

# Optimal Stochastic Routing in Low Duty-cycled Wireless Sensor Networks

[Invited Paper]

Dongsook Kim  
Department of Electrical Engineering and  
Computer Science  
University of Michigan, Ann Arbor, MI  
kimds@umich.edu

Mingyan Liu  
Department of Electrical Engineering and  
Computer Science  
University of Michigan, Ann Arbor, MI  
mingyan@umich.edu

## ABSTRACT

We study a routing problem in wireless sensor networks where sensors are duty-cycled. When sensors alternate between on and off modes, delay encountered in packet delivery due to loss in connectivity can become a critical problem, and how to achieve delay-optimality is non-trivial. For instance, when sensors' sleep schedules are uncoordinated, it is not immediately clear whether a sensor with data to transmit should wait for a particular neighbor (who may be on a short route) to become available/active before transmission, or simply transmit to an available/active neighbor to avoid waiting. To obtain some insight into this problem, in this paper we formulate the above problem as an optimal stochastic routing problem, where the randomness in the system comes from random duty cycling, as well as the uncertainty in packet transmission due to channel variations. Similar framework has been used in prior work which results in optimal routing algorithms that are sample-path dependent, also referred to as opportunistic in some cases. We show such algorithms are no longer optimal when duty cycling is introduced. We first develop and analyze an optimal centralized stochastic routing algorithm for randomly duty-cycled wireless sensor network, and then simplify the algorithm when local sleep/wake states of neighbors are available. We further develop a distributed algorithm utilizing local sleep/wake states of neighbors which performs better than some existing distributed algorithms such as ExOR.

## Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Wireless communication*; C.2.2 [Computer-Communication Networks]: Network Protocols—*Routing protocols*; G.1.6 [Mathematics of Computing]: Optimization; G.3 [Mathematics of Computing]: Probability and Statistics—*Markov Processes, Stochastic processes*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WICON '08, November 17-19, 2008, Maui, Hawaii, USA  
Copyright 2008 ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

## 1. INTRODUCTION

For the past decade or so, wireless sensor networks have been extensively studied for a variety of applications, many of which require remote and autonomous operations of the sensors that are battery powered and not easily renewable. As a result, energy conservation and efficiency has long been a critical research issue. Of different energy conservation approaches, low duty-cycling, the act of periodic turning off the sensors not in active use, has been considered as one of the most effective. Its main drawback is the temporary unavailability of sensors which can adversely affect both the coverage and connectivity of the network.

In this study, we are interested in designing good routing algorithms (measured by low delay) for wireless sensor networks in the presence of very low duty cycles. In particular, we will consider a class of random sleep schedules where sensors go to sleep independent of each other and for a random duration given by a certain probability distribution. In such a scenario, when a node does not have future information on other nodes' sleep schedules but only which of its neighbors are *currently* available, its routing decision (the selection of a neighbor to relay a packet) must properly balance the immediate availability of a node against the future performance of the corresponding route.

Low duty-cycling creates significant uncertainty in the availability and connectivity of the network. Prior work in routing has primarily focused on the uncertainty caused by mobility and time-varying channel quality<sup>1</sup>. The goal of the present paper is to take both sources of uncertainty into account in designing the routing algorithm.

Generally speaking, to deal with uncertainty, one could either choose to perform routing in a deterministic way by selecting a route independent of the sleep state or the success/failure state of the network (an open-loop approach)<sup>2</sup>, or try to utilize information available to the nodes in making a closed-loop routing decision. Traditionally, most routing algorithms fall under the former category, see for instance [3, 5, 6, 9–12, 14]. More recently, there have been a number of stochastic routing (also referred to as opportunistic routing) algorithms proposed in the literature [2, 8, 13] to address the uncertainty in transmission. The key idea un-

<sup>1</sup>We will not consider mobility as a factor in this paper by assuming a quasi-static sensor network.

<sup>2</sup>Note that this does not mean that a deterministic routing cannot take into account average statistics, e.g., the average quality of a link; many routing protocols indeed do so.

derlying this class of approaches is to make routing decisions *after* having observed the outcome of an earlier transmission, i.e., after knowing which down stream nodes have or have not successfully received the transmission, thereby making a closed-loop decision. Given different realizations of these transmission events, the actual route taken by a packet can be different, thus the term *event-based routing* or *sample-path dependent routing* [8], or *opportunistic routing* [2]. It was shown in [8] that there exists an optimal Markov routing policy in the form of an index policy in a wireless network, while a conceptually very similar, sub-optimal but more practical routing algorithm called ExOR was proposed in [2]. This type of routing algorithms has a clear advantage over traditional deterministic routing in that it takes into account state information available to the nodes.

In this paper we will adopt the event-based/opportunistic routing idea and extend it to the case of low duty cycles. In particular, we will follow closely the stochastic decision framework developed in [8]. The model used in this paper is thus an extension to [8] in that it captures the randomness of topology caused by duty-cycling in addition to the randomness in channel conditions. The objective is to seek an optimal routing policy in such networks with respect to performance metrics such as transmission cost and delay. In subsequent sections we will formally define this optimization problem. Various policies are then explored and characterized for optimality. The main contributions of this paper are as follows.

1. As a benchmark we develop and analyze a centralized optimal stochastic algorithm for randomly duty-cycled wireless sensor network.
2. We develop a centralized stochastic routing algorithm with reduced state space which performs near-optimal when local sleep/wake states of neighbors are available.
3. We further develop a distributed algorithm utilizing local sleep/wake states of neighbors which performs better than some existing distributed algorithms such as ExOR, etc.

The rest of this paper is organized as follows. Section 2 provides the description of the network model. Based on the specified model, we consider the centralized stochastic routing problem with the information of duty-cycles of nodes in the network in Section 4. In Section 5, we present a centralized stochastic routing algorithm and in Section 6 we develop a distributed algorithm to compute a policy that resembles the near-optimal centralized algorithm. The performance of these algorithms is evaluated in Section 7. We conclude the paper in Section 8.

## 2. DESCRIPTION OF THE MODEL AND PROBLEM FORMULATION

We consider a static wireless ad-hoc or sensor network where nodes are randomly duty-cycled independently from one another. At a high level, our central problem is to find a good (in terms of delay or certain cost measure) route from a source node to a destination node. In a non-duty cycled static network, a typical method is to associate a measure/cost with each link in the network and perform shortest path routing. For instance, if such a cost is unit, then one ends up with a minimum hop-count route; if such

a cost indicates the expected number of transmissions over a link (by using a predefined transmission success probability), then the resulting route has the least number of expected transmissions.

