# Optimal Anycast Technique for Delay-Sensitive Energy-Constrained Asynchronous Sensor Networks

Joohwan Kim*, Xiaojun Lin†, and Ness B. Shroff‡
*†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

***Abstract*—In wireless sensor networks, asynchronous sleep-wake scheduling protocols can significantly reduce energy consumption without incurring the communication overhead for clock synchronization used in typical sleep-wake scheduling protocols. However, the savings could come at a significant cost in delay performance. Recently, researchers have attempted to exploit the inherent broadcast nature of the wireless medium to reduce this delay with virtually no additional energy cost. These schemes are called "anycasting," where each sensor node forwards the packet to the first node that wakes up among a set of candidate next-hop nodes. In this paper, we develop a delay-optimal anycasting scheme under periodic sleep-wake patterns. Our solution is computationally simple and fully distributed. We show that periodic sleep-wake patterns result in the smallest delay among all wake-up patterns under given energy constraints. Simulation results illustrate the benefit of our proposed schemes over the state-of-the art.**

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

## I. Introduction

The most efficient method to save energy in wireless sensor networks (WSNs) is to put nodes to sleep when there is no need to relay or transmit packets. Such mechanisms are called *sleep-wake scheduling* and have been used to dramatically reduce energy consumption in energy-constrained WSNs. However, it is well known that sleep-wake scheduling can significantly increase the packet-delivery delay because, at each hop, an event-reporting packet has to wait for its next-hop node to wake up. Such additional delays can be detrimental to delay-sensitive applications, such as Tsunami/fire detection, environmental monitoring, security surveillance, etc. In this paper, we study how to improve this tradeoff between energy-savings and delay, by using a new technique called "*anycasting*" (to be described later) that exploits the broadcast nature of the wireless medium.

We focus on asynchronous sleep-wake scheduling, where nodes do not synchronize their clocks with other nodes and thus wake up independently. Asynchronous sleep-wake scheduling is simpler to implement, and it does not consume energy required for synchronizing sleep-wake schedules across the network. However, because nodes do not know the wake-up schedules of other nodes, asynchronous sleep-wake scheduling could result in large one-hop delays.

This work was supported in part by ARO Awards W911NF-07-10376 (SA08-03) and W911NF-08-1-0238, and NSF Awards 0626703-CNS, 0635202-CCF, 0721236-CNS, and 0721477-CNS.

Recently, *anycast packet-forwarding schemes* have been used to substantially reduce the one-hop delay under asynchronous sleep-wake scheduling [1]–[6]. Note that in traditional packet-forwarding schemes, nodes forward their packets to their designated next-hop nodes. In contrast, in anycast-based forwarding schemes, nodes maintain multiple candidates of next-hop nodes and forward their packet to the *first* candidate node that wakes up. Hence, an anycast forwarding scheme can substantially reduce the one-hop delay over traditional schemes, especially when nodes are densely deployed, as is the case for many WSN applications. (See the example in Section I and Fig. 1 of [6] that illustrates the advantage of anycasting over the traditional schemes.) However, the reduction in the one-hop delay may not necessarily lead to the reduction in the expected end-to-end delay experienced by a packet because the first candidate node that wakes up may not have a small expected end-to-end delay to the sink. Hence, the anycast forwarding policy (with which nodes decide whether or not to forward a packet to an awake node) needs to be carefully designed.

In our prior work [5], [6], we developed a distributed anycast forwarding policy that simultaneously minimizes the expected end-to-end delays from all nodes to the sink, when the wake-up rates of the nodes are given. (The wake-up rate represents the frequency with which a node wakes up.) However, the delay-optimal anycast policy in [5], [6] is based on the assumption that nodes wake up according to a Poisson process (i.e., the wake-up intervals of a node are i.i.d. exponential random variables). Hence, the following open questions need to be considered: (1) If we can control the wake-up patterns (subject to given wake-up rates) in addition to the anycast forwarding policy, is there a wake-up pattern that results in optimal delay performance? and (2) If such a pattern exists, which forwarding policy is delay-optimal for the wake-up pattern? These questions are more complex than the one in [5], [6] because we can no longer exploit the memoryless property of a Poisson Process. In this paper, we extend the results in [5], [6] to address these questions.

## II. System Model

We consider an event-driven WSN with $N$ sensor nodes. Let $\mathcal{N}$ be the set of all nodes. We assume in this paper that event information is reported to a single sink node $s$, but the analysis can be readily extended to the scenario with multiple sink nodes. Each node $i$ has a set $\mathcal{N}_i$ of neighboring nodes to which node $i$ is able to directly transmit packets.

The lifetime of an event-driven WSN under asynchronous sleep-wake scheduling consists of two phases: *the configuration phase* and *the operation phase*. When sensor nodes are deployed, the configuration phase begins, during which the nodes determine their packet-forwarding and sleep-wake scheduling policies. It is also during this phase that the optimization on these policies (which we will study in this paper) is carried out. Once the optimal policies are determined, the operation phase begins, during which the nodes apply the policies determined in the configuration phase to perform their main functions: detecting events and reporting the event information. Specifically, during this phase, sensor nodes alternate between sleeping and waking up independently of other nodes. Consider a node that wakes up and hears a request from a neighboring node for relaying the event-reporting packets. If it is an eligible next-hop node based on the packet-forwarding policy, it receives the packet and then finds a new next-hop node to forward the packet. If the node successfully forwards the packets, it returns to sleep and follows the sleep-wake scheduling policy again.

### A. Basic Forwarding and Sleep-Wake Scheduling Protocols

We first introduce the basic packet-forwarding and sleep-wake scheduling protocols that are used in the operation phase.

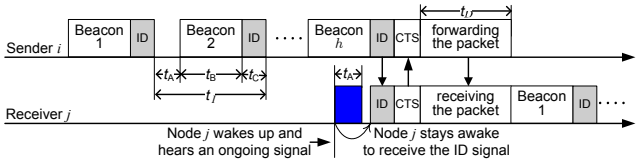



Fig. 1. System model

*Packet-Forwarding Protocol:* When a node $i$ has a packet to deliver to the sink, it must wait for its neighboring nodes to wake up. Under asynchronous sleep-wake scheduling, we simply assume that the clocks are not synchronized. Hence, the sending node $i$ does not know exactly when its neighboring nodes will wake up (although it may have some statistical information of their wake-up patterns and wake-up rates).[1] Fig. 1 describes the protocol with which sending node $i$ transmits its packet to one of its neighboring nodes. As soon as node $i$ is ready to transmit the packet, it sends a beacon signal (Beacon 1 in Fig. 1) of duration $t_B$, and ID signal of duration $t_C$, and then listens for acknowledgements (CTS: Clear-To-Send) for duration $t_A$. The sending node repeats this sequence until it hears an acknowledgement. The ID signal contains the identity of the sending node and the sequence number of the last beacon signal. When a node wakes up and senses the $h$-th beacon signal, it will stay awake to decode the following ID signal, in which case we say that the node receives the $h$-th ID signal. (If a node wakes up in the middle of the ID signal, it must stay awake to decode the next ID signal.) Then, such a node has two choices. **Choice 1:** If the node chooses to receive the packet, it responds with a CTS message containing its identity during the acknowledgement period $t_A$ that immediately follows the ID signal. Once the sending node hears the CTS, it forwards the packet to the awake node during the data transmission period $t_D$. **Choice 2:** If the awake node decides not to receive the packet, it goes back to sleep. For simplicity of notation, let $t_I = t_B + t_C + t_A$, which denotes the duration of each beacon-ID signaling iteration (See Fig. 1).

