ of node i that minimizes $f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i)$.

We first show that in order to minimize $f(\cdot, \cdot, \cdot)$, the optimal priority assignment $\vec{b}_i^*$ can be completely determined by the neighboring delay vector $\vec{\pi}_i$.

Proposition 1: Let $\vec{b}_i^*$ be the priority assignment that gives higher priorities to neighboring nodes with smaller delays, i.e., for each pair of nodes j and k that satisfy $b_{ij}^* < b_{ik}^*$, the inequality $D_k \leq D_j$ holds. Then, for any given $\mathcal{F}_i$, we have $f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i^*) \leq f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i)$ for all possible $\vec{b}_i$.

The detailed proof is provided in Appendix A. The intuition behind Proposition 1 is that when multiple nodes send acknowledgments, selecting the node with the smallest delay should minimize the expected delay. Therefore, priorities must be assigned to neighboring nodes according to their (given) delays D_j, independent of the awake probabilities and forwarding sets. In the sequel, we use $b_i^*(\vec{\pi}_i)$ to denote the optimal priority assignment for given neighboring delay vector $\vec{\pi}_i$, i.e., for all nodes j and k in $\mathcal{C}_i$, if $b_{ij}^*(\vec{\pi}_i) < b_{ik}^*(\vec{\pi}_i)$, then $D_k \leq D_j$. For

[4]The value-iteration algorithm has some similarity to the Bellman–Ford shortest-path routing algorithm. In fact, we can regard the Bellman–Ford algorithm as a special case of the value-iteration algorithm in which each node has only one next-hop node.

ease of notation, we define the value of the local delay function with this optimal priority assignment as

$$\hat{f}(\vec{\pi}_i, \mathcal{F}_i) \triangleq f\left(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i^*(\vec{\pi}_i)\right). \tag{11}$$

The following properties characterize the structure of the optimal forwarding set.

Proposition 2: For a given $\vec{\pi}_i$, let $\mathcal{J}_1$, $\mathcal{J}_2$, and $\mathcal{J}_3$ be mutually disjoint subsets of $\mathcal{C}_i$ satisfying $b_{ij_2}^*(\vec{\pi}_i) < b_{ij_1}^*(\vec{\pi}_i)$ for all nodes $j_1 \in \mathcal{J}_k$ and $j_2 \in \mathcal{J}_{k+1}$ $(k = 1, 2)$. Let

$$D_{J_k} = \frac{\sum_{j\in\mathcal{J}_k} D_j p_j \prod_{k\in\mathcal{J}_k : b_{ij}^*(\vec{\pi}_i) < b_{ik}^*(\vec{\pi}_i)} (1-p_k)}{1 - \prod_{j\in\mathcal{J}_k}(1-p_j)}$$

denote the weighted average delay for the nodes in $\mathcal{J}_k$ for $k = 1$, 2, 3. Then, the following properties related to $\hat{f}(\vec{\pi}_i, \cdot)$ hold:

(a) $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \Leftrightarrow D_{J_3} + t_{\mathrm{D}} < \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \Leftrightarrow D_{J_3} + t_{\mathrm{D}} < \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$.

(b) $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) = \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \Leftrightarrow D_{J_3} + t_{\mathrm{D}} = \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \Leftrightarrow D_{J_3} + t_{\mathrm{D}} = \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$.

(c) If $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1)$, then $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$.

(d) If $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) = \hat{f}(\vec{\pi}_i, \mathcal{J}_1)$, then $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3) \leq \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$, and the equality holds only when $D_{j_2} = D_{j_3}$ for all $j_2 \in \mathcal{J}_2$ and $j_3 \in \mathcal{J}_3$.

The proof is provided in Appendix B. While Proposition 2 looks complex, it can be interpreted in a straightforward manner as follows. Note that nodes in $\mathcal{J}_1$ (or $\mathcal{J}_2$, correspondingly) have lower delay (and higher priority) than nodes in $\mathcal{J}_2$ (or $\mathcal{J}_3$, correspondingly). Properties (a) and (b) provide a test to decide whether to include the higher-delay nodes of $\mathcal{J}_3$ into the forwarding set. Specifically, Property (a) implies that adding the lower priority nodes of $\mathcal{J}_3$ into the current forwarding set $\mathcal{F}_i = \mathcal{J}_1$ decreases the delay if and only if the weighted average delay of the neighboring nodes in $\mathcal{J}_3$ plus t_{D} is smaller than the current delay. Similarly, Property (b) implies that adding the lower-priority node of $\mathcal{J}_3$ does not change the current delay if and only if the weighted average delay for the nodes in $\mathcal{J}_3$ plus t_{D} is equal to the current delay.

On the other hand, Properties (c) and (d) state that if the forwarding set already includes the higher-delay nodes in $\mathcal{J}_3$, then it should also include the lower-delay nodes in $\mathcal{J}_2$. Specifically, Property (c) implies that, if adding the lower-priority nodes of $\mathcal{J}_3$ decreases the current delay, then adding the nodes of $\mathcal{J}_2$ (that have higher priorities than the nodes of $\mathcal{J}_3$) together with the nodes of $\mathcal{J}_3$ will further reduce the current delay. Finally, Property (d) implies that if adding the lower-priority nodes of $\mathcal{J}_3$ do not change the delay, and the weighted average delay of the nodes in $\mathcal{J}_2$ is smaller than that of the nodes in $\mathcal{J}_3$, then adding the nodes of $\mathcal{J}_2$ together with the nodes of $\mathcal{J}_3$ will decrease the current delay. Otherwise, if adding the lower-priority nodes of $\mathcal{J}_3$ do not change the delay, and the weighted average delay of the nodes in $\mathcal{J}_2$ is equal to that of the nodes in $\mathcal{J}_3$, adding the nodes of $\mathcal{J}_2$ together with the nodes of $\mathcal{J}_3$ will not change the current delay.

Using Proposition 2, we can obtain the following main result.

Proposition 3: Let $\mathcal{F}_i^* = \arg\min_{\mathcal{F}_i\subset\mathcal{C}_i} \hat{f}(\vec{\pi}_i, \mathcal{F}_i)$. Then, $\mathcal{F}_i^*$ has the following structural properties:

(a) $\mathcal{F}_i^*$ must contain all nodes j in $\mathcal{C}_i$ that satisfy $D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\mathrm{D}}$.

(b) $\mathcal{F}_i^*$ cannot contain any nodes j in $\mathcal{C}_i$ that satisfy $D_j > \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\mathrm{D}}$.

(c) If there is a node j in $\mathcal{F}_i^*$ that satisfies $D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\mathrm{D}}$, then we have $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^* \setminus \{j\})$.

The proof is provided in Appendix C. Proportion 3 implies that there must be a threshold value such that the nodes whose delay is smaller than the value should belong to the optimal forwarding set $\mathcal{F}_i^*$, and the other nodes should not. Hence, we can characterize the optimal forwarding set as $\mathcal{F}_i^* = \{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\mathrm{D}}\} \cup \mathcal{G}$, where $\mathcal{G}$ is a subset of $\{j \in \mathcal{C}_i | D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\mathrm{D}}\}$. Note that if there exists a node j such that $D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\mathrm{D}}$, then $\mathcal{F}_i^*$ is not unique. Intuitively, this means that, if such a node j wakes up first, there is no difference in the overall delay whether node i transmits a packet to this node or waits for the other nodes in $\mathcal{F}_i^*$ to wake up. On the other hand, it would be desirable to use the smallest optimal forwarding set, i.e., $\{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\mathrm{D}}\}$, in order to reduce the possibility that multiple nodes send duplicated acknowledgments. Hence, in this paper, we restrict our definition of the optimal forwarding set $\mathcal{F}_i^*$ to the following

$$\mathcal{F}_i^* = \left\{ j \in \mathcal{C}_i | D_j < \min_{\mathcal{F}\subset\mathcal{C}_i} \hat{f}(\vec{\pi}_i, \mathcal{F}) - t_{\mathrm{D}} \right\}.$$

(Recall that $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) = \min_{\mathcal{F}\subset\mathcal{C}_i} \hat{f}(\vec{\pi}_i, \mathcal{F})$.) Under this definition, the optimal forwarding set is unique and nonempty.