In our scenario, these nodes are not always available due to duty-cycling, and not available all at the same time. Since a node can potentially obtain the information on whether each of its neighbors is available when a packet needs to be transmitted, a routing decision (i.e., the selection of the next hop relay node) must be made as to whether one should select the least-cost node among all wake nodes, or to wait for a particular node to wake up who has the least-cost among all nodes (wake and asleep), or some variations of these. In this context, it is not immediately clear what principles a good routing algorithm should employ.

To address this problem, we will start by considering a centralized system, where at each instance of time (we assume discrete time) some central agent has the full knowledge of which subset of nodes have already received the message, and which subset of nodes are currently awake. The central agent cannot foresee future sleep state of the nodes, but knows the current state. The routing decision at each time step then reduces to the question of among this set of nodes that have already received the message, which one should be selected as the relay node to retransmit the message, and whether we should simply do nothing, wait for one time step and reconsider the decision at the next time. This is the routing decision problem we seek to address in this paper. For this centralized version of the problem we will derive the structural properties of the optimal routing policy and construct an algorithm that computes such a policy. To reduce the computational complexity we will further propose a sub-optimal routing algorithm and is considerably simpler. We then consider a distributed implementation of this sub-optimal algorithm, whereby each node only has access to local information: who among its neighbors have received the message, and who among its neighbors are currently awake or asleep. A node then must decide, based on such local information whether it should serve as a relay for the message it receives. Such a distributed implementation is accomplished via packet exchange and certain local information update procedure.

### 2.1 Assumptions

- We will focus on the routing of a single message originated from somewhere in the network and has a single destination node.
- We consider a discrete time system, where in each time step (or time slot) a node is active/awake with a time-invariant probability, independent of other time slots and other nodes. For simplicity in our derivation we will assume that this *active* probability is the same for all node, though they need not be. The complement of active probability is also called the *sleep* probability.
- Any node that has successfully received the message will remain awake. This assumption is adopted for simplicity in presentation in our analysis.
- The lossy wireless medium is modeled by a pair-wise time-invariant transmission success probability  $q_{ij}$  between two nodes  $i$  and  $j$ . If this probability is nonzero,

then node  $j$  is called a “neighbor” of node  $j$ . This probability does not have to be symmetric.

- A transmission and its ACK from successful receivers occur within a single time slot.

## 2.2 Notations

A summary list of notations used in Model (M) in this paper is as follows.

$N$  is the number of nodes in the network.

$\Omega = \{1, \dots, N\}$  is the set of all nodes. So,  $|\Omega| = N$ .

$I$  is a nonexistent node which represents the idle action.

$q_{ij}$  is the transmission success probability from node  $i$  to node  $j$ , given that both nodes are awake.

$p$  is the active probability for all nodes.

$(W, A)$  refers to a state of the system, where  $W \subseteq \Omega$  and  $A \in \{0, 1\}^N$ .  $W$  is defined as the set of nodes that have received the message.  $A$  is defined as the sequence of sleep(0)/active(1) status of all nodes. In particular, node  $i$  is awake if it has received a message as stated in assumptions: Given  $A = \{a_1, a_2, \dots, a_N\}$ ,  $a_i = 1$  for all  $i \in W$ .

$F(W)$  denotes a feasible set of all possible sleep/active states  $A$  induced by  $W$  so that  $A$  is consistent with  $W$ . More specifically, given  $W$ , there are a total of  $2^{N-|W|}$  sets of  $A$ 's in  $F(W)$  where  $a_i = 1$  for all  $i \in W$  and  $a_i \in \{0, 1\}$  for all  $i \in \Omega - W$ .

$F(W|W', A')$  for  $W \subset W', A' \in F(W')$  denotes a subset of sleep/active states  $A \in F(W)$ , such that that  $A$  is identical to  $A'$  except that  $a_i \in \{0, 1\}$  for all  $i \in W' - W$ .

$F(W|W', A')$  for  $W \supset W', A' \in F(W')$  denotes a subset of sleep/active states  $A \in F(W)$ , such that that  $A$  is identical to  $A'$  except that  $a_i = 1$  for all  $i \in W - W'$ . We see that there is only one such  $A$  in this set.

$T : 2^\Omega \rightarrow 2^N$  is defined as a mapping from  $W$  to a vector  $T(W) = \{w_1, w_2, \dots, w_N\}$ ,  $W \subseteq \Omega$  where each element  $w_i = 1$  if node  $i$  has received the message, and 0 otherwise.

$P^i(W', A'|W, A)$  indicates the probability of state  $(W', A')$  reached from state  $(W, A)$  by choosing  $i$  for transmission,  $i \in W$ . Let  $T(W) = \{w_1, w_2, \dots, w_N\}$  and  $A = \{a_1, a_2, \dots, a_N\} \in F(W)$ . Also,  $T(W') = \{w'_1, w'_2, \dots, w'_N\}$  and  $A' = \{a'_1, a'_2, \dots, a'_N\} \in F(W')$ . If a node  $i$  is chosen for transmission, the transition probability is given by

$$P^i(W', A'|W, A) = \left( \prod_{\forall j: w_j=0, a_j=1, w'_j=1} q_{ij} \right) \cdot \left( \prod_{\forall j: w_j=0, a_j=1, w'_j=0} 1 - q_{ij} \right) \cdot \left( \prod_{\forall j: a_j=0, w'_j=1} 0 \right) \cdot p^{I_{\overline{W'}} - I_{\overline{W}}} (1 - p)^{N - I_{\overline{W}}}, \text{ for } \forall i \in W,$$

where  $I_{\overline{W'}}$  is the number of 1's in  $T(W')$ , and  $I_{\overline{W}}$  is the number of 1's in  $A'$ . If the idle node  $I$  is chosen,

$$P^I(W', A'|W, A) = \begin{cases} p^{I_{\overline{W'}} - I_{\overline{W}}} (1 - p)^{N - I_{\overline{W}}}, & \text{if } W' = W \\ 0, & \text{otherwise.} \end{cases}$$

$R : 2^\Omega \rightarrow \mathbb{R}$  is the reward functions. Specially, we denote  $R_i = R(\{i\})$ .

$\pi$  is a Markov policy such that  $\pi$  depends only on the current state  $(W, A)$ . We write  $\pi(W, A) = i$  to indicate that policy  $\pi$  transmits at node  $i$  when in state  $(W, A)$ ,  $i \in W$ . We write  $\pi(W, A) = I$  to indicate policy  $\pi$  choose the

idle/wait action. We write  $\pi(W, A) = r$  to indicate policy  $\pi$  retires and receives reward  $R(W) = r$  when in state  $(W, A)$ .

