On Maximizing the Lifetime of Delay-Sensitive Wireless Sensor Networks with Anycast

Joohwan Kim*, Xiaojun Lin†, Ness B. Shroff‡, and Prasun Sinha§
*†School of Electrical and Computer Engineering, Purdue University
‡§Departments of ECE and CSE, The Ohio State University
Email:{*jhkim, †linx}@purdue.edu, ‡shroff@ece.osu.edu, §prasun@cse.ohio-state.edu

***Abstract*—Sleep-wake scheduling is an effective mechanism to prolong the lifetime of energy-constrained wireless sensor networks. However, it incurs an additional delay for packet delivery when each node needs to wait for its next-hop relay node to wake up, which could be unacceptable for delay-sensitive applications. Prior work in the literature has proposed to reduce this delay using anycast, where each node opportunistically selects the first neighboring node that wakes up among multiple candidate nodes. In this paper, we study the joint control problem of how to optimally control the sleep-wake schedule, the anycast candidate set of next-hop neighbors, and anycast priorities, to maximize the network lifetime subject to a constraint on the expected end-to-end delay. We provide an efficient solution to this joint control problem. Our numerical results indicate that the proposed solution can substantially outperform prior heuristic solutions in the literature, especially under the practical scenarios where there are obstructions in the coverage area of the wireless sensor network.**

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

I. Introduction

Sleep-wake scheduling is an effective mechanism to prolong the lifetime of energy-constrained sensor networks. In this paper, we are particularly interested in event-driven wireless sensor networks, where events occur occasionally. Therefore, by putting nodes to sleep when there are no events, the energy consumption of the sensor nodes can be significantly reduced.

In the literature, synchronized sleep-wake scheduling protocols have been proposed in [1]–[3]. In these protocols, sensor nodes periodically or aperiodically exchange synchronization information with neighboring nodes. However, these synchronous protocols could incur additional communication overhead, and consume a considerable amount of energy. In this work, we are interested in asynchronous sleep-wake scheduling protocols such as those proposed in [4], [5]. In these protocols, the sleep-wake schedule at each node is independent of that of other nodes, and thus no synchronization is required. However, due to the lack of knowledge of the sleep-wake schedule of other nodes, it incurs additional delays for packet delivery when each node needs to wait for its next-hop node to wake up. 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.

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).

Prior work in the literature has proposed the use of *anycast* to reduce this event reporting delay [6]–[10]. In contrast to traditional sleep-wake scheduling, where each sending node wakes up a particular next-hop relay node, in anycast each sending node tries to wake up a group of neighboring nodes in a candidate set, and the sending node then picks the first node that wakes up to relay packets. Roughly speaking, if each neighboring node wakes up once every T time, by selecting a candidate set of n nodes, the time needed before the first node wakes up is on average around $\frac{T}{n}$ (assuming that the sleep-wake schedules of the n nodes are independent). Thus, the delay to wake up the next-hop neighbors can be significantly reduced. On the other hand, the end-to-end delay not only depends on the per-hop delay, but also the end-to-end path that packet traverses. Hence, the set of candidate nodes must be carefully chosen because it will also affect the possible routing paths.

The existing anycast schemes in the literature have mainly focused on the so-called "MAC-layer anycast" problem, i.e., they try to find the candidate set at each node such that some local measure of delay is minimized. For the routing path, they either use a separate routing algorithm [8], [9], or rely on geographical information [6], [7], [10]. Thus, the interactions between the choice of the candidate set and the routing path was not systematically studied, and it is then unclear whether such approaches will minimize the actual end-to-end delay. In this paper, we directly optimize the system with respect to the end-to-end delay. In particular, we formulate the joint control problem of how to optimally control the sleep-wake schedule, the anycast candidate set of neighboring nodes, and anycast priorities among neighboring nodes, to maximize the network lifetime subject to a constraint on the end-to-end delay. We provide an efficient solution to this joint control problem, and as a part of solution, we also show how to optimally choose the candidate set in order to minimize the end-to-end delay for all nodes.

The rest of this paper is organized as follows. In Section II, we describe the system model and introduce the lifetime-maximization problem that we intend to solve. In Section III, we analyze the end-to-end delay under anycast, and we develop an optimal distributed anycast algorithm that mini-

mizes the end-to-end delay of all nodes. In Section IV, we solve the lifetime-maximization problem. 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.

We assume that the sensor network employs sleep-wake scheduling to improve energy-efficiency and to prolong the network lifetime. With sleep-wake scheduling, 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 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. If a node wakes up and does not sense any beacon signal or any 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 where nodes do not synchronize their wake-up times, which is easier to implement than other schemes that require global synchronization [1]–[3]. 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 in 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 when a node has to wait for its next-hop node to wake up. To reduce this delay, we use an anycast forwarding scheme described in Fig. 1. Let $\mathcal{C}_i$ denote the set of nodes in the transmission range of node i. Suppose that node i has a packet, and it needs to pick up a node in $\mathcal{C}_i$ 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 a *forwarding set*, which is denoted by $\mathcal{F}_i$. In addition, each node j is also assumed to maintain a list of nodes i that use node j as a forwarder (i.e., $j \in \mathcal{F}_i$). As shown in Fig. 1, node i starts sending a beacon signal and an ID signal, successively. All nodes in $\mathcal{C}_i$ 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 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 some 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. We let t_R denote the resolution period, during which time node i identifies which nodes have sent acknowledgements, and, if there are multiple awake nodes, 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 turn to sleep before the next beacon signal. In this paper, we set $t_{\text{active}} = t_A + t_C$ so that the node that wakes up right before the first beacon signal also has the same chance of detecting the beacon signal as nodes that wake up between two beacon signals.

