Optimal Stochastic Routing in Low Duty-Cycled Wireless Sensor Networks

Dongsook Kim and Mingyan Liu [1]

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 it 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 networks, and then simplify the algorithm to work with only local information. We further develop a distributed algorithm utilizing only local sleep/wake information of neighbors. This algorithm is shown to perform better than some existing distributed opportunistic routing algorithms such as ExOR.

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 efficient design of such networks at all levels, from material to circuit to protocol, has long been a critical research issue. Of different energy conservation approaches, low duty-cycling, the act of periodically 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. This in turn can cause delay in sensing, detection, and packet delivery (routing).

In this study, we are interested in designing good routing algorithms (measured by low delay and low energy consumption) 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. For instance, we may pre-determine a best route based on average performance (say delay or number of transmissions needed) using prior statistics, and at each hop of this route the upstream node simply waits for the downstream node to become available. Alternatively, we can make a state-dependent decision depending on which

[1]D. Kim is with Samsung Electronics Co. LTD, Telecommunication R&D Center, Korea. M. Liu is with Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122. This work was done while D. Kim was at the University of Michigan, and through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. An earlier version of this paper appeared at ACM WICON, November 2008, Maui, Hawaii.

set of neighboring nodes are available. An extreme example of this latter method is to forward the packet to the earliest available neighbor.

Low duty-cycling creates significant uncertainty in the availability and connectivity of the network, which is both a challenge and an opportunity that can be potentially exploited. Prior work in routing has primarily focused on the uncertainty caused by time-varying channel quality [2]. The goal of the present paper is to take both sources of uncertainty into account in designing a good 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)[3], 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 [1–8], and thus do not react to transmission failures actively. More recently, there have been a number of stochastic routing (also referred to as opportunistic routing) algorithms proposed in the literature [9–11] to address the uncertainty in transmission. The key idea underlying 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 routes taken by different packets will be different, thus the term *event-based routing* or *sample-path dependent routing* [9], or *opportunistic routing* [10].

It was shown in [9] that there exists an optimal Markov routing policy in the form of an index policy in a wireless network. Specifically, there exists a priority-ordering of nodes that can be computed off-line; a node continues to transmit till a higher-priority neighbor receives the packet successfully and becomes the next relay. A conceptually very similar but sub-optimal, though more practical, routing algorithm called ExOR was proposed in [10]. Compared to [9] ExOR has a different relay selection criterion; a node selects a relay among its neighbors based on a metric called estimated transmission count (ETX) which is the smallest estimated number of transmissions it takes to reach the destination along any possible path. Zhong et al. in [11] further improved ExOR by using a metric called expected any-path transmissions (EAX) in making relay selection decisions. EAX captures the expected number of transmissions needed to successfully deliver a data packet to a destination under opportunistic routing whereas ETX is the expected number of transmissions along a best path with the largest delivery probability. This work may be considered as an alternative implementation of the algorithm in [9]. This class of routing algorithms like the ones cited above 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 low duty-cycled sensor networks. In particular, we will follow closely the stochastic decision framework developed in [9]. Within this framework, transmissions are costly and a certain reward is obtained if the packet reaches a certain node. The objective is to find a routing algorithm that maximizes the total expected reward less the total cost. It was shown in [9] that there exists an optimal Markov policy for this problem with time-invariant transmission success probabilities, in the form of a priority policy. As we will show, optimal policies for the problem considered in [9] are not in general optimal for low duty-cycled sensor networks because they do not take into account the current sleep state of nodes. In particular, a sender may be forced to wait when a subset of its

[2]Uncertainty can also be caused by mobility, which we will not consider in this paper by assuming a quasi-static sensor network.

[3]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.

neighbors are asleep, a scenario that does not arise in the setting of [9].

The model used in this paper is an extension to [9] 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 the next section 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 decentralized and distributed algorithm utilizing local sleep/wake states of neighbors. This algorithm is shown to perform better than existing distributed opportunistic algorithms such as ExOR, both ETX-based [10] and EAX-based [11].

The remainder of this paper is organized as follows. Section 2 provides the description of the network model with assumptions and definitions. Based on the specified model, we consider the centralized stochastic routing problem in Section 4. In Section 5, we present two centralized stochastic routing algorithms, one exactly optimal and the other near-optimal. 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 numerically evaluated in Section 7 and is also compared to ExOR. We conclude the paper in Section 8.

2 Model Description and Problem Formulation

We consider a static wireless ad-hoc or sensor network where nodes are duty-cycled independently from one another. We will limit our attention to the delivery of a single message (or packet, a term used interchangeably) from a source node to a destination node in this model, but our performance evaluation of the resulting policy in Section 7 is not restricted to a single message. We note that this done primarily for the simplicity of presentation; the same policy can be equally applied to the routing of multiple packets.

At a high level, our problem is to find a good (in terms of delay and transmission cost) 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. Similar measures can also be defined to take into account factors such as energy consumption.

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 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. This effectively results in a decentralized routing problem: a node must decide, based on such local information whether it should serve as a relay for the message it receives. Because such decisions are made by individual nodes in a decentralized fashion, it is possible that multiple nodes may decide to relay the same message. Such a distributed implementation is accomplished via packet exchange and certain local information update procedure.

2.1 Assumptions

Below we summarize the main assumptions used in our analysis.

- We will focus on the unicast routing of a single message originated from somewhere in the network with a pre-specified destination. Under the stochastic routing framework, since the routing is sample-path dependent, each message may follow a different path. It should be noted that the exact same framework can be used to solve the more general, *anycast* problem where the message is considered successfully delivered if it reaches at least one of a set of destination nodes.

- 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 nodes, 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 practice, we only need to ensure that the node who is designated as the relay should stay awake till the next hop/relay receives the message successfully.

- The lossy wireless medium is modeled by a pair-wise time-invariant transmission success probability p_{ij} between the sender i and receiver j, independent of other transmission attempts. If this success probability is nonzero or above a given threshold, then j is called a "neighbor" of i. This probability does not have to be symmetric between two nodes.

- A transmission and its acknowledgment (ACK) from successful receivers occur within a single time slot. ACKs are assumed error-free.

2.2 Notations

A summary list of notations used in this paper is as follows.

N is the number of nodes in the network.

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

I denotes a nonexistent node; this is used to represent 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 the assumption: Given $A = \{a_1, a_2, \cdots, a_N\}$, $a_i = 1$ for all $i \in W$.

$F(W)$ denotes a feasible set of all possible sleep/active states A induced by W, such 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 $i \in W - W'$. Note that there is only one such A in this set.

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

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

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

where $1_{\overline{w}'}$ is the number of 1's in $T(W')$, and $1_{\overline{a}'}$ is the number of 1's in A'. If the idling action (or node) I is chosen,

$$P^I(W', A'|W, A) = \begin{cases} p^{1_{\overline{a}'} - 1_{\overline{w}'}} (1-p)^{N - 1_{\overline{a}'}}, \text{ if } W' = W \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

$R : 2^\Omega \to \mathbb{R}$ is a reward function. To simplify notation, we will denote $R_i = R(\{i\})$. Since we are considering the unicast problem, in general rewards of all nodes except for the destination node are zero.

π is a Markov policy such that π depends only on the current state (W, A). We write $\pi(W, A) = i$ to indicate that policy π select node i to transmit (as the relay) when in state (W, A), $i \in W$. We write $\pi(W, A) = I$ to indicate that policy π selects the idle/wait action. We write $\pi(W, A) = r$ to indicate that policy π retires and receives reward $R(W)$ when in state (W, A). For convenience we often write $\pi(W, A) = r_i$ to denote that policy π retires and receives the reward of node i, $R_i, i \in W$.