*Remark:* In the above basic protocol, we have ignored the possibility of collisions, which can be due to either multiple awake nodes or multiple sending nodes. In our on-line technical report [7, Section V], we describe an extended packet-forwarding protocol that addresses these collision scenarios using random or deterministic back-offs. However, in a low duty-cycle WSN, chances are small that multiple neighboring nodes wake up at the same beacon signal. Due to this reason, we use the basic protocol for our analysis and study the effect of collisions using simulation in Section V.

*Sleep-Wake Scheduling Protocol:* In order to save energy, each node wakes up infrequently and goes back to sleep if there is no activity in the neighborhood. Note that if the duration for which the node stays awake is shorter than $t_A$, the node may stay awake only within an acknowledgement period $t_A$ and miss on-going beacon-ID signals. In order to avoid such a case, we assume that nodes must stay awake for at least $t_A$. Since a longer awake duration results in higher energy consumption, we set the awake duration to be exactly equal to $t_A$. The next time to wake up is determined by the sleep-wake scheduling policy of the node.

### B. Sleep-Wake Scheduling and Anycast Forwarding Policies

In this subsection, we define the sleep-wake scheduling and anycast forwarding policies that are computed during the configuration phase and applied during the operation phase. These policies affect the end-to-end delay experienced by a packet, and the energy consumption of the network.

**Sleep-Wake Scheduling Policy:** We assume that each node $i$ wakes up according to a stationary and ergodic point process. We use two control variables $r_i$ and $w_i$ to denote such a wake-up process.

Wake-up Rate: We define the wake-up rate $r_i$ of node $i$ as the expected number of times that node $i$ wakes up per unit time. Let $\vec{r} = (r_1, r_2, \cdots, r_N)$ be the global wake-up rate (or simply the wake rate). Note that a higher wake-up rate consumes energy faster.

[1] It may be possible for neighboring nodes to synchronize their clocks when they are forwarding event-reporting packets. However, since events occur rarely compared to the wake-up rates, by the time that the next event occurs, their clocks will drift substantially.

Wake-up Pattern: For any stationary and ergodic wake-up process of a node $i$, by re-scaling time, we can convert it to a process with a wake-up rate of 1. We call this scaled process as the wake-up pattern $w_i$ of node $i$. Let $\vec{w} = (w_1, w_2, \cdots, w_N)$ denote the global wake-up pattern (or simply the wake-up pattern). If a node $i$ chooses *a periodic wake-up pattern* $w_i = w_{\text{per}}$ and a wake-up rate $r_i$, then it will wake up every $1/r_i$ time, i.e., the wake-up intervals are given by $1/r_i$. If the Poisson wake-up pattern is chosen, the intervals will be i.i.d exponential random variables with mean $1/r_i$.

Suppose that a node has an event-reporting packet to forward at a given time $t$. We assume that the wake-up processes of its neighboring nodes are independent of each other. The assumption is reasonable because under asynchronous sleep-wake scheduling, the node with the packet has not synchronized its clock with its neighboring nodes since it relayed a packet last time. Hence, due to the random drift of clock offsets, it is difficult for the node to know the exact timing when its neighboring nodes will wake up.

**Anycast Forwarding Policy:** Suppose that a sending node $i$ has sent the $h$-th beacon-ID signal, and a set $X \subset \mathcal{N}_i$ of the neighboring nodes wakes up and receives the ID signal. We let $f_{i,h}(X)$ denote the corresponding decision of the sending node $i$, which is to be specified next. We let $f_{i,h}(X) = j$ if the sending node $i$ decides to transmit the packet to node $j \in X$, and we let $f_{i,h}(X) = i$ if the sending node $i$ decides to send out the $(h+1)$-st beacon-ID signal, i.e., the packet remains at node $i$. This forwarding decision is inconsistent with the packet-forwarding protocol decribed in Subsection II-A, in which the sending node is restricted to transmit the packet whenever it receives a CTS. However, we only use this general setting to find the optimal forwarding decisions and then show that such optimal decisions can be implemented by our packet-forwarding protocol. Let $f_i = \{f_{i,1}, f_{i,2}, \cdots\}$ denote the anycast forwarding policy of node $i$ (or simply the anycast policy of node $i$). We further denote by $f = \{f_1, f_2, \cdots, f_N\}$ the global anycast forwarding policy (or simply the anycast policy).

### C. Performance Metrics and Optimization

In this section, we define the notion of the end-to-end delay. We then formulate the problem of minimizing the end-to-end delay by jointly controlling the anycast forwarding policy and the sleep-wake scheduling policy.[2]

**Expected end-to-end delay:** During the operation phase, we define the end-to-end delay as the delay from the time when a source node detects an event and generates the event-reporting packet (or packets) to the time the *first* packet is received at the sink. For applications that use a single packet to carry the event information, the above definition captures the actual delay for reporting the event information. For applications that use multiple packets, if the nodes that relayed the first packet stay awake for a while, the delay to relay subsequent packets will be much smaller than that experienced by the first packet. (For instance, these subsequent packets may be sent a few nodes behind the first packet, and hence they can reach the sink soon after the first packet reaches the sink.) Hence, the actual event-reporting delay can still be approximated by the delay experienced by the first packet.

The sleep-wake scheduling policy $(\vec{r}, \vec{w})$ and anycast forwarding policy $f$ fully determine the stochastic process with which the first packet traverses the network from the source node to the sink. Hence, we use $D_i(\vec{r}, \vec{w}, f)$ to denote the expected end-to-end delay from node $i$ to the sink under the joint policy $(\vec{r}, \vec{w}, f)$. For simplicity, from now on, we simply call the expected end-to-end delay from node $i$ to the sink as "the delay from node $i$".

*Delay-Minimization Problem:* The objective of this paper is to find the optimal joint policy $(\vec{w}, f)$ that solves the following delay-minimization problem for given wake-up rate $\vec{r}$:

$$\min_{\vec{w}, f} D_i(\vec{r}, \vec{w}, f). \tag{1}$$

Note that $\vec{r}$ controls the duty cycle of the sensor network, which in turn controls the energy expenditure. Hence, the problem can also be viewed as minimizing the delays for a given energy budget. In Sections III and IV, we develop an algorithm that solves this problem for all nodes $i$, i.e., our solution can simultaneously minimize the delays from all nodes.

## III. Delay-Optimal Anycast Policy for a Given Sleep-Wake Scheduling Policy

As a preliminary step to solving the delay-minimization problem, in this section we first fix a sleep-wake scheduling policy $(\vec{r}, \vec{w})$ and study delay-optimal anycast policies for the fixed policy $(\vec{r}, \vec{w})$. We will then focus on the case when the wake-up patterns of nodes are periodic, and show that the delay-optimal forwarding policy for the periodic wake-up patterns has a much simpler form.

