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 an arrival to occur. 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 the practical scenarios where there are obstructions, e.g., a lake or a mountain, in the coverage area of wireless sensor networks.**

***Index Terms*—Anycast, Sleep-wake scheduling, Sensor network, Energy-efficiency, Delay**

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 will focus on event-driven asynchronous sensor networks, where events occur rarely. This is an important class of sensor networks and 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 actual data transmission and reception. The fraction of total energy consumption for actual 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 procedure could incur additional communication overhead, and consume a considerable amount of energy. On-demand sleep-wake scheduling protocols have been proposed in [7], [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 work, we are interested in asynchronous sleep-wake scheduling protocols such as those proposed in [9], [10]. In these protocols, the sleep-wake schedule at each node is independent of that of other nodes, and thus the nodes do not require either a synchronization procedure or a secondary low-power receiver. However, because it is not practical for each node to have complete knowledge of the sleep-wake schedule of other nodes, it incurs additional delays along the path to the sink because each node needs to wait for its next-hop node to wake up before it can transmit. This delay could be unacceptable for delay-sensitive applications, such as fire detection or tsunami alarm, which require that the event reporting delay be small.

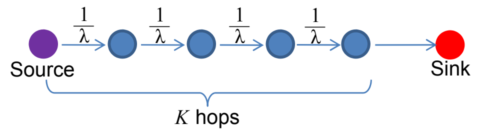
Prior work in the literature has proposed the use of *anycast* packet-forwarding schemes to reduce this event reporting delay [11]–[15]. Under traditional packet-forwarding schemes, every node has one designated next-hop relaying node in the neighborhood, and it has to wait for the next-hop node to

This work has been partially supported by the National Science Foundation through awards CNS-0626703, CNS-0721477, CNS-0721434, CCF-0635202, and ARO MURI Award No. W911NF-07-10376 (SA08-03). An earlier version of this paper has appeared in the Proceedings of IEEE INFOCOM, 2008 [1].

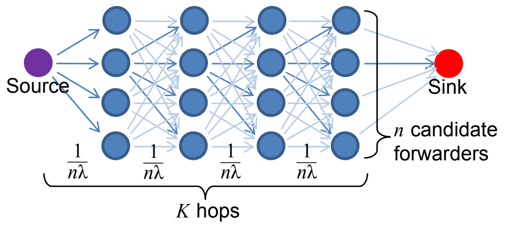
J. Kim and X. Lin are with School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA (Email:{jhkim,linx}@purdue.edu)

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 (Email: shroff@ece.osu.edu).

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



(a) Data forwarding without anycast: the expected delay at each hop is $\frac{1}{\lambda}$.



(b) Data forwarding with anycast: the expected delay at each hop is $\frac{1}{n\lambda}$.

Fig. 1. Example of anycast date-forwarding: anycast can reduce the expected one-hop delay and the expected end-to-end delay by n times.

wake up 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 *a forwarding set*). A sending node can forward the packet to the first node that wakes up in the forwarding set. The example in Fig. 1 well illustrates the advantage of anycast in sensor networks with asynchronous sleep-wake scheduling. Assume that each node wakes up independently according to a Poisson process with a common rate λ, i.e., the intervals between every two successive wake-up events of any given node are i.i.d. and exponentially distributed with mean $\frac{1}{\lambda}$. In the figure, a packet generated at a source node needs to be relayed by K hops in order to reach a sink. Without anycast, the expected one-hop delay is $\frac{1}{\lambda}$, and the expected end-to-end delay is $\frac{K}{\lambda}$ (see Fig. 1(a)). Now, if we employ an anycast packet-forwarding scheme, assume that each node has n nodes in the forwarding set as shown in Fig. 1(b), i.e., each node has n candidates to be the next-hop node. Then, the expected one-hop delay and the end-to-end delay decrease to $\frac{1}{n\lambda}$ and $\frac{K}{n\lambda}$, respectively. Hence, anycast reduces the event-reporting delay by n times in this example.

Although anycast clearly reduces the event-reporting delay, it leads to a number of challenging control and optimization problems. The first challenge is for each node to determine its anycast forwarding policy to minimize the end-to-end packet-delivery delay. Note that in practice sensor networks are often not laid out with a layered structure as in Fig. 1(b). Therefore, each individual node must choose its anycast forwarding policy (e.g., the forwarding set) distributively, with minimal knowledge of the global network topology. The existing anycast schemes in the literature [11], [12], [15] address this problem using geographical information. However, these heuristic solutions do not minimize the end-to-end delay (We provide comparisons in Section V).

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 [11], [12], [15].

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 networks 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 above 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 sub-phases, i.e., *the sleeping sub-phase* and *the event-reporting sub-phase*. In the sleeping sub-phase, 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 sub-phase. Finally, when an event occurs, the information needs to be passed on to the sink as soon as possible, which becomes the event-reporting sub-phase. It is in this event reporting sub-phase 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

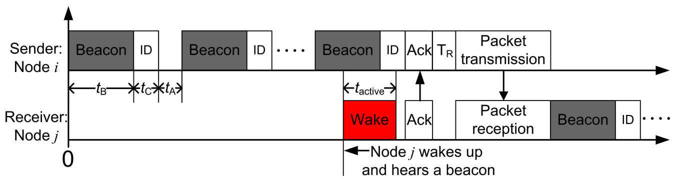


Fig. 2. System Model

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 multi-hop relaying. We assume in this paper that there is a single sink, however, the analysis can be generalized to the case with multiple sinks (see Section III-D).

We assume that the sensor network employs sleep-wake scheduling to improve energy-efficiency. We first introduce a basic sleep-wake scheduling protocol as follows. 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. (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 IV of our online technical report [16].) 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 followed by an ID signal (carrying the sender information). Let t_B and t_C be the duration of the beacon signal and the ID signal, respectively. When node j wakes up and senses a 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 the 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. 2. 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 as 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. 2, node i starts sending a beacon signal and an ID signal, successively. All nodes in $\mathcal{C}_i$ 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 acknowledgement after the ID signal ends. After each ID signal, node i checks whether there is any acknowledgement from the nodes in $\mathcal{F}_i$. If no acknowledgement is detected, node i repeats the beacon-ID-signaling and acknowledgement-detection processes until it hears one. On the other hand, if there is an acknowledgement, it may take additional time for node i to identify which node acknowledges the beacon-ID signals, especially when there are multiple 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 acknowledgements. 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 acknowledgement-detection processes to find the next forwarder. Since nodes consume energy when awake, t_{active} should be as small as possible. However, t_{active} has to be larger than t_A because otherwise a neighboring node could wake up after an ID signal and could return to sleep before the next beacon signal. In this paper, we set $t_{\text{active}} = t_A$ so that nodes cannot miss on-going beacon-ID signals and also can reduce the energy consumption for staying awake.

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 set, 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 though 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. 2. If no nodes in $\mathcal{F}_i$ have heard the first $m-1$ beacon and ID signals, then node i transmits the m-th 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 m-th 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 m-th 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. It should be noted that, 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 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 Set: 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, *the optimal choices of forwarding sets at different nodes are correlated.* We use a matrix $\mathbf{A}$ to represent the forwarding set of all nodes collectively, as follows:

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

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 acknowledgement 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:

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

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.

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 pre-configured 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

[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.