$V^\pi(W, A)$  is the expected reward when starting in state  $(W, A)$  under policy  $\pi$ .

## 2.3 Problem Formulation

*Problem 1.* We consider the centralized routing problem where a central controller, who knows the current set of nodes with the message and the set that are awake, chooses among three actions at each time instant: (1) select a node among nodes that have the message for the next transmission; (2) wait for the next time step; and (3) terminate the routing process. The transmission from a node  $i$ ,  $i \in \Omega$ , costs  $c_i > 0$  and is the local broadcast to its active neighbors. The idle action is denoted by  $I$  and  $c_i = \alpha \geq 0$ , indicating a penalty on idle waiting. This transmission is successfully received by a neighbor  $j$  with a time-invariant probability  $p_{ij}$  given node  $j$  is active during that time slot. Each transmission event is assumed to be independent of another. The objective is to choose the best action at each time step so as to maximize the total expected reward less cost:

$$E\{R(S_f) - \sum_{t=1}^{\tau-1} c_{i(t)}\},$$

where  $\tau$  is the stopping time when the transmission process is terminated,  $S_f$  is the state at  $\tau$ , and  $i(t)$  is the node (including idle action) chosen by the policy at time  $t$ .

## 3. PRELIMINARIES

When nodes are always awake (i.e.,  $p = 1$ ), which is a special case of Problem 1, the authors of [8] showed that an optimal Markov policy for this problem is both a priority policy and an index policy. The first few definitions below are reproduced from [8] for this paper to be self-contained. We then present an example to illustrate these concepts are not able to capture the extra dynamics caused by node sleeping. This motivates us to generalize the notion of priority policies and index policies.

*Definition 1.* [8] A Markov policy  $\pi$  is a priority policy if there is a strict priority ordering of the nodes s.t.  $\forall i \in \Omega$  we have  $\pi(S \cup \{i\}) = \pi(\{i\}) = i$  or  $r_i$ ,  $\forall S \subseteq \Omega_i$ , where  $\Omega_i$  is the set of nodes of priority lower than  $i$ .

*Definition 2.* [8] A function  $f : 2^\Omega \rightarrow \mathbb{R}$  is an index function on  $\Omega$  if  $f$  satisfies

$$f(S) = \max_{i \in S} f(\{i\}), \quad \forall S \subseteq \Omega.$$

*Definition 3.* [8] A priority policy  $\pi$  is called an index policy if  $V^\pi(\cdot)$  is an index function on  $\Omega$ .

In the following, we use a simple example to show that an optimal policy may not be found in the class of priority policies for Problem 1.

*Example 1.* We consider a system depicted in Figure 1, where  $\Omega = \{1, 2, 3, 4, 5\}$  and  $p = 0.1$ . Assume that  $R_i = 0$  except node 5 which has a reward  $R_5 > 0$ . For simplicity we also assume that  $c_i = 1$  for  $i \in \Omega \cup \{I\}$ . An active node  $i$  is denoted by  $ia$  and a sleeping node  $i$  by  $is$ . Since

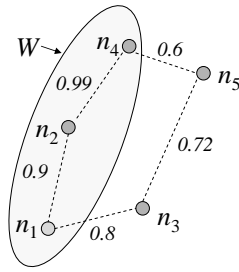


Figure 1: System for an Example 1.

nodes in  $W$  are assumed to be awake, we are only concerned with those in  $\Omega - W$ , i.e.,  $A \in F(W)$ . Let  $W = \{1, 2, 4\}$  as shown in the Figure 1. Let  $\pi^*$  to be an optimal Markov policy. We have  $\pi^*(W, \{3a, 5a\}) = 4$ ,  $\pi^*(W, \{3a, 5s\}) = 1$ ,  $\pi^*(W, \{3s, 5a\}) = 4$ , and  $\pi^*(W, \{3s, 5s\}) = I$  using stochastic dynamic programming; the details of the calculation can be found in [7]. Let us focus on  $A = \{3a, 5s\}$ . In this case, node 1 is the highest priority node among nodes 1, 2, and 4. Now, suppose  $W = \{1, 2\}$ . For the sleep/wake states in  $F(W|\{1, 2, 4\}, \{3a, 5s\})$ , we obtain  $\pi^*(W, \{3a, 4a, 5s\}) = 2$  and  $\pi^*(W, \{3a, 4s, 5s\}) = 1$  by a similar calculation. When node 4 is asleep, node 1 is the highest priority node as expected. On the other hand, when node 4 is active, node 2 is the highest priority node among node 1 and node 2. In other words, node 1 is not always the highest priority node among nodes 1, 2, and 4; instead, nodes' priorities may change with the sleep state.

As can be seen from the above example, removing a node like 4 from the set  $W = \{1, 2, 4\}$  has a significant impact on the resulting optimal policy, even though it is not the highest priority node given  $A = \{3a, 5s\}$ . This is because node 4 is the highest priority node in  $W$  given other sleep/wake states such as  $\{3a, 5a\}$  and  $\{3s, 5a\}$ . To summarize, given  $W$ , if a node  $i$  is the highest priority node in  $W$  for some feasible sleep/wake state, then the priority ordering in  $W - \{i\}$  are not always preserved under other sleep/wake states. Thus if we remove node  $i$  we need to recalculate the priority ordering of nodes in  $W - \{i\}$ . By contrast, in the case when  $p = 1$ , this priority ordering is preserved no matter which node we remove from the set  $W$ . This is the primary difference between Problem 1 and that considered in [8] both from a conceptual and a computational point of view.

Motivated by the above example, it is necessary to generalize the preceding definitions in the context of our problem.

**Definition 4.** Consider a Markov policy  $\pi$  such that  $\pi(W, A_i) = n_i \in W \cup \{I\}, \forall i \in \{1, \dots, m\}$  for  $W \subseteq \Omega$  and  $\forall A_i \in F(W)$  where  $m = 2^{N-|W|}$ . This policy is called a Generalized(G)-priority policy if the following condition holds: Define  $N_W = \bigcup_{i=1}^m n_i - \{I\}$  and for  $\forall S \subseteq W - N_W$ , we have

$$\pi(W, A_i) = \pi(S \cup N_W, A) = n_i, \\ \forall A \in F(S \cup N_W|W, A_i), \forall i \in \{1, \dots, m\}.$$

Here the condition on  $A$  is simply to ensure that the sleep state  $A$  is consistent with state  $A_i$  (it is identical to  $A_i$  except for nodes in  $W - S - N_W$  what are unspecified). What this definition says is that a policy is a G-priority policy if there exists a set  $N_W$  of priority nodes within  $W$  whose priorities are strictly higher than the rest regardless of the sleep state,

but whose priority ordering among themselves can only be determined for a specific sleep state. This set consists of nodes that would have been selected in at least one sleep state.