A. Sleep-wake Schedule, Forwarding Set, and Priority

In this model, there are three control variables that affect the network lifetime and the end-to-end delay experienced by a packet.

1) Sleep-Wake Schedule: The sleep-wake schedule is determined by the 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, it leads to higher energy consumption at node j so that the network lifetime may decrease. In the rest of the paper, it is more convenient to work with the notion of awake probability which is a function of λ_j.

Suppose that node i sends the first beacon signal at time 0, as in Fig. 1. If no nodes in $\mathcal{F}_i$ have heard the first $m-1$ beacon and ID signals, then node i transmits the m-th beacon and ID

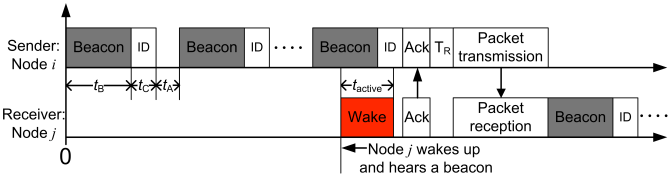

Fig. 1. System Model

signals 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 signals, the probability that node $j \in \mathcal{C}_i$ wakes up and hears these signals is $p_j = 1-e^{-\lambda_j(t_B+t_C+t_A)}$. We call p_j the *awake probability* of node j. It should be noted that, due to the memoryless property of a Poisson random process, p_j is same for each beacon-ID signaling iteration, m.

Note that there is a one-to-one mapping between the awake probability p_j and waking-up frequency λ_j. Hence, an 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, choosing the correct forwarding set is not easy. 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 vertices $V = \mathcal{N}$, and 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. Otherwise, we call it *disconnected to* node j. For convenience, we call a node that is 'connected (disconnected) to sink s' simply as '*connected (disconnected)*.' An acyclic path is the path which 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: When multiple nodes send an acknowledgement after the same ID signal, the source node i needs to pick one of them as a forwarder. We assume that node i assigns priorities to all nodes in $\mathcal{C}_i$, and will pick the node with the highest priority among these nodes that wake up. Clearly, the priority assignment will also affect the expected delay. We use a matrix $\mathbf{B}$ to represent the global priority decision, as follows:

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

where $b_{ij} \in \{1, \cdots, |\mathcal{C}_i|\}$ if $j \in \mathcal{C}_i$, and $b_{ij} = 0$ otherwise. This b_{ij} represents the priority of node j from the viewpoint of node i. We call this matrix $\mathbf{B}$ the priority matrix. The priority matrix $\mathbf{B}$ further satisfies $b_{ij_1} \neq b_{ij_2}$ for all distinct nodes $j_1, j_2 \in \mathcal{C}_i$. Among the awake nodes, the node j that satisfies $b_{ij} > b_{ik}$ for all the other nodes k will be chosen as a forwarder. Note that even though only the nodes in a forwarding set need priorities, we here assign priorities to all nodes to make priority matrix $\mathbf{B}$ an independent control variable from forwarding matrix $\mathbf{A}$. We let $\mathcal{B}$ denote the set of all possible priority matrices.

B. Performance Metrics

We next describe the performance metrics that we are interested in.

1) End-to-End Delay: We assume that the end-to-end delay for event delivery is dominated by the cumulative sum of the delay for each hop to wake up and to relay a packet to its next-hop neighbor. This is a reasonable approximation in many event-driven networks. Note that when an event occurs in an event-driven sensor network, the first packet generated by the event usually suffers most of the delay because at every hop it has to wait for nodes to wake up so that it can be relayed. Once the first packet goes through, the sensor nodes can stay awake for a while, hence the delay of the subsequent packets are often much smaller. Therefore, in this paper, we define the end-to-end delay as the delay incurred by the first packet, which is the sum of the delay for each hop to wake up and to relay the packet to its next-hop neighbor.

Given $\mathbf{A}$, $\mathbf{B}$, and $\vec{p}$, the stochastic process with which a packet traverses the network from the source node to the sink is completely specified, and can be described by a Markov process. 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 awake probability vector $\vec{p}$, forwarding matrix $\mathbf{A}$, and priority matrix $\mathbf{B}$ 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 to sink s under forwarding matrix $\mathbf{A}$, packets from the node cannot reach sink s. Therefore, 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'.

2) Network Lifetime: We assume that node i consumes u_i unit of energy each time it wakes up. Let Q_i be the energy available to node i. Then, the expected lifetime of node i is $\frac{Q_i}{u_i \lambda_i}$. By introducing the power consumption ratio $e_i = u_i/Q_i$, we can express the lifetime of node i as

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

Here we have used the definition of the awake probability $p_i = 1 - e^{-\lambda_j(t_B+t_C+t_A)}$ from (1). Note that in this definition of lifetime we have chosen not to account for the energy consumption by data transmission. This is a reasonable approximation for those event-driven sensor networks where events occur very rarely, in which case the energy consumption of the sensor nodes is dominated by the energy consumed during the sleep-wake scheduling.

We assume that the *network lifetime* is determined by the shortest lifetime of all nodes. In other words, the network

lifetime for a given awake probability vector $\vec{p}$ is given by $T(\vec{p}) = \min_{i\in\mathcal{N}} T_i(\vec{p})$. The methodology of the paper may be extended to handle other definitions of lifetime, e.g., when the sensor network is considered operational if less than a certain percentage of nodes are alive. However, we leave this more general definition of lifetime for future work.