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

2.3 Problem Formulation

Problem 1 *We consider the transmission of a message or packet in a low duty-cycled wireless network of N nodes, where each node is active with probability p, described above. At each time instant the central controller chooses among three actions: (1) select a node among nodes that have the packet to perform the next transmission; (2) wait for the next time step; and (3) terminate the routing process. It acts at the beginning of each time slot with the knowledge of the set of nodes which have received the message and the set of current active nodes in the network. The transmission from a node i costs $c_i > 0$ and is the local broadcast to its active neighbors. The idle action, denoted by $i = I$, costs $c_I = \alpha \geq 0$, 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 right action at each time step and the right time to terminate the process so as to maximize the total expected reward less cost:*

$$E\{R(W_\tau) - \sum_{t=1}^{\tau-1} c_{i(t)}\}, \tag{3}$$

where τ is the stopping time when the transmission process is terminated, W_τ is the set of nodes with the message at τ, 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 [9] have shown that an optimal Markov policy is both a priority policy and an index policy; this will be referred to as *Lott's algorithm* throughout our discussion. The first few definitions below are reproduced from [9] for this paper to be self-contained. These explain what a priority or an index policy is. We then present an example to illustrate they are not able to capture the extra dynamics introduced by node sleeping. This motivates us to define generalized versions of priority policies and index policies, respectively.

Definition 1 *[9] A Markov policy π 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, $\forall S \subseteq \Omega_i$, where Ω_i is the set of nodes of priority lower than i.*

Definition 2 *[9] A function $f : 2^\Omega \to \mathbb{R}$ is an index function on Ω if f satisfies*

$$f(S) = \max_{i \in S} f(\{i\}), \quad \forall S \subseteq \Omega. \tag{4}$$

Definition 3 *[9] A priority policy π is called an index policy if $V^\pi(S)$, which is the expected total reward less cost under policy π given state S, is an index function on Ω.*

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

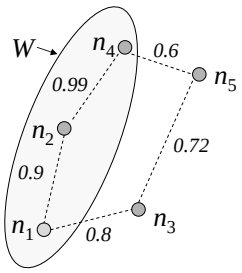


Figure 1: System for an Example 1.

Example 1 *We consider a system depicted in Figure 1, where* $\Omega = \{1, 2, 3, 4, 5\}$ *and nonzero transmission success probabilities between nodes. 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* $\forall i \in \Omega$*. Let us consider first the case where nodes are not duty cycled. An optimal policy can be found by applying Lott's algorithm. For instance, when* $W = \Omega$*, it is trivial to see that the optimal action is to retire and receive* R_5*. Any* W *that includes node 5 results in the same decision as above; node* 5 *will thus be considered to have the* highest priority *among all nodes. When* $W = \{1, 2, 3, 4\}$*, the optimal decision is for node* 3 *to transmit. Similarly, the optimal decision given any* W *that includes node* 3 *is always to select node 3 for transmission. Node* 3 *thus has the highest priority among all nodes except for* 5*. If we take nodes* 5 *and* 3 *away from the set* W*, then node* 4 *becomes the optimal decision, with the next highest priority, regardless of the membership of the rest of the set. Eventually, by repeating this process until* W *becomes empty, we obtain an ordered list of nodes, in descending order of their priorities. For this particular example, the priorities are such that the ordered list is nodes 5, 3, 4, 1, 2 from the highest to the lowest. The result is called a priority policy because there exists such a priority list that is independent of the actual state of the system, and that the optimal decision is based on this priority list: choose the highest priority node among* W *for the next transmission, and continue to transmit till a node of even higher priority receives the message.*

Now, we consider the case where nodes are duty cycling with active probability $p = 0.1$*. In addition, we assume that the idling cost is 1, i.e.,* $c_I = 1$*. To facilitate the discussion, an active node* i *is denoted by* ia *and a sleeping node* i *by* is*. As mentioned in the previous section, nodes in* W *are assumed to be awake. Therefore, we only need to consider nodes in* $\Omega - W$ *for their on/off states. Let* $W = \{1, 2, 4\}$ *as shown in Figure 1. Let* π^* *to be an optimal Markov policy. We have* $\pi^*(W, \{3a, 5a\}) = 4$*,* $\pi^*(W, \{3a, 5s\}) = 1$*,* $\pi^*(W, \{3s, 5a\}) = 4$*, and* $\pi^*(W, \{3s, 5s\}) = I$*. The detailed calculations can be found in Appendix A. Let us focus on* $A = \{3a, 5s\}$*. In this case, node* 1 *seems to be the highest priority node among nodes* $1, 2$*, and* 4*. Now, suppose* $W = \{1, 2\}$*. We obtain* $\pi^*(W, \{3a, 4a, 5s\}) = 2$ *and* $\pi^*(W, \{3a, 4s, 5s\}) = 1$ *by the calculation similarly done in Appendix A. When node 4 is in sleep mode, 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 between nodes 1 and 2. In other words, node 1 is not always the highest priority node among nodes* $1, 2$*, and* 4*; it depends on the sleep states of other nodes.*

Remark 1 *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* $W - \{i\}$ *is in general not preserved under other sleep/wake states. Thus if we remove node* i*, we will 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 [9] both from a conceptual and a computational point of view.

The above example suggests that it is necessary to generalize the preceding definitions in the context of our problem.

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

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

where 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 $f : 2^{\Omega} \times 2^{N} \to \mathbb{R}$ is an Generalized(G)-index function on 2^{Ω} if f satisfies*

$$f(W, A) = \max_{W' \subseteq W, A' \in F(W'|W,A)} f(W', A'), \quad \forall W \subseteq \Omega, \forall A \in F(W) \ . \tag{5}$$

Definition 6 *A priority policy π is called an Generalized(G)-index policy if $V^{\pi}(W, A)$ is an G-index function on Ω.*

We end this section by noting two special-case interpretations of Problem 1 depending on what we use as costs.

3.0.1 The case of $c_I = 0$

If the idle cost is zero, there is no penalty on waiting. In this case, there is no loss of optimality to always wait till all nodes are awake (a positive probability event) and then make a decision on who is to transmit. If we only consider the problem in this particular sleep state (all awake), i.e., we wait in other states, then the problem becomes identical to the one studied and solved in [9].

3.0.2 The case of $c_i = c_I = c$

If all costs are the same, the problem can be regarded as finding a policy which minimizes delay. Assuming the transmission of a packet consumes a certain amount of time and so does waiting, each cost can be translated into a time unit. Therefore, the problem is to find a policy that minimizes the sum of the time slots taken.

4 Analysis of Problem 1

In this section, we analyze Problem 1 and derive structural properties of an optimal policy π^*. As mentioned earlier, we will take a centralized point of view and assume that at each time instant, a decision-maker has complete information on the time-invariant transition probabilities and the

current sleep/wake state. We will then use these properties to construct optimal and sub-optimal routing policies. In a later section we will discuss distributed implementations of these.