**Definition 5.** A function  $f : 2^\Omega \times 2^N \rightarrow \mathbb{R}$  is an Generalized(G)-index function on  $2^\Omega$  if  $f$  satisfies

$$f(W, A) = \max_{\tilde{W} \subseteq W, \tilde{A} \in F(\tilde{W}|W, A)} f(\tilde{W}, \tilde{A}), \forall W \subseteq \Omega, \forall A \in F(W).$$

**Definition 6.** A priority policy  $\pi$  is called an Generalized(G)-index policy if  $V^\pi(\cdot)$  is an G-index function on  $\Omega$ .

Note there are two special case interpretations of Problem 1 depending on what we use as costs. If the idle cost is zero, then there is no loss of optimality to always wait till all nodes are awake (a positive probability event) to make a decision. Given that we only consider the problem in this particular sleep state (all awake), the problem reduces to the one studied in [8]. If all costs are the same, the problem can be regarded as finding a policy which minimizes delay.

## 4. ANALYSIS OF PROBLEM 1

In this section, we analyze Problem 1 and derive structural properties of an optimal policy  $\pi^*$ . Specifically, we show that there exists an optimal G-index policy for Problem 1 in Theorem 1. The idea behind its proof is to show that an optimal Markov policy with certain properties is a G-priority policy, which is in turn a G-index policy by proving that the expected reward function is a G-index function. This method follows closely the framework developed in [8] although there are technical differences due to the introduction of sleep states. Throughout the paper most proofs are omitted due to space limit; they can be found in [7].

Lemma 1 below is essentially the same as given in [8], but adapted to our notation. It shows the properties of an optimal Markov policy that if all supersets that can be reached from a state have optimal expected reward values and the actions at the state for all sleep states are optimal, then the expected reward value at the state is optimal.

**Lemma 1** Let  $\pi^*$  be an optimal Markov policy for Problem 1. Suppose we are given  $W_1$  and  $A_1 \in F(W_1)$ , and let  $\pi$  be a Markov policy with the following properties:

$$V^\pi(W, A) = V^{\pi^*}(W, A), \quad \forall W \supset W_1, \forall A \in F(W), \\ \pi(W_1, A_1) = \pi^*(W_1, A_1), \quad \forall A_1 \in F(W_1).$$

Then

$$V^\pi(W_1, A_1) = V^{\pi^*}(W_1, A_1).$$

The next lemma shows the monotonicity of an optimal Markov policy.

**Lemma 2** In Problem 1, let  $\pi^*$  be an optimal Markov policy. Let  $W_1, W_2 \subseteq \Omega$  and  $W_2 \subseteq W_1$ . Then, for  $A_1 \in F(W_1)$ ,  $V^{\pi^*}(W_2, A_2) \leq V^{\pi^*}(W_1, A_1)$  where  $A_2 \in F(W_2|W_1, A_1)$ .

The next lemma shows the G-priority structure of an optimal Markov policy.

**Lemma 3** Let  $\pi^*$  be an optimal Markov policy for Problem 1. Then, there exists a Markov policy  $\pi$  which has the following properties.

1. For all  $W \subseteq \Omega$  where  $|W| \geq 2$  and all possible  $A_i \in F(W) = \{A_1, \dots, A_m\}$ ,  $m = 2^{N-|W|}$ ,

$$\begin{aligned} \pi(W, A_i) = n_i \in W \cup \{I\} &\Rightarrow \pi(W - \{j\}, A) = n_i, \\ \forall j \in W - \cup_{i=1}^m n_i, \forall A \in F(W - \{j\} | W, A_i), \\ \pi(W, A_i) = r_{n_i}, n_i \neq I &\Rightarrow \pi(W - \{j\}, A) = r_{n_i}, \\ \forall j \in W - \cup_{i=1}^m n_i, \forall A \in F(W - \{j\} | W, A_i). \end{aligned}$$

2. For all  $W \subseteq \Omega$  where  $|W| \geq 2$  and all possible  $A_i \in F(W)$ , and  $\pi(W, A_i) = n_i \in W \cup \{I\}$  or  $r_{n_i}, n_i \neq I$  for  $i \in \{1, \dots, m\}$ ,

$$\begin{aligned} V^\pi(W - \{j\}, A) &= V^\pi(W, A_i) \\ &= V^{\pi^*}(W, A_i) = V^{\pi^*}(W - \{j\}, A), \\ \forall j \in W - \cup_{i=1}^m n_i, \forall A \in F(W - \{j\} | W, A_i). \end{aligned}$$

3.  $\pi$  is an optimal Markov policy.

**Lemma 4** For any optimal Markov policy  $\pi^*$ ,  $V^{\pi^*}(\cdot)$  is a G-index function on  $\Omega \cup \{I\}$ .

**Theorem 1** There is an optimal Markov policy  $\pi^*$  for Problem 1 which is a G-index policy.

PROOF. By Lemma 3, there exists a Markov policy  $\pi^*$  which is an optimal Markov policy.  $V^{\pi^*}(\cdot)$  is a G-index function by Lemma 4. This says that the optimal decision on the resulting set after removing some nodes that are not in  $\cup_i n_i$  from  $W$  remains the same. Thus the conditions in Definition 4 are satisfied. Thus  $\pi^*$  is a G-priority policy. Since  $\pi^*$  is a G-priority policy and its  $V^{\pi^*}(\cdot)$  is a G-index function,  $\pi^*$  is a G-index policy according to Definition 6.  $\square$

## 5. OPTIMAL AND SUB-OPTIMAL ROUTING ALGORITHMS

### 5.1 An Optimal Centralized Algorithm for Problem 1

We present an algorithm to compute the optimal G-index policy for Problem 1. Compared to the brute-forth dynamic programming, our algorithm utilizes the properties of G-index policy stated in Lemma 3 to reduce the amount of computation. Let node  $d$  be the destination. The procedure starts with  $W = \Omega$  and  $A = \{1, \dots, 1\}$ . Its optimal action and reward value are straight-forward, which are

$$V(\Omega, A) = R_d \text{ and } \pi(\Omega, A) = r_d.$$

From the properties 1 and 2 in Lemma 3, we know

$$V(\Omega - \{j\}, A) = R_d \text{ and } \pi(\Omega - \{j\}, A) = d,$$

for  $\forall A \in F(\Omega - \{j\})$  if  $j \neq d$ . Thus, we only need to calculate  $V(\Omega - \{d\}, A)$  for  $\forall A \in F(\Omega - \{d\})$ .