3) Problem Formulation: The objective of this paper is to choose awake probability vector $\vec{p}$, forwarding matrix $\mathbf{A}$, and priority matrix $\mathbf{B}$ to maximize the network lifetime, subject to the constraint that the expected delay from each node to sink s is below the maximum allowable delay, i.e.,

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

where ξ^* is the maximum allowable delay.

III. Minimization of End-to-End delays for given awake probabilities

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

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 node j in $\mathcal{C}_i$ becomes a forwarder right after the h-th beacon-ID signals is equal to the probability that no nodes in $\mathcal{C}_i$ have woken up for the past $h-1$ beacon-ID-signaling iterations, and 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 $(t_B+t_C+t_A)h + t_R + t_P + D_j(\vec{p},\mathbf{A},\mathbf{B})$. 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 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:

$$\begin{aligned} & D_i(\vec{p},\mathbf{A},\mathbf{B}) \\ = & \sum_{h=1}^{\infty}\sum_{j\in\mathcal{F}_i(\mathbf{A})}[(t_I h + t_D + D_j(\vec{p},\mathbf{A},\mathbf{B}))P_{j,h}] \\ = & \ t_D + \frac{t_I}{1-\prod_{j\in\mathcal{F}_i(\mathbf{A})}(1-p_j)} \\ & + \frac{\sum_{j\in\mathcal{F}_i(\mathbf{A})} D_j(\vec{p},\mathbf{A},\mathbf{B})p_j \prod_{k\in\mathcal{F}_i(\mathbf{A}):b_{ij}<b_{ik}}(1-p_k)}{1-\prod_{j\in\mathcal{F}_i(\mathbf{A})}(1-p_j)}. \end{aligned} \quad (2)$$

We call (2) *the local delay relationship*, which must hold for all nodes i except the sink s. (Recall $D_s(\vec{p},\mathbf{A},\mathbf{B}) = 0$ regardless of the delay of neighboring nodes.)

B. The Optimal Forwarding Set and Priority Assignment

In this subsection, we first consider the hypothetical scenario where a node i knows the delays D_j from its neighboring nodes j to sink s, and such delays D_j are fixed. Under this hypothesis, we will study how node i should adjust its own forwarding set and priority assignment to minimize the expected delay from node i to sink s. Then, in the next subsection, we will use the insight from the result to determine the optimal forwarding matrix $\mathbf{A}$, and priority matrix $\mathbf{B}$.

Consider that a node i has multiple neighboring nodes with fixed delay. Similar to (2), we can calculate the expected delay from node i to sink s for a given neighboring delay vector $\vec{\pi}_i = (D_j, j\in\mathcal{C}_i)$, forwarding set $\mathcal{F}_i$, and priority assignment $\vec{b}_i$ as

$$\begin{aligned} & f(\vec{\pi}_i,\mathcal{F}_i,\vec{b}_i) \\ \triangleq & \ t_D + \frac{t_I + \sum_{j\in\mathcal{F}_i} D_j p_j \prod_{k\in\mathcal{F}_i:b_{ij}<b_{ik}}(1-p_k)}{1-\prod_{j\in\mathcal{F}_i}(1-p_j)}. \end{aligned} \quad (3)$$

We call the function $f(\cdot,\cdot,\cdot)$ *the local delay function*.

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 delay, i.e., for each pair of nodes j and k satisfying $b_{ij}^* < b_{ik}^*$, the inequality $D_k \le D_j$ holds. Then, for any given $\mathcal{F}_i$,

$$f(\vec{\pi}_i,\mathcal{F}_i,\vec{b}_i^*) \le f(\vec{\pi}_i,\mathcal{F}_i,\vec{b}_i) \quad (4)$$

for all possible $\vec{b}_i$.

The detailed proof is provided in Appendix A in our on-line technical report [11]. The intuition behind Proposition 1 is that when multiple nodes sends acknowledgements, selecting the node with the smallest delay should minimize the expected delay. Therefore, priorities must be assigned to neighboring nodes according to their (known) delays D_j, independent of 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 $\hat{f}(\vec{\pi}_i, \mathcal{F}_i) \triangleq f(\vec{\pi}_i, \mathcal{F}_i, \vec{b^*_i}(\vec{\pi}_i))$.

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 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 $j_3 \in \mathcal{J}_3$.

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 $P_{J_k} = 1 - \prod_{j\in\mathcal{J}_k}(1-p_j)$, i.e., the probability that any node in $\mathcal{J}_k$ wakes up. Then, the local delay function can be expressed as

$$\hat{f}(\vec{\pi}_i, \cup_{k=1}^K \mathcal{J}_k) = t_D + \frac{t_I + \sum_{k=1}^K D_{J_k} P_{J_k} \prod_{l=1}^{k-1}(1-P_{J_l})}{1 - \prod_{k=1}^K (1-P_{J_k})}$$

for $K = 1, 2, 3$. Then, by algebraic manipulation, we can establish Properties (a)-(d). Details are again available in Appendix B in [11]. ■

The interpretation of Proposition 2 is straightforward. For example, Property (a) implies that adding 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 in $\mathcal{J}_3$ plus t_D is smaller than 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) For all nodes j in $\mathcal{C}_i$ that satisfy $D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}^*_i) - t_D$, the following relationship holds,

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

We can prove Proposition 3 by using the result of Proposition 2. (See the detail provided in Appendix C in [11].)