Our system of Problem 1 can be modeled as a two-dimensional finite state Markov chain. That is, each decision is made based on the current state (W, A). Without loss of optimality, we will limit our attention to Markov policies. One may use stochastic dynamic programming to find an optimal Markov policy. Suppose s and d are the source and destination nodes, respectively. We can then use the following set of dynamic programming equations:

$$\begin{aligned} V(\Omega, A) &= R_d; \\ V(W, A) &= \max_{i \in W \cup \{I\}} \left\{ -c_i + \sum_{W' \supseteq W} \sum_{A' \in F(W')} P^i(W', A'|W, A_j) V(W', A'), R_i \right\}, \\ &\qquad \forall W \subset \Omega, A \in F(W) \ , \end{aligned} \tag{6}$$

and the optimal reward is given by the expected value of $V(\{s\}, A)$ over all possible states $A \in F(\{s\})$.

However, the computational complexity involved in this approach is very high. For instance, suppose that the number of nodes in the network is N and $|W| = n$. Given W, there are 2^{N-n} A's in $F(W)$ and $n+1$ actions, one for each node in W plus I. For each pair (W, A_i), $A_i \in F(W)$, its optimal value function requires the optimal value functions for other sleep/wake states $(W, A_j), \forall A_j \in F(W)$. All these optimal value functions are solved simultaneously by setting the action for each (W, A_j). Thus, the number of such combinations is $(n+1)^{2^{(N-n)}}$ for given W. And there are $\frac{N!}{n!(N-n)!}$ W's for $|W| = n$. Therefore, the total number of calculations is

$$\sum_{n=1}^{N} \frac{N!}{n!(N-n)!} (n+1)^{2^{(N-n)}}. \tag{7}$$

As N grows, the complexity grows rapidly. For this reason, instead of applying stochastic dynamic programming directly, we will investigate the structural properties of an optimal Markov policy, which are then used to construct algorithms with lower complexity.

We next show that there exists an optimal G-index policy for Problem 1 in Theorem 1. The idea behind the proof of Theorem 1 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. We then propose an algorithm to find an optimal G-index policy and discuss its computational complexity. While this method follows closely the framework developed in [9], there are intricate technical differences and additional difficulties due to the introduction of sleep states.

Unless otherwise noted, all missing proofs may be found in the appendix.

The proof of Theorem 1 utilizes some useful lemmas presented next. Lemma 1 below shows that an optimal Markov policy has the property 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 π^* be an optimal Markov policy for Problem 1. Suppose we are given W_1 and $A_1 \in F(W_1)$, and let π be a Markov policy with the following properties:*

$$V^{\pi}(W, A) = V^{\pi^*}(W, A), \quad \forall W \supset W_1, \forall A \in F(W), \tag{8}$$

$$\pi(W_1, A_1) = \pi^*(W_1, A_1), \quad \forall A_1 \in F(W_1). \tag{9}$$

Then

$$V^{\pi}(W_1, A_1) = V^{\pi^*}(W_1, A_1). \tag{10}$$

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

Lemma 2 *In Problem 1, let π^* 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 properties of an optimal Markov policy, specifically the G-priority structure.

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

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

$$\begin{aligned} & \pi(W, A_i) = n_i \in W \cup \{I\} \\ \Rightarrow \quad & \pi(W - \{j\}, A) = n_i, \quad \forall j \in W - \cup_{i=1}^{m} n_i, \ \forall A \in F(W - \{j\}|W, A_i), \qquad (11) \\ & \pi(W, A_i) = r_{n_i}, n_i \neq I \\ \Rightarrow \quad & \pi(W - \{j\}, A) = r_{n_i}, \quad \forall j \in W - \cup_{i=1}^{m} n_i, \ \forall A \in F(W - \{j\}|W, A_i). \qquad (12) \end{aligned}$$

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 $i \in \{1, \cdots, m\}$,*

$$\begin{aligned} V^{\pi}(W - \{j\}, A) = V^{\pi}(W, A_i) = V^{\pi^*}(W, A_i) = V^{\pi^*}(W - \{j\}, A), \\ \forall j \in W - \cup_{i=1}^{m} n_i, \forall A \in F(W - \{j\}|W, A_i). \end{aligned} \tag{13}$$

3. *π is an optimal Markov policy.*

The following lemma shows that an optimal markov policy has the expected reward that is a G-index function.

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

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

Proof: By Lemma 3, there exists a Markov policy π^* 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 $\bigcup_i n_i$ from W remains the same. Thus the conditions in Definition 4 are satisfied. Thus π^* is a G-priority policy. Since π^* is a G-priority policy and its $V^{\pi^*}(\cdot)$ is a G-index function, π^* is a G-index policy according to Definition 6. ∎

5 Optimal and Sub-Optimal Routing Algorithms

5.1 An Optimal Centralized Algorithm for Problem 1

Compared to using brute-force dynamic programming, we can utilize the properties of G-index policy stated in Lemma 3 to reduce the amount of computation. The key idea is that for a given set W, once we identify the set of all highest-priority nodes $N_W = \bigcup n_i - \{I\}$, where n_i is the

highest-priority node for sleep state $A_i \in F(W)$, then removing non-highest-priority nodes from the set W will not change the optimal action or the maximum reward, resulting in savings in computation. By contrast, direct computation using (6) would require computing the rewards of all supersets of W.

More specifically, the procedure starts with $W = \Omega$ and $A = \{1, \cdots, 1\}$. Its optimal action and maximum reward are straight-forward, which are

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

From Properties 1 and 2 in Lemma 3, we know

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

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 $A_i \in F(\Omega - \{d\})$. Again denote by $N_{\Omega - \{d\}} = \cup_i n_i - \{I\}$ the set of highest priority nodes in $\Omega - \{d\}$. Using properties of Lemma 3, we have

$$\pi(S \cup N_{\Omega - \{d\}}, A) = n_i, \forall S \subset \Omega - \{d\}, A \in F(S \cup N_{\Omega - \{d\}} | \Omega - \{d\}, A_i).$$

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

We now formally describe the above procedure in Algorithm 1. Note that this algorithm is presented for a single destination, but can be easily extended to the case of multiple destinations.

Algorithm 1 *Define sets W, $F(W)$, N_W and a queue Q, as follows. Each entry in queue Q contains an ordered set of nodes $S \in \Omega$. Each is interpreted as a possible set of nodes that have* not *received the packet. We will denote by Q_b the first entry in the queue (head of the queue). W is the complement of Q_b with respect to Ω, i.e., $W = \Omega - \{Q_b\}$, 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 Q_b (with all ones for the nodes in W): $F(W) = \{A_1, A_2, ..., A_m\}$ where $m = 2^{|Q_b|}$. N_W is the set of highest priority nodes in W, each for a given $A_j \in F(W)$.*

The initial case $W = \Omega$ is trivial and already known: $V(W, A) = R_d$ and $\pi(W, A) = r_d$.

We now start with $W = \Omega - \{d\}$. Q contains only one entry $\{d\}$, the destination node. $F(W)$ contains two sets: 1's for all nodes except for d, which is 0 in one set and 1 in the other. The algorithm proceeds as follows.

1. *Take the set Q_b, find $W = \Omega - Q_b$. If W is empty, go to step 5.*

2. *For each state (W, A_j), $1 \leq j \leq m$, define the reward of taking action i (either a node for transmission or idle), $i \in W \cup \{I\}$ and each $A_j \in F(W)$, as follows:*

$$V_i(W, A_j) = \max\{-c_i + \sum_{W' \supseteq W} \sum_{A' \in F(W')} P^i(W', A' | W, A_j) V(W', A'), R_i\} \ , \qquad (14)$$

where $c_I = \alpha$. This results in $|W|+1$ linear equations for $|W|+1$ unknowns for each (W, A_j) (note that the quantities $V(W', A')$ will have been computed in previous iterations). Solve these to obtain $V_i(W, A_j)$, $i \in W \cup \{I\}$, $A_j \in F(W)$.