By solving the associated set of linear equations, we obtain  $\pi(\Omega - \{d\}, A)$  for  $\forall A \in F(\Omega - \{d\})$ . Suppose  $\pi(\Omega - \{d\}, A_i) = n_i$  for each  $i$  s.t.  $A_i \in F(\Omega - \{d\})$ . We will denote by  $D(\Omega - \{d\}) = \cup_i \{n_i\}$  the set of highest priority nodes in  $W$ . Again, by the properties of Lemma 3, we have

$$\begin{aligned} \pi(S \cup D(\Omega - \{d\}), A) &= n_i, \\ \forall S \subset \Omega - \{d\}, A \in F(S \cup D(\Omega - \{d\}) | \Omega - \{d\}, A_i). \end{aligned}$$

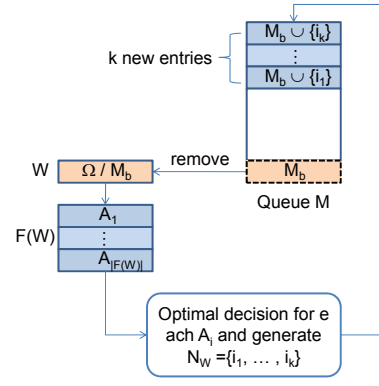


Figure 2: The diagram of Algorithm 1.

Therefore, the reward functions that need to be calculated are  $V(\Omega - \{d\} - \{n_i\}, A)$ , for  $\forall A \in F(\Omega - \{d\} - \{n_i\})$ . The subsequent steps are done similarly.

We now formally describe the above procedure in Algorithm 1. Figure 2 illustrates how Algorithm 1 works. While it is presented for a single destination, we note that it can be easily extended to the case of multiple destinations.

*Algorithm 1.* Define sets  $W$ ,  $F(W)$ ,  $N_W$  and a queue  $M$ , as follows.

Each entry in queue  $M$  contains the set of nodes  $S \in \Omega$  which have not received the packet. Specially, denote by  $M_b$  the head of line of  $M$ .  $W$  is the complement of  $M_b$  with respect to  $\Omega$ , which is  $W = \Omega - M_b$  meaning the set of nodes which have received packet.  $F(W)$  is the set of all feasible active(1)/sleep(0) states of the nodes in  $M_b$  and all ones for the nodes in  $W$ . That is,  $F(W) = \{A_1, A_2, \dots, A_k\}$  where  $k = 2^{|M_b|}$ .  $N_W$  is the set of highest priority nodes in  $W$  for every  $A_i \in F(W)$ .

Since the case where  $W = \Omega$  is trivial, we start with  $W = \Omega - \{d\}$ . Initially, the queue  $M = \{M_b\} = \{\{d\}\}$  contains a destination  $d$ ; the action taken by an optimal G-index policy  $\pi$  on the destination  $d$  is to retire and receive  $R_d$  regardless of sleep states.  $F(W)$  contains two sets which include ones for all nodes except for  $d$  which is zero in one set and one in the other.  $N_W$  is initially empty.

The algorithm proceeds as follows.

1. For each  $i \in W$  and each  $A_j \in F(W)$ , let  $\pi_i^j$  be an G-index policy with the same priority list as  $\pi$  for the nodes in  $M_b$ , with  $i$  as the next highest priority node after  $M_b$ , and with the priority of the nodes,  $W - \{i\}$  arbitrary, but lower than  $i$ . Compute  $V_i^{\pi_i^j}(W, A_j)$  for all  $j$ ,  $1 \leq j \leq k$  from

$$\begin{aligned} V_i^{\pi_i^j}(W, A_j) &= \max\{-c_i + \\ &\sum_{W' \supseteq W} \sum_{A' \in F(W')} P^i(W', A' | W, A_j) V_{\pi_i^j(W', A')}^{\pi_i^j}(W', A'), R_i\}. \end{aligned}$$

2. For the idle action  $I$ , let  $\pi_I$  be an index policy which is similarly defined as in step 1 except no actual trans-

mission takes place. Thus,

$$V_i^{\pi^j}(W, A_j) = \max\{-\alpha + \sum_{A' \in F(W')} P^I(W, A'|W, A_j) V_{\pi_j^I(W, A')}^{\pi^j}(W, A'), R_i\}.$$

3. For each set of choices of a node  $i_j \in W \cup \{I\}$  for  $A_j$ ,  $1 \leq j \leq k$ , denoted by  $\mathbf{i} = \{i_1, i_2, \dots, i_k\}$ ,  $V_{i_j}^{\pi^j}(W, A_j)$  are solved by  $k$  linear equations. Choose  $\mathbf{i}$  with the highest values of  $V_{i_j}^{\pi^j}(W, A_j)$ 's. Ties are broken with more  $I$ s in  $\mathbf{i}$ , otherwise arbitrarily.
4.  $N_W$  includes all distinct  $y \in \mathbf{i}$ , which is not equal to  $I$ . For each node in  $N_W$ , append it to the set  $M_b$  and place the resulting set on top of the queue  $M$ .
5. Finally, remove  $M_b$  from the bottom of the queue  $M$ . If  $M$  is empty, stop. Otherwise, go to step 1).

**Theorem 2** For Problem 1, Algorithm 1 produces an optimal  $G$ -index policy.

## 5.2 A Sub-Optimal Algorithm

We next propose a sub-optimal algorithm that reduces the complexity of Algorithm 1. We consider the same problem and again we will assume a centralized view, with the only difference that we will now assume that the central controller only knows  $W$  but not  $A$ . Accordingly the system has a simple state given by  $W$ . Note that with this modification the idle action  $I$  is never chosen. Below we redefine a few notations below for this simpler system.

$P^i(W'|W, A)$  denotes the probability of state  $W'$  reached from state  $W$  by choosing  $i \in W$  for transmission, when nodes' sleep/wake status is  $A$  at the moment.

If a node  $i$  is chosen for transmission, the transition probability is defined as

$$P^i(W'|W, A) = \left( \prod_{\forall j: w_j=0, a_j=1, w'_j=1} q_{ij} \right) \cdot \left( \prod_{\forall j: w_j=0, a_j=1, w'_j=0} 1 - q_{ij} \right) \cdot \left( \prod_{\forall j: a_j=0, w'_j=1} 0 \right), \text{ for } \forall i \in W.$$

$\pi$  is a Markov policy such that  $\pi$  depends only on the current state  $W$ .

$V^\pi(W)$  is the expected reward when starting in state  $W$  under policy  $\pi$ .