From Proportion 3, 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 $\{j \in \mathcal{C}_i | D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}^*_i) - t_D\}$. This means that if there exists a node j such that $D_j = \hat{f}(\vec{\pi}_i, \mathcal{F}^*_i) - t_D$, $\mathcal{F}^*_i$ is not unique. In other words, 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.

Since the optimal forwarding set consists of nodes whose delay is smaller than or equal to some threshold value, the simplest solution to find the optimal forwarding sets is to run an exhaustive 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 $\mathcal{F}_{i,k} = \{j \in \mathcal{C}_i | b^*_{ij}(\vec{\pi}_i) \geq k\}$. If there are multiple optimal forwarding sets, we only need to find one of them. In this paper, we chose to use the set $\mathcal{F}^*_i = \{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}^*_i) - t_D\}$ as the optimal forwarding set because it is the first one that we can obtain in the exhaustive search. Therefore, we redefine the optimal forwarding set $\mathcal{F}^*_i$ as the forwarding set that satisfies both $\mathcal{F}^*_i = \arg\min_{\mathcal{F}_i \subset \mathcal{C}_i} \hat{f}(\vec{\pi}_i, \mathcal{F}_i)$ and $\mathcal{F}^*_i = \{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}^*_i) - t_D\}$. Note that with this definition, the optimal forwarding set is unique. Then, the following lemma helps us to find the optimal forwarding set more quickly.

***Lemma** 1:* For all $\mathcal{F} \subset \mathcal{C}_i$ that satisfies $\mathcal{F} = \{j \in \mathcal{C}_i | D_j < \hat{f}(\vec{\pi}_i, \mathcal{F}) - t_D\}$, $\mathcal{F}^*_i \subset \mathcal{F}$.

Proof: From Proposition 3 (a) and the definition of $\mathcal{F}^*_i$, all nodes $k \in \mathcal{F}^*_i$ satisfy $D_k < \hat{f}(\pi_i, \mathcal{F}^*_i) - t_D \leq \hat{f}(\pi_i, \mathcal{F}_i) - t_D$, for any subset $\mathcal{F}_i \subset \mathcal{C}_i$. Since $\mathcal{F} \subset \mathcal{C}_i$, we obtain $D_k < \hat{f}(\pi_i, \mathcal{F}) - t_D$ for all nodes $k \in \mathcal{F}^*_i$. Hence, $\mathcal{F}^*_i \subset \mathcal{F}$. ■

Lemma 1 implies that when we exhaustively 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 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 $b^*_{il}(\vec{\pi}_i) = k - 1$, $D_l \geq \hat{f}(\vec{\pi}_i, \mathcal{F}_{i,k}) - t_D$.

It should be noted that the optimal forwarding set is time-invariant due to the memoryless property of a Poisson random process. Specifically, the expected time for each node j in $\mathcal{C}_i$ to wake up is always the same as t_I/p_j regardless of how long the source node have waited. Therefore, the strategy to minimize the expected delay is also time-invariant.

C. Globally Optimal Forwarding and Priority Matrices

We next use the insight of Section III-B to develop an algorithm computing the globally optimal forwarding and priority matrices for given $\vec{p}$. This algorithm has the flavor of the distributed Bellman-Ford's algorithm for finding the shortest paths. At each iteration, each node uses the delay estimates from the previous iteration to update the forwarding set and the priority assignment. We will show that the algorithm converges in N iterations, and the resulting $\mathbf{A}$ and $\mathbf{B}$ minimize the expected delay $D_i(\vec{p}, \mathbf{A}, \mathbf{B})$.

The algorithm is presented next.

The OPT-DELAY Algorithm

Step (1) At iteration 0, each node i sets

$$D_i^{(0)} = \begin{cases} 0 & \text{if } i = s, \\ \infty & \text{otherwise.} \end{cases}$$

and $\mathcal{F}_i^{(0)} = \emptyset$. Each node arbitrarily assigns priorities to neighboring nodes.

Step (2) At iteration h (≥ 1), each node i sets $\vec{b}_i^{(h)} = \vec{b}_i^*(\vec{\pi}_i^{(h-1)})$, where $\vec{\pi}_i^{(h-1)} = (D_j^{(h-1)}, j \in \mathcal{C}_i)$.

Step (3) Each node i updates $\mathcal{F}_i^{(h)}$ by finding the optimal forwarding set for $\vec{\pi}_i^{(h-1)}$ and also updates $D_i^{(h)}$ as follows

$$D_i^{(h)} = \hat{f}(\vec{\pi}_i^{(h-1)}, \mathcal{F}_i^{(h)}). \tag{5}$$

Step (4) If $D_i^{(h)} = D_i^{(h-1)}$ for all nodes $i \in \mathcal{N}$, this algorithm terminates. Otherwise, each node increases h by one and goes back to **Step (2)**.