Given a sleep-wake scheduling policy $(\vec{r}, \vec{w})$, the delay-minimization problem can be formulated as a stochastic shortest path (SSP) problem [8, Chapter 2], where the sensor node that has a packet corresponds to the "state", and the delay corresponds to the "cost" that we intend to minimize. The sink $s$ corresponds to the terminal state, where no further cost (delay) will be incurred. Let $i_0, i_1, i_2, \cdots, i_L = s$ be the sequence of nodes that relay the packet from the source node $i_0$ to the sink $s$ in $L$ steps. Note that under anycasting, the sequence is random because each node has a set of candidate next-hop nodes and does not know which of them will wake up first to receive the packet. Then, the end-to-end delay $D_i(\vec{r}, \vec{w}, f)$ from each node $i_0$ to the sink can be expressed as

$$D_i(\vec{r}, \vec{w}, f) = E\Big\{ \sum_{l=0}^{L} D_{\text{hop},i_l}(\vec{r}, \vec{w}, f_{i_l}) \Big\}, \tag{2}$$

where $D_{\text{hop},i_l}(\vec{r}, \vec{w}, f_{i_l})$ is the expected one-hop delay at node $i_l$, and the expectation is taken with respect to the random

[2] As mentioned earlier, our goal during the configuration phase is to design the system to minimize the delay of interest during the operation phase.

sequence $i_1, i_2, \cdots, i_L$. Given the sleep-wake scheduling policy $(\vec{r}, \vec{w})$, let $D_i^*(\vec{r}, \vec{w}) \triangleq \min_f D_i(\vec{r}, \vec{w}, f)$ be the minimum expected delay from node $i$. Then, according to the Bellman equation [8, Section 2.2], for all nodes $i$, the minimum delay $D_i^*(\vec{r}, \vec{w})$ must satisfy

$$D_i^*(\vec{r}, \vec{w}) = \min_{f_i} \Big( D_{\text{hop},i}(\vec{r}, \vec{w}, f_i) + \sum_{j \in \mathcal{N}_i} q_{i,j}(\vec{r}, \vec{w}, f_i) D_j^*(\vec{r}, \vec{w}) \Big), \tag{3}$$

where $q_{i,j}(\vec{r}, \vec{w}, f_i)$ is the probability that node $j$ is chosen as the next-hop node of node $i$ under the forwarding policy $f_i$. Further, using the following value-iteration algorithm [8, Section 1.3], we can find the delay-optimal forwarding policy that achieves $D_i^*(\vec{r}, \vec{w})$ for all nodes $i$:

**Value Iteration Algorithm:** At the initial iteration $k = 0$, all nodes $i$ set their initial delay values $D_i^{(0)}$ to $\infty$, and the sink $s$ sets its delay value $D_s^{(0)}$ to zero. At each iteration $k = 1, 2, \cdots$, every node $i$ collects the delay values $D_j^{(k-1)}$ from its neighboring nodes $j$ and then updates its delay value $D_i^{(k)}$ by solving

$$D_i^{(k)} = \min_{f_i} \Big( D_{\text{hop},i}(\vec{r}, \vec{w}, f_i) + \sum_{j \in \mathcal{N}_i} q_{i,j}(\vec{r}, \vec{w}, f_i) D_j^{(k-1)} \Big). \tag{4}$$

Let $f_i^{(k)}$ be the forwarding policy of node $i$ that minimizes (4). Then, according to [8, Proposition 2.2.2], the delay value $D_i^{(k)}$ of each node $i$ converges to the minimum delay $D_i^*(\vec{r}, \vec{w})$, i.e., $\lim_{k\to\infty} D_i^{(k)} = D_i^*(\vec{r}, \vec{w})$, and the corresponding forwarding policy $f^{(k)} = \{f_1^{(k)}, f_2^{(k)}, \cdots\}$ also converges to the delay-optimal forwarding policy, i.e., $\lim_{k\to\infty} f^{(k)} \in \arg\min_f D_i(\vec{r}, \vec{w}, f)$ for all nodes $i$.

The key step in this value iteration algorithm is how every node $i$ solves the sub-problem in (4) at each iteration $k$. Note that this subproblem is equivalent to the following problem: we need to find a forwarding policy of node $i$ that minimizes the expected delay from node $i$ when the delays from neighboring nodes $j$ to the sink are given by $D_j^{(k-1)}$, and the sleep-wake scheduling policy is given by $(\vec{r}, \vec{w})$. In the next two subsections, we will study how to solve this sub-problem.

### *A. Necessary Conditions for the Optimal Anycast Policy*

To solve the above sub-problem, we focus on a node $i$ that has a packet. For ease of exposition, let the delays from neighboring nodes $j$ be denoted by $D_j = D_j^{(k-1)}$ $(j \in \mathcal{N}_i)$, which is equal to $D_j^{(k-1)}$ for iteration $k$ in the value-iteration algorithm. Without loss of generality, we assume that the node $i$ has neighboring nodes $1, 2, \cdots, N_i$ ($N_i = |\mathcal{N}_i|$), and their delays are sorted in increasing order, i.e., $D_1 \le D_2 \le \cdots \le D_{N_i} < \infty$. (We exclude the neighboring nodes with infinite delays.) To avoid confusion, we further assume that the index $i$ of the sending node is larger than $N_i + 1$.

After the sending node $i$ sends out the $h$-th beacon signal, it has to choose either to **transmit** the packet to one of the awake nodes or to **wait** for the other node to wake up by sending the next beacon signal. We call this moment the decision stage $h$ (or simply stage $h$) and denote the set of the awake nodes at this moment by $X_h$. By definition, $f_{i,h}(X_h) = j$ $(j \in X_h)$ implies that node $i$ decides to transmit to node $j$, and $f_{i,h}(X_h) = i$ implies that node $i$ decides to wait and send the $(h+1)$-th beacon signal. Since stage 0 is the moment when node $i$ is about to send the first beacon signal, we set $X_0 = \emptyset$ and $f_{i,0}(X_0) = i$. Given the expected delay $D_1, D_2, \cdots, D_{N_i}$, the sub-problem (4) can also be modeled as an infinite horizon dynamic programming (DP) problem [8, Chapter 1], where the states at stage $0, 1, \cdots$ are given by the sets $X_0, X_1, \cdots$ of awake nodes. This state transition terminates whenever the sending node transmits the packet to an awake node $j$. When this happens, the packet will be relayed by the node $j$ and eventually arrive at the sink after $D_j$ time (the delay from node $j$). We denote this terminal state by state 0. Under the following assumption, the state transition $X_0, X_1, \cdots$ can be expressed more simply:

**Assumption 1**: If an awake node is not chosen as the next-hop node, we assume that the node stays awake to remain eligible to be chosen as the next-hop node at following stages. Under this assumption, the state transition must satisfy $X_0 \subseteq X_1 \subseteq \cdots$.

*Remark:* Assumption 1 not only simplifies the analysis, but it also clearly leads to smaller delay, compared with the case where an awake node can return to sleep when it is not immediately chosen as the next-hop node. However, one could argue that keeping nodes awake consumes more energy. In Section III-B, we will show the following: when the neighboring nodes wake up periodically, the delay-optimal anycast forwarding policy achieves the minimum delay without Assumption 1, and thus the awake nodes in fact do not need to stay awake. But for now, we use the assumption to simplify the analysis.