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 adaption policies are outside the scope of our paper. In other words, we mainly focus on the optimization of the anycast and sleep-wake scheduling policies 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 as 'the delay from node i.'

Our first objective is to solve the following *delay-minimization problem*: $\min_{\mathbf{A},\mathbf{B}} \; D_i(\vec{p}, \mathbf{A}, \mathbf{B})$. 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 $\frac{Q_i}{\mu_i \lambda_i}$. 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 introduction, 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 19.2mW of power while awake (i.e., in an active mode) and 40.8mW when transmitting at 250kbps [17]. Assume that nodes stay awake only 1 percent of the time, and each node has to deliver 50M bytes 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 $6054.9W \cdot s$ (365 days/year $\times$ 24 hours/day $\times$ 60 minutes/hour $\times$ 60 seconds/minute $\times 0.01 \times 19.2mW$).[2] In contrast, the energy consumption due to packet transmission for a year is about $66.8W \cdot s$ (50 M byte $\times$ 1024 Kbytes/Mbyte $\times$ 8 bits/byte / 250Kbps $\times$ 40.8 mW), which is only 1 percent of the energy consumption due to waking up.

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

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

$$T_i(\vec{p}) = \frac{1}{e_i \lambda_i} = \frac{t_B + t_C + t_A}{e_i \ln \frac{1}{(1-p_i)}}. \quad (2)$$

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

$$T(\vec{p}) = \min_{i \in \mathcal{N}} T_i(\vec{p}). \quad (3)$$

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:

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

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 (3) 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 [18]–[20]. In Section IV, we solve the lifetime-maximization problem **(P)** with the lifetime definition in (3), 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 **(P)**.

The delay-minimization problem is an instance of the stochastic shortest path (SSP) problem [21, Chapter 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 incurred. 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 $h \geq K$. 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 $D_i(\vec{p}, \mathbf{A}, \mathbf{B}) = E\left[\sum_{k=0}^{\infty} d_{i_k}(\vec{p}, \mathbf{A}, \mathbf{B})\right]$.

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 h-th 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 iterations, and that node j wakes up at the h-th beacon-ID signals while all nodes with a higher priority than node j remain sleeping at the h-th 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 $(t_B + t_C + t_A)h + t_R + t_P + D_j(\vec{p}, \mathbf{A}, \mathbf{B})$, 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. 2). For ease of notation, we define the iteration period $t_I \triangleq t_B + t_C + t_A$ and the data transmission period $t_D \triangleq t_R + t_P$. 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:

$$q_{i,j}(\vec{p}, \mathbf{A}, \mathbf{B}) \triangleq \sum_{h=1}^{\infty} P_{j,h} = \frac{p_j \prod_{k \in \mathcal{F}_i(\mathbf{A}): b_{ij} < b_{ik}} (1-p_k)}{1 - \prod_{j \in \mathcal{F}_i(\mathbf{A})} (1-p_j)}. \quad (4)$$

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

$$\begin{aligned} d_i(\vec{p}, \mathbf{A}, \mathbf{B}) &\triangleq \sum_{h=1}^{\infty} \sum_{j \in \mathcal{F}_i(\mathbf{A})} [(t_I h + t_D) P_{j,h}] \\ &= t_D + \frac{t_I}{1 - \prod_{j \in \mathcal{F}_i(\mathbf{A})} (1-p_j)}. \quad (5) \end{aligned}$$

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:

$$\begin{aligned} D_i(\vec{p}, \mathbf{A}, \mathbf{B}) &= \sum_{h=1}^{\infty} \sum_{j \in \mathcal{F}_i(\mathbf{A})} [(t_I h + t_D + D_j(\vec{p}, \mathbf{A}, \mathbf{B})) P_{j,h}] \quad (6) \\ &= d_i(\vec{p}, \mathbf{A}, \mathbf{B}) + \sum_{j \in \mathcal{F}_i(\mathbf{A})} q_{i,j}(\vec{p}, \mathbf{A}, \mathbf{B}) D_j(\vec{p}, \mathbf{A}, \mathbf{B}). \quad (7) \end{aligned}$$

We call (7) *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 (4) and (5), 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 $\vec{\pi}_i = (D_j, j \in \mathcal{C}_i)$, the forwarding set $\mathcal{F}_i$, and the priority assignment $\vec{b}_i$, let

$$\begin{aligned} & f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i) \\ \triangleq \quad & t_D + \frac{t_I + \sum_{j \in \mathcal{F}_i} p_j \prod_{k \in \mathcal{F}_i : b_{ij} < b_{ik}} (1-p_k) D_j}{1 - \prod_{j \in \mathcal{F}_i} (1-p_j)}. \quad (8) \end{aligned}$$

We call the function $f(\cdot, \cdot, \cdot)$ *the local delay function*. With the local delay function, we can express the local delay relationship (7) 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 value-iteration [21, Section 1.3] as follows. Start from some appropriately chosen initial values of the delay $D_i^{(0)}$. At each iteration $k = 1, 2, \cdots$, each node i takes the delay value of other nodes from the $(k-1)$-th iteration, and updates its own policy and delay value using

$$D_i^{(k)} = \min_{\mathcal{F}_i, \vec{b}_i} f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i), \quad (9)$$

where $\vec{\pi}_i = (D_j^{(k-1)}, j \in \mathcal{C}_i)$. As we will show later, the value iterations 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 (9). 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 (9) 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 $(k-1)$-st 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 acknowledgements, 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 ease of notation, we define the value of the local delay function with this optimal priority assignment as

$$\hat{f}(\vec{\pi}_i, \mathcal{F}_i) \triangleq f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i^*(\vec{\pi}_i)). \tag{10}$$

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) $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \Leftrightarrow D_{J_3} + t_D < \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \Leftrightarrow D_{J_3} + t_D < \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$.
(b) $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) = \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \Leftrightarrow D_{J_3} + t_D = \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \Leftrightarrow D_{J_3} + t_D = \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)$.
(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, its interpretation is actually quite straightforward. 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) states 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 $D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$.
(b) $\mathcal{F}_i^*$ cannot contain any nodes j in $\mathcal{C}_i$ that satisfy $D_j > \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$.
(c) If there is a node j in $\mathcal{F}_i^*$ that satisfies $D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$, the following relationship holds,

$$\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^* \setminus \{j\}).$$

Proof: We prove this proposition by contradiction. In order to prove Property (a), assume that there exists a node j such that $D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$ 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 $D_j < \hat{f}(\vec{\pi}_i, \mathcal{J}_1) - t_D$, 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 do not change the delay analysis. Since $\mathcal{F}_i^* = \mathcal{J}_1 \cup \mathcal{J}_2$ is optimal, we must have $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) \leq \hat{f}(\vec{\pi}_i, \mathcal{J}_1)$. **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, $D_{J_3} = \hat{f}(\vec{\pi}_i, \mathcal{J}_1) - t_D = \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) - t_D > D_j = D_{J_2}$. 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 $D_j > \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$. Let $\mathcal{J}_1 = \{l \in \mathcal{F}_i^* | D_l \leq \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D\}$ and $\mathcal{J}_3 = \{l \in \mathcal{F}_i^* | D_l > \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D\}$. Then, the weighted average delay in $\mathcal{J}_3$ is larger than $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$, i.e.,$\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) - t_D < D_{J_3}$. 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 $D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$. 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 $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$, 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. ∎

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 $\mathcal{F}_i^* = \{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D\} \cup \mathcal{G}$, where $\mathcal{G}$ is a subset of $\{j \in \mathcal{C}_i | D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D\}$. Note that if there exists a node j such that $D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$, 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., $\{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D\}$, in order to reduce the possibility that multiple nodes send duplicated acknowledgements. Hence, in this paper, we restrict our definition of the optimal forwarding set $\mathcal{F}_i^*$ to the following

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

(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.