Since the optimal forwarding set consists of nodes whose delay is smaller than some threshold value, the simplest solution to find the optimal forwarding set is to run a linear search from the highest priority (i.e., $k = |\mathcal{C}_i|$) to the lowest priority (i.e., $k = 1$) to find the k that minimizes $\hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k})$, where $\mathcal{F}_{i,k} = \{j \in \mathcal{C}_i | b_{ij}^*(\vec{\pi}_i) \geq k\}$. The following lemma provides a stopping condition, which means that we do not need to search over all $k = 1, \cdots, |\mathcal{C}_i|$.

Lemma 1: For all $\mathcal{F} \subset \mathcal{C}_i$ that satisfies $\mathcal{F} = \{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}) - t_{\mathrm{D}}\}$, the optimal forwarding set $\mathcal{F}_i^*$ must be contained in $\mathcal{F}$, i.e., $\mathcal{F}_i^* \subset \mathcal{F}$.

Proof: From Proposition 3(a), all nodes $k \in \mathcal{F}_i^*$ satisfy $D_k < \hat{f}(\pi_i, \mathcal{F}_i^*) - t_{\mathrm{D}}$. By the definition of $\mathcal{F}_i^*$, we have $\hat{f}(\pi_i, \mathcal{F}_i^*) \leq \hat{f}(\pi_i, \mathcal{F}')$, for any subset $\mathcal{F}'$ of neighboring nodes, i.e., $\mathcal{F}' \subset \mathcal{C}_i$. Since $\mathcal{F}$ (that satisfies $\mathcal{F} = \{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}) - t_{\mathrm{D}}\}$) is also a subset of $\mathcal{C}_i$, the threshold values of $\mathcal{F}$ and $\mathcal{F}_i^*$ satisfy $\hat{f}(\pi_i, \mathcal{F}_i^*) - t_{\mathrm{D}} \leq \hat{f}(\pi_i, \mathcal{F}) - t_D$. Hence, $\mathcal{F}_i^* \subset \mathcal{F}$. ■

Lemma 1 implies that when we linearly search for the optimal forwarding set from $k = |\mathcal{C}_i|$ to $k = 1$, we can stop searching if we find the first (largest) k such that for all nodes $j \in \mathcal{F}_{i,k}$, $D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k}) - t_{\mathrm{D}}$, and for all nodes $l \notin \mathcal{F}_{i,k}$, $D_l \geq \hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k}) - t_{\mathrm{D}}$. Since all neighboring nodes are prioritized by their delays, we do not need to compare the delays of all neighboring node with the threshold value. Hence, the stopping condition can be further simplified as follows: node i searches for the largest k such that for node j with $b_{ij}^*(\vec{\pi}_i) = k$, $D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k}) - t_{\mathrm{D}}$, and for node l with $b_{il}^*(\vec{\pi}_i) = k - 1$, $D_l \geq \hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k}) - t_{\mathrm{D}}$. We refer to this line search algorithm as the LOCAL-OPT algorithm and illustrate the mechanism in Fig. 2. The complexity for finding the optimal forwarding set is $\mathcal{O}(|\mathcal{C}_i| \log |\mathcal{C}_i| + |\mathcal{C}_i|) \approx \mathcal{O}(|\mathcal{C}_i| \log |\mathcal{C}_i|)$, where the complexities for sorting the delays of $|\mathcal{C}_i|$ neighboring nodes and for running the linearly search are $\mathcal{O}(|\mathcal{C}_i| \log |\mathcal{C}_i|)$ and $\mathcal{O}(|\mathcal{C}_i|)$, respectively.

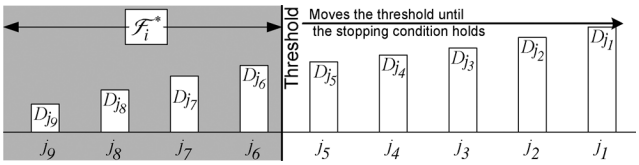


Fig. 2. The LOCAL-OPT algorithm moves the threshold from the highest-priority node j_9 to the smallest-priority node j_1 until the stopping condition is satisfied.

LOCAL-OPT Algorithm

1: **Node** i receives the delay values $\vec{\pi}_i$ of neighboring nodes

2: Assigns the optimal priorities $\vec{b}_i^*$ according to Proposition 1, and let j_k be the index of the neighboring node with priority k.

3: **Initial Setup**: $\text{prod} \leftarrow 1$ and $\text{sum} \leftarrow 0$

4: **for** $k = |\mathcal{C}_i|$ to 1 **do**

5: $\quad \text{sum} \leftarrow \text{sum} + D_{j_k} \cdot p_{j_k} \cdot \text{prod}$

6: $\quad \text{prod} \leftarrow \text{prod} \cdot (1 - p_{j_k})$

7: $\quad$ Compute (9): $f \leftarrow t_{\text{D}} + ((t_I + \text{sum})/(1 - \text{prod}))$

8: $\quad$ **if** $k > 1$ and $D_{j_{k-1}} \geq f - t_{\text{D}}$ **then**

9: $\quad\quad$ **break**

10: $\quad$ **end if**

11: **end for**

12: $\mathcal{F}_i^* \leftarrow \{j_k, j_{k+1}, \cdots, j_{|\mathcal{C}_i|}\}$

13: **return** $(\mathcal{F}_i^*, \vec{b}_i^*)$ and $\min_{\mathcal{F}, \vec{b}} f(\vec{\pi}_i, \mathcal{F}, \vec{b}) = f$

It should be noted that the optimal forwarding set is time-invariant due to the memoryless property of the Poisson random wake-up process. Specifically, the expected time for each node j in $\mathcal{C}_i$ to wake up is always t_I/p_j regardless of how long the sending node has waited. Therefore, the strategy to minimize the expected delay is also time-invariant, i.e., the forwarding set is not affected by the sequence number of the current beacon signal.

C. Globally Optimal Forwarding and Priority Matrices

Using the LOCAL-OPT algorithm, each node i can solve (10) and find $\mathcal{F}_i^{(h)}$ and $\vec{b}_i^{(h)}$ at each iteration h. We now show that the value-iteration algorithm converges to the optimal solution within N iterations.

Let $\mathbf{A}^{(h)}$ be the forwarding matrix that corresponds to $\mathcal{F}_i^{(h)}$ for all nodes $i \in \mathcal{N}$, i.e., $a_{ij}^{(h)} = 1$ if $j \in \mathcal{F}_i^{(h)}$, or $a_{ij}^{(h)} = 0$, otherwise. Similarly, let $\mathbf{B}^{(h)}$ be the priority matrix in which the transpose of the ith row is $\vec{b}_i^{(h)}$. Then, the following proposition shows the convergence of the value-iteration algorithm.

Proposition 4: For given $\vec{p}$, the delay value $D_i^{(h)}$ of each node i and the global anycast policy $(\mathbf{A}^{(h)}, \mathbf{B}^{(h)})$ converge to the minimum value $D_i^*(\vec{p})$ of the delay-minimization problem in (2) and the corresponding optimal solution $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$, respectively, as $h \to \infty$.

The proof is provided in Appendix D.

Proposition 4 shows that the converged anycast policy can minimize the delays from all nodes simultaneously. Furthermore, since the set $\mathcal{A} \times \mathcal{B}$ of admissible policies $(\mathbf{A}, \mathbf{B})$ is a finite set, Proposition 4 also implies that after some iteration h', the policy $(\mathbf{A}^{(h)}, \mathbf{B}^{(h)})$ and such an optimal policy will agree for $h > h'$.

We next show the convergence of the value-iteration algorithm within N iterations, i.e., $(\mathbf{A}^{(N)}, \mathbf{B}^{(N)}) = (\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$. To show this, we need the following result.