Since the number of possible states at each stage increases exponentially with the number $N_i$ of neighboring nodes ($2^{N_i}$ states at each stage), it is more convenient to deal with a simpler transition model as follows. Note that if node $i$ decides to transmit the packet to one of the awake nodes in $X_h$, clearly it should choose the node $j$ with the smallest delay $D_j$ among all the awake nodes in order to minimize the delay from the next-hop node. Hence, at each stage $h$, node $i$ only needs to remember the awake node with the smallest delay. In other words, if a delay-optimal policy is applied, only the awake node with the smallest delay affects the state transition dynamics. We denote this node by $x_h = \arg\min_{j \in X_h} D_j$. If no nodes are awake ($X_h = \emptyset$), we simply set $x_h = N_i + 1$. (For example, since $X_0 = \emptyset$, the initial state is always given by $x_0 = N_i + 1$.) From now on, we can use a simpler state transition model $x_0, x_1, x_2, \cdots$ to solve the sub-problem (4) without any loss of optimality. Due to the same principle, we abuse notations slightly, and use $f_{i,h}(x_h)$ to denote the decision of node $i$ at state $x_h$ as follows: $f_{i,h}(x_h) = x_h$ if the sending node $i$ decides to transmit the packet to node $x_h$, and $f_{i,h}(x_h) = i$ if the node $i$ decides to wait. We denote the terminal state by $x_h = 0$. Under Assumption 1, the state transition must satisfy $x_0 \ge x_1 \ge, \cdots$ because $X_0 \subseteq X_1 \subseteq \cdots$ and $D_1 \le D_2 \le \cdots, \le D_{N_i}$.

We next consider the state transition probability. Let $P_{x,x'}^{(h)}$ be the state transition probability from state $x_{h-1}=x$ to state $x_h=x'$, given that node $i$ decides to wait at state $x_{h-1}=x$, i.e., $P_{x,x'}^{(h)} \triangleq \Pr\big(x_h=x'|x_{h-1}=x \text{ and } f_{i,h-1}(x)=i\big)$. Let $p_{j,h}$ be the conditional probability that node $j$ wakes up at stage $h$ conditioned on not having woken up at earlier stages. Using $p_{j,h}$, we can express the state transition probability as

$$P_{x,x'}^{(h)} = \begin{cases} \prod_{j=1}^{x'-1} p_{x',h}(1-p_{j,h}) & \text{if } x'<x, \\ \prod_{j=1}^{x'-1}(1-p_{j,h}) & \text{if } x'=x. \end{cases} \quad (5)$$

The state transition probability conditioned on $f_{i,h-1}(x)=x$ is trivial because, if the sending node decides to transmit the packet to node $x$, the next state must be 0. We say that state $x_h=x$ is *admissible* if $\Pr(x_h=x|f_{i,h'}(x_{h'})=i,\forall h'<h)>0$.

In our dynamic programming problem, the cost to be minimized is delay. Let $g(x_h,f_{i,h}(x_h))$ be the one-step delay between stages $h$ and $h+1$ when decision $f_{i,h}$ is used at state $x_h$. If the sending node $i$ sends out the next beacon signal ($f_{i,h}(x_h)=i$), the delay incurred by this decision is the beacon-ID signaling duration $t_I$. If node $i$ transmits the packet, the packet will be transmitted to the next-hop node $x_h$ for the packet transmission period $t_D$ and will arrive at the sink $D_{x_h}$ time later. Hence, the delay incurred by this decision is $t_D+D_{x_h}$. Once the packet reaches the sink, there will be no more delay to be incurred. Hence, the one-step delay can be expressed as

$$g(x_h,f_{i,h}(x_h)) = \begin{cases} t_I & \text{if } f_{i,h}(x_h)=i, \quad (6) \\ t_D+D_{x_h} & \text{if } f_{i,h}(x_h)=x_h. \quad (7) \end{cases}$$

for $x_h\neq 0$ and $g(x_h,f_{i,h}(x_h))=0$ for $x_h=0$. Using the above state transition probability and the one-step delay, we can represent the sub-problem (4) as the following dynamic program (DP) problem [8, Chapter 1]: given the delays $D_j$ from the neighboring nodes $j$, we want to find the anycast forwarding policy $f_i$ of node $i$ that minimizes the overall cost (delay) function $d_{f_i}=\lim_{\bar{h}\to\infty} E\left\{\sum_{h'=0}^{\bar{h}-1} g(x_{h'},f_{i,h'}(x_{h'}))\right\}$, where $x_0,x_1,x_2,\cdots$ are the states visited, and the expectation is taken with respect to these states. Define the optimal delay function $d^*$ and the optimal forwarding policy $f_i^*$ of the sending node $i$ as $d^*\triangleq\min_{f_i} d_{f_i}$ and $f_i^*=\arg\min_{f_i} d_{f_i}$, respectively. Then, $d^*$ and $f_i^*$ in this sub-problem corresponds to $D_i^{(k)}$ and $f_i^{(k)}$ in (4), respectively.

To solve this DP problem, we define $d^{(h)}(x_h)$ as the expected delay from state $x_h\geq 1$ at stage $h$, given that the optimal forwarding policy is applied afterward, i.e.,

$$d^{(h)}(x_h) \triangleq \min_{f_{i,h},f_{i,h+1},\cdots} \left(\lim_{\bar{h}\to\infty} E\left\{\sum_{h'=h}^{\bar{h}-1} g(x_{h'},f_{i,h'}(x_{h'}))\right\}\right)$$

where $x_{h+1},x_{h+2},\cdots$ are the states to be visited after stage $h$, and the expectation is taken with respect to these states. By definition, it immediately follows that $d^{(0)}(N_i+1)=d^*$. The delay function $d^{(h)}(x_h)$ can be interpreted as the minimum expected delay from state $x_h$. Suppose that the sending node $i$ at state $x_h$ decides to transmit (**TX**) the packet to node $x_h$ ($f_{i,h}(x_h)=x_h$). By (6), the minimum expected delay $d_{\text{TX}}^{(h)}(x_h)$ conditioned on this decision is $t_D+D_{x_h}$. If node $i$ decides to **wait** ($f_{i,h}(x_h)=i$), the minimum expected delay $d_{\text{wait}}^{(h)}(x_h)$ conditioned on this decision is given by

$$d_{\text{wait}}^{(h)}(x_h) = t_I + \sum_{x_{h+1}=1}^{x_h} P_{x_h,x_{h+1}}^{(h+1)} d^{(h+1)}(x_{h+1}), \quad (8)$$

*assuming that the optimal decisions $f_{i,h+1}^*,f_{i,h+2}^*,\cdots$ are applied afterward.* Using these conditional delays, we can express $d^{(h)}(x_h)$ by the following Bellman equation [8, Equation (1.3)]:

$$d^{(h)}(x_h) = \min(d_{\text{wait}}^{(h)}(x_h), d_{\text{TX}}^{(h)}(x_h)). \quad (9)$$

Note that by setting $D_{N_i+1}=\infty$, we can still use (9) even when $x_h=N_i+1$, in which case $d^{(h)}(N_i+1)$ is always equal to $d_{\text{wait}}^{(h)}(N_i+1)$. (In other words, if no nodes are awake, the only choice left is to send the next beacon-ID signal.)

From (9), we can immediately obtain the optimal forwarding decision at stage $h$ as follows:

$$f_{i,h}^*(x_h) = \begin{cases} x_h & \text{if } D_{x_h} \leq d_{\text{wait}}^{(h)}(x_h) - t_D, \\ i & \text{otherwise.} \end{cases} \quad (10)$$

Clearly, whenever node 1 has woken up, the optimal decision is to forward the packet to node 1. Hence, the optimal forwarding decision must satisfy

$$f_{i,h}^*(1)=1 \text{ and } d^{(h)}(1)=t_D+D_1 \text{ for all } h. \quad (11)$$

Then, the following proposition holds, which will be used for later analysis:

***Proposition** 1:* For all stages $h$ and admissible states $x_h=x',x''$, if $x'<x''$, then we have $d_{\text{wait}}^{(h)}(x')\leq d_{\text{wait}}^{(h)}(x'')$, and thus $d^{(h)}(x')\leq d^{(h)}(x'')$ from (9).