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 $\mathcal{F} = \{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}) - t_D\}$, 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 $D_k < \hat{f}(\pi_i, \mathcal{F}_i^*) - t_D$. 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 $\mathcal{F} = \{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}) - t_D\}$) is also a subset of $\mathcal{C}_i$, the threshold values of $\mathcal{F}$ and $\mathcal{F}_i^*$ satisfy $\hat{f}(\pi_i, \mathcal{F}_i^*) - t_D \leq \hat{f}(\pi_i, \mathcal{F}) - t_D$. 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 $j \in \mathcal{F}_{i,k}$, $D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k}) - t_D$, and for all nodes $l \notin \mathcal{F}_{i,k}$, $D_l \geq \hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k}) - t_D$. 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 $b_{ij}^*(\vec{\pi}_i) = k$, $D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k}) - t_D$, and for node l with $b_{il}^*(\vec{\pi}_i) = k - 1$, $D_l \geq \hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k}) - t_D$. The detail is provided in the LOCAL-OPT algorithm. 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.

The LOCAL-OPT Algorithm

Step (1) Node i sorts the delays $\vec{\pi}_i$ of its neighboring nodes.
Step (2) Node i assigns different priorities $b_{i,j}^*$ $(1 \leq b_{i,j}^* \leq |\mathcal{C}_i|)$ to the neighboring nodes j such that for any neighboring nodes j_1 and j_2, if $b_{i,j_1}^* > b_{i,j_2}^*$, then $D_{j_1} \leq D_{j_2}$.
Step (3) Let $b^{-1}(k)$ be the index of the neighboring node with priority k, i.e., $b_{i,b^{-1}(k)} = k$.
Step (4) Initial Setup: $k \leftarrow |\mathcal{C}_i|$, prod $\leftarrow 1$, and sum $\leftarrow 0$.
Step (5) Compute f in the following order:

$$\text{5a) sum} \leftarrow \text{sum} + D_{b^{-1}(k)} \cdot p_{b^{-1}(k)} \cdot \text{prod},$$

$$\text{5b) prod} \leftarrow \text{prod} \cdot (1 - p_{b^{-1}(k)}),\ \text{and 5c)} f \leftarrow t_D + \frac{t_I + \text{sum}}{\text{prod}}.$$

Step (6a) If $k > 1$ and $D_{b^{-1}(k-1)} < f - t_D$, then decrease k by one and go back to Step (5).
Step (6b) Else, this algorithm terminates and returns

$$\mathcal{F}_i^*(\vec{\pi}_i) = \{b^{-1}(l); l = k, \cdots, |\mathcal{C}_i|\} \text{ and } \min_{\mathcal{F} \subset \mathcal{C}_i} \hat{f}(\vec{\pi}_i, \mathcal{F}) = f.$$

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

We next use the insight of Section III-B to develop an algorithm for computing the optimal anycast policy $(\mathbf{A}, \mathbf{B})$ for given $\vec{p}$. This algorithm can be viewed as performing the value-iteration [21, Section 1.3] as discussed at the beginning of this section. Initially (iteration 0), each node i sets its delay value $D_i^{(0)}$ to infinity, and only the sink sets $D_s^{(0)}$ to zero. Then, at every iteration h, each node i uses the delay values $D_j^{(h-1)}$ of neighboring nodes j from the previous iteration $(h-1)$ to update the current forwarding set $\mathcal{F}_i^{(h)}$ and the current priority assignment $\vec{b}_j^{(h)}$ according to the value iteration (9). We will show that when the value iterations are synchronized, the algorithm converges in N iterations, and it returns the optimal anycast policy that minimizes the delays of all nodes simultaneously. Furthermore, we will show that even if the value iterations and the policy updates of nodes are not synchronized, the anycast policy of each node still converges to the optimal anycast policy (although, as is the case with most asynchronous algorithms, the convergence within N iterations is not guaranteed).

The complete algorithm is presented next.

The GLOBAL-OPT Algorithm

Step (1) At iteration $h = 1$: sink node s sets $D_s^{(0)}$ to 0, and other nodes i set $D_i^{(0)}$ to ∞.
Step (2) At iteration h, each node i runs the LOCAL-OPT algorithm with the input $\vec{\pi}_i^{(h-1)} = (D_j^{(h-1)}, j \in \mathcal{C}_i)$.
Step (3) Using the output of the LOCAL-OPT algorithm (6b), each node i updates the anycast forwarding policy and the delay value as follows:

$$\begin{aligned} \text{POLICY ITERATION:} \quad & \mathcal{F}_i^{(h)} \leftarrow \mathcal{F}_i^*(\vec{\pi}_i^{(h-1)}), \\ & \vec{b}_i^{(h)} \leftarrow \vec{b}_i^*(\vec{\pi}_i^{(h-1)}) \\ \text{VALUE ITERATION:} \quad & D_i^{(h)} \leftarrow \min_{\mathcal{F} \subset \mathcal{C}_i} \hat{f}(\vec{\pi}_i^{(h-1)}, \mathcal{F}). \end{aligned} \quad (11)$$

Step (4a) If $h = N$, this algorithm terminates and returns $\mathcal{F}_i^* = \mathcal{F}_i^{(N)}$ and $D_i^*(\vec{p}) = D_i^{(N)}$.
Step (4b) If $h < N$, $h \leftarrow h + 1$ and goes back to Step (2).

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 i-th row is $\vec{b}_i^{(h)}$. Then, the following proposition shows the convergence of the GLOBAL-OPT algorithm:

***Proposition** 4:* For given $\vec{p}$, the delay values $D_i^{(h)}$ converge to $D_i^*(\vec{p})$, i.e., $D_i^{(h)} \longrightarrow D_i^*(\vec{p})$ as $h \to \infty$. Furthermore, the anycast policy $(\mathbf{A}^{(h)}, \mathbf{B}^{(h)})$ also converges to an anycast policy $(\mathbf{A}^*, \mathbf{B}^*)$ as $h \to \infty$ such that $D_i(\vec{p}, \mathbf{A}^*, \mathbf{B}^*) = D_i^*(\vec{p})$ for all nodes i.

Proof: The GLOBAL-OPT 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 [21]), 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 (4). 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 (5). 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 on page 95 of [21]. Hence, according to Proposition 2.2.2 of [21], 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 $h \to \infty$. ■

Let $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$ be the converged anycast policy, i.e., $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \triangleq \lim_{h\to\infty}(\mathbf{A}^{(h)}, \mathbf{B}^{(h)})$. Then, Proposition 4 shows that this converged anycast policy corresponds to an optimal anycast policy that minimizes the delays from all nodes simultaneously, i.e., $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) = \arg\min_{\mathbf{A},\mathbf{B}} D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ for all nodes i.

We next show the convergence of the GLOBAL-OPT 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 on the structure of the optimal forwarding set.