Proposition 5: For any awake probability vector $\vec{p}$, the anycast policy $(\mathbf{A}^{(h)}, \mathbf{B}^{(h)})$ at each iteration h does not incur any cycle in the routing paths, i.e., $g(\mathbf{A}^{(h)})$ is acyclic.

The detailed proof is provided in Appendix E. From Propositions 4 and 5, the converged policy $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$ is delay-optimal and cycle-free. The existence of this optimal cycle-free policy leads to the following proposition.

Proposition 6: For given $\vec{p}$, the value-iteration algorithm converges within N iterations, i.e., $(\mathbf{A}^{(N)}, \mathbf{B}^{(N)}) = (\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$.

Proof: Since there exists at least one optimal cycle-free policy, based on the proof in [26, p. 107], for all nodes i, the delay value $D_i^{(h)}$ converges to $\min_{(\mathbf{A},\mathbf{B})} D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ within N iterations. Then, according to [26, p. 99, Property 2.2.2(c)], the policy $(\mathbf{A}^{(h)}, \mathbf{B}^{(h)})$ also converges to the optimal anycast policy $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$ within N iterations. ■

Proposition 6 shows that the anycast policy $(\mathbf{A}^{(N)}, \mathbf{B}^{(N)})$ that the value-iteration algorithm returns corresponds to the optimal policy $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$. Furthermore, the graph $g(\mathbf{A}^*(\vec{p}))$ is acyclic. The complexity of this algorithm at each node i is given by $\mathcal{O}(N \cdot |\mathcal{C}_i| \log |\mathcal{C}_i|)$.

The value-iteration algorithm is a synchronous algorithm that requires all nodes to execute the value-iteration in locked steps. In fact, an asynchronous version of value-iteration algorithm can also be shown to converge, although the convergence will typically require more than N iterations. Furthermore, the value-iteration algorithm can be applicable to the case where there are multiple sink nodes. The details are in our technical report [21].

IV. Maximization of Network Lifetime

In the previous section, we solved the delay-minimization problem. In this section, we use the result to develop a solution to the lifetime-maximization problem $(\mathbf{P})$. From (3), the lifetime T_i and the awake probability p_i have a one-to-one mapping. Hence, we convert Problem $(\mathbf{P})$ to the following equivalent problem that controls $\overrightarrow{T} = (T_1, T_2, \cdots, T_N)$, $\mathbf{A} \in \mathcal{A}$, and $\mathbf{B} \in \mathcal{B}$:

$$(\mathbf{P1}) \quad \max_{\vec{T} \in (\mathbb{R}^+)^N, \mathbf{A}, \mathbf{B}} \quad \min_{i \in \mathcal{N}} T_i \tag{12}$$

$$\text{subject to} \quad D_i(\vec{p}, \mathbf{A}, \mathbf{B}) \leq \xi^*, \ \forall i \in \mathcal{N} \tag{13}$$

$$p_i = 1 - e^{-\frac{t_I}{e_i T_i}}, \quad \forall i \in \mathcal{N} \tag{14}$$

where $\mathbb{R}^+ = (0, \infty)$. For any given $\vec{p}$, $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$ is the optimal anycast policy that minimizes the delay from all nodes,

i.e., $D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \leq D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ for all $(\mathbf{A}, \mathbf{B})$. Hence, we can rewrite Problem $(\mathbf{P1})$ as follows:

$$(\mathbf{P2}) \max_{\vec{T} \in (\mathbb{R}^+)^N} \min_{i \in \mathcal{N}} T_i$$
$$\text{subject to} \quad D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \leq \xi^*, \ \forall i \in \mathcal{N}$$
$$p_i = 1 - e^{-\frac{t_I}{e_i T_i}}, \ \forall i \in \mathcal{N}. \quad (15)$$

Problem $(\mathbf{P2})$ can be further simplified by the following proposition.

Proposition 7: If $\overrightarrow{T}^* = (T_1^*, T_2^*, \cdots, T_N^*)$ is an optimal solution to Problem $(\mathbf{P2})$, then so is $\overrightarrow{T}$ such that $\overrightarrow{T} = (T_i = \min_k T_k^*, i \in \mathcal{N})$. In other words, according to the lifetime definition (4), it is no worse in terms of both the network lifetime and the delay to let all nodes set their lifetime to the shortest lifetime.

Proof: Since both solutions have the same objective value under our network lifetime definition in (4), it is sufficient to show that if $\overrightarrow{T}^*$ is in the feasible set, so is $\overrightarrow{T}$. Let $\vec{p}^*$ and $\vec{p}$ be the awake probability vectors that correspond to $\overrightarrow{T}^*$ and $\overrightarrow{T}$, respectively, by (14). Since p_i monotonically decreases as T_i increases, and $\overrightarrow{T} \preceq \overrightarrow{T}^*$, we have $\vec{p}^* \preceq \vec{p}$. (The symbol "$\preceq$" denotes component-wise inequality, i.e., if $\overrightarrow{T} \preceq \overrightarrow{T}^*$, then $T_i \leq T_i^*$ for all $i \in \mathcal{N}$.)

We next show that the delay $D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$ from each node i is a nonincreasing function with respect to each component of $\vec{p}$. For given $(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$, assume that node j increases its awake probability p_j to p_j'. Let $\vec{p}'$ be the corresponding global awake probability vector. Since the increased awake probability p_j' does not increase the one-hop delay of nodes for the fixed anycast policy $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$, we have

$$D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \geq D_i(\vec{p}', \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$$
$$\geq D_i(\vec{p}', \mathbf{A}^*(\vec{p}'), \mathbf{B}^*(\vec{p}')). \quad (16)$$

The last inequality in (16) is due to the delay-optimality of $(\mathbf{A}^*(\vec{p}'), \mathbf{B}^*(\vec{p}'))$ for $\vec{p}'$. Hence, the delay $D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$ is nonincreasing with respect to each component of $\vec{p}$. Since $\vec{p}^* \preceq \vec{p}$, for all nodes i, we have $D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \leq D_i(\vec{p}^*, \mathbf{A}^*(\vec{p}^*), \mathbf{B}^*(\vec{p}^*))$. Hence, if $\overrightarrow{T}^*$ satisfies (15), so does $\overrightarrow{T}$. ∎

Using the above proposition, we can rewrite problem $(\mathbf{P2})$ into the following problem with one-dimensional variable T that corresponds to the network lifetime:

$$(\mathbf{P3}) \max_{T \in \mathbb{R}^+} T$$
$$\text{subject to} \quad \max_{i \in \mathcal{N}} D_i(\vec{p}, \mathbf{A}(\vec{p}), \mathbf{B}(\vec{p})) \leq \xi^*$$
$$p_i = 1 - e^{-\frac{t_I}{e_i T}}, \quad \forall i \in \mathcal{N}.$$

If T^* is the solution to Problem $(\mathbf{P3})$, then $(\vec{p}^*, \mathbf{A}(\vec{p}^*), \mathbf{B}(\vec{p}^*))$, where $p_i^* = 1 - e^{-(t_I/e_i T^*)}$ corresponds to the solution to the original problem $(\mathbf{P})$.

Note that $\max_{i \in \mathcal{N}} D_i(\vec{p}, \mathbf{A}(\vec{p}), \mathbf{B}(\vec{p}))$ is nonincreasing with respect to each component of $\vec{p}$. (See the proof of Proposition 7.) Since each component of $\vec{p}$ is a decreasing function of T, $\max_{i \in \mathcal{N}} D_i(\vec{p}, \mathbf{A}(\vec{p}), \mathbf{B}(\vec{p}))$ is increasing as T increases. Hence, we can develop an efficient binary search algorithm (see the Binary Search Algorithm that follows) for computing the optimal value of T^* such that $\max_{i \in \mathcal{N}} D_i(\vec{p}^*, \mathbf{A}(\vec{p}^*), \mathbf{B}(\vec{p}^*)) = \xi^*$.