The detailed proof is provided in [7, Appendix A]. This proposition implies that, the better state the sending node is in, the better delay performance the node will have.

We have shown that for an arbitrary sleep-wake process the delay value $d^{(h)}$ and the optimal forwarding decision $f_{i,h}^*$ must satisfy the necessary conditions in (9) and (10), respectively. If there is a reference stage $\bar{h}$ such that the minimum delay $d^{(\bar{h})}(x_{\bar{h}})$ is known for all admissible states $x_{\bar{h}}$, we can then use (8) and (9) as a backward iteration from stage $\bar{h}$ to stage 0, and can solve the sub-problem (4) via the value iteration algorithm. Consequently, we can solve the SSP problem in (2) as well. In the next subsection, we will show that such a reference stage $\bar{h}$ exists when the neighboring nodes wake up periodically. Using this property, we also construct the optimal forwarding policy under the periodic wake-up patterns of the neighboring nodes.

### *B. Optimal Anycast Policy for Periodic Wake-Up Processes*

We now construct the optimal anycast policy of the sending node $i$ when the given wake-up patterns of neighboring nodes are periodic, i.e., each neighboring node $j$ wakes up every $1/r_j$ time unit. Then, each node $j$ must wake up by stage

$h_{j,\max} \triangleq \lfloor \frac{1/r_j}{t_I} \rfloor$. Under Assumption 1, node $j$ must be awake after stage $h_{j,\max}$. Let $\bar{h} \triangleq h_{1,\max}$. Since node 1 must wake up no later than stage $\bar{h}$, the state $x_{\bar{h}}$ must be 1 if the packet has not been forwarded until stage $\bar{h}-1$, i.e., $P^{(\bar{h})}_{x_{\bar{h}-1},1} = 1$ for $x_{\bar{h}-1} \neq 0$. Then, by (11), the minimum delay $d^{(\bar{h})}(1)$ at stage $\bar{h}$ is $t_D + D_1$ and $x_{\bar{h}} = 1$ is the only admissible state in this stage. Hence, we can find the optimal forwarding policy $f_i^*$ using the backward recursive algorithm (9) and (10) from stage $\bar{h}$ to stage 0. (We will study the conditional awake probability $p_{j,h}$ in (15) in order to obtain the state transition probability $P^{(h+1)}_{x_h,x_{h+1}}$, which is required to run the recursive algorithm.) The resulting complexity is $\mathcal{O}(N_i^2\bar{h})$ because there are $\bar{h}$ stages, at each stage the minimal delay $d^{(h)}(x_h)$ must be computed for all possible states $x_h$, and computing each $d^{(h)}(x_h)$ requires $x_h$ number of summations (See (8)).

Although we have found the optimal forwarding policy $f_i^*$, it is difficult to implement such a policy because of the following reasons. First, the optimal forwarding policy requires the knowledge of the list ($X_h$ or $x_h$) of awake nodes at each stage $h$. It can be difficult to acquire this information during a short period $t_A$ between two beacon-ID signals because of collisions. Second, the optimal policy is based on Assumption 1, which requires that an awake node stay awake even if it is not immediately chosen as the next-hop node. However, if the node is not chosen as the next-hop node in the end, the additional energy that it has spent to remain awake is then wasted.

To resolve these implementation problems, we develop an alternative policy $\hat{f}_i$ and show that it also achieves optimal performance. Note that at each stage $h$, there is a set of nodes that must be awake under the periodic wake-up patterns. Specifically, at stage $h$ at least the nodes $j$ such that $h_{j,\max} \leq h$ must be awake under Assumption 1. Among these nodes, let $x_{h,\max}$ be the index of the node with the smallest delay. Since the delays are assumed to be sorted in increasing order, we have $x_{h,\max} = \arg\min_{j=1,2,\cdots,N_i+1:h_{j,\max}\leq h} D_j$. Since at least node $x_{h,\max}$ must be awake, state $x_h$ must satisfy $x_h \leq x_{h,\max}$. Note that $x_{h,\max}$ can be $N_i+1$ if no nodes $j \in \mathcal{N}_i$ have $h_{j,\max} \leq h$ (i.e., if $\Pr(X_h = \emptyset) > 0$). To summarize, under the periodic wake-up processes, the admissible states $x_h$ must satisfy the following conditions:

$$\begin{aligned} &(1)\, x_0 = N_i + 1, \text{ and } x_h = 1 \text{ for } h \geq \bar{h}, \\ &(2)\, x_h \leq x_{h,\max} \text{ and } x_h \leq x_{h-1} \text{ for } h > 0. \end{aligned} \tag{12}$$

Let $\overline{d}^{(h)}_{\text{wait}} \triangleq d^{(h)}_{\text{wait}}(x_{h,\max})$. According to Proposition 1, the condition $x_h \leq x_{h,\max}$ leads to $d^{(h)}_{\text{wait}}(x_h) \leq \overline{d}^{(h)}_{\text{wait}}$. In other words, state $x_{h,\max}$ can be interpreted as the worst admissible state at stage $h$ because $\overline{d}^{(h)}_{\text{wait}}$ is the worst conditional delay. We now define the following alternative anycast policy $\hat{f}_i$ as follows:

$$\hat{f}_{i,h}(x_h) = \begin{cases} x_h & \text{if } D_{x_h} \leq \overline{d}^{(h)}_{\text{wait}} - t_D, \\ i & \text{otherwise.} \end{cases} \tag{13}$$

The main difference between $\hat{f}_i$ and the optimal policy $f_i^*$ is that the decision criteria $d^{(h)}_{\text{wait}}(x_h) - t_D$ in (10) is replaced by the state-independent criteria $\overline{d}^{(h)}_{\text{wait}} - t_D$. Next, we will show that the alternative policy $\hat{f}_i$ is easier to implement and it agrees with the optimal forwarding policy $f_i^*$, i.e., $\hat{f}_{i,h}(x_h) = f^*_{i,h}(x_h)$ for all $h = 0, 1, \cdots, \bar{h}$ and all admissible states $x_h$. To this end, we need to study the conditional delay $d^{(h)}_{\text{wait}}(x_h)$, which plays a key role in the decision criteria (10). We first state the following proposition that allows us to relax Assumption 1 without loss of optimality (to be discussed after Proposition 3).

***Proposition** 2:* For all $h = 1, 2, \cdots, \bar{h}$, and all admissible states $x_{h-1}$ and $x_h$, the following holds:

$$d^{(h-1)}_{\text{wait}}(x_{h-1}) \geq d^{(h)}_{\text{wait}}(x_h). \tag{14}$$

The detailed proof is provided in [7, Appendix B]. The result of Proposition 2 can be interpreted as follows: as more stages pass by, the neighboring nodes are more likely to wake up, and the conditional delay $d^{(h)}_{\text{wait}}$ then decreases.

*Remark:* From Proposition 2, we can infer that $\overline{d}^{(h-1)}_{\text{wait}} \geq \overline{d}^{(h)}_{\text{wait}}$ because it is a special case of (14) when $x_{h-1} = x_{h-1,\max}$ and $x_h = x_{h,\max}$. This implies that the decision criteria $\overline{d}^{(h)}_{\text{wait}} - t_D$ in (13) decreases with $h$. Hence, if the node $x_h$ with the smallest delay among awake nodes is not chosen as a next-hop node at stage $h$ (i.e., $D_{x_h} > \overline{d}^{(h)}_{\text{wait}} - t_D$), it can not be the next-hop node afterwards under the alternative policy $\hat{f}_i$. Thus, the alternative forwarding policy in fact does not need the awake node to remain awake.