***Proposition** 5:* For any awake probability vector $\vec{p}$, there exists some optimal anycast policy $(\mathbf{A}', \mathbf{B}')$ that solves the delay-minimization problem for all nodes and does not incur any cycle in routing paths, i.e., $g(\mathbf{A}')$ is acyclic.

Proof: It suffices to show that for any $\vec{p}$, $\mathbf{A} \in \mathcal{A}$, and $\mathbf{B} \in \mathcal{B}$ such that $g(\mathbf{A})$ is cyclic (i.e., there is a cyclic path from a node to the sink), we can find $\mathbf{A}' \in \mathcal{A}$ and $\mathbf{B}' \in \mathcal{B}$ that satisfies the following properties:
(a) $g(\mathbf{A}')$ is acyclic, and
(b) $D_i(\vec{p}, \mathbf{A}', \mathbf{B}') \le D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ for all nodes i.
We first show how to construct such a policy $(\mathbf{A}', \mathbf{B}')$ for given $\mathbf{A}$ and $\mathbf{B}$, and then show that such $\mathbf{A}'$ and $\mathbf{B}'$ satisfy Properties (a) and (b).

Assume that all nodes are connected to the sink under the forwarding matrix $\mathbf{A}$. (We will consider the other case later.) Then, every node i has at least one neighboring node in the forwarding set $\mathcal{F}_i(\mathbf{A})$. Now, let every node i select its new forwarding set among the nodes in $\mathcal{F}_i(\mathbf{A})$. Each node then finds the optimal forwarding set $\mathcal{F}_i^*$ and the optimal priority assignment $\vec{b}_i^*(\vec{\pi}_i)$ for the delays $\vec{\pi}_i = (D_j(\vec{p}, \mathbf{A}, \mathbf{B}), j \in \mathcal{F}_i(\mathbf{A}))$ using the LOCAL-OPT algorithm. Let $\mathbf{A}'$ and $\mathbf{B}'$ be the new global forwarding matrix and the new global priority matrix, respectively, that correspond to the forwarding set $\mathcal{F}_i^*$ and the priority assignment $\vec{b}_i^*(\vec{\pi}_i)$ of all nodes $i \in \mathcal{N}$.

We next prove that the new anycast policy $(\mathbf{A}', \mathbf{B}')$ satisfies Property (a). Let $\vec{b}_i$ be the priority assignment of node i under the priority matrix $\mathbf{B}$. By the local-delay relationship in (6), we have $f(\vec{\pi}_i, \mathcal{F}_i(\mathbf{A}), \vec{b}_i) = D_i(\vec{p}, \mathbf{A}, \mathbf{B})$. Since $\mathcal{F}_i^*$ is the optimal forwarding set for given $\vec{\pi}_i$, we also have $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) \le f(\vec{\pi}_i, \mathcal{F}_i(\mathbf{A}), \vec{b}_i)$. Combining the above, we have

$$\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) \le D_i(\vec{p}, \mathbf{A}, \mathbf{B}). \quad (12)$$

According to Proposition 3, for a neighboring node $j \in \mathcal{F}_i(\mathbf{A})$ to be in the new forwarding set $\mathcal{F}_i^*$, the delay $D_j(\vec{p}, \mathbf{A}, \mathbf{B})$ must be smaller than $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*) - t_D$. From (12), it follows that all nodes j in $\mathcal{F}_i^*$ must satisfy $D_j(\vec{p}, \mathbf{A}, \mathbf{B}) < D_i(\vec{p}, \mathbf{A}, \mathbf{B}) - t_D$. Hence, the delay value $D_j(\vec{p}, \mathbf{A}, \mathbf{B})$ decreases by at least t_D along all paths under the new forwarding matrix $\mathbf{A}'$. This implies that there could not exist any cyclic path. Hence, Property (a) follows.

We next prove by contradiction that the new anycast policy $(\mathbf{A}', \mathbf{B}')$ also satisfies Property (b), i.e., $D_i(\vec{p}, \mathbf{A}', \mathbf{B}') \le D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ for all nodes i. From (12), it suffices to show that $D_i(\vec{p}, \mathbf{A}', \mathbf{B}') \le \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*)$ holds for all nodes i. Assume in contrary that there exists a node i such that

$$D_i(\vec{p}, \mathbf{A}', \mathbf{B}') > \hat{f}(\vec{\pi}_i, \mathcal{F}_i^*). \quad (13)$$

From (10), we can rewrite the right hand side $\hat{f}(\vec{\pi}_i, \mathcal{F}_i^*)$ as $f(\vec{\pi}_i, \mathcal{F}_i^*, \vec{b}_i^*(\vec{\pi}_i))$. Let $\vec{\pi}_i'$ be the delays of nodes in $\mathcal{F}_i(\mathbf{A})$ under the new anycast policy $(\mathbf{A}', \mathbf{B}')$, i.e., $\vec{\pi}_i' = (D_j(\vec{p}, \mathbf{A}', \mathbf{B}'), j \in \mathcal{F}_i(\mathbf{A}))$. Then, comparing (8) and (7), we can also express the left hand side $D_i(\vec{p}, \mathbf{A}', \mathbf{B}')$ as $f(\vec{\pi}_i', \mathcal{F}_i^*, \vec{b}_i^*(\vec{\pi}_i))$. Since the local delay function $f(\cdot, \mathcal{F}_i^*, \vec{b}_i^*(\vec{\pi}_i))$ is monotonic with respect to each element of the first argument, there must exist at least one neighboring node j in $\mathcal{F}_i^*$ such that $D_j(\vec{p}, \mathbf{A}, \mathbf{B}) < D_j(\vec{p}, \mathbf{A}', \mathbf{B}')$ to satisfy (13). Since $\hat{f}(\vec{\pi}_j, \mathcal{F}_j^*) \leq D_j(\vec{p}, \mathbf{A}, \mathbf{B})$ from (12), such node j satisfies

$$D_j(\vec{p}, \mathbf{A}', \mathbf{B}') > \hat{f}(\vec{\pi}_j, \mathcal{F}_j^*). \tag{14}$$

By applying the same method iteratively, we can find a sequence of such nodes j such that all of which satisfy (14). Since all paths under $\mathbf{A}'$ are acyclic and connected to the sink s, the sequence must converge to sink s. However, the delay of sink s is always zero, i.e., $D_s(\vec{p}, \mathbf{A}', \mathbf{B}') = D_s(\vec{p}, \mathbf{A}, \mathbf{B}) = 0$, which is a contradiction to (14). Hence, Property (b) follows.

We have shown that if all nodes are connected to the sink under the forwarding matrix $\mathbf{A}$, there exists an alternative policy $(\mathbf{A}', \mathbf{B}')$ that satisfies Properties (a) and (b). In the case where some nodes are disconnected from the sink under the forwarding matrix $\mathbf{A}$, we simply exclude these nodes from the node set $\mathcal{N}$ and then use the above procedure to construct the alternative anycast policy $(\mathbf{A}', \mathbf{B}')$ for the connected nodes only. Then, for the connected nodes, the policy $(\mathbf{A}', \mathbf{B}')$ must satisfy Properties (a) and (b) according to the earlier proof. Since the disconnected nodes are still disconnected under the alternative policy, their delays will remain infinite, and there are no cyclic paths from these node to the sink. Hence, Properties (a) and (b) also hold for the entire network. ■

From the existence of the optimal cycle-free anycast policy, we can show the following proposition:

***Proposition** 6:* For given $\vec{p}$,
(a) The GLOBAL-OPT algorithm converges within N iterations, i.e., $(\mathbf{A}^{(N)}, \mathbf{B}^{(N)}) = (\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$
(b) The GLOBAL-OPT algorithm does not incur any cyclic paths, i.e., $g(\mathbf{A}^*(\vec{p}))$ is acyclic.