3. *Update the optimal reward and action as follows:*

$$V(W, A_j) = \max_{i \in W \cup \{I\}} V_i(W, A_j) \ , \tag{15}$$

$$\pi(W, A_j) = \arg \max_{i \in W \cup \{I\}} V_i(W, A_j) = n_j \ . \tag{16}$$

In obtaining the optimal action n_j in (16), ties are broken randomly between nodes, and the idle action is chosen if ties occur between a node and the idle action. Note that retiring is not considered here because W starts from $\Omega - \{d\}$ and decreases (goes backward) over the iterations.

4. *Compute $N_W = \cup_{1 \le j \le m} \{n_j\} - \{I\}$. N_W now contains a set of distinct nodes of the highest priority among all nodes in W. Note that N_W can be empty. For each node $n_j \in N_W$, add a new entry $Q_b \cup \{n_j\}$ to the queue Q.*

5. *Use Eqns (11)-(13) to obtain the action and reward for state $(W - \{j\}, A), \forall j \in W - \cup_{i=1}^{m} n_i, \ \forall A \in F(W - \{j\}|W, A_i)$.*

6. *Remove the current Q_b entry from the queue, and point Q_b to the next entry. If the queue becomes empty, terminate the algorithm. Otherwise go to step 1.*

7. *Upon termination, compute the reward*

$$V(\{s\}) = \sum_{A_j \subset F(\{s\})} V(\{s\}, A_j) P(A_j). \tag{17}$$

It should be fairly easy to see that the above procedure generates an optimal G-index policy π for Problem 1, and the reward functions $V(W, A)$ are solutions to the set of dynamic programming equations (6). This is because the procedure essentially computes (6) backward while exploiting the property of a G-index policy in steps 4-6. For this reason we simply state the following theorem.

Theorem 2 *Algorithm 1 produces an optimal G-index policy for Problem 1.*

5.2 A Sub-Optimal Algorithm

While Algorithm 1 can provide a useful benchmark, its computational complexity remains very high and can only be used in small-size problems. In this section we present an approximation that significantly simplifies the computation. Consider the following modification to the system: suppose we will use W rather than (W, A) to represent the state. Equivalently, suppose that the decision-maker has the knowledge of the nodes that have received the message but *no* information on the sleep/wake status of any node. Our approximation combines the optimal solution to this problem, which is known and easy to compute, with a greedy use of the extra knowledge A.

We first redefine some notations for use in this new setting.

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

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

$$P^i(W'|W, A) = \left(\prod_{\forall j: w_j = 0, a_j = 1, w'_j = 1} q_{ij} \right) \cdot \left(\prod_{\forall j: w_j = 0, a_j = 1, w'_j = 0} 1 - q_{ij} \right) \cdot \left(\prod_{\forall j: w_j = 0, a_j = 0, w'_j = 1} 0 \right), \quad \text{for } \forall i \in W \ . \tag{18}$$

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

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

Without nodes' active/sleep information, the problem is reduced to the one studied in [9] with a modification to the state transition probability. This is because under the above assumptions the decision-maker cannot differentiate transmission failures caused by channel errors from the ones by duty cycling. Hence, the sleep/wake activity of nodes is reflected in transition probabilities 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 [9] (Lott's algorithm) to generate an optimal index policy for this modified problem. Specifically, under this model the optimal expected reward given state W is

$$\tilde{V}(W) = \max_{i \in W} \{-c_i + \sum_{W' \supseteq W} \left(\sum_{A \in F(W)} P^i(W'|W, A)P(A) \right) \tilde{V}(W'), R_i\} \ , \tag{19}$$

and the optimal policy is given by a deterministic priority ordering of nodes that can be computed offline as mentioned earlier. Note that under this model the idle action is never chosen.

Below we present an algorithm that both utilizes and outperforms Lott's algorithm for Problem 1. Specifically, the decision maker uses 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. This significantly simplifies the computation compared to Algorithm1, and at the same time allows it to outperform Lott's algorithm.

Algorithm 2 *The sets W, $F(W) = \{A_1, A_2, ..., A_m\}$, N_W, and the queue Q are defined similarly as in Algorithm 1. This 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 as shown in (19). The on-line part of the algorithm proceeds similarly as in Algorithm 1 as follows. Again we will start from Q containing a single entry $\{d\}$.*

1. *Take the set Q_b, find $W = \Omega - Q_b$. If W is empty, go to step 5.*

2. *For each (W, A_j), $1 \leq j \leq m$, compute the reward of taking action i (either a node for transmission or idle), $i \in W \cup \{I\}$ and each $A_j \in F(W)$, as follows:*

$$V_i^{'}(W, A_j) = \max\{-c_i + \sum_{W' \supseteq W} P^i(W'|W, A_j)\tilde{V}(W'), R_i\}. \tag{20}$$

3. *Update the reward and action as follows:*

$$V^{'}(W, A_j) = \max_{i \in W \cup \{I\}} V_i'(W, A_j) \ , \tag{21}$$

$$\pi^{'}(W, A_j) = \arg \max_{i \in W \cup \{I\}} V_i'(W, A_j) = n_j \ . \tag{22}$$

   In obtaining the optimal action n_j in (22), ties are broken randomly between nodes, and the idle action is chosen if ties occur between a node and the idle action.

4. *Compute $N_W = \cup_{1 \leq j \leq m}\{n_j\} - \{I\}$. For each node $n_j \in N_W$, add a new entry $Q_b \cup \{n_j\}$ to the queue Q.*

5. *Remove the current Q_b entry from the queue, and point Q_b to the next entry. If the queue becomes empty, terminate the algorithm. Otherwise go to step 1.*

Unlike Lott's algorithm, Algorithm 2 takes an action dependent on A. It recomputes the priorities of nodes in W with consideration of sleep/wake status at the time of transmission and selects a node with the highest modified priority for the next transmission. This algorithm cannot perform better than Algorithm 1 as the latter is optimal. However, below we show that it is at least as good as Lott's Algorithm.

Theorem 3 *Algorithm 2 performs at least as good as Lott's algorithm for Problem 1.*

Proof: As before, $F(W) = \{A_1, A_2, \cdots, A_m\}$. Using (19), we have

$$\begin{aligned}
\tilde{V}(W) &= \max_{i\in W}\{-c_i + \sum_{W'\supseteq W}\left(\sum_{j=1}^{m} P^i(W'|W,A_j)P(A_j)\right)\tilde{V}(W'), R_i\} \\
&= \max_{i\in W}\{\sum_{j=1}^{m}\left(-c_i + \sum_{W'\supseteq W} P^i(W'|W,A_j)\tilde{V}(W')\right)P(A_j), R_i\} \\
&\leq \max_{i\in W}\{\sum_{j=1}^{m}\left(-c_{n_j} + \sum_{W'\supseteq W} P^{n_j}(W'|W,A_j)\tilde{V}(W')\right)P(A_j), R_i\} \\
&= \sum_{j=1}^{m}\left(-c_{n_j} + \sum_{W'\supseteq W} P^{n_j}(W'|W,A_j)\tilde{V}(W')\right)P(A_j) \\
&= \sum_{j=1}^{m} V'(W,A_j)P(A_j)\ , \qquad (23)
\end{aligned}$$

where the inequality is due to the fact that action n_j maximizes the term within the summation per (22), and the third equality is due to the fact that retiring is not optimal.