The next proposition states that the alternative policy achieves optimal performance.

***Proposition** 3:* For $h = 0, 1, \cdots, \bar{h}$ and all admissible states $x_h$, the optimal and the alternative forwarding policies agree, i.e., $f^*_{i,h}(x_h) = \hat{f}_{i,h}(x_h)$.

The detailed proof is provided in [7, Appendix C]. From Proposition 3, the alternative forwarding policy $\hat{f}_i$ and the optimal forwarding policy $f_i^*$ are the same at all admissible states. Hence, there is no degradation in delay performance when the alternative policy is applied.

Using Proposition 2 and Proposition 3, we can easily implement the alternative (optimal) policy. As stated in Section II, the sending node $i$ broadcasts beacon-ID signals until a CTS is received. Recall that in the original optimal policy, all awake nodes must respond with a CTS at each stage, and the sending node decides whether to wait or to send the packet. In the alternative policy, each awake node can decide whether it should respond with a CTS. Specifically, each neighboring node $j$ maintains an integer parameter $h_j^{(i)} = \max_{h:D_j \leq \overline{d}^{(h)}_{\text{wait}} - t_D} h$, which denotes the last stage when node $j$ can be the next-hop node. If node $j$ wakes up and receives an ID signal, it recognizes the index $i$ of the sending node and the sequence number $h$ from the ID signal. If $h > h_j^{(i)}$, then node $j$ returns to sleep. If $h \leq h_j^{(i)}$, node $j$ responds with a CTS. If there is only one node that responds with a CTS, the sending node should always send the packet to this node. When multiple eligible nodes $j$ such that $D_j < \overline{d}^{(h)}_{\text{wait}} - t_D$ wake up at the

same stage, a collision resolution protocol like the one in [7, Appendix E] can be employed. (Note that such collision is rare in low duty-cycle WSN.) In this case, the node with the smallest delay among the awake nodes will be chosen as the next-hop node. *The resulting forwarding policy will be the same as the alternative policy* $\hat{f}_{i,h} = f^*_{i,h}$. In the end of [7, Section III-B], we provide the so-called LOCAL-OPT algorithm that implements the alternative forwarding policy. This algorithm finds the last stage $h_j^{(i)}$ and the conditional delay $\overline{d}_{\text{wait}}^{(h)}$ and has a computational complexity of $\mathcal{O}(N_i\bar{h})$, which is lower than the complexity $\mathcal{O}(N_i\bar{h})$ of the original backward recursive algorithm using (9) and (10).

### C. Convergence of the Value-Iteration Algorithm

Let $\vec{w}_{\text{per}}$ denote the global sleep-wake scheduling policy where each node wakes up periodically. In Section III-B, we have solved the sub-problem (4) in the value iteration algorithm when the wake-up patterns are given by $\vec{w}_{\text{per}}$. As a result, the value-iteration algorithm solves $\min_f D_i(\vec{r}, \vec{w}_{\text{per}}, f)$ for all nodes $i$. In this subsection, we study the convergence properties of the value-iteration algorithm.

Recall that $D_i^*(\vec{r}, \vec{w}) = \min_f D_i(\vec{r}, \vec{w}, f)$, and $N$ is the number of nodes. Then, the next proposition states the convergence of the value-iteration algorithm:

***Proposition** 4:* At the end of the $N$-th iteration, we must have $D_i^{(N)} = D_i(\vec{r}, \vec{w}_{\text{per}}, f^{(N)}) = D_i^*(\vec{r}, \vec{w}_{\text{per}})$.

In this paper, we only provide the insight of the proof. First we can show that there is an optimal forwarding policy that minimizes (2) for all nodes and under which packets do not pass through the same node twice. Then, according [8, Page 106, Finite Termination of Value Iteration], the existence of such an acyclic optimal forwarding policy leads to the convergence within $N$ iterations. The detailed proof is provided in [7, Proposition 4].

From Proposition 4, every node needs to run the LOCAL-OPT algorithm for only $N$ iterations, and the last forwarding policy $f^{(N)}$ is delay-optimal when all nodes wake up periodically. Hence, the overall complexity experienced by each node $i$ is $O(N_i\bar{h}N)$. This computation overhead only occurs at the configuration phase. Note that the value-iteration algorithm is fully distributed.

## IV. Optimal Wake-up Pattern

In the previous section, we have solved the delay-minimization problem under the periodic wake-up patterns. In this section, we show that among all wake-up patterns with the same wake-up rates, the periodic wake-up patterns can minimize the delays from all nodes.

### A. Fundamental Properties of Wake-up Patterns

We begin by studying the fundamental properties of the wake-up patterns. We define the residual time $R_j$ as the interval from the time that the sending node $i$ starts sending beacon-ID signals to the next wake-up time of node $j$. We further define the function $F_{R_j}(y)$ as the *cdf* of the residual time. Note that since nodes wake up independently of other nodes under asynchronous sleep-wake scheduling, the residual time $R_j$ is independent of those of other nodes. Let $F^*_{R_j}(y)$ be the *cdf* of the residual time $R_j$ when the given node $j$ uses the periodic wake-up patterns. Since the node wakes up every $\frac{1}{r_j}$ time in a periodic wake-up process with a random offset, the residual time $R_j$ is uniformly distributed in $[0, \frac{1}{r_j}]$. Hence, the *cdf* of the residual time under the periodic wake-up process is $F^*_{R_j}(y) \triangleq r_j y 1_{\{0 \le y \le \frac{1}{r_j}\}} + 1_{\{y > \frac{1}{r_j}\}}$. The following proposition then shows the essential properties of the *cdf* of the residual time.

***Proposition** 5:* For any stationary and ergodic wake-up process with rate $r_j$, the *cdf* $F_{R_j}(y)$ of the residual time $R_j$ satisfies the following properties:

**(a)** $F_{R_j}(y) \le F^*_{R_j}(y)$,

**(b)** $dF_{R_j}(y) \le dF^*_{R_j}(y)$ for $0 \le y \le \frac{1}{r_j}$.

The detailed proof is provided in [7, Section IV-A]. Proposition 5 shows that for all $0 \le y \le 1/r_j$, the *cdf* $F_{R_j}(y)$ and the derivative $\frac{dF_{R_j}(y)}{dy}$ are maximized when the wake-up pattern is periodic. [3]

Recall that the awake probability $p_{j,h}$ is defined as the conditional awake probability that the given node $j$ wakes up and receives the $h$-th beacon-ID signal, conditioned on that it has not woken up at earlier beacon-ID signals. In order to receive the ID signal $h$, the residual time $R_j$ until node $j$ wakes up must be in the interval $[(h-1)t_I, ht_I]$, i.e., $p_{j,h} = \Pr\{R_j \in ((h-1)t_I, ht_I] \mid R_j \in [0, (h-1)t_I)\}$. Using the cdf $F_{R_j}(y)$, we can express the awake probability as

$$p_{j,h} = \frac{F_{R_j}(ht_I) - F_{R_j}((h-1)t_I)}{1 - F_{R_j}((h-1)t_I)}. \tag{15}$$

Then, from Proposition 5, we obtain the following two important properties of wake-up processes.