Proof: To show that convergence occurs in N iterations, we need Proposition 5, which states that there must exist at least one optimal anycast policy that does not incur any cycles. Hence, based on the proof on page 107 of [21], 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 Property 2.2.2 (c) on page 99, 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.

We next prove that Property (b) holds. According to Property (a), all nodes i satisfy $\mathcal{F}_i^{(N+1)} = \mathcal{F}_i^{(N)}$ and $D_i^{(N+1)} = D_i^{(N)}$. Note that $\mathcal{F}_i^{(N+1)}$ is the optimal forwarding set for the delays $\pi_i^{(N)} = (D_j^{(N)}, j \in \mathcal{C}_j)$, and $D_i^{(N+1)}$ is the corresponding optimized value $\hat{f}(\pi_i^{(N)}, \mathcal{F}_i^{(N+1)})$. According to Proposition 3, neighboring nodes j in $\mathcal{F}_i^{(N+1)}$ must satisfy $D_j^{(N)} < D_i^{(N+1)} - t_D$, which leads to $D_j^{(N)} < D_i^{(N)} - t_D$ for all neighboring nodes j in $\mathcal{F}_i^{(N)}$. This implies that under anycast policy $(\mathbf{A}^{(N)}, \mathbf{B}^{(N)})$, the delay value $D_i^{(N)}$ decreases by at least t_D along each possible routing path. Hence, Property (b) follows. ■

Proposition 6 shows that the anycast policy $(\mathbf{A}^{(N)}, \mathbf{B}^{(N)})$ that the GLOBAL-OPT 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 GLOBAL-OPT algorithm is a synchronous algorithm that requires all nodes to execute the value-iteration in locked steps. In fact, an asynchronous version of GLOBAL-OPT algorithm can also be shown to converge, although the convergence will typically require more than N iterations.

***Proposition** 7:* Assume that at each iteration, every node i independently chooses either to follow Step (2) and Step (3) in the GLOBAL-OPT algorithm or to maintain its current policy and delay value, i.e., $\mathcal{F}_i^{(h)} = \mathcal{F}_i^{(h-1)}$, $\vec{b}_i^{(h)} = \vec{b}_i^{(h-1)}$, and $D_i^{(h)} = D_i^{(h-1)}$. Further, assume that every node updates its policy and delay value infinitely often. Then, the policy and the delay value converge to the optimal policy and the minimum delay, respectively, as $h \to \infty$, i.e., $\lim_{h\to\infty}(\mathbf{A}^{(h)}, \mathbf{B}^{(h)}) = \arg\min_{\mathbf{A},\mathbf{B}} D_i(\vec{p}, \mathbf{A}, \mathbf{B})$, $\lim_{h\to\infty} D_i^{(h)} = \min_{\mathbf{A},\mathbf{B}} D_i(\vec{p}, \mathbf{A}, \mathbf{B})$.

Proof: The proof again follows from the standard results of Proposition 1.3.5 in [21]. ■

Since the asynchronous version of GLOBAL-OPT algorithm does not converge in N steps, for practical implementation, we use an expiration time t_{exp} to force the algorithm to terminate. The expiration time t_{exp} leads to a tradeoff between optimality and the duration of the configuration phase. If t_{exp} increases, the latest anycast policy will be closer to the optimal policy, but the execution time will also increase. (Recall that the asynchronous GLOBAL algorithm runs in the configuration phase that occurs at the very beginning of the deployment of sensor nodes.) Hence, the parameter t_{exp} must be chosen carefully, with this tradeoff in mind.

D. The case with multiple sink nodes

Our results can be easily generalized to the case when there are multiple sink nodes and the event-reporting packets can be collected by any sink nodes. In this case, we can simply set $D_s(\vec{p}, \mathbf{A}, \mathbf{B}) = 0$ for all sink nodes s. Then the same delay-minimization algorithm GLOBAL-OPT can be used. The resulting anycast policy will minimize the end-to-end delay to reach any sink node.

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 **(P)**. From (2), 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 $\overrightarrow{T} = (T_1, T_2, \cdots, T_N)$, $\mathbf{A}$, and $\mathbf{B}$:

$$\begin{aligned}
(\mathbf{P1}) \quad & \max_{\overrightarrow{T}, \mathbf{A}, \mathbf{B}} \quad \min_{i \in \mathcal{N}} T_i, && (15)\\
\text{subject to} \quad & D_i(\vec{p}, \mathbf{A}, \mathbf{B}) \leq \xi^*, \quad \forall i \in \mathcal{N} && (16)\\
& p_i = 1 - e^{-\frac{t_I}{e_i T_i}}, \quad \forall i \in \mathcal{N} && (17)\\
& T_i \in (0, \infty), \quad \forall i \in \mathcal{N}\\
& \mathbf{A} \in \mathcal{A}, \quad \mathbf{B} \in \mathcal{B}.
\end{aligned}$$

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.,

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

$$
\begin{aligned}
(\mathbf{P2}) \quad & \max_{\vec{T}} \quad \min_{i\in\mathcal{N}} T_i, \\
\text{subject to} \quad & D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \le \xi^*, \quad \forall i \in \mathcal{N} \qquad (18)\\
& p_i = 1 - e^{-\frac{t_I}{e_i T_i}}, \quad \forall i \in \mathcal{N} \\
& T_i \in (0, \infty), \quad \forall i \in \mathcal{N}
\end{aligned}
$$

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

***Proposition** 8:* If $\vec{T}^* = (T_1^*, T_2^*, \cdots, T_N^*)$ is the optimal solution to Problem $(\mathbf{P2})$, then so is $\vec{T}$ such that $\vec{T} = (T_i = \min_k T_k^*, i \in \mathcal{N})$. In other words, according to the lifetime definition (3), 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 (3), it is sufficient to show that if $\vec{T}^*$ is in the feasible set, so is $\vec{T}$. Let $\vec{p}^*$ and $\vec{p}$ be the awake probability vectors that correspond to $\vec{T}^*$ and $\vec{T}$, respectively, by (17). Since p_i monotonically decreases as T_i increases, and $\vec{T} \preceq \vec{T}^*$, we have $\vec{p}^* \preceq \vec{p}$. (The symbol '$\preceq$' denotes component-wise inequality, i.e., if $\vec{T} \preceq \vec{T}^*$, then $T_i \le T_i^*$ 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 non-increasing 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

$$
\begin{aligned}
D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) &\ge D_i(\vec{p}\,', \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \\
&\ge D_i(\vec{p}\,', \mathbf{A}^*(\vec{p}\,'), \mathbf{B}^*(\vec{p}\,')). \qquad (19)
\end{aligned}
$$