Binary Search Algorithm

1: **Initial Setup: The sink** sets $T^{(1)}$ to a half of the maximum possible lifetime and sets $T_{\text{record}} \leftarrow 0$

2: **for** $m = 1$ to $m_{\max}$ **do**

3: **Every node** i computes $p_i^{(m)} = 1 - e^{-(t_I/(T^{(m)} e_i))}$, and

4: runs the value-iteration algorithm for $(p_j^{(m)}, j \in \mathcal{C}_i)$.

5: **Nodes** i that satisfy $D_i^* > D_j^*$ for all neighboring nodes j send feedback of their delay values D_i^* to the sink.

6: **The sink** sets $D_{\max} \leftarrow \max_i D_i^*$, and

7: **if** $D_{\max} > \xi^*$ **then**

8: $T^{(m+1)} \leftarrow T^{(m)} - (T^{(1)}/2^m)$.

9: **else**

10: $T^{(m+1)} \leftarrow T^{(m)} + (T^{(1)}/2^m)$ and $T_{\text{record}} \leftarrow T^{(m)}$.

11: **end if**

12: **end for**

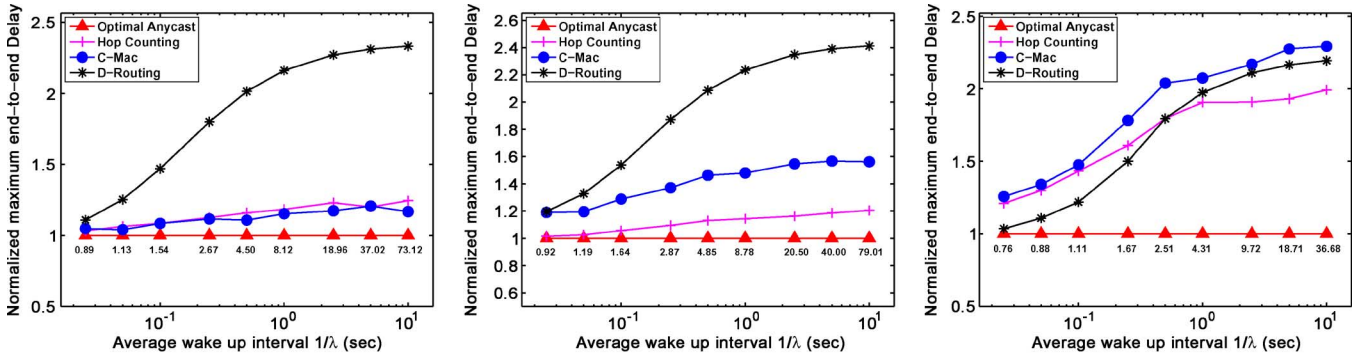
13: **return** $T^* \leftarrow T_{\text{record}}$

After $m_{\max}$ iterations of the Binary Search Algorithm, the difference between the optimal lifetime T^* and the algorithm output T_{record} is smaller than $T^{(1)}/2^{m_{\max}}$. If we want to make this difference less than ϵ, the complexity of the Binary Search Algorithm is $\mathcal{O}(\log(1/\epsilon) \cdot N \cdot \max_i |\mathcal{C}_i| \log \max_i |\mathcal{C}_i|)$. Note that, in **Line 5**, only those nodes that do not belong to the forwarding sets of any other nodes need to send the feedback delay to the sink s. (There must exist at least one such node because of the acyclic property of the value-iteration algorithm in Proposition 6.) According to Property (a) in Proposition 3, if node j belongs to the forwarding set of node i under the value-iteration algorithm, the delay of node j plus t_{D} is smaller than that of node i. Since sink s only needs to know the maximum delay, there is no need for such nodes j to feedback their delays, which reduces the communication overhead.

V. Simulation Results

In this section, we provide simulation results to compare the performance of the optimal anycast algorithm and the following algorithms.

C-MAC: The *C-MAC* algorithm proposed in [14] is an anycast-based heuristic that exploits geographic information to reduce the delay from each node. Let d_i be the Euclidean distance from node i to sink s. Furthermore, let r_{ij} be the geographical progress toward the sink, i.e., if node i forwards the packet to node j, the progress is defined as $r_{ij} = d_i - d_j$. If a node has a packet, let D be the one-hop delay from node i to a next-hop node, and let R be the progress between two nodes. Since node i selects the next-hop node probabilistically, both D and R are random variables. The objective of the C-MAC algorithm is to find the forwarding set that minimizes the expectation of normalized one-hop delay, i.e., $E[D/R]$. The idea behind this



(a) Nodes with the same wake-up rate are uniformly deployed

(b) Nodes with the same wake-up rate are non-uniformly deployed

(c) Nodes with the different wake-up rates are non-uniformly deployed

Fig. 3. The maximum end-to-end delay under each algorithm normalized by that under "Optimal anycast" when (a) 400 nodes with the same wake-up rate λ are uniformly deployed, (b) 391 nodes with the same wake-up rate λ are deployed forming a connectivity hole, and (c) 391 nodes with different wake-up rates are deployed forming a connectivity hole. The numbers beneath the line of "Optimal Anycast" are the delay values (in seconds) under the optimal anycast algorithm.

algorithm is to minimize the expected delay per unit distance of progress, which might help to reduce the actual end-to-end delay.[5]

Hop-counting Algorithm: For comparison, we have developed a heuristic hop-counting algorithm that exploits the hop count (the minimum number of hops to reach the sink) of neighboring nodes to reduce the end-to-end delay. The objective of this algorithm is to minimize the time for a packet to advance one hop closer to the sink. This algorithm is inspired by the original hop-counting algorithms in [19].[6] If an h-hop node i has a packet to transmit, it waits until any $(h-1)$- or h-hop neighboring node wakes up. If an $(h-1)$-hop node wakes up first, then the packet is transmitted to the $(h-1)$-hop node. If an h-hop node j wakes up first, node i has to decide whether it transmits the packet to node j or it waits for an $(h-1)$-hop node to wake up. Such a decision is made by comparing the corresponding expected delays. If node j is chosen, the expected delay is given by $t_{\rm D}+(t_I/(1-\prod_{j'\in C_j^{(h-1)}}(1-p_{j'})))+t_{\rm D}$. (The three terms in the summation correspond to the time to transmit the packet to node j, the expected time for node j to wait for another $(h-1)$-hop neighboring node to wake up, and the time to transmit the packet to the $(h-1)$-hop node, respectively, where $C_j^{(h-1)}$ is the set of $(h-1)$-hop neighboring nodes of node j.) If node i waits for an $(h-1)$-hop node, the expected delay is $(t_I/(1-\prod_{j'\in C_i^{(h-1)}}(1-p_{j'})))+t_{\rm D}$. Hence, node i chooses the decision with the smaller expected delay.

Deterministic Routing (D-Routing): By *deterministic routing*, we mean that each node has only one designated next-hop forwarding node. To find the delay-optimal routing path, we use the well-known Bellman–Form algorithm, in which the length of each link (i,j) is given by the expected one-hop delay $t_I/p_j+t_{\rm D}$. Comparing this algorithm with the others, we will study how exploiting path diversity can help to reduce the end-to-end delay.

A. Case 1: Uniformly Deployed Homogeneous Nodes

We first simulate a wireless sensor network with 400 uniformly deployed nodes over a 1×1 km^2 area with the sink s located at the lower left corner. We assume that the transmission range from each node i is a disc with radius 100 m. The parameters t_I and $t_{\rm D}$ are set to 6 and 30 ms, respectively. We also assume that all nodes are homogeneous, i.e., all nodes have the same wake-up rate λ. For each value of the wake-up interval $1/\lambda$, we measure the expected end-to-end delay from all nodes under each algorithm. Among the expected delays of all nodes, we pick the maximum expected delay under each algorithm and report it in Fig. 3(a), where the x-axis represents different average wake-up intervals $1/\lambda$, and the y-axis represents the maximum expected end-to-end delay. For ease of comparison, we normalized the maximum end-to-end delay under each algorithm by that under our "Optimal anycast" algorithm, i.e., the value-iteration algorithm and the LOCAL-OPT algorithm. The numbers beneath the line of "Optimal anycast" are the average end-to-end delay value (in seconds) under the optimal anycast algorithm. From Fig. 3(a), we observe that all algorithms that exploit path diversity significantly reduce the delay compared with the deterministic routing algorithm. We also observe that the performance of the optimal anycast, the hop-counting, and the C-MAC algorithms are very close. This result implies that the hop count and progress are strongly correlated with the end-to-end delay when nodes are deployed uniformly.