Given the modified model described above (i.e., without nodes' active/sleep information), Since the nodes' sleep information is not available to the decision maker, this problem is essentially the same as the one studied in [8] with a modification to the state transition probability. This is because the decision maker cannot differentiate transmission failures caused by channel errors from the ones by duty-cycling. Hence, sleep/wake activity of nodes is reflected in transition probability measured on average, i.e.,  $P^i(W'|W) = \sum_{A \in F(W)} P^i(W'|W, A)P(A)$ .

With these transition probabilities, one can use the algorithm developed in [8] (referred to as Lott's Algorithm below) to generate an optimal index policy for this modified problem. Specifically, the expected reward given  $W$  when  $i$  is transmitting is calculated by

$$V_i^{\pi^i}(W) = \max\{-c_i + \sum_{W' \supseteq W} \left( \sum_{A \in F(W)} P^i(W'|W, A)P(A) \right) V_{\pi_i^i(W')}^{\pi^i}(W'), R_i\}.$$

Below we present an algorithm that both utilizes and outperforms Lott's Algorithm for Problem 1. Specifically, the decision maker maintains the simple state  $W$  to calculate the expected reward, but it makes the routing decision by taking into account the current sleep/wake state  $A$  available. This significantly simplifies the computation compared to Algorithm 1.

*Algorithm 2.* The sets  $W$ ,  $F(W) = \{A_1, A_2, \dots, A_k\}$ ,  $N_W$ ,  $M_b$  and a queue  $M$  are defined the same as in Algorithm 1.

The algorithm consists of two parts: an off-line part and an on-line part. The off-line part obtains the expected reward values  $\tilde{V}(W)$  for all  $W \subseteq \Omega$  by Lott's Algorithm. The on-line part of the algorithm proceeds as follows.

1. For each  $i \in W$ , let  $\pi$  be a policy with the same priority list as the policy generated by Lott's Algorithm for the nodes of  $M_b$  with  $i$  as the next highest priority node after  $M_b$ ,  $W - \{i\}$  arbitrary, but lower than  $i$ . Compute  $V_i^\pi(W, A_j)$  for all  $j$ ,  $1 \leq j \leq k$  from

$$V_i^\pi(W, A_j) = \max\{-c_i + \sum_{W' \supseteq W} P^i(W'|W, A_j)\tilde{V}(W'), R_i\}.$$

2. When selecting the idle action its value is computed as:

$$V_I^\pi(W, A_j) = \max\{-\alpha + P^I(W|W, A_j)\tilde{V}(W'), R_I\}.$$

3. For  $A_j$ , choose a node  $i_j \in W \cup \{I\}$  with highest values of  $V_{i_j}^\pi(W, A_j)$ ,  $1 \leq j \leq k$ , denoted by  $\mathbf{i} = \{i_1, i_2, \dots, i_k\}$ . Ties are broken arbitrarily.
4. For each distinct  $y \in \mathbf{i}$ , which is not equal to  $I$ , append  $\{y\} \cup M_b$  at the top of  $M$ . Remove  $M_b$  from the bottom of  $M$ .
5. If  $M$  is empty, stop. If not, go to step 1.

**Corollary 1** Algorithm 2 performs at least as good as Lott's Algorithm for Problem 1.

## 6. DISTRIBUTED IMPLEMENTATION

In this section, we present a practical routing protocol that implements Algorithm 2 in a distributed way. We will adopt opportunistic-like forwarding used in [2] in our algorithm, where nodes are not assumed to have perfect information on  $W$  and  $A$ . Specifically, nodes periodically exchange a HELLO (also referred to as a beacon below) packet when they are awake. From these exchanges nodes infer about their neighbors' sleep status when making a decision on whether they should forward a received packet.

Our stochastic routing protocol, referred to as SRP below, consists of two elements: priority update and forwarder selection. Nodes are initialized with the off-line computed priorities using Lott's algorithm (also called off-line priorities). As they obtain their neighbors' sleep state they can choose to recalculate and update these priorities. In the forwarder selection step a node decides for itself whether it should become a forwarder and retransmit the packet it received based on current priorities. Below we present these two elements in more detail.

## 6.1 Priority Update Procedure

We will assume that the off-line computation of  $\{\tilde{V}_i^\pi\}$  by Lott's Algorithm is completed, such that each node has its own  $\tilde{V}_i^\pi$  as well as  $\tilde{V}_j^\pi$  for all nodes  $j$  in its neighbor set  $N_i$ . This can be accomplished using the Dijkstra-like distributed algorithm proposed in [8], in which case this computation is off-line only in the sense that this computation is done prior to the execution of SRP. We also assume that an on duration is larger than a beacon interval  $T$ .

1. When node  $i$  goes to sleep, it turns off the radio and does nothing.
2. Upon waking up, node  $i$  sets the beacon counter  $n$  to zero, the beacon transmission time  $t_0^i$  to current time, and immediately transmits a beacon packet containing value  $V^0(i)$  which is set to  $\tilde{V}_i^\pi$ .  $V_i^0(j)$  is initialized to  $\tilde{V}_j^\pi$  for all  $j \in N_i$ ; the set  $A_i$  that contains all active neighbors is initialized to be an empty set. The set  $C_i$  of forwarder candidates contains the set of neighbors  $j$ 's who have  $\tilde{V}_j^\pi > \tilde{V}_i^\pi$ .
3. Node  $i$  then increments  $n$  by one, and set the next beacon transmission time  $t_n^i$  to  $t_{n-1}^i + T$ , where  $T$  is the (constant) beacon interval.
4. Between  $t_{n-1}^i$  and  $t_n^i$ , if node  $i$  receives a beacon packet from some neighbor  $j$ , it updates  $V_i^{n-1}(j)$  with the new value contained in the packet and records its update time. Also, node  $j$  is added to  $A_i$  if it is not already in the set.
5. Right before the  $n$ -th beacon transmission, at time  $t_n^i$ , node  $i$  recalculates the priorities as follows. If a beacon packet from node  $j$  was last received at a time earlier than  $t_n^i - \beta T$ , where  $\beta$  a constant multiplier and  $\beta T$  sets a threshold on how long a neighbor has not been heard from before assuming it's asleep, then node  $j$  is assumed to be in sleep mode and is removed from  $A_i$ . For those nodes in  $A_i$ , set  $V_i^n(j) = V_i^{n-1}(j)$ . Otherwise, set  $V_i^n(j) = \tilde{V}_j^\pi$  for a sleep node  $j$ . Include in  $C_i$  all neighbors that qualify as a possible forwarder and their current priorities. Denote by  $q_{ij|C_i, A_i}^*$  the probability that node  $j$  receives successfully while nodes with higher priorities in  $A_i \cap C_i$  fail. Denote nodes with higher priorities than node  $j$  by  $\{A_i \cap C_i\}_j^+ \subset A_i \cap C_i$ . Then,

$$q_{ij|C_i, A_i}^* = q_{ij} \prod_{k \in \{A_i \cap C_i\}_j^+} (1 - q_{ik}).$$

Using this probability, node  $i$  updates  $V^n(i)$  as follows.

$$V^n(i) = \frac{-c_i + \sum_{j \in A_i \cap C_i} q_{ij|C_i, A_i}^* V_i^n(j)}{1 - \sum_{j \in A_i \cap C_i} (1 - q_{ij})}.$$

Node  $i$  then transmits a beacon packet with  $V^n(i)$  to its neighbors.