***Proposition** 6:* Let $p^*_{j,h}$ be the awake probability of node $j$ when node $j$ wakes up periodically. Then, for $h = 1, \cdots, h_{j,\max}$

**(a)** $p^*_{j,h-1} < p^*_{j,h}$, and **(b)** $p^*_{j,h} \ge p_{j,h}$.

*Proof:* (a) Since node $j$ must wake up by stage $h_{j,\max}$, the awake probability is one for $h = h_{j,\max}$. Hence, Property (a) holds for this case. For $1 \le h < h_{j,\max}$, the numerator in (15) is a constant, and the denominator decreases with $h$. Hence, Property (a) still holds.

(b) By Proposition 5(a), the denominator is minimized under the periodic wake-up pattern. Further, by Proposition 5(b), the numerator is maximized under the periodic wake-up pattern. Hence, we directly obtain Property (b). ■

Property (a) implies that under the periodic wake-up pattern, the awake probability $p^*_{j,h}$ increases with respect to the number $h$ of the beacon-ID signals sent. Property (b) implies that the conditional awake probability is maximized when the neighboring node wakes up periodically.

[3] Proposition 5 is closely related to the standard results for renewal processes that periodic renewal processes have the smallest mean residual time [9, Chapter 5.2]. These standard results often require the wake-up intervals to be independent, while Proposition 5 does not require such an assumption. Since we were unable to find a result in the literature that covered the non-independent case, we have provided a full proof in [7, Section IV-A].

### B. Optimality of Periodic Wake-up Patterns

Using the properties of the periodic wake-up patterns, we show that the periodic wake-up patterns result in the smallest delay from all nodes. To show this, we first revisit the subproblem that we have solved in Section III-A and in Section III-B.

Consider two scenarios:

**(Scenario 1)** Each neighboring node $j$ wakes up periodically every $1/r_j$ time. The optimal forwarding policy $f_i^*$ that we obtained in Section III-B is applied. For this scenario, we use the same notations that are used for the optimal forwarding policy, e.g, $d_{\text{wait}}^{(h)}(x_h)$, $d^{(h)}(x_h)$, $P_{x_{h-1},x_h}^{(h)}$, $h_{j,\max}$, and $x_{h,\max}$. Recall that the packet at the sending node is forwarded no later than stage $\bar{h} = h_{1,\max}$.

**(Scenario 2)** The wake-up process of each neighboring node $j$ is arbitrary, but the wake-up rate is still given by $r_j$. We denote by $\tilde{f}_i$ the optimal forwarding policy for the given wake-up processes of the neighboring nodes. To differentiate from Scenario 1, we put a tilde ($\sim$) on all notations in this scenario, *e.g.*, $\tilde{d}_{\text{wait}}^{(h)}(x_h)$, $\tilde{d}^{(h)}(x_h)$, $\tilde{P}_{x_{h-1},x_h}^{(h)}$, etc. Similarly, node $j$ must have woken up no later than stage $\tilde{h}_{j,\max}$, and let $\tilde{x}_{h,\max}$ be the node with the smallest delay among the nodes that must be awake at stage $h$. By simply setting $\tilde{h}_{j,\max} = \infty$ and $\tilde{x}_{h,\max} = N_i + 1$, we can still use these notations for the wake-up processes under which there is no such a finite limit point. For instance, if all neighboring nodes $j$ follow the Poisson wake-up pattern, then the residual times until they wake up are independent exponential random variables, and we thus have $\tilde{h}_{j,\max} = \infty$ for $j \in \mathcal{N}_i$ and $\tilde{x}_{h,\max} = N_i + 1$ for all $h \geq 0$. Since the awake probability is maximized when nodes wake up periodically, it follows that $h_{j,\max} \leq \tilde{h}_{j,\max}$ and $x_{h,\max} \leq \tilde{x}_{h,\max}$. Further, the optimal policy $\tilde{f}_i$ must satisfy the necessary conditions (9) and (10).

We now compare the delays from both scenarios.

***Proposition** 7:* $d^{(h)}(x_h) \leq \tilde{d}^{(h)}(x_h)$ for $h = 0, 1, \cdots, \bar{h}$ and $x_h \leq x_{h,\max}$,

*Proof:* We prove this by induction. By (11), we must have $d^{(\bar{h})}(1) = \tilde{d}^{(\bar{h})}(1) = t_I + t_D + D_1$. At stage $\bar{h}$, node 1 must be awake under the periodic wake-up process (i.e., $x_{\bar{h},\max} = 1$). Hence, Proposition 7 holds for $h = \bar{h}$.

Assume that $d^{(h)}(x_h) \leq \tilde{d}^{(h)}(x_h)$ holds for $h = h'+1, h'+2, \cdots, \bar{h}$ and $x_h \leq x_{h,\max}$. We then show that this also holds for $h = h'$. From (8), we have the following inequality:

$$\tilde{d}_{\text{wait}}^{(h')}(x_{h'}) - t_I = \sum_{x_{h'+1}=1}^{x_{h'}} \tilde{P}_{x_{h'},x_{h'+1}}^{(h'+1)} \tilde{d}^{(h'+1)}(x_{h'+1})$$

$$\geq \sum_{x_{h'+1}=1}^{x_{h'}} \tilde{P}_{x_{h'},x_{h'+1}}^{(h'+1)} d^{(h'+1)}(x_{h'+1}) \quad (16)$$

$$\geq \sum_{x_{h'+1}=1}^{x_{h'}} P_{x_{h'},x_{h'+1}}^{(h'+1)} d^{(h'+1)}(x_{h'+1}) \quad (17)$$

To obtain (16), we have used the induction hypothesis. The inequality in (17) can be understood as follows: according to Proposition 6(b), neighboring nodes are more likely to wake up under the periodic wake-up patterns, and thus the delay is also minimized under the periodic wake-up pattern. (See the detailed proof of Proposition 7 in [7]).

Since (17) is equal to $d_{\text{wait}}^{(h')}(x_{h'}) - t_I$, we have $d_{\text{wait}}^{(h')}(x_{h'}) \leq \tilde{d}_{\text{wait}}^{(h')}(x_{h'})$. Then, from (9), we have $d^{(h')}(x_{h'}) \leq \tilde{d}^{(h')}(x_{h'})$. Hence, Proposition 7 holds for $h = h'$. By induction, this also holds for $h = 0, 1, \cdots, \bar{h}$. ■

From Proposition 7, we can infer that $d^{(0)}(N_i + 1) \leq \tilde{d}^{(0)}(N_i+1)$, which implies $D_i^{(k)} \leq \tilde{D}_i^{(k)}$ in the value iteration algorithm. Hence, when the delays from the neighboring nodes are given, the delay from the sending node $i$ is minimized when the neighboring nodes wake up periodically and the corresponding optimal forwarding policy is applied.