The last inequality in (19) 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 non-increasing with respect to each component of $\vec{p}$. Since $\vec{p}^* \preceq \vec{p}$, for all nodes i, we have $D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \le D_i(\vec{p}^*, \mathbf{A}^*(\vec{p}^*), \mathbf{B}^*(\vec{p}^*))$. Hence, if $\vec{T}^*$ satisfies (18), so does $\vec{T}$. ■

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

$$
\begin{aligned}
(\mathbf{P3}) \quad \max \quad & T, \\
\text{subject to} \quad & \max_{i\in\mathcal{N}} D_i(\vec{p}, \mathbf{A}(\vec{p}), \mathbf{B}(\vec{p})) \le \xi^* \\
& p_i = 1 - e^{-\frac{t_I}{e_i T}}, \quad \forall i \in \mathcal{N} \\
& T \in (0, \infty).
\end{aligned}
$$

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

Note that $\max_{i\in\mathcal{N}} D_i(\vec{p}, \mathbf{A}(\vec{p}), \mathbf{B}(\vec{p}))$ is non-increasing with respect to each component of $\vec{p}$. (See the proof of Proposition 8.) 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 for computing the optimal value of T^* such that $\max_{i\in\mathcal{N}} D_i(\vec{p}^*, \mathbf{A}(\vec{p}^*), \mathbf{B}(\vec{p}^*)) = \xi^*$

The Binary Search Algorithm for Problem (P3)

Step (1) Let $k = 1$. Initially, sink s sets $T^{(1)}$ to a half of the maximum possible lifetime and sets $T_{record} = 0$.

Step (2) Nodes run the GLOBAL-OPT algorithm for given $\vec{p}^{(k)} = (p_i^{(k)} = 1 - e^{-\frac{t_I}{T^{(k)} e_i}}, i \in \mathcal{N})$.

Step (3) When the GLOBAL-OPT algorithm terminates, each node i finds the optimal anycast policy and the optimal delay value $D_i^{(N)}$. Only nodes j that are not in the other node's forwarding set, i.e., $j \notin \mathcal{F}_i^*(\mathbf{A}^*(\vec{p}^{(k)}))$ for all nodes i, send feedback of their delay values $D_j^{(N)}$ to sink s.

Step (4) Let $D_{\max}$ be the maximum feedback delay value arrived at sink s.

- If $D_{\max} > \xi^*$, then sink s sets $T^{(k+1)} = T^{(k)} - \frac{T^{(1)}}{2^k}$.
- If $D_{\max} < \xi^*$, then sink s memorizes $T_{record} = T^{(k)}$ and sets $T^{(k+1)} = T^{(k)} + \frac{T^{(1)}}{2^k}$.

Step (5) If $k = k_{\max}$, return T_{record} as the solution to **(P3)**. Otherwise, increases k by one, and goes back to **Step (2)**.

After $k_{\max}$ iterations, the difference between the optimal lifetime T^* and the algorithm output T_{record} is smaller than $\frac{T^{(1)}}{2^{k_{\max}}}$. If we want to make this difference less than ϵ, the complexity of the Binary Search Algorithm is $\mathcal{O}(\log \frac{1}{\epsilon} \cdot N \cdot \max_i |\mathcal{C}_i| \log \max_i |\mathcal{C}_i|)$. Note that in **Step (3)** 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 GLOBAL-OPT algorithm in Proposition 6.) According to Property (a) in Proposition 3, if node j belongs to the forwarding set of node i under the GLOBAL-OPT 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.

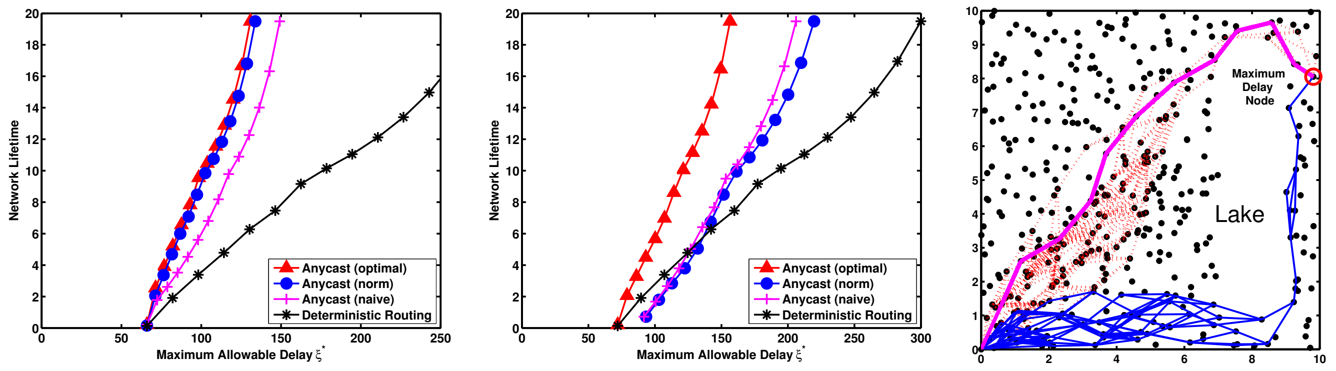
V. Simulation Results

In this section, we provide simulation results to illustrate the performance advantage of our optimal anycast algorithm. We simulate a wireless sensor network with 400 nodes deployed randomly over a 10-by-10 area with a given distribution, and the sink s located at $(0, 0)$. We assume that the transmission range from each node i is a disc with radius 1.5, i.e., if the distance between node j and node i is less than 1.5, then $j \in \mathcal{C}_i$. The parameters t_I and t_D are set to 1 and 5, respectively. We also assume that the power consumption ratio e_i is identical for all nodes i.

A. Existing Algorithms Proposed in the Literature

In this subsection, we review some existing algorithms that we will compare with our optimal algorithm.

Normalized-latency Anycast Algorithm: The *normalized-latency* algorithm proposed in [15] 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



(a) Lifetime when nodes are uniformly deployed (b) Lifetime when nodes are not uniformly deployed (c) Node deployment

Fig. 3. (a, b) The network lifetime subject to different allowable delay ξ^* (a) when nodes are uniformly deployed and (b) when nodes are not uniformly distributed. (c) Node deployment and routing paths under different forwarding algorithms when $p_i = 0.5$: The dotted lines illustrate all routing paths under the optimal anycast algorithm, the thick solid lines illustrate the unique routing path under the deterministic routing path, and thin solid lines illustrate all routing paths under the normalized-latency anycast algorithm

to sink s. Further, 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 normalized latency algorithm is to find the forwarding set that minimizes the expectation of normalized one-hop delay, i.e., $E[\frac{D}{R}]$. The idea behind this algorithm is to minimize the expected delay per unit distance of progress, which might help to reduce the actual end-to-end delay.

Naive Anycast Algorithm: The *naive* algorithm proposed in [15] is also an anycast-based heuristic algorithm that exploits geographic information. Under this algorithm, each node includes all neighboring nodes with positive progress in the forwarding set.