This shows that when averaged over all possible sleep states, Algorithm 2 performs at least as good as Lott's Algorithm. ∎

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 [10] 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 packet in the sequel) packet when they are awake. From these exchanges nodes infer 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 priorities computed using Lott's algorithm; these will be referred to as the *offline priorities* for clarity but it is not necessarily an offline process. [9] proposed an efficient Dijkstra-like distributed algorithm for a node to compute its priority. As nodes obtain their neighbors' sleep state they can choose to recalculate and update these priorities during the priority update stage. In the forwarder selection step a node decides for itself whether it should become a forward and retransmit the packet it received based on current priorities. Below we present these two elements in more detail.

6.1 Priority Update Procedure

An active node i transmits a short HELLO packet periodically[4]. This HELLO packet contains explicit information on measured channel quality and implicitly conveys the fact that the sender of the HELLO packet is active. In addition, it contains the current value of node i's priority, denoted by $V^l(i)$ for the l-th updating period, calculated as follows.

The initial value $V^0(i)$ for all i is obtained using Lott's Algorithm. Recall that the optimal policy obtained by Lott's Algorithm is an index policy (i.e., $\tilde{V}^{\tilde{\pi}}(W) = \tilde{V}^{\tilde{\pi}}(\{i\})$ if i is the highest priority node under $\tilde{\pi}$ in W). As part of initialization, we assign $V^0(i) = \tilde{V}^{\tilde{\pi}}(\{i\})$ to node i at the start of the algorithm; without ambiguity $\tilde{V}^{\tilde{\pi}}(\{i\})$ is also written as $\tilde{V}(i)$ below for simplicity of notation.

This quantity is then updated before node i sends out each beacon within a single wake period, and is reset to $V^0(i) = \tilde{V}(i)$ upon waking up from a sleep period. Specifically, right before the l-th beacon transmission at time t_l^i, node i updates $V^l(i)$ and includes this value in the beacon packet. Note that the transmission times of the beacon packets are unsynchronized among nodes in the network; a node's beacon transmission times are only relevant to its latest wake-up time. Thus, t_l^i for node i might be different from t_l^j for node j. Node i recalculates $V^l(i)$ based on updates received from active neighbors during the time interval $[t_{l-1}^i, t_l^i]$. In addition, node i maintains a candidate set denoted by C_i; this is a subset of all i's neighbors whose current priorities are higher than i's. In other words, C_i contains all possible forwarders. Initially, C_i contains the nodes with higher initial priorities (determined by $V^0(\cdot)$) than i's. This set then gets updated with priority updates.

The more precise details are given in the following description of the priority update procedure followed by a given node i. We have assumed that the computation of $\{\tilde{V}(i)\}$ by Lott's Algorithm is completed prior to running SRP, such that each nodes has its own $\tilde{V}(i)$ as well as $\tilde{V}(j)$ for all neighboring nodes j.

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 l 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 $\tilde{V}(i)$. It initializes $V_i^0(j)$ to $\tilde{V}(j)$ for all j in its neighboring set. The set A_i that contains all active neighbors is initialized to be empty. The set C_i of forwarder candidates contains a neighbors j if $\tilde{V}(j) > \tilde{V}(i)$.

3. Node i then increments l by one, and sets the next beacon transmission time t_l^i to $t_{l-1}^i + T$, where T is the (constant) beacon interval.

4. Between t_{l-1}^i and t_l^i, if node i receives a beacon packet from some neighbor j, it updates $V_i^{l-1}(j)$ 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 l-th beacon transmission at time t_l^i, node i recalculates the priorities as follows. If a beacon packet from node j was last received at a time earlier than $t_l^i - \beta T$, where β is a constant multiplier and βT sets a threshold on how long a neighbor has not been heard from before assuming it is asleep, then node j is assumed to be in sleep mode and is removed from A_i. For those nodes in A_i, set $V_i^l(j) = V_i^{l-1}(j)$. Otherwise, set $V_i^l(j) = \tilde{V}(j)$

[4]HELLO packets are commonly used for neighborhood discovery, a mechanism employed by virtually all routing protocols to maintain fresh information on which nodes are one's neighbors. In this sense our protocol simply utilizes an existing mechanism and the exchanged state information gets a free ride.

for a sleep node j. Include in C_i all neighbors that qualify as possible forwarders and their current priorities.

Denote by $q^*_{ij|C_i,A_i}$ the probability that node j receives successfully from node i while nodes with higher priorities in $A_i \bigcap C_i$ fail. Denote the set of nodes with higher priorities than node j by $\{A_i \bigcap C_i\}^+_j \subset A_i \bigcap C_i$. Then,

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

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

$$V^l(i) = \frac{-c_i + \sum_{j \in A_i \bigcap C_i} q^*_{ij|C_i,A_i} V^l_i(j)}{1 - \sum_{j \in A_i \bigcap C_i} (1 - q_{ij})}.$$

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

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

Remark 2 *The relationship between T and an "on" duration: We assume that an on duration is larger than a beacon interval T. The length of an on duration obviously affects the accuracy of the recalculation of $V^l(i)$.*

6.2 Forwarder Selection Procedure

When a forwarder, say node i, sends out the message, it contains a list of potential forwarders C_i. When node j receives the message within its l-th beacon interval, $[t^j_{l-1}, t^j_l]$, it first checks to see if it is included in the set C_i. If it is, it waits for a certain time period to see if it hears any ACKs from higher priority nodes. This time period is randomly chosen but inversely related to its own priority position in C_i. If it does, then node j will not transmit the message. If it fails to receive any ACK from higher priority nodes during the period, it transmits the message containing its list of forwarder candidates in the message. Details of this forwarder selection procedure are provided in the following algorithm. This algorithm is performed whenever node generates a message or receives it from one of its neighbors.

1. A node j while awake stays in the listening mode. When it receives a message, say from node i, it obtains the list of candidate forwarders C_i. If it is on the list, go to step 2. Otherwise, it does not forward the message and remains in the listening mode.

2. If node j is the first on the list C_i, it becomes the forwarder, sends out an ACK and transmits the message right away, with its updated candidate list C_j. Otherwise, node j sets a timer D_1 proportional to its position on the list C_i.

3. Within this timer D_1, if node j receives an ACK from a higher priority node, say node k, on C_i, it transmits a duplicate of this ACK with the identity of node k. If node j does not receive any ACK from a higher priority node before timer D_1 expires, upon timer expiration node j it becomes the forwarder, sends an ACK and transmits the message with its updated candidate list C_j.

4. A forwarder j sets a timer D_2 upon transmitting the message. If it does not receive any ACK before the timer expires, it times out and retransmits the message with an updated list C_j, up to a certain maximum number. (This rule applies to the source node as well.)

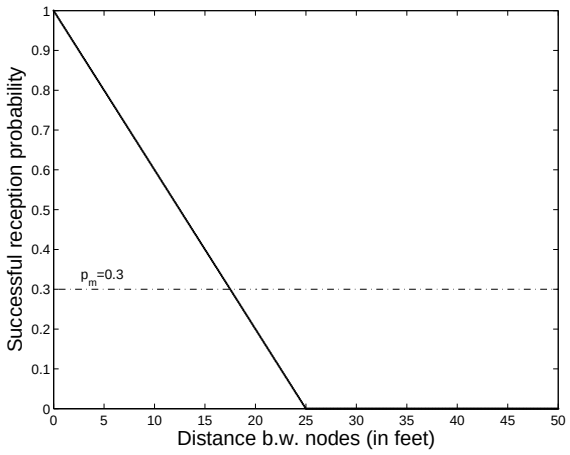


Figure 2: Delivery success probability w.r.t. distance.