B. Case 2: Homogeneous Nodes With a Connectivity Hole

We next simulate a topology where there is a connectivity hole in the sensor field as shown in Fig. 4(a). This is motivated by practical scenarios, where there are obstructions in the sensor field, e.g., a lake or a mountain where sensor nodes cannot be deployed. The simulation result based on this topology is provided in Fig. 3(b). From this figure, we observe that the optimal anycast algorithm substantially outperforms the C-MAC. Fig. 4(a) provides us with the intuition for this performance gap. We plot the routing paths from the nodes with the largest

[5]The progress-based C-MAC algorithm may fail when a packet reaches a node that does not have any neighboring nodes with positive progress. To avoid this *deadlock* problem, we modified the original C-MAC algorithm slightly such that the packet is then forwarded using the hop-counting algorithm for a small number of hops in order to escape the region.

[6]Note that the algorithms in [19] and [20] do not directly apply to anycast in asynchronous sleep–wake scheduling. The model in [19] and [20] requires the assignment of a state-dependent cost metric to each link, and the algorithms there choose a next-hop node that incurs the minimum cost metric for the current state. If we interpret the current state as the set of nodes that are awake, the optimal anycast decision may still be to wait for other nodes that are not awake. However, for these nodes, their cost cannot be defined for the current state because their exact wake-up time (in the future) is unknown.

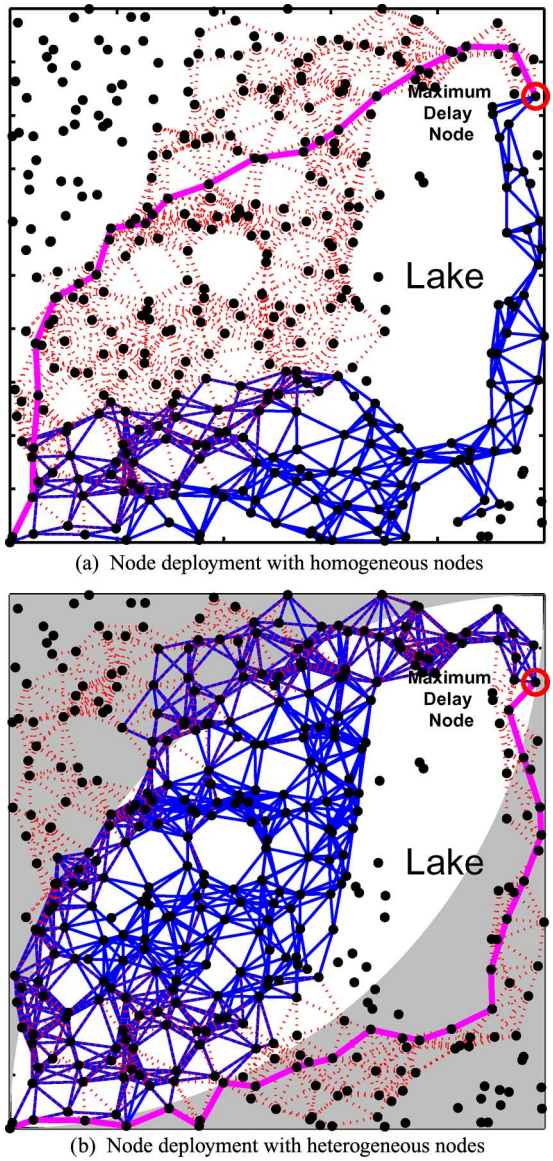


Fig. 4. (color online) Node deployment and routing paths when (a) all nodes have the same wake-up rate λ (homogeneous), and (b) the nodes in the shaded area have the wake-up rate 3λ, while the other nodes have the wake-up rates of λ (heterogeneous). The dotted lines illustrate frequently used routing paths under the optimal anycast algorithm, the thick solid lines in red illustrate the frequently used routing paths under the deterministic routing algorithm, and thin solid lines in blue illustrate the frequently used routing paths under (a) the C-MAC algorithm and (b) the hop-counting algorithm.

delay. The dotted lines (above the hole) illustrate the frequently used routing paths under the optimal anycast algorithm. The thick solid lines (in red, above the hole) illustrate the unique routing path under the deterministic routing algorithm. The thin solid lines (in blue, below the hole) illustrate the frequently used routing paths under the C-MAC algorithm. In our optimal algorithm, in order to reduce the delay, a packet is first forwarded to neighbors with negative progress but smaller end-to-end delay. However, under the C-MAC, all packet are forwarded only to nodes with positive progress, and hence they take longer detours. Therefore, the result of Fig. 4(a) shows that when the node distribution is not uniform, the correlation between progress and delay becomes much weaker. Thus, the anycast-based heuristic algorithms depending only on geographical information could perform poorly. Unlike the C-MAC algorithm, we observe that the connectivity hole does not significantly affect the performance of the hop-counting algorithm, as the hop-count is a global metric that depends on connectivity of nodes, rather than their geographical locations.

C. Case 3: Heterogeneous Nodes With a Connectivity Hole

So far, we have assumed that all nodes have the same wake-up rate λ. We now simulate the case of heterogeneous wake-up rates: We set the nodes in the shaded area in Fig. 4(b) to have the wake-up rate 3λ while the other nodes still have the wake-up rate λ. This kind of diversity can occur when nodes are deployed with different amount of initial energy or are deployed at different times. In this case, the nodes in the unshaded area have to reduce their wake-up rates to save energy. The simulation result based on this setting is provided in Fig. 3(c). From this figure, we observe that the optimal anycast algorithm can significantly reduce the delay compared with the hop-counting algorithm. We find the reason from Fig. 4(b), in which the thin solid lines (in blue) now illustrate the frequently used routing paths under the hop-counting algorithm. Compared to the routing paths in Fig. 4(a), our optimal algorithm now sends packets through the nodes near the border to take advantage of the high wake-up rates for delay reduction. However, the hop-counting algorithm still uses similar routing paths as those of the optimal algorithm in Fig. 4(a), where the wake-up rates are homogeneous. This result shows that when the wake-up rates are heterogeneous, the correlation between the hop count and delay becomes weak, and forwarding packets to a larger hop count node [e.g., the routing paths that circumvent the unshaded area in Fig. 4(b)] may help to reduce the end-to-end delay. We next plot the network lifetime in Fig. 5(a) by solving the lifetime-maximization problem. (We assume that the initial amount of energy in the shaded area is given by one-third of that in the unshaded area, and nodes consume the same amount of energy each time they wake up.) From the result, we observe that our algorithm can extend the network lifetime over the other algorithms. Finally, to observe the impact of node density to delay-performance, we simulate Case 3 again by changing the number of deployed nodes. Fig. 5(b) shows that our algorithm outperforms the other algorithms regardless of node density.