We next apply this result to the SSP problem in (2). Assume that each node $i$ can control the wake-up patterns $\vec{w}_i$ of its neighboring nodes $j$, as well as its forwarding policy $f_i$. Then, to minimize (2) with respect to $(\vec{w}, f)$, every node $i$ should carry out the following value-iteration algorithm, which is a generalized version of (4): for $k = 1, 2, \cdots$, $D_i^{(k)} = \min_{\vec{w}_i, f_i}(D_{\text{hop},i}(\vec{r}, \vec{w}_i, f_i) + \sum_{j \in \mathcal{N}_i} q_{i,j}(\vec{r}, \vec{w}_i, f_i) D_j^{(k-1)})$. In this equation, the expected one-hop delay $D_{\text{hop},i}(\vec{r}, \vec{w}_i, f_i)$ and the probability $q_{i,j}(\vec{r}, \vec{w}_i, f_i)$ that node $i$ forwards the packet to node $j$ depend only on $\vec{w}_i$ (instead of $\vec{w}$). This is because the wake-up patterns of the other nodes than the neighboring nodes do not affect the one-hop delay and the transition probability from node $i$. From Proposition 7, $D_i^{(k)}$ is maximized when $\vec{w}_i$ is given by $\vec{w}_{\text{per}}$ and the corresponding optimal forwarding policy is chosen. Hence, the following proposition holds:

***Proposition** 8:* $\min_f D_i(\vec{r}, \vec{w}_{\text{per}}, f) = \min_{\vec{w},f} D_i(\vec{r}, \vec{w}, f)$ for all nodes $i$.

Let $f^*(\vec{r})$ be the optimal forwarding policy for a given sleep-wake scheduling policy $(\vec{r}, \vec{w}_{\text{per}})$. From Proposition 4, $f^*(\vec{r})$ is equal to $f^{(N)}$ in the value-iteration algorithm. Then, Proposition 8 implies that $(\vec{w}_{\text{per}}, f^*(\vec{r}))$ is the solution to the delay-minimization problem (1).

## V. Simulation Results

In this section, we provide simulation results to evaluate the delay performance of the proposed solution. To simulate more realistic scenarios, we randomly deploy 690 nodes in a 1 km-by-1km area with obstructions as shown in Fig 2(b). We set the transmission range to 70 m and the duration $t_I$ and $t_D$ to 6 ms and 30 ms, respectively.

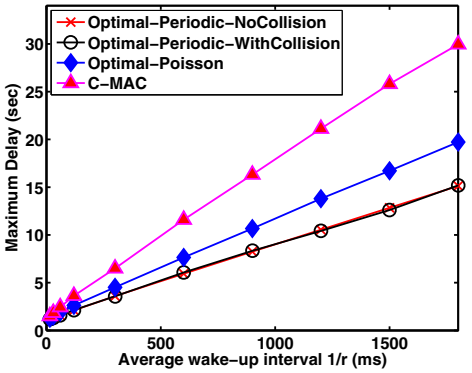
We will compare the delay performance of the following algorithms:

**Optimal-Periodic-NoCollision:** This corresponds to the optimal anycast forwarding policy with periodic wake-up patterns, and the effect of collision is ignored. We obtain the expected delay simply from the output of the value iteration algorithm in (4).

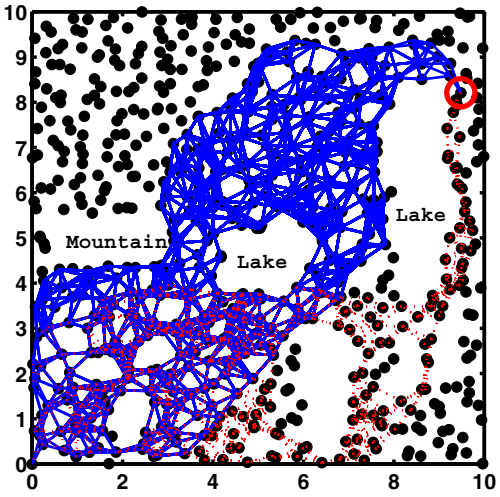
**Optimal-Periodic-WithCollision:** This corresponds to the optimal anycast policy with periodic wake-up patterns. We simulate the policy with the collision resolution component in [7, Appendix D, Deterministic Backoff]

**Optimal-Poisson:** This corresponds to the optimal anycast forwarding policy in [5] with Poisson wake-up patterns. We also simulate the policy with the same collision resolution component in Optimal-Periodic-WithCollision.

**CMAC (Convergent MAC):** This corresponds to the heuristic algorithm with Poisson wake-up pattern that was proposed in




(a) Delay Comparison




(b) Node Deployment and Possible Paths

Fig. 2. (a) Maximum delay under different wake-up rate $r$ and (b) Node deployment and the possible routing paths for 300 ms average wake-up interval under Optimal-Periodic-NoCollision (blue solid lines) and CMAC (red dotted lines). The paths under Optimal-Periodic-NoCollision pass through the network diagonally while the paths under CMAC circumvent the lake.

[2]. CMAC uses geographical information to choose the packet forwarding policy. Let $D$ and $R$ be the random variables that denote the one-hop delay and process in reducing the Euclidean distance to the sink when a packet is forwarded to the next-hop node. Then, under CMAC, each node $i$ selects the set of eligible next-hop nodes that can maximize the expected normalized-latency $E[D/R]$. Since the performance advantage of CMAC over other existing anycast-based heuristics has been extensively studied in [2], we only compare the performance of our optimal algorithm to that of CMAC.

To simulate these algorithms, we generate 50 packets at each node and take the average on the measured delay.

In Fig. 2(a), we compare the maximum expected end-to-end delay over all nodes under different wake-up rates $r$. We observe that 'Optimal-Periodic-NoCollision' and 'Optimal-Periodic-WithCollision' significantly reduce the end-to-end delay compared with the other algorithms. This is consistent with our result that the periodic wake-up pattern is delay-optimal. We also observe the significant performance gap between 'CMAC' and 'Optimal-Periodic-WithCollision.' To explain this performance gap, we show in Fig. 2(b) the possible routing paths under both algorithms. Under CMAC, packets tend to be forwarded to the nodes with higher progress. However, overall the packets may take longer paths to go around the obstructions. In contrast, under 'Optimal-Periodic-Withcollision,' the next-hop nodes are chosen by delay. Hence, it is possible for a packet to be first forwarded to nodes with negative progress, if doing so reduces the delay beyond the next-hop node. For example, in Fig. 2(b), 'Optimal-Periodic-WithCollision' results in paths that are shorter than those under 'CMAC.' From Fig. 2(b), we can infer that if there is no strong correlation between distance and delay (e.g. where there are obstructions), the heuristic anycast solutions such as CMAC can perform poorly. Finally, we can observe from Fig. 2(a) that the performance gap between 'Optimal-Periodic-NoCollision' and 'Optimal-Periodic-WithCollision' is negligible over average wake-up intervals (from 30 ms to 1800 ms). Hence, as long as collisions are resolved properly, they will not significantly impact the performance of our proposed solution at reasonable wake-up rates.

## VI. Conclusion

In this paper, we have studied the optimal anycast forwarding and sleep-wake scheduling policies that minimize the end-to-end delay. We have shown that among all wake-up patterns with the same wake-up rate, the periodic wake-up pattern maximizes the probability that a neighboring node wakes up at each beacon signal. Using this result, we have developed the optimal anycast forwarding algorithms for periodic wake-up patterns and have shown that the algorithms guarantee the minimum end-to-end delay of all nodes for given wake-up rates (which correspond to given energy budgets). Through simulation results, we have illustrated the benefits of using asynchronous periodic sleep-wake scheduling.

## References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] J. Kim, X. Lin, and N. B. Shroff, "Optimal Anycast Technique for Delay-Sensitive Energy-Constrained Asynchronous Sensor Networks," *Technical Report, http://web.ics.purdue.edu/~kim309/Kim08tech3.pdf*, 2008.

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

[9] L. Kleinrock, *Queueing Systems vol. 1: Theory*. Wiley-Interscience, 1 ed., 1975.