7 Performance Evaluation

We performed extensive MATLAB simulation to evaluate the performance of the proposed algorithm. The simulated system closely follows most of the assumptions listed earlier in this paper, but is not restricted to single-message routing. Here we reiterate some of the more relevant ones. 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 linear function of the distance between the nodes as shown in Figure 2. This distribution is based on measurements on Rene Motes using medium transmission power reported by Ganesan et al in [12]. In general, a node with non zero reception probability is regarded as a neighbor. However, we also eliminate nodes with poor reception probability (those lower than a threshold p_m) from a neighboring set. 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. A node that has received a message does not go back to sleep again till the simulation ends.

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*: Under this scenario the problem reduces to finding a delay-optimal path from a source to a destination. Note that the term *delay* used in this paper accounts for the number of time units taken to reach the destination considering both hop counts and retransmissions caused by channel errors. With this cost scenario we may also find a path that minimizes energy consumption, given that the normalized energy consumption in transmission is roughly the same as that in idle waiting.

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. Because both transmissions and waiting are costly, there may be a tradeoff between minimizing the number of transmissions and minimizing delay. For instance, a path may incur the smallest number of transmissions (e.g., a shortest path when all transmission success probabilities are equal) but may involve a large amount of waiting. The combined cost may render this path less desirable. The tradeoff between transmission energy consumption and delay can be adjusted through setting the respective costs. The intention of using a random transmission cost is so that this cost

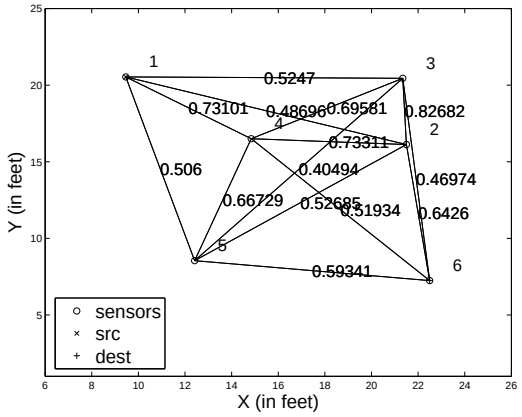


Figure 3: Topology 1: 6 nodes. Source node: 7, destination node: 18.

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.

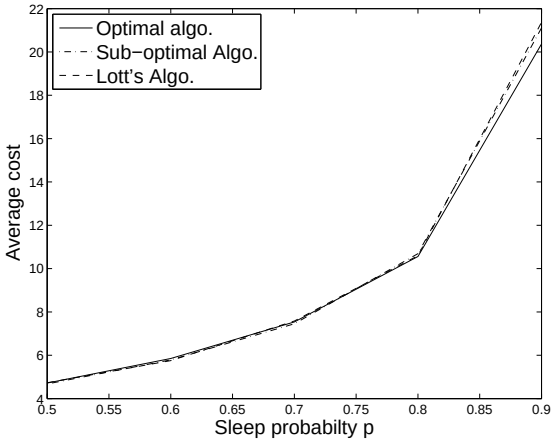
3. *Random cost for transmission and zero cost for idle action*: In this case the problem looks for a cost-efficient path without worrying about penalty on waiting. Since there is no penalty on waiting, it is optimal to wait till all nodes are awake and the apply Lott's algorithm. The third scenario is meant for applications that are extremely delay-tolerant.

7.1 The effect of sleep information on optimality

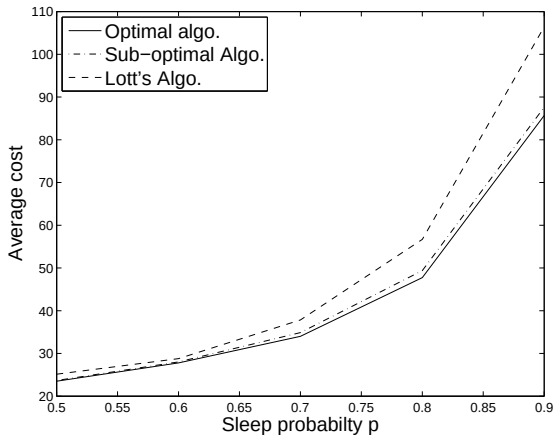
In the previous sections, it was shown that Algorithm 1, referred to as the Optimal Algorithm in the remainder of this section, generates an optimal G-index policy for Problem 1. Unfortunately, its computational complexity is high and is thus not scalable. We therefore use a small network to compare its performance with .... The network consists of 6 sensor nodes with average node degree 4.6 and $p_m = 0.3$ as shown in Figure 3, referred to as Topology 1. Using this topology, we first examine how much performance degradation will result if we ignore sleep information. In Figure 4 we compare Algorithm 1, Lott's algorithm which requires and uses no sleep information, and Algorithm 2 (also referred to as the sub-optimal algorithm in the remainder of this section) that greedily utilizes the current sleep state in making forwarding decisions.

Figure 4 depicts the average costs of paths taken by these algorithms when different cost distributions are applied. When all costs are the same and normalized to unit as in cost scenario 1, average path cost is identical to average delay. When p is relatively small up to 0.8, average delays of all three algorithms are virtually indistinguishable as shown in Figure 4(a). As p becomes very high (0.9), the optimal algorithm shows a slight advantage. This suggests that if we are only interested in delay, then all three algorithms perform very closely. This is because the optimal and sub-optimal algorithms are discouraged from waiting too long and will try to transmit sooner, which results in similar behavior as under Lott's algorithm which does not wait at all.

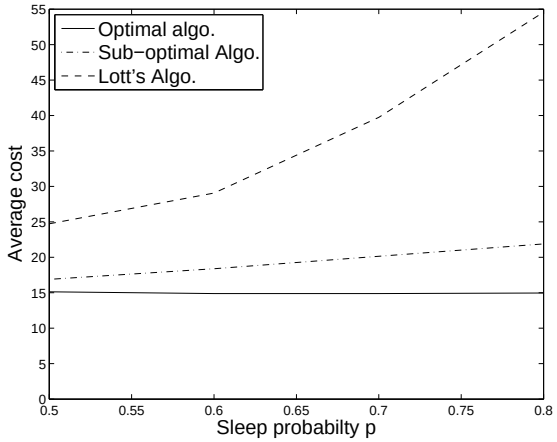
In cost scenario 2, transmission costs are uniformly generated over $[1, 7]$ while idle cost is fixed at 4. As shown in Figure 4(b), it is quite remarkable that the sub-optimal algorithm performs nearly as good as the optimal one. Since the transmission costs can be significant in this case, the optimal and suboptimal algorithms make more judicious decisions on waiting to saving excessive transmission. Lott's algorithm, being oblivious to sleep state, results in more wasteful (less efficient) transmissions.



(a) Scenario 1 with unit costs.