VI. Conclusion

In this paper, we develop an anycast packet-forwarding scheme to reduce the event-reporting delay and to prolong the lifetime of wireless sensor networks employing asynchronous sleep–wake scheduling. Specifically, we study two optimization problems. First, when the wake-up rates of the sensor nodes are given, we develop an efficient and distributed algorithm to minimize the expected event-reporting delay from all sensor nodes to the sink. Second, using a specific definition of the network lifetime, we study the lifetime-maximization problem to optimally control the sleep–wake scheduling policy and the anycast policy in order to maximize the network lifetime subject to an upper limit on the expected end-to-end delay. Our numerical results suggest that the proposed solution can substantially outperform prior heuristic solutions in the literature under practical scenarios where there are obstructions in the coverage area of the wireless sensor network. For future

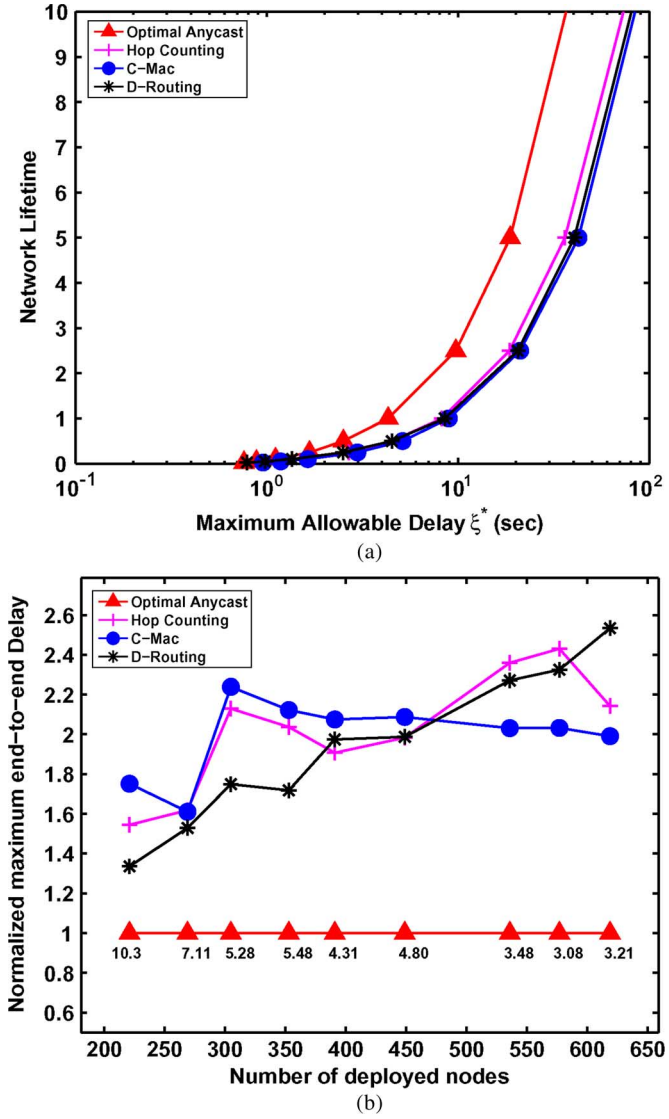


Fig. 5. (a) The network lifetime subject to different allowable delay ξ^*, and (b) the normalized end-to-end delay under different node density for the case of Fig. 3(c). In (a), the network lifetime under "D-routing" almost overlaps with that under "C-MAC."

work, we plan to generalize our solution to take into account non-Poisson wake-up processes and other lifetime definitions.

Appendix A
Proof of Proposition 1

Proof: We consider any $\vec{b}_i$ with which there exists a pair of nodes j_1 and j_2 such that $b_{ij_1} < b_{ij_2}$ and $D_{j_1} < D_{j_2}$. Without loss of generality, we assume that $\mathcal{C}_i = \{1, 2, \cdots, K\}$, and we sort the nodes such that $b_{ik} > b_{i,k+1}$ for $k = 1, 2, \cdots, K-1$. Then, there must exist a pair of nodes l and m such that $m = l+1$ and $D_m < D_l$. Let $\vec{b}'_i$ be the priority assignment when we interchange the priorities of nodes l and m, i.e., $b'_{il} = b_{im}$, $b'_{im} = b_{il}$, and $b'_{ij} = b_{ij}$ if $j \neq l, m$. For any forwarding set $\mathcal{F}_i$, let $p'_j = p_j \mathbf{1}_{\{j \in \mathcal{F}_i\}}$, where $\mathbf{1}_{\{j \in \mathcal{F}_i\}} = 1$ if $j \in \mathcal{F}_i$ and $\mathbf{1}_{\{j \in \mathcal{F}_i\}} = 0$, otherwise. Then, we can rewrite (9) as follows: $f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i) = t_D + ((t_I + \sum_{j=1}^{K} D_j p'_j \prod_{k=1}^{j-1}(1-p'_k))/(1 - \prod_{j=1}^{K}(1-p'_j)))$.

Using the above,

$$
\begin{aligned}
&f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i) - f\left(\vec{\pi}_i, \mathcal{F}_i, \vec{b}'_i\right) \\
&= \frac{1}{1 - \prod_{j=1}^{K}(1-p'_j)} \\
&\quad \times \Bigg[D_l p'_l \prod_{j=1}^{l-1}(1-p'_k) + D_m p'_m \prod_{j=1}^{l}(1-p'_k) \\
&\qquad - D_m p'_m \prod_{j=1}^{l-1}(1-p'_k) - D_l p'_l \prod_{j=1}^{l-1}(1-p'_k)(1-p'_m) \Bigg] \\
&= \frac{p'_l p'_m \prod_{j=1}^{l-1}(1-p'_k)}{1 - \prod_{j=1}^{K}(1-p'_j)}(D_l - D_m) \geq 0. \qquad (17)
\end{aligned}
$$

In other words, the local delay function does not increase after we switch the priorities. If we repeatedly apply the above priority-switching procedure on the node with the smallest delay, the node will eventually be assigned the highest priority. In the meantime, the local delay function will not increase. Similarly, we can apply the iterative switching procedures on the node with the second smallest delay, the node with the third smallest delay, and so forth. In the end, the priority assignment will be equal to $\vec{b}^*_i$, and the local delay function value will not increase, i.e., $f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}^*_i) \leq f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i)$. Therefore, for all $\vec{b}_i$, $f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}^*_i) \leq f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i)$ holds. ■

Appendix B
Proof of Proposition 2

Proof: This proposition can be shown by noting that each node set $\mathcal{J}_k$ ($k = 1, 2, 3$) can be regarded as a node with delay D_{J_k} and awake probability $P_{J_k} = 1 - \prod_{j \in \mathcal{J}_k}(1-p_j)$. The probability P_{J_k} is the probability that any node in $\mathcal{J}_k$ wakes up. Then, from (9), we have

$$
\begin{aligned}
&\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \qquad (18) \\
&\Leftrightarrow \frac{t_I + \sum_{j \in \mathcal{J}_1 \cup \mathcal{J}_3} D_j p_j \prod_{k \in (\mathcal{J}_1 \cup \mathcal{J}_3): b^*_{ij} < b^*_{ik}}(1-p_k)}{1 - \prod_{j \in \mathcal{J}_1 \cup \mathcal{J}_3}(1-p_j)} \\
&\quad < \frac{t_I + \sum_{j \in \mathcal{J}_1} D_j p_j \prod_{k \in \mathcal{J}_1: b^*_{ij} < b^*_{ik}}(1-p_k)}{1 - \prod_{j \in \mathcal{J}_1}(1-p_j)} \\
&\Leftrightarrow \frac{t_I + D_{J_1}P_{J_1} + D_{J_3}P_{J_3}(1-P_{J_1})}{1-(1-P_{J_1})(1-P_{J_3})} < \frac{t_I + D_{J_1}P_{J_1}}{P_{J_1}} \\
&\Leftrightarrow D_{J_3} < \frac{t_I + D_{J_1}P_{J_1}}{P_{J_1}} = \hat{f}(\vec{\pi}_i, \mathcal{J}_1) - t_{\mathrm{D}} \qquad (19) \\
&\Leftrightarrow D_{J_3}(1-(1-P_{J_1})(1-P_{J_3}) - P_{J_3}(1-P_{J_1})) \\
&\quad < t_I + D_{J_1}P_{J_1} \\
&\Leftrightarrow D_{J_3} < \frac{t_I + D_{J_1}P_{J_1} + D_{J_3}P_{J_3}(1-P_{J_1})}{1-(1-P_{J_1})(1-P_{J_3})} \\
&\Leftrightarrow D_{J_3} < \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) - t_{\mathrm{D}}. \qquad (20)
\end{aligned}
$$