Deterministic Routing Algorithm: By *deterministic routing*, we mean that each node has only one designated next-hop forwarding node. Therefore, deterministic routing can be viewed as a special case of anycast, in which the size of the forwarding set at each node is restricted to one. Therefore, instead of finding the optimal forwarding set $\mathcal{F}_i^{(h)} = \arg\min_{\mathcal{F}\subset\mathcal{C}_i} \hat{f}(\vec{\pi}_i^{(h-1)}, \mathcal{F})$ in **Step (3)** of the GLOBAL-OPT algorithm, we update $\mathcal{F}_i^{(h)}$ according to $\mathcal{F}_i^{(h)} = \arg\min_{\mathcal{F}\subset\mathcal{C}_i:|\mathcal{F}|=1} \hat{f}(\vec{\pi}_i^{(h-1)}, \mathcal{F})$. After the above modification, the GLOBAL-OPT algorithm becomes one that finds the optimal next hop under deterministic routing. Note that this modified algorithm is equivalent to the well-known Bellman-Ford shortest path algorithm, in which the length of each link (i, j) is given by $t_I/p_j + t_D$. Let $D_i(\vec{p})$ denote the minimum delay from node i under deterministic routing. Then, $D_i^{(h)}$ under the modified algorithm converges to $D_i(\vec{p})$.

In this simulation, in order to compare the network lifetime under the different algorithms, we run the binary search algorithm for Problem **(P3)**, replacing the GLOBAL-OPT algorithm in **Step (3)** with the above mentioned algorithms.

B. *Performance Comparison*

We first simulate the case when sensor nodes are distributed uniformly in a 10-by-10 area. In Fig. 3(a), we compare the network lifetime under the different algorithms, where the x-axis represents different maximum allowable delays ξ^* in our original Problem **(P)**, and the y-axis represents the maximum lifetime for each ξ^*. The curve labeled 'Anycast (optimal)' represents the lifetime under the optimal anycast algorithm, i.e., the GLOBAL-OPT algorithm. The curves labeled 'Anycast (norm)' and 'Anycast (naive)' represent the lifetime under the normalized-latency anycast algorithm, and under the naive anycast algorithm, respectively. The curve labeled 'Deterministic routing' represents the lifetime under the deterministic routing algorithm. From Fig. 3(a), we observe that all anycast algorithms significantly extend the lifetime compared to the deterministic routing algorithm. We also observe that the performance of the optimal and the normalized-latency algorithm is very close. Note that the normalized-latency algorithm gives preference to nodes with larger progress, while our optimal algorithm gives preference to nodes with smaller delays. The results in Fig. 3(a) seem to suggest that there is a correlation between progress and delay when nodes are deployed uniformly. Finally, the reason for the performance gap between the optimal and the naive algorithms is that transmitting a packet to a neighbor with small progress is often not a good decision if a node with higher progress is expected to wake up during the packet transmission time.

We next simulate a topology where there is a hole in the sensor field as shown in Fig. 3(c). 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 other algorithms (including the normalized-latency anycast algorithm). Fig. 3(c) provides us with the intuition for this performance gap. We plot the routing paths from the nodes with the largest delay. The dotted lines (above the hole) illustrate all routing paths under the optimal anycast algorithm.

The thick solid lines (above the hole) illustrate the unique routing path under the deterministic routing algorithm. The thin solid lines (below the hole) illustrate all routing paths under the normalized-latency anycast algorithm. The routing paths under the naive anycast algorithm are omitted because they are similar to those under the normalized-latency anycast algorithm. In our optimal algorithm, in order to reduce the delay, a packet is first forwarded to neighbors with negative progress but smaller delay. However, under the normalized-latency algorithm, all packet are forwarded only to nodes with positive progress, and hence they take longer detours. Therefore, the result of Fig. 3(c) shows that when the node distribution is not uniform, there may not be a strong correlation between progress and delay. Thus, the anycast-based heuristic algorithms depending only on geographical information could perform poorly.

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 a 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 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 exists 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 (8) as follows: $f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i) = t_D + \frac{t_I + \sum_{j=1}^{K} D_j p'_j \prod_{k=1}^{j-1}(1-p'_k)}{1-\prod_{j=1}^{K}(1-p'_j)}$.

Using the above,

$$\begin{aligned}
&f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}_i) - f(\vec{\pi}_i, \mathcal{F}_i, \vec{b}'_i)\\
&= \frac{1}{1-\prod_{j=1}^{K}(1-p'_j)} \Bigg[D_l p'_l \prod_{j=1}^{l-1}(1-p'_k) + D_m p'_m \prod_{j=1}^{l}(1-p'_k)\\
&\qquad - D_m p'_m \prod_{j=1}^{l-1}(1-p'_k) - D_l p'_l \prod_{j=1}^{l-1}(1-p'_k)(1-p'_m) \Bigg]\\
&= \frac{p'_l p'_m \prod_{j=1}^{l-1}(1-p'_k)}{1-\prod_{j=1}^{K}(1-p'_j)} (D_l - D_m) \geq 0. \qquad (20)
\end{aligned}$$

In other words, the local delay function does not increase after we switches 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, the following relationship holds:

$$\begin{aligned}
&\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1) \qquad (21)\\
&\Leftrightarrow \frac{t_I + \sum_{j \in \mathcal{J}_1 \cup \mathcal{J}_3} D_j p_j \prod_{k \in (\mathcal{J}_1 \cup \mathcal{J}_3): b^*_{ij} < b^*_{ik}}(1-p_k)}{1 - \prod_{j \in \mathcal{J}_1 \cup \mathcal{J}_3}(1-p_j)}\\
&\qquad < \frac{t_I + \sum_{j \in \mathcal{J}_1} D_j p_j \prod_{k \in \mathcal{J}_1: b^*_{ij} < b^*_{ik}}(1-p_k)}{1 - \prod_{j \in \mathcal{J}_1}(1-p_j)}\\
&\Leftrightarrow \frac{t_I + D_{J_1}P_{J_1} + D_{J_3}P_{J_3}(1-P_{J_1})}{1-(1-P_{J_1})(1-P_{J_3})} < \frac{t_I + D_{J_1}P_{J_1}}{P_{J_1}}\\
&\Leftrightarrow D_{J_3} < \frac{t_I + D_{J_1}P_{J_1}}{P_{J_1}} = \hat{f}(\vec{\pi}_i, \mathcal{J}_1) - t_D \qquad (22)\\
&\Leftrightarrow D_{J_3}(1-(1-P_{J_1})(1-P_{J_3}) - P_{J_3}(1-P_{J_1}))\\
&\qquad < t_I + D_{J_1}P_{J_1}\\
&\Leftrightarrow D_{J_3} < \frac{t_I + D_{J_1}P_{J_1} + D_{J_3}P_{J_3}(1-P_{J_1})}{1-(1-P_{J_1})(1-P_{J_3})}\\
&\Leftrightarrow D_{J_3} < \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) - t_D. \qquad (23)
\end{aligned}$$

This proves Property (a). The proof of Property (b) is similar (by replacing all '$<$' by '$=$').