6. While node  $i$  continues to be awake, repeat steps 3-5.

## 6.2 Forwarder Selection Procedure

The forwarder selection procedure is provided in the following. This algorithm is performed whenever node  $i$  generates a message or receives it from one of its neighbors.

1. Recall that  $V(i)$  and  $\{V_i(j)\}_{j \in N_i}$  are set to current priority values calculated by the priority update procedure. The current active neighbors of node  $i$ ,  $A_i$ , is also given in priority update.
2. When node  $i$  receives a message, it obtains the list of candidate forwarders. If it is on the list, go to step 3. Otherwise, it does not forward the message and returns to the receiving mode.
3. If node  $i$  is listed as a potential forwarder, it calculates a time period  $D$  based on its priority on the list. If it is the  $k$ -th highest priority node on the list with a total of  $M$  nodes on the list, it randomly selects  $D$  as proportional to  $k - 1$ . Or an ACK is repeated like the multiple duplicated ACKs as robust acknowledgement introduced by [13].
4. If node  $i$  receives ACKs from higher priority nodes, it transmits an ACK with the identity (ID) of the highest priority node, and it does not forward the message. During the period  $D$ , if node  $i$  does not receive an ACK from any of the higher priority nodes, node  $i$  decides to forward and transmits an ACK with its own ID. The message contains the priority list of the next forwarders according to  $V(i)$ ,  $\{V_i(j)\}_{j \in N_i}$ ,  $A_i$ .
5. If node  $i$  decides not to forward under the policy  $\pi$  and receives no ACK during  $M \cdot T_s$  period, it goes to step 3, unless it was already repeated for  $R$  times. If so, the message is removed.
6. If node  $i$  has transmitted the message, it waits ACKs from neighbors for at most  $R \cdot T_s$ . If it receives no ACK, it retransmits the message.

## 7. PERFORMANCE EVALUATION

We have performed extensive MATLAB simulation to evaluate the performance of the proposed algorithms. The simulated system closely follows the set of assumptions listed earlier in this paper. The lossy channel model we adopted in the simulation is based on pair-wise distance. Specifically, we assume that the success probability that a node receives a message from any node is given by a decreasing linear function of the distance between the nodes, as shown in Figure 3. This distribution is based on the measurements on Rene Motes using medium transmission power reported by Ganesan et al in [4]. A node with reception probability above a threshold  $p_m$  is regarded as a neighbor. Each sensor node is duty-cycled with a sleep probability  $p_s$ , and the discrete time unit is chosen large enough for a transmission and ACKs to occur. A source and a destination are randomly selected among nodes in the network. We assume that the network is connected when all nodes are awake, thus in time any destination may be reached from any source.

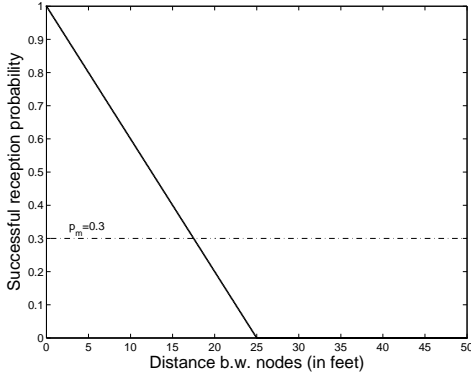


Figure 3: Delivery success probability w.r.t. distance.

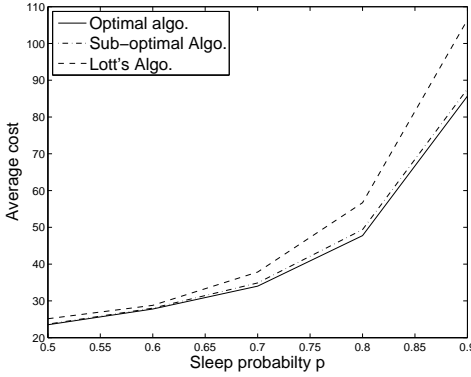


Figure 4: Performance comparison of the centralized algorithms on Topology 1 (Scenario 2).

Throughout this section, we consider three different scenarios depending on how the transmission cost and idle penalty are determined.

1. *Unit cost for both transmission and idle action:* As pointed out earlier in this scenario the problem reduces to finding a delay-optimal path from a source to a destination.
2. *Random cost for transmission and nonzero cost for idle action:* With this cost scenario the problem finds a path that minimizes the total cost. The intention of using a random transmission cost is so that this cost may represent the fact that some transmissions are more costly if the transmitting node has relatively low residual energy, or if all its neighbors are located far away thereby physically requiring more energy.
3. *Random cost for transmission and zero cost for idle action.*

## 7.1 The effect of sleep information on optimality

In the previous sections, it was shown that Algorithm 1, referred to as the Optimal Algorithm in the remainder of this section, generates an optimal G-index policy for Problem 1. Unfortunately, its computational complexity is very high and thus is not really usable even for a small network.

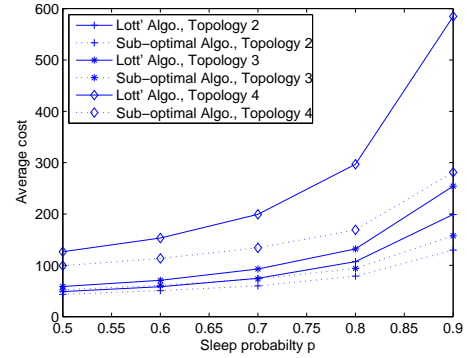


Figure 5: The effect of average degree of nodes on the performance of Sub-optimal and Lott's Algorithms (scenario 3).