This proves Property (a). The proof of Property (b) is similar (by replacing all "<" by "="). Using the similar methods, we can also prove Properties (c) and (d). (See the detailed proof in [21, Appendix B].) ■

APPENDIX C
PROOF OF PROPOSITION 3

Proof: We prove this proposition by contradiction. In order to prove Property (a), assume that there exists a node j such that $D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\rm D}$ and $j \notin \mathcal{F}_i^*$. There are two cases. **Case 1**: If $b_{ij}^*(\vec{\pi}_i) < b_{ik}^*(\vec{\pi}_i)$ for all nodes k in $\mathcal{F}_i^*$, let $\mathcal{J}_1 = \mathcal{F}_i^*$ and $\mathcal{J}_3 = \{j\}$. Then, since $D_j < \hat{f}(\vec{\pi}_i, \mathcal{J}_1) - t_{\rm D}$, we have $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1)$ by Property (a) in Proposition 2. This contradicts to the fact that $\mathcal{F}_i^*$ is the optimal forwarding set. **Case 2**: If there exists node k in $\mathcal{F}_i^*$ such that $b_{ik}^*(\vec{\pi}_i) < b_{ij}^*(\vec{\pi}_i)$, let $\mathcal{J}_1 = \{l \in \mathcal{F}_i | b_{il}^*(\vec{\pi}_i) > b_{ij}^*(\vec{\pi}_i)\}$, $\mathcal{J}_2 = \{j\}$, and $\mathcal{J}_3 = \{l \in \mathcal{F}_i | b_{il}^*(\vec{\pi}_i) < b_{ij}^*(\vec{\pi}_i)\}$. Note that $\mathcal{J}_1 \cup \mathcal{J}_3 = \mathcal{F}_i^*$. If $\mathcal{J}_1$ is an empty set, we assume a virtual node 0 such that $b_{i0}^*(\vec{\pi}_i) = b_{ij}^*(\vec{\pi}_i)+1$, $D_0 < D_j$, and $p_0 \to 0$, and let $\mathcal{J}_1 = \{0\}$. The idea behind this assumption is that introducing a hypothetical node that wakes up with infinitesimal probability does not change the delay analysis. Since $\mathcal{F}_i^* = \mathcal{J}_1 \cup \mathcal{J}_2$ is optimal, we must have $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) \le \hat{f}(\vec{\pi}_i, \mathcal{J}_1)$. **Case 2-1**: If $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1)$, then by Property (c) in Proposition 2, $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$. This is a contradiction because $\mathcal{J}_1 \cup \mathcal{J}_3 = \mathcal{F}_i^*$ is by assumption the optimal forwarding set. **Case 2-2**: If $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) = \hat{f}(\vec{\pi}_i, \mathcal{J}_1)$, then by Property (b) in Proposition 2, $D_{J_3} = \hat{f}(\vec{\pi}_i, \mathcal{J}_1) - t_{\rm D} = \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) - t_D > D_j = D_{J_2}$. By Property (d) in Proposition 2, $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$. This is also a contradiction. Therefore, such node j must be in $\mathcal{F}_i^*$.

In order to prove Property (b), suppose in contrary that there exists a node j in $\mathcal{F}_i^*$ such that $D_j > \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\rm D}$. Let $\mathcal{J}_1 = \{l \in \mathcal{F}_i^* | D_l \le \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\rm D}\}$ and $\mathcal{J}_3 = \{l \in \mathcal{F}_i^* | D_l > \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\rm D}\}$. Then, the weighted average delay in $\mathcal{J}_3$ is larger than $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\rm D}$, i.e., $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) - t_{\rm D} < D_{J_3}$. By Properties (a) and (b) in Proposition 2, we have $f(\vec{\pi}_i, \mathcal{J}_1) < f(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$. Since $\mathcal{F}_i^* = \mathcal{J}_1 \cup \mathcal{J}_3$, this leads to a contradiction. Therefore, such a node j must not be in $\mathcal{F}_i^*$.

To prove Property (c), let node j in $\mathcal{F}_i^*$ have the highest priority among nodes that satisfy $D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\rm D}$. We then need to show that $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^* \setminus \{j\})$. We let $\mathcal{J}_1 = \mathcal{F}_i^* \setminus \{j\}$ and $\mathcal{J}_3 = \{j\}$. From Property (b), $\mathcal{J}_1$ does not contain any nodes with a higher delay than $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_{\rm D}$, which implies node j is the highest-priority node in $\mathcal{F}_i^*$. Then, by Property (b) in Proposition 2, we have $\hat{f}(\vec{\pi}_i, \mathcal{J}_1) = \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$, where $\mathcal{J}_1 \cup \mathcal{J}_3 = \mathcal{F}_i^*$ and $\mathcal{J}_1 = \mathcal{F}_i^* \setminus \{j\}$. Therefore, the result of Property (c) follows. ∎

APPENDIX D
PROOF OF PROPOSITION 4

Proof: The value-iteration algorithm is a classic value-iteration for solving Shortest Stochastic Problem (SSP) problems. Specifically, we map the delay-minimization problem to the SSP problem as follows. Consider the following Markov chain that corresponds to the process with which a packet is forwarded under a given anycast policy. There are states $1, 2, \cdots, N$, and s, where state i represents that a packet is in node i. A state transition occurs from state i to state j when node i forwards the packet to node j. (We do not consider self-transitions.) State s is the *absorbing* state (it is also called the *termination state* in [26]), where state transition ends. Under the anycast forwarding policy $(\mathbf{A}, \mathbf{B})$, a packet at node i will be forwarded to neighboring node j with probability $q_{i,j}(\vec{p}, \mathbf{A}, \mathbf{B})$ given by (5). The cost associated with the transition from node i to any neighboring node corresponds to the expected one-hop delay $d_i(\vec{p}, \mathbf{A}, \mathbf{B})$ given by (6). The total cost, which is the expectation of the accumulated costs from initial state i to sink s, corresponds to the expected end-to-end delay $D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ from node i to the sink s. Then, the evolution of $D_i^{(h)}$ corresponds to the value iteration in [26, p. 95]. Hence, according to [26, Proposition 2.2.2], we have $D_i^{(h)} \to D_i^*(\vec{p})$ and $(\mathbf{A}^{(h)}, \mathbf{B}^{(h)}) \to (\mathbf{A}^*, \mathbf{B}^*)$ such that $D_i(\vec{p}, \mathbf{A}^*, \mathbf{B}^*) = D_i^*(\vec{p})$ for all nodes i, as $h \to \infty$. ∎

APPENDIX E
PROOF OF PROPOSITION 5

Proof: To prove this proposition, we first show by contradiction that $D_i^{(h)}$ does not increase with h. Assume that there is a node i whose delay value $D_i^{(h')}$ increases at some iteration $h' > 0$, i.e., $D_i^{(h')} > D_i^{(h'-1)}$. Then, there must exist a neighboring node j in $\mathcal{F}_i^{(h'-1)}$ whose delay value $D_j^{(h'-1)}$ has increased at iteration $h'-1$. (Otherwise, i.e., if $D_j^{(h'-1)} \le D_j^{(h'-2)}$ for all nodes $j \in \mathcal{F}_i^{(h'-1)}$, taking $\mathcal{F}_i^{(h')} = \mathcal{F}_i^{(h'-1)}$ at iteration h' leads to at least the same delay value of node i, i.e., $D_i^{(h')} \le D_i^{(h'-1)}$.) Applying the same method iteratively, we can find a sequence of nodes $i = i_{h'}, i_{h'-1}, \cdots, i_1$ such that $i_{h-1} \in \mathcal{F}_{i_h}^{(h-1)}$ and $D_{i_h}^{(h)} > D_{i_h}^{(h-1)}$ for $h = 1, 2, \cdots, h'$. If node i_1 is the sink, then this is a contradiction because $D_s^{(h)} = 0$ for all h. If node i_1 is not the sink, then we have $D_{i_1}^{(0)} = \infty$. Then, node i_1 cannot be selected as an eligible forwarder of node i_2 at iteration 1, i.e., $i_1 \notin \mathcal{F}_{i_2}^{(1)}$, which is also a contradiction. Hence, for all nodes i, $D_i^{(h)}$ does not increase with h.