To analyze the OPT-DELAY algorithm, we will use the following notations. We define the subgraph $g_i(\mathbf{A}) = G(V_i(\mathbf{A}), E_i(\mathbf{A}))$ as the graph with vertices $V_i(\mathbf{A}) = \{j \in V(\mathbf{A}) | \, i \text{ is connected to } j \text{ in } g(\mathbf{A})\}$ and edges $E_i(\mathbf{A}) = \{(j,k) \in E(\mathbf{A}) | \{j,k\} \subset V_i(\mathbf{A})\}$. By convention, node i is connected to itself, i.e., $i \in V_i(\mathbf{A})$ for all $\mathbf{A} \in \mathcal{A}$. This subgraph $g_i(\mathbf{A})$ shows all possible paths from node i under forwarding matrix $\mathbf{A}$. For any forwarding matrix $\mathbf{A}$, the number of distinct acyclic paths in $g(\mathbf{A})$ is finite when the total number of nodes is finite. Let $|g(\mathbf{A})|$ be the maximum length of acyclic paths in $g(\mathbf{A})$. Then, the following proposition states an important property for analyzing the OPT-DELAY algorithm.

***Proposition** 4:* For any $\vec{p}$, $\mathbf{A} \in \mathcal{A}$, and $\mathbf{B} \in \mathcal{B}$ such that $g_i(\mathbf{A})$ is cyclic, there exist $\mathbf{A}' \in \mathcal{A}$ and $\mathbf{B}' \in \mathcal{B}$ such that $g_i(\mathbf{A}')$ is acyclic, $|g_i(\mathbf{A}')| \leq |g_i(\mathbf{A})|$, and

$$D_i(\vec{p}, \mathbf{A}', \mathbf{B}') \leq D_i(\vec{p}, \mathbf{A}, \mathbf{B}).$$

The detailed proof is provided in Appendix D in [11]. Proposition 4 implies that for any forwarding and priority matrices that cause a cyclic path from any node i to sink s, there always exist other forwarding and priority matrices with which all paths from node i to sink s are acyclic, and the delay from node i with the new matrices is equal to or smaller than the delay with the original matrices. This is intuitively true because it will incur higher delay if the packets have to traverse loops.

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 $\vec{b}_i^{(h)}$. Let $\mathbf{A}^*(\vec{p})$ and $\mathbf{B}^*(\vec{p})$ be the forwarding and priority matrices when the OPT-DELAY algorithm converges. (Note that $\vec{p}$ is fixed and given.)

The following proposition provides the key properties of the algorithm.

***Proposition** 5:* The algorithm has the following properties:

1) At iteration h, $g(\mathbf{A}^{(h)})$ is an acyclic graph.
2) The OPT-DELAY algorithm converges within N iterations.
3) For given $\vec{p}$, $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) = \arg\min_{\mathbf{A},\mathbf{B}} D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ for all nodes $i \in \mathcal{N}$.

Proof: In this paper, we show the basic ideas of the proof. The detailed version of proof is provided in Appendix E in [11].

We first show that $D_i^{(h+1)} \leq D_i^{(h)}$ for $h \geq 1$ and all nodes i. Suppose in contrary that there exists node i such that $D_i^{(h)} < D_i^{(h+1)}$. Then, from [11], we can show that there must exist node $i_1 \in \mathcal{C}_i$ such that $D_{i_1}^{(h-1)} < D_{i_1}^{(h)}$. Repeating this procedure, we can find a sequence of nodes $i_2, \cdots, i_k, \cdots, i_{h-1}$ such that $i_k \in \mathcal{C}_{i_{k-1}}$ and $D_{i_k}^{(h-k)} < D_{i_k}^{(h-k+1)}$. For node i_{h-1}, $D_{i_{h-1}}^{(1)} < D_{i_{h-1}}^{(2)}$. However, if sink s is in i_{h-1}'s transmission range, i.e., $s \in \mathcal{C}_{i_{h-1}}$, then $D_{i_{h-1}}^{(k)} = D_{i_{h-1}}^{(1)} = t_D + t_I/p_s$ at any iteration k, because node i_{h-1} can deliver the packet directly to sink s. If sink s is not in $\mathcal{C}_{i_{h-1}}$, $D_{i_{h-1}}^{(2)} \leq D_{i_{h-1}}^{(1)} = \infty$. This leads to a contradiction. Thus, $D_i^{(h+1)} \leq D_i^{(h)}$ for all nodes $i \in \mathcal{N}$ and iteration $h > 1$.

We now prove the first property. Suppose in contrary that there is a cyclic path in $g(\mathbf{A}^{(h)})$. Let the sequence of nodes along this cyclic path be $i_1, i_2, \cdots, i_K$, and $i_{K+1} = i_1$, i.e., $i_{k+1} \in \mathcal{F}_{i_k}^{(h)}$ for $k = 1, 2, ..., K$. Then, from [11], we can show that the delays along the cyclic path satisfy

$$D_{i_1}^{(h)} > D_{i_2}^{(h)} > \cdots > D_{i_K}^{(h)} > D_{i_1}^{(h)}.$$

This is a contradiction. Therefore, $g(\mathbf{A}^{(h)})$ is an acyclic graph.

Let $\mathcal{A}_i^{(h)} = \{\mathbf{A} \in \mathcal{A} | |g_i(\mathbf{A})| \leq h\}$. $\mathcal{A}_i^{(h)}$ denotes the set of forwarding matrices with which the maximum number of hops along acyclic paths from node i to sink s is less than h. We now show that

$$D_i^{(h)} \leq \min_{\mathbf{A} \in \mathcal{A}_i^{(h)}, \mathbf{B}} D_i(\vec{p}, \mathbf{A}, \mathbf{B}). \tag{6}$$

We prove by induction. At iteration 1, $\mathcal{F}_i^{(1)} = \{s\}$ and $D_i^{(1)} = t_D + t_I/p_s$ if sink $s \in \mathcal{C}_i$. Otherwise, $\mathcal{F}_i^{(1)} = \emptyset$ and $D_i^{(1)} = \infty$. Now consider the right-hand-side of (6). If $\mathcal{A}_i^{(1)}$ is not an empty set, this means that node i has a direct path to sink s, which implies that its expected delay is $t_D + t_I/p_s$. If $\mathcal{A}_i^{(1)}$ is an empty set, this means that there is no path for node i to reach sink s within 1 hop, which implies that the expected delay to reach sink s within 1 hop is infinite. Therefore, (6) holds at iteration 1.