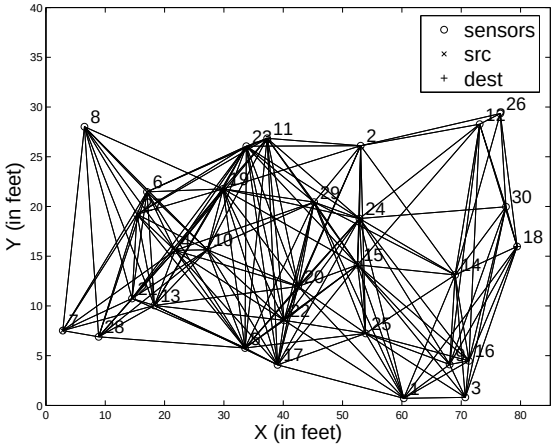


(b) Scenario 2 with random cost and nonzero idle action penalty.

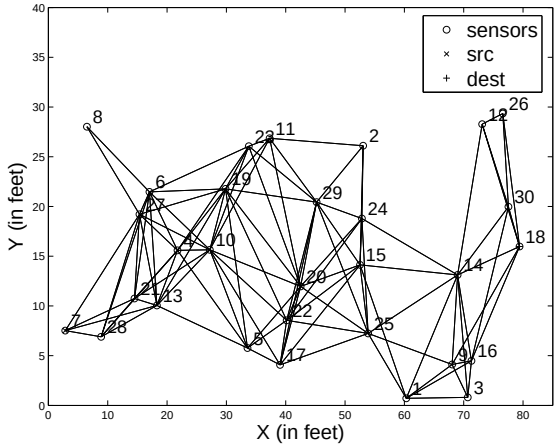


(c) Scenario 3 with random cost and zero idle action penalty.

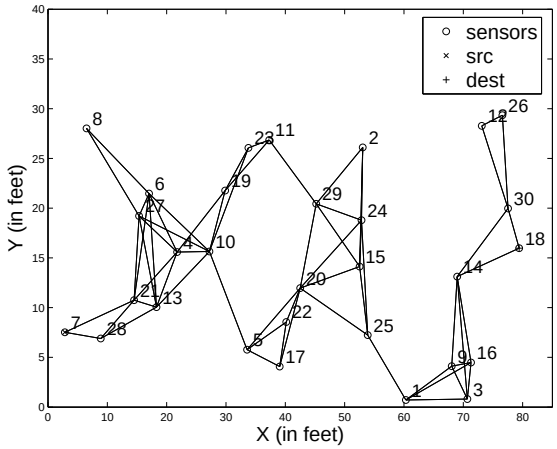
Figure 4: Performance comparison of the centralized algorithms on Topology 1



(a) Topology 2 with average node degree = 12.33 when $p_m = 0$.



(b) Topology 3 with average node degree = 7.13 when $p_m = 0.3$.



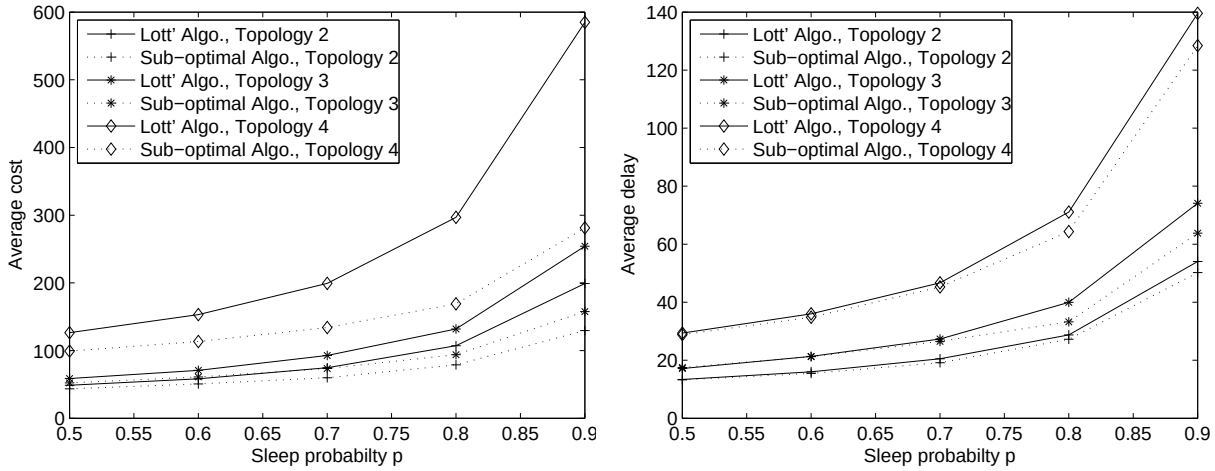
(c) Topology 4 with average node degree = 4.13 when $p_m = 0.5$.

Figure 5: Topologies with 30 sensor nodes. In all cases the source node is 7, destination node 18.

In cost scenario 3, transmission costs are generated by the same distribution as above but no costs are imposed on the idle action. Figure 4(c) shows that the average costs of the optimal algorithm and sub-optimal algorithm are almost unaffected by the increase in sleep probability by taking a large number of idle actions and waiting for the right moment to transmit. In particular, the average cost of the optimal algorithm is exactly the same while waiting delay increases exponentially as p increases. On the other hand, the cost of Lott's algorithm rises quickly since it does not take sleep state into account which results in many wasted transmissions.

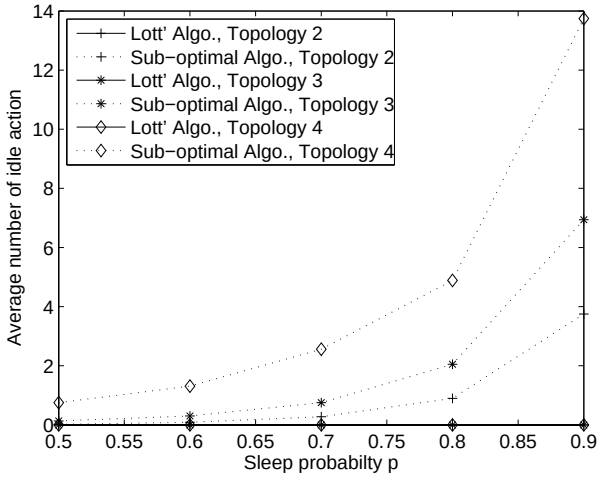
7.2 The effect of node degree

If a node has more neighbors, given the same sleep probability it is more likely to have wake neighbors. However, even in a highly connected network, a best neighbor is not always on. Thus, whether to transmit now or wait for better neighbors to be on is not a straight-forward question to answer depending on which neighbors are awake at the time of transmission. We focus on the performance comparison of Lott's algorithm and the sub-optimal algorithm when increasing the average node degree in the next set of results. We consider a network where $N = 30$ sensor nodes



(a) Average cost.

(b) Average delay.