We did manage for sizes up to  $N = 6$ . The network topology under consideration is a small network of 6 sensor nodes with average node degree 4.6 and  $p_m = 0.3$  referred to as Topology 1. Using this topology, we first examine how much performance degradation is incurred if we ignore sleep information. In Figure 4 we compare Algorithm 1, Lott's Algorithm, and Algorithm 2 (also referred to as the sub-optimal algorithm in the remainder of this section).

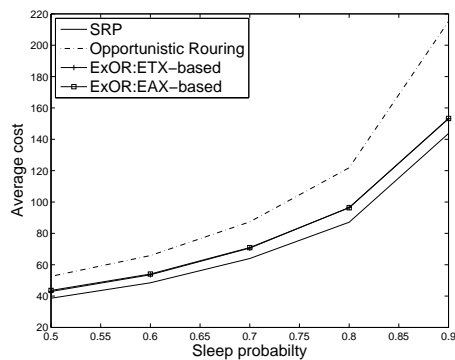
Figure 4 depicts the average costs of paths taken by these algorithms under the second cost scenario. Here the costs are uniformly generated over  $[1, 7]$  while idle cost is fixed at 4. It is clearly shown that the Sub-optimal Algorithm performs very close to the optimal. This indicates that the Sub-optimal Algorithm which is much simpler and requires only local sleep/wake information than the optimal algorithm works sufficiently well in such a small network.

## 7.2 The effect of node degree

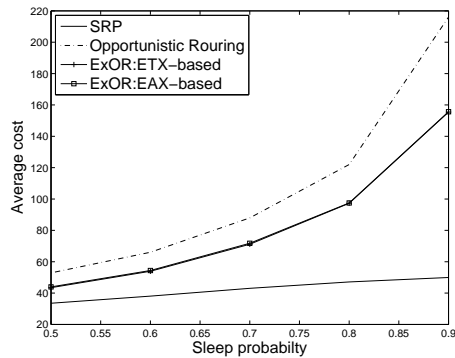
We next examine the performance comparison between Lott's Algorithm and our Sub-optimal Algorithm when increasing the average node degree. Recall that the former is oblivious to the sleep state. We consider three network topologies where  $N = 30$  sensor nodes are deployed with different  $p_m = \{0, 0.3, 0.5\}$ .  $p_m$  determines the set of neighbors and so does node degree. The first topology (Topology 2) has 12.33 average node degree when  $p_m = 0$ . The second, Topology 3, has less average node degree, 7.13, by increasing  $p_m$  into 0.3. The last one, Topology 4, has 4.13 average node degree as  $p_m$  becomes 0.5, which is the least connected.

Using the third cost scenario, Figure 5 shows Sub-optimal Algorithm improves significantly compared to Lott's Algorithm, and the improvement increases as  $p$  increases. That is, the Sub-optimal Algorithm is more effective when duty-cycling is heavy. Simulation results (not shown here) also show that the delay performance of Sub-optimal Algorithm is slightly better than that of Lott's Algorithm. Since Lott's Algorithm ignores the sleep information, it does not take idle actions. This implies that it generally uses more transmissions to reach the destination whereas Sub-optimal Algorithm waits for a better wake set of neighbors, resulting in fewer number of transmissions.

## 7.3 Performance of the distributed protocol SRP



(a) Average cost. (scenario 2)



(b) Average cost. (scenario 3)

**Figure 6: Comparison between the decentralized algorithms and ExORs.**

We now evaluate the performance of SRP on Topology 3 with 30 nodes and  $p_m = 0.3$ . Recall that the SRP's access to sleep state is limited to a node's 1-hop neighbors, obtained from beacons broadcasted every  $T$  time units, set to 2 in the simulation. Node's sleep schedule follows a geometric distribution with a mean of 4. The performance of SRP is compared with a few variations of ExOR with different forwarder selection metrics: 1) the number of hops to best-path and loss rate [1], 2) ETX [2], and 3) EAX [13].

Figures 6(a) and 6(b) depict the average cost of these algorithms when nodes' costs are distributed uniformly with a mean 4 and idle cost is zero. ExOR, which is known to outperform traditional routing where packets are sent to the pre-computed path with the smallest costs, performs the worst among them in the figure. Other versions of ExOR using ETX and EAX metrics performs better than the original ExOR. On the other hand, the average cost of SRP is the minimum with the largest delay. Overall, our algorithms outperform ExORs in terms of average cost with reasonable delay performance.

## 8. CONCLUSION

We studied a routing problem in wireless sensor networks where sensors are randomly duty-cycled. We used a stochastic routing framework to analyze the structural properties of the optimal routing algorithm. We further developed a distributed algorithm utilizing local sleep/wake states of neighbors which performs better than some existing distributed algorithms such as ExOR.

## 9. REFERENCES

- [1] S. Biswas and R. Morris. Opportunistic routing in multi-hop wireless networks. In *Workshop on Hot Topics in Networks (HotNets-II)*, Nov. 2003.
- [2] S. Biswas and R. Morris. ExOR: Opportunistic multi-hop routing for wireless networks. In *Conference of the Special Interest Group on Data Communication (SIGCOMM)*, Aug. 2005.
- [3] D. Ferrara, L. Galluccio, A. Leonardi, G. Morabito, and S. Palazzo. Macro: An integrated mac/routing protocol for geographic forwarding in wireless sensor networks. In *Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Mar. 2005.
- [4] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical report ucla/csd-tr 02-0013, Feb. 2002.
- [5] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE Transactions on Networking*, 11(1), 2003.
- [6] D. B. Johnson and D. A. Maltz. *Dynamic source routing in ad hoc wireless networks*. Mobile computing, Kluwer Academic Publishers, 1996.
- [7] D. Kim. *Low Duty-Cycled Wireless Sensor Networks: Connectivity and Opportunistic Routing*. PhD thesis, Univ. of Michigan, 2008.
- [8] C. Lott and D. Teneketzis. Stochastic routing in ad-hoc networks. *IEEE Transactions on Automatic Control*, 51(1), 2006.
- [9] T. Melodia, D. Pompili, and I. F. Akyildiz. Optimal local topology knowledge for energy efficient geographical routing in sensor networks. In *Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Mar. 2004.
- [10] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications Journal*, Oct. 1996.
- [11] C. E. Perkins and P. Bhagwat. Highly dynamic destination sequenced distance vector routing (dsv) for mobile computers. In *Conference of the Special Interest Group on Data Communication (SIGCOMM)*, Oct. 1994.
- [12] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2004.
- [13] Z. Zhong and S. Nelakuditi. On the efficacy of opportunistic routing. In *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2007.
- [14] N. Zhou, H. Wu, and A. A. Abouzeid. Reactive routing overhead in networks with unreliable nodes. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCOM)*, Aug. 2003.