Next, assume that the induction hypothesis (6) holds at iteration h, i.e.,

$$D_i^{(h)} \leq \min_{\mathbf{A} \in \mathcal{A}_i^{(h)}, \mathbf{B}} D_i(\vec{p}, \mathbf{A}, \mathbf{B}) \qquad \forall i \in \mathcal{N}. \tag{7}$$

Then, using Proposition 4, we can show that (7) also holds at iteration $h+1$. (See the detail in Appendix E in [11].) Hence, (6) holds for all nodes.

We next prove that $D_i(\vec{p}, \mathbf{A}^{(h)}, \mathbf{B}^{(h)}) \leq D_i^{(h)}$. At iteration 1, $\mathcal{F}_i^{(1)} = \{s\}$ if $s \in \mathcal{C}_i$. Otherwise, $\mathcal{F}_i^{(1)} = \emptyset$. Hence,

$$D_i(\vec{p}, \mathbf{A}^{(1)}, \mathbf{B}^{(1)}) = \begin{cases} t_D + t_I/p_s & \text{if } s \in \mathcal{C}_i, \\ \infty & \text{otherwise.} \end{cases}$$

Thus, $D_i(\vec{p}, \mathbf{A}^{(1)}, \mathbf{B}^{(1)}) = D_i^{(1)}$ for all nodes i, and so $D_i(\vec{p}, \mathbf{A}^{(h)}, \mathbf{B}^{(h)}) \leq D_i^{(h)}$ holds at iteration 1.

Next assume that the induction hypothesis

$$D_i(\vec{p}, \mathbf{A}^{(l)}, \mathbf{B}^{(l)}) \leq D_i^{(l)} \quad \forall i \in \mathcal{N} \tag{8}$$

holds for all $l \leq h$. Then, from [11], we can show that (8) also holds for $l = h + 1$. Hence, $D_i(\vec{p}, \mathbf{A}^{(h)}, \mathbf{B}^{(h)}) \leq D_i^{(h)}$ holds for all i and h.

From the previous results, we conclude that

$$D_i(\vec{p}, \mathbf{A}^{(h)}, \mathbf{B}^{(h)}) \leq D_i^{(h)} \leq \min_{\mathbf{A} \in \mathcal{A}_i^{(h)}, \mathbf{B}} D_i(\vec{p}, \mathbf{A}, \mathbf{B}) \tag{9}$$

The maximum length of an acyclic path is equal to or less than N. Therefore, $\mathcal{A}_i^{(N)} = \mathcal{A}$ for all nodes i. Since $\mathbf{A}^{(h)} \in \mathcal{A}$, at iteration N, we obtain

$$D_i(\vec{p}, \mathbf{A}^{(N)}, \mathbf{B}^{(N)}) = D_i^{(N)} = \min_{\mathbf{A}, \mathbf{B}} D_i(\vec{p}, \mathbf{A}, \mathbf{B})$$

from (9). Hence, the algorithm must converge in at most N iterations, and $\mathbf{A}^{(h)}$, $\mathbf{B}^{(h)}$, and $D_i^{(h)}$ converge to the optimal forwarding matrix, the optimal priority matrix, and the minimum expected delay from node i to sink s, respectively. ■

Proposition 5 shows that there always exists $(\mathbf{A}, \mathbf{B})$ that can minimize the delay from *all* nodes at the same time, and $(\mathbf{A}(\vec{p}), \mathbf{B}(\vec{p}))$ corresponds to such a solution. Furthermore, the graph $g(\mathbf{A}^*(\vec{p}))$ is acyclic. The complexity of this algorithm is given by $\mathcal{O}(N)$. Moreover, this algorithm can be implemented in a fully distributed fashion.

IV. Solution to The Lifetime-maximization Problem

In this section, we solve the original lifetime-maximization problem **(P)**, using the results in previous sections. By letting $q_i = \ln(1 - p_i)^{-e_i}$, we can rewrite problem $(\mathbf{P})$ as

$$\begin{aligned} (\mathbf{P1}) \quad \max_{\vec{q}, \mathbf{A}, \mathbf{B}} \quad & \min_{i \in \mathcal{N}} \frac{t_I}{q_i}, \\ \text{subject to} \quad & D_i(\vec{p}, \mathbf{A}, \mathbf{B}) \leq \xi^*, \quad \forall i \in \mathcal{N} \\ & p_i = 1 - e^{-q_i/e_i}, \quad \forall i \in \mathcal{N} \\ & q_i \in (0, \infty), \quad \forall i \in \mathcal{N} \\ & \mathbf{A} \in \mathcal{A}, \quad \mathbf{B} \in \mathcal{B}. \end{aligned} \tag{10}$$

Since for any given $\vec{p}$, $\mathbf{A}^*(\vec{p})$ and $\mathbf{B}^*(\vec{p})$ are the optimal forwarding matrix and the optimal priority matrix, respectively, that minimize the delay from all nodes, we have $D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \leq D_i(\vec{p}, \mathbf{A}, \mathbf{B})$ for all $\mathbf{A}$ and $\mathbf{B}$. Hence, we can rewrite problem **(P1)** as follows:

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

Problem **(P2)** can be further simplified with the following proposition.