Using this result, we show that $g(\mathbf{A}^{(h)})$ is acyclic. According to Proposition 3, a neighboring node j in $\mathcal{F}_i^{(h)}$ must satisfy $D_j^{(h-1)} < D_i^{(h)} - t_{\rm D}$. From the preceding proof, we also have $D_j^{(h-1)} < D_i^{(h-1)} - t_{\rm D}$. This implies that under anycast policy $(\mathbf{A}^{(h)}, \mathbf{B}^{(h)})$, the delay value $D_i^{(h-1)}$ decreases by at least $t_{\rm D}$ along each possible routing path. Hence, the result follows. ∎

REFERENCES

[1] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "On maximizing the lifetime of delay-sensitive wireless sensor networks with anycast," in *Proc. IEEE INFOCOM*, Pheonix, AZ, Apr. 2008, pp. 807–815.

[2] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," *Comput. Netw.*, vol. 43, pp. 317–337, Oct. 2003.

[3] W. Ye, H. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[4] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. SenSys*, Nov. 2003, pp. 171–180.

[5] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in *Proc. IPDPS*, Apr. 2004, pp. 224–231.

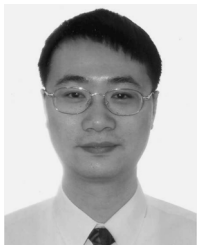
[6] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.

[7] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *Proc. MobiCom*, 2001, pp. 272–287.

[8] M. Nosovic and T. Todd, "Low power rendezvous and RFID wakeup for embedded wireless networks," presented at the IEEE Comput. Commun. Workshop, 2000.
[9] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE Trans. Mobile Comput.*, vol. 1, no. 1, pp. 70–80, Jan.–Mar. 2002.
[10] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proc. SenSys*, Nov. 2004, pp. 95–107.
[11] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, "A unifying link abstraction for wireless sensor networks," in *Proc. SenSys*, Nov. 2005, pp. 76–89.
[12] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for ad hoc and sensor networks: Energy and latency performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 349–365, Oct.–Dec. 2003.
[13] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for ad hoc and sensor networks: Multihop performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 337–348, Oct.–Dec. 2003.
[14] S. Liu, K.-W. Fan, and P. Sinha, "CMAC: An energy efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks," in *Proc. SECON*, San Diego, CA, Jun. 2007, pp. 11–20.
[15] R. R. Choudhury and N. H. Vaidya, "MAC-layer anycasting in ad hoc networks," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 75–80, Jan. 2004.
[16] S. Jain and S. R. Das, "Exploiting path diversity in the link layer in wireless ad hoc networks," in *Proc. WoWMoM*, Jun. 2005, pp. 22–30.
[17] P. Larsson and N. Johansson, "Multiuser diversity forwarding in multihop packet radio networks," in *Proc. IEEE WCNC*, 2005, vol. 4, pp. 2188–2194.
[18] S. Biswas and R. Morris, "ExOR: Opportunistic multi-hop routing for wireless networks," in *Proc. ACM SIGCOMM*, Oct. 2005, vol. 35, pp. 133–144.
[19] M. Rossi and M. Zorzi, "Integrated cost-based MAC and routing techniques for hop count forwarding in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 4, pp. 434–448, Apr. 2007.
[20] M. Rossi, M. Zorzi, and R. R. Rao, "Statistically Assisted Routing Algorithm (SARA) for hop count based forwarding in wireless sensor networks," *Wireless Netw.*, vol. 14, pp. 55–70, Feb. 2008.
[21] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "Minimizing delay and maximizing lifetime for wireless sensor networks with anycast," Purdue University, Tech. Rep., 2008 [Online]. Available: http://web.ics.purdue.edu/~kim309/Kim08tech2.pdf
[22] "IRIS OEM module datasheet," Crossbow Technology, Tech. Rep. [Online]. Available: http://www.xbow.com
[23] J.-H. Chang and L. Tassiulas, "Routing for maximum system lifetime in wireless ad-hoc networks," in *Proc. 37th Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, Oct. 1999, pp. 1191–1200.
[24] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *Proc. IEEE INFOCOM*, Mar. 2000, vol. 1, pp. 22–31.
[25] Y. T. Hou, Y. Shi, and H. D. Sherali, "Rate allocation in wireless sensor networks with network lifetime requirement," in *Proc. IEEE/ACM MobiHoc*, 2004, pp. 67–77.
[26] D. P. Bertsekas, *Dynamic Programming and Optimal Control Vol. 2*, 3rd ed. Belmont, MA: Athena Scientific, 2007.

Joohwan Kim (S'07) received the B.S. degree from Yonsei University, Seoul, Korea, in 2004, and the M.S. degree from Purdue University, West Lafayette, IN, in 2006.

He is currently a Ph.D. candidate of Electrical and Computer Engineering at Purdue University. His research interests range over the various areas of wireless communication networks: scheduling, routing, power control, network pricing, and wireless resource optimization in sensor and mobile ad hoc networks.



Xiaojun Lin (S'02–M'05) received the B.S. degree from Zhongshan University, Guangzhou, China, in 1994, and the M.S. and Ph.D. degrees from Purdue University, West Lafayette, IN, in 2000 and 2005, respectively.

He is currently an Assistant Professor of electrical and computer engineering at Purdue University. His research interests are resource allocation, optimization, network pricing, routing, congestion control, network as a large system, cross-layer design in wireless networks, and mobile ad hoc and sensor networks.

Ness B. Shroff (S'91–M'93–SM'01–F'07) received the Ph.D. degree from Columbia University, New York, NY, in 1994.

He joined Purdue University, West Lafayette, IN, as an Assistant Professor after receiving the Ph.D. degree. At Purdue, he became Professor of the School of Electrical and Computer Engineering in 2003, and Director of the Center for Wireless Systems and Applications (CWSA) in 2004. In 2007, he joined The Ohio State University, Columbus, as the Ohio Eminent Scholar of Networking and Communications and Professor of electrical and computer engineering and computer science and engineering. His research interests span the areas of wireless and wireline communication networks. He is especially interested in fundamental problems in the design, performance, control, and security of these networks.

Dr. Shroff received the IEEE INFOCOM 2008 Best Paper Award, the IEEE INFOCOM 2006 Best Paper Award, the IEEE IWQoS 2006 Best Student Paper Award, the 2005 Best Paper of the Year Award for the *Journal of Communications and Networking*, the 2003 Best Paper of the Year Award for *Computer Networks*, and the NSF CAREER Award in 1996 (his INFOCOM 2005 paper was also selected as one of two runner-up papers for the Best Paper Award). He has served on the editorial boards of the IEEE/ACM TRANSACTIONS ON NETWORKING, *Computer Networks*, and IEEE COMMUNICATIONS LETTERS. He was the Technical Program Co-Chair of IEEE INFOCOM '03, IEEE CCW '99, the Program Co-Chair for the symposium on high-speed networks, Globecom 2001, and the panel co-chair for ACM MobiCom '02. He was also a Co-Organizer of the NSF workshop on Fundamental Research in Networking, held in Airlie House, Warrenton, VA, in 2003. In 2008, he served as the Technical Program Co-Chair of ACM MobiHoc and as General Co-Chair of WICON.

Prasun Sinha received the B.Tech. degree from the Indian Institute of Technology—Delhi, New Delhi, India, in 1995, the M.S. degree from Michigan State University in 1997, and the Ph.D. degree from the University of Illinois, Urbana-Champaign, in 2001.

He worked at Bell Labs, Lucent Technologies, Holmdel, NJ, as a Member of Technical Staff from 2001 to 2003. Since 2003, he has been an Assistant Professor with the Department of Computer Science and Engineering, The Ohio State University, Columbus. His research focuses on design of network protocols for sensor networks and mesh networks.