We now show Property (c) and (d). If $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) < \hat{f}(\vec{\pi}_i, \mathcal{J}_1)$, we have $D_{J_3} < \frac{t_I + D_{J_1}P_{J_1}}{P_{J_1}}$ by Property (a). In addition, since D_{J_k} is the weighted average delay in $\mathcal{J}_k$, and nodes in $\mathcal{J}_k$ have higher priorities than nodes in $\mathcal{J}_{k+1}$, we have $D_{J_2} \leq D_{J_3}$. Let $\omega_{1,3} \triangleq 1-(1-P_{J_1})(1-P_{J_3})$, $\omega_{1,2,3} \triangleq 1-(1-P_{J_1})(1-P_{J_2})(1-P_{J_3})$, and $\chi \triangleq \left(\frac{P_{J_2}(1-P_{J_1})}{\omega_{1,2,3}}\right)$. With these notations, we have

$$\begin{aligned}
&\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_3) - \hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3)\\
&= \frac{t_I + \sum_{k=1}^{3} D_{J_k}P_{J_k}\prod_{l=1}^{k-1}(1-P_{J_l})}{\omega_{1,2,3}}\\
&\quad - \frac{t_I + D_{J_1}P_{J_1} + D_{J_3}P_{J_3}(1-P_{J_1})}{\omega_{1,3}}
\end{aligned}$$

$$
\begin{aligned}
&= (t_I + D_{J_1}P_{J_1})\left(\frac{1}{\omega_{1,2,3}} - \frac{1}{\omega_{1,3}}\right) + \frac{D_{J_2}P_{J_2}(1-P_{J_1})}{\omega_{1,2,3}} \\
&\quad + D_{J_3}P_{J_3}(1-P_{J_1})\left(\frac{1-P_{J_2}}{\omega_{1,2,3}} - \frac{1}{\omega_{1,3}}\right) \\
&= \chi\left(-(t_I + D_{J_1}P_{J_1})\frac{1-P_{J_3}}{\omega_{1,3}} + D_{J_2} - \frac{D_{J_3}P_{J_3}}{\omega_{1,3}}\right) \\
&< \chi\left(-\frac{D_{J_3}P_{J_1}(1-P_{J_3}) + D_{J_3}P_{J_3}}{\omega_{1,3}} + D_{J_2}\right) \qquad (24) \\
&= \chi(-D_{J_3} + D_{J_2}) \le 0. \qquad (25)
\end{aligned}
$$

We have used $D_{J_3} < \frac{t_I + D_{J_1}P_{J_1}}{P_{J_1}}$ to obtain (24) and $D_{J_2} \le D_{J_3}$ to obtain (25). Property (b) then follows.

Finally, if $\hat{f}(\vec{\pi}_i, \mathcal{J}_1 \cup \mathcal{J}_3) = \hat{f}(\vec{\pi}_i, \mathcal{J}_1)$, we have $D_{J_3} = \frac{t_I + D_{J_1}P_{J_1}}{P_{J_1}}$ by Property (b). Then, the inequality '$<$' in (24) changes to equality '$=$', which corresponds to Property (d). In particular, equality in (25) holds only if $D_{J_2} = D_{J_3}$, i.e., $D_{j_2} = D_{j_3}$ for all $j_2 \in \mathcal{J}_2$ and $j_3 \in \mathcal{J}_3$. ■

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 *Proceedings of IEEE INFOCOM*, (Pheonix, AZ), April 2008.

[2] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-Saving Protocols for IEEE 802.11-Based Multi-Hop Ad Hoc Networks," *Computer Networks*, 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 Transactions on Networking*, vol. 12, pp. 493–506, June 2004.

[4] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. SenSys*, pp. 171–180, November 2003.

[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*, pp. 224–231, April 2004.

[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.

[8] M. Nosovic and T. Todd, "Low power rendezvous and RFID wakeup for embedded wireless networks," in *Annual IEEE Computer Communications Workshop*, 2000.

[9] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proc. SenSys*, pp. 95–107, November 2004.

[10] J. Polastre, J. Hill, P. Levis, J. Zhao, D. Culler, and S. Shenker, "A Unifying Link Abstraction for Wireless Sensor Networks," in *Proc. SenSys*, pp. 76–89, November 2005.

[11] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance," *IEEE transactions on Mobile Computing*, vol. 2, pp. 349–365, October 2003.

[12] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for Ad hoc and Sensor Networks: Multihop Performance," *IEEE transactions on Mobile Computing*, vol. 2, pp. 337–348, October 2003.

[13] R. R. Choudhury and N. H. Vaidya, "MAC-Layer Anycasting in Ad Hoc Networks," *SIGCOMM Computer Communication Review*, vol. 34, pp. 75–80, January 2004.

[14] S. Jain and S. R. Das, "Exploiting Path Diversity in the Link Layer in Wireless Ad Hoc Networks," in *Proc. WoWMoM*, pp. 22–30, June 2007.

[15] 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), June 2007.

[16] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "Minimizing Delay and Maximizing Lifetime for Wireless Sensor Networks with Anycast," *Technical Report, Purdue University, http://web.ics.purdue.edu/˜kim309/Kim08tech2.pdf*, 2008.

[17] "IRIS OEM Module Datasheet," tech. rep., Crossbow Technology, http://www.xbow.com/Products/.

[18] J. Chang and L. Tassiulas, "Routing for maximum system lifetime in wireless ad-hoc networks," in *37th Annual Allerton Conference on Communication, Control, and Computing*, (Monticello, IL), October 1999.

[19] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *Proceedings of IEEE INFOCOM*, pp. 22–31.

[20] Y. T. Hou, Y. Shi, and H. D. Sherali, "Rate allocation in wireless sensor networks with network lifetime requirement," in *Proceedings of IEEE/ACM MobiHoc*, pp. 67 – 77, 2004.

[21] D. P. Bertsekas, *Dynamic Programming and Optimal Control vol. 2*. Athena Scientific, 3 ed., 2007.

PLACE PHOTO HERE

Joohwan Kim (S'07) received his B.S. degree from Yonsei University, Seoul, Korea, in 2004, and his M.S. degree from Purdue University, West Lafayette, Indiana, in 2006. He is currently a Ph.D. Candidate of Electrical and Computer Engineering at Purdue University.
His research interests range over the various area of wireless communication networks: Scheduling, Routing, Power controlling, Network pricing and Wireless resource optimization in sensor and mobile ad hoc networks.

PLACE PHOTO HERE

Xiaojun Lin (S'02 / M'05) received his B.S. from Zhongshan University, Guangzhou, China, in 1994, and his M.S. and Ph.D. degrees from Purdue University, West Lafayette, Indiana, in 2000 and 2005, respectively. He is currently an Assistant Professor of Electrical and Computer Engineering at Purdue University.
Dr. Lin's research interests are resource allocation, optimization, network pricing, routing, congestion control, network as a large system, cross-layer design in wireless networks, mobile ad hoc and sensor networks.

PLACE PHOTO HERE

Ness B. Shroff (S'91 / M'93 / SM'01 / F'07) Ness B. Shroff received his Ph.D. degree from Columbia University, NY in 1994 and joined Purdue university immediately thereafter as an Assistant Professor. At Purdue, he became Professor of the school of Electrical and Computer Engineering in 2003 and director of CWSA in 2004, a university-wide center on wireless systems and applications. In 2007, he joined The Ohio State University as the Ohio Eminent Scholar of Networking and Communications and Professor of ECE and CSE.
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.

PLACE PHOTO HERE

Prasun Sinha received his Ph.D. from University of Illinois, Urbana-Champaign in 2001, M.S. from Michigan State University in 1997, and B. Tech from IIT Delhi in 1995. He worked at Bell Labs, Lucent Technologies as a Member of Technical Staff from 2001 to 2003. Since 2003, he is an Assistant Professor in Department of Computer Science and Engineering at Ohio State University. His research focusses on design of network protocols for sensor networks and mesh networks.