***Proposition** 6:* If $\vec{q}^*$ is the optimal solution to problem $(\mathbf{P2})$, then so is $\vec{q}$ such that $\vec{q} = (q_i = \max_k q_k^*, i \in \mathcal{N})$, i.e., we can let every node have the same q_i.

Proof: Since both solutions have the same objective value, it is sufficient to show that if $\vec{q}^*$ is in the feasible set, so is $\vec{q}$. Let $\vec{p}*$ and $\vec{p}$ be the awake probability vectors that correspond to $\vec{q}^*$ and $\vec{q}$, respectively, by (10). Since p_i is monotonically increasing as q_i increases, and $\vec{q}^* \preceq \vec{q}$, we have $\vec{p}^* \preceq \vec{p}$. (The symbol '$\preceq$' denotes componentwise inequality, i.e., if $\vec{q} \preceq \vec{p}$, then $q_i \leq p_i$ for all i, where q_i and p_i are the i-th components of $\vec{q}$ and $\vec{p}$, respectively.)

Note 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 $\vec{p}$. (See Appendix F in [11].) Since $\vec{p}^* \preceq \vec{p}$, for all nodes i, we have $D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \leq D_i(\vec{p}^*, \mathbf{A}^*(\vec{p}^*), \mathbf{B}^*(\vec{p}^*))$. Hence, if $\vec{q}^*$ is in the feasible set, so is $\vec{q}$. ■

Using the above proposition, we can rewrite problem **(P2)** into a problem with one variable q,

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

If q^* is the solution to problem **(P3)**, then $(\vec{p}^*, \mathbf{A}(\vec{p}^*), \mathbf{B}(\vec{p}^*))$ $(p_i^* = 1 - e^{-q^*/e_i})$ 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 a non-increasing function of p_i. (See the proof of Proposition 6.) Since $\vec{p}$ is an increasing vector of q, the simplest solution to Problem **(P3)** is to linearly search q such that $\max_{i \in \mathcal{N}} D_i(\vec{p}, \mathbf{A}(\vec{p}), \mathbf{B}(\vec{p})) = \xi^*$ where $p_i = 1 - e^{-q/e_i}$.

We develop an efficient binary search algorithm for computing the optimal value of q.

The Binary Search Algorithm for Problem (P3)

Step (1) Initially, sink s sets $p^{(1)} = 0.5$ and $k = 1$.

Step (2) Sink s sets $q^{(k)} = \ln(1 - p^{(k)})^{-\max_{i \in \mathcal{N}} e_i}$.

Step (3) Nodes run the OPT-DELAY algorithm for given $\vec{p}^{(k)} = (p_i^{(k)} = 1 - e^{-q^{(k)}/e_i}, i \in \mathcal{N})$.

Step (4) After N iterations, the optimal forwarding set and the optimal priority assignment under $\vec{p}^{\,(k)}$ are found. 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 delays $D_j(\vec{p}^{(k)}, \mathbf{A}^*(\vec{p}^{(k)}), \mathbf{B}^*(\vec{p}^{(k)}))$ to sink s.

Step (5) Let $D_{\max}$ be the maximum feedback delay arrived at sink s.

- If $D_{\max} > \xi^* + \epsilon$, then sink s sets $p^{(k+1)} = p^{(k)} + 0.5^{k+1}$, increases k by one, and goes back to **Step (2)**.
- If $D_{\max} < \xi^* - \epsilon$, then sink s sets $p^{(k+1)} = p^{(k)} - 0.5^{k+1}$, increases k by one, and goes back to **Step (2)**.
- If $D_{\max} \in [\xi^* - \epsilon, \xi^* + \epsilon]$, then the algorithm terminates, and returns $q^{(k)}$ as the optimal solution to Problem **(P3)**.

The reason that we take $q^{(k)}$ with respect to the maximum e_i in **Step (2)** is because this makes all $p_i^{(k)}$ less than or equal to $p^{(k)}$. (Note that we only search $p^{(k)}$ over $(0, 1]$.) In

Step (4), only such a node j that does not belong to any other forwarding set needs to send the feedback delay to the sink s because the node with the maximum delay does not belong to any other forwarding set according to Property (a) in Proposition 3. Since sink s only needs to know the maximum delay, there is no need for the other nodes to feedback their delays.

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 uniform distribution, and the sink s is located at $(0, 0)$. We assume that the transmission range from each node i is a disc with radius 1.5, i.e., $j \in \mathcal{C}_i$, if the distance between node j and node i is less than 1.5. The parameters t_I and t_D are set to 1 and 5, respectively. We also assume that 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 [10] is an anycast-based heuristic that exploits geographic information to reduce the delay from each node. Let d_i be the distance from node i to sink s, and let r_{ij} be the progress from node i to node j toward sink s, i.e., $r_{ij} = d_i - d_j$. 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, which might help to reduce the actual end-to-end delay.

Naive Anycast Algorithm: The *naive* algorithm proposed in [10] 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 OPT-DELAY algorithm, we update $\mathcal{F}_i^{(h)}$ according to

$$\mathcal{F}_i^{(h)} = \arg\min_{\mathcal{F}\subset\mathcal{C}_i:|\mathcal{F}|=1} \hat{f}(\vec{\pi}_i^{(h-1)}, \mathcal{F}), . \tag{11}$$

After the above modification, the OPT-DELAY 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 OPT-DELAY algorithm in **Step (3)** with the above mentioned algorithms.

B. Performance Comparison

In Fig. 2, we compare the network lifetime under the different algorithms, where x-axis represents different maximum allowable delays ξ^* in our original Problem **(P)**, and 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 OPT-DELAY 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.

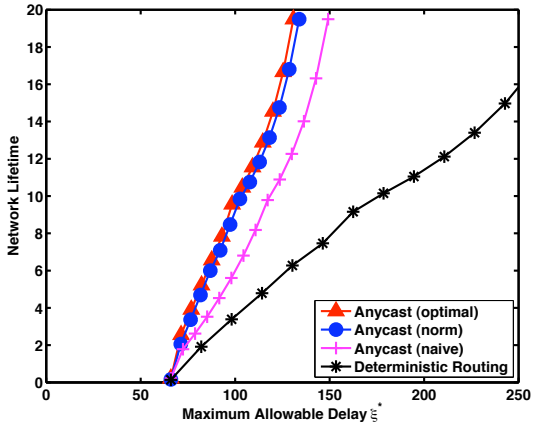

Fig. 2. The network lifetime according to different allowable delay ξ^* when nodes are uniformly deployed.

From Fig. 2, 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. 2 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 soon.

We next simulate a topology where there is a hole in the sensor field as shown in Fig. 3. 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. 4.

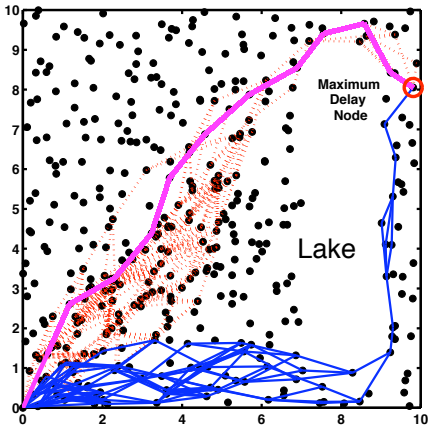

Fig. 3. Node deployment and routing paths under different forwarding algorithms when $p_i = 0.5$: The dotted lines illustrate all routing paths under the optimal anycast algorithm, the thick solid lines illustrate the unique routing path under the deterministic routing path, and thin solid lines illustrate all routing paths under the normalized-latency anycast algorithm

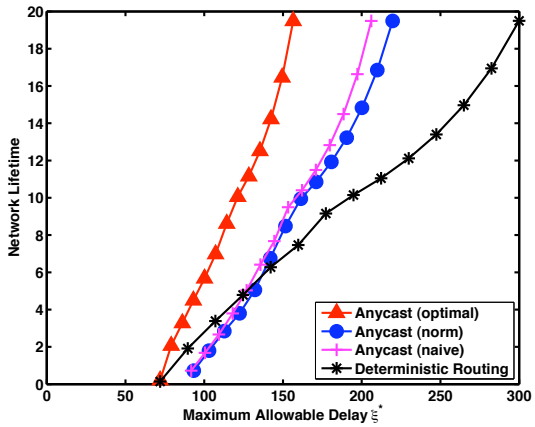

Fig. 4. The network lifetime according to different allowable delay ξ^* when nodes are not uniformly distributed.

From this figure, we observe that the optimal anycast algorithm substantially outperforms the other algorithms including the normalized-latency anycast algorithm. Fig 3 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 (above 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 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 study how to use anycast to reduce the end-to-end delay and to prolong the lifetime of wireless sensor networks employing asynchronous sleep-wake scheduling. In particular, we study the joint control problem of how to optimally control the sleep-wake schedule, the anycast candidate set of next-hop neighbors, and the anycast priorities, in order to maximize the network lifetime subject to a upper limit on the expected end-to-end delay. We provide an efficient solution to this joint control problem, and as a part of the solution, we also show how to optimally choose the anycast candidate set to minimize the end-to-end delay from all sensor nodes. 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.

The algorithms that we have developed can be easily applied to energy-constrained event-driven wireless sensor networks. In future work, we plan to extend the result to the case with non-Poisson sleep-wake patterns, and to handle more general notions of network lifetime.

References

[1] 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, no. 3, pp. 493–506, June 2004.

[2] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proc. SenSys'03*, November 2003, pp. 171–180.

[3] 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'04*, April 2004, pp. 224–231.

[4] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for WIreless Sensor Networks," in *Proc. SenSys'04*, November 2004, pp. 95–107.

[5] J. Polastre, J. Hill, P. Levis, J. Zhao, D. Culler, and S. Shenker, "A Unifying Link Abstraction for Wireless Sensor Networks," in *Proc. SenSys'05*, November 2005, pp. 76–89.

[6] 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, no. 4, pp. 349–365, October 2003.

[7] ——, "Geographic Random Forwarding (GeRaF) for Ad hoc and Sensor Networks: Multihop Performance," *IEEE transactions on Mobile Computing*, vol. 2, no. 4, pp. 337–348, October 2003.

[8] R. R. Choudhury and N. H. Vaidya, "MAC-Layer Anycasting in Ad Hoc Networks," *SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 75–80, January 2004.

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

[10] 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'07*, San Diego, CA, June 2007.

[11] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "On Maximizing the Lifetime of Delay-Sensitive Wireless Sensor Networks with Anycast," *Technical Report, Purdue University, http://web.ics.purdue.edu/~kim309/Kim08tech.pdf*, 2007.