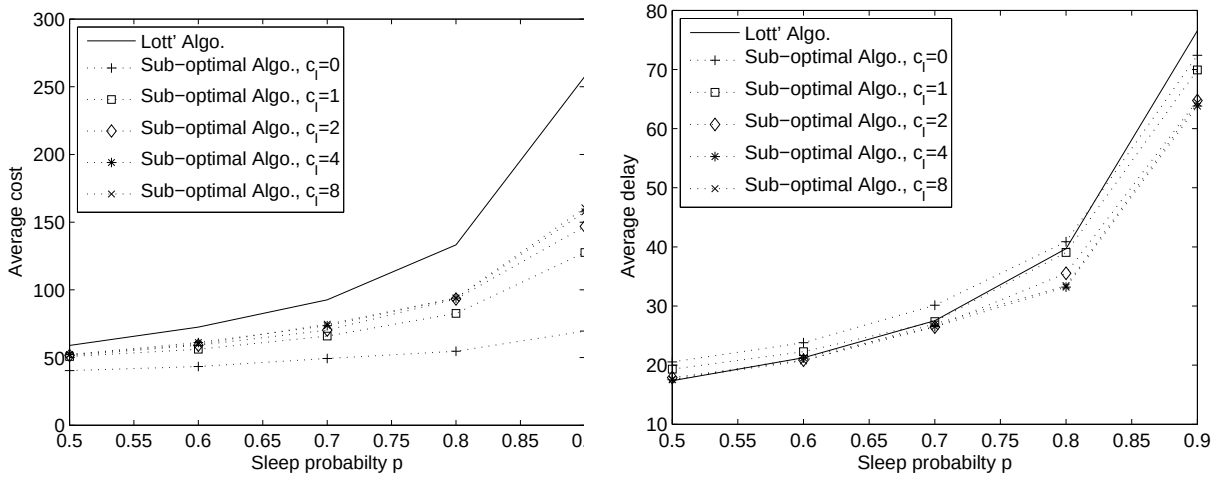
(c) Average number of idle actions.

Figure 6: The effect of average degree of nodes on the performance of sub-optimal and Lott's algorithms (cost scenario 3).

are deployed with different $p_m = \{0, 0.3, 0.5\}$ as shown in Figure 5. p_m determines the set of neighbors and so does node degree. In all topologies the source node is 7 and the destination node 18.

Using the third cost scenario, both algorithms improve as the degree increases, much to be expected. Figure 6(a) shows that the sub-optimal algorithm improves more quickly compared to Lott's algorithm, and the improvement is more pronounced with larger p. This suggests that the sub-optimal algorithm is more effective when duty-cycling is heavy. This is because there are sufficient number of wake neighbors around, which makes idle action unnecessary.

Figure 6(b) shows that the delay performance of the two algorithms are quite close, with the sub-optimal algorithm slightly better. This is a somewhat surprising result, because there is no penalty on idling so one would expect the sub-optimal algorithm to fully trade off delay for less transmission, while under Lott's algorithm transmission occurs every time step. What this suggests is that as the sleep probability increases, even though Lott's algorithm keeps busy, a lot of its transmissions either fall on deaf ears (neighbors are asleep) which does not help reduce delay, or they result in longer routes (wake neighbors happen to lead to bad/long routes) which



(a) Average cost.

(b) Average delay.

Figure 7: The effect of idle cost on the performance of the sub-optimal algorithm on Topology 3.

in turn can increase delay. Figure 6(c) shows the amount of idle action taken by the sub-optimal algorithm (Lott's algorithm takes no idle actions).

7.3 The role of idle costs

As described above, Lott's Algorithm is invariant to changes in idle cost. In this subsection, we examine more closely the sub-optimal algorithm on Topology 3 while varying the idle cost by selecting it from the set $c_I = \{0, 1, 2, 4, 8\}$.

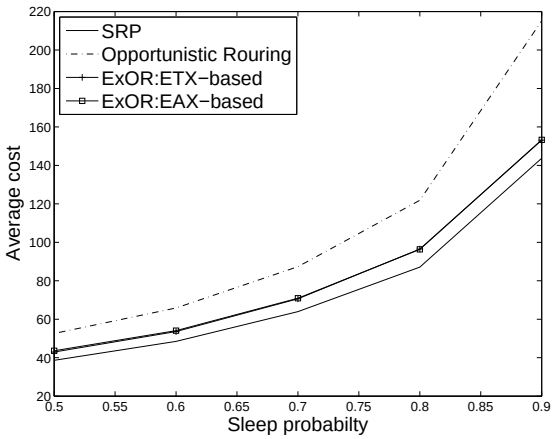
As shown in Figure 7, as c_I grows the average cost tends to increase but the average delay decreases. c_I is therefore a design parameter that can be used to tune this tradeoff. Specifically, one may try to find a certain c_I^* to satisfy some cost efficiency and delay constraint.

7.4 The performance of the distributed protocol SRP

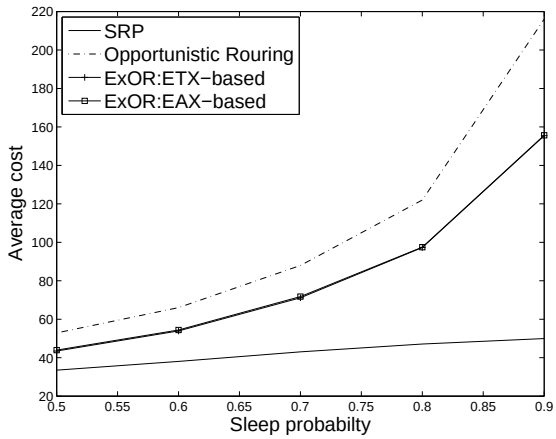
We next evaluate the performance of SRP on Topology 3 with 30 nodes and $p_m = 0.3$ as illustrated in Figure 5(b). As described in Section 6, the distributed algorithm's access to sleep state is limited to a node's 1-hop neighbors, which is obtained from the beacons broadcasted by neighbors every T time units. In our simulation, T is set to 2. Each node's sleep schedule is generated by a geometric distribution with mean length of on periods of 4 (the average off period is then determined from the sleep probability).

Given the scenarios of cost distributions introduced earlier, we examine the performance of SRP described in Section 6 with respect to three variations of ExOR with different forwarder selection metrics: 1) the number of hops to best-path and loss rate [13], 2) ETX [10], and 3) EAX [11]. In each simulation 300 packets are randomly generated during 3000 time units of simulated time. Each node has a finite queue so that the total delay takes into account queueing in addition to hop counts and the amount of waiting.

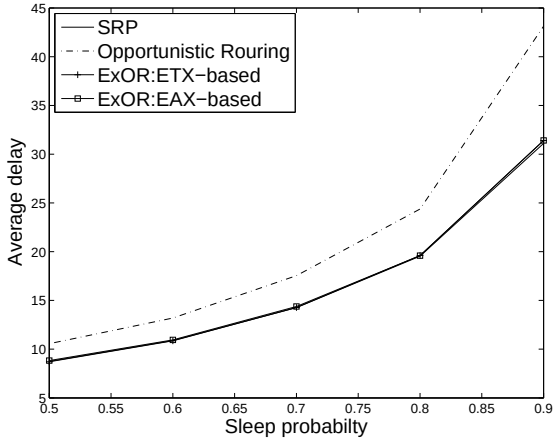
Figure 8(a) depicts the average cost of these algorithms when transmission costs are distributed uniformly with a mean 4 and idle cost is 4. ExOR (labeled as opportunistic routing in the figures), which is known to outperform traditional routing where packets are sent to the pre-computed path with the smallest costs, performs the worst among the set in the figure. Using ETX and EAX metrics performs better than the original ExOR. Figures 8(b) and 8(c) compare the cost and delay



(a) Average cost (scenario 2).



(b) Average cost (scenario 3).



(c) Average delay (scenario 3).

Figure 8: Performance comparison between SRP and three variants of ExOR.

of these protocols under scenario 3 with the same distribution for transmission costs and zero idle cost. We see that the average cost of SRP is the smallest, while its delay performance is virtually the same as ETX and EAX. This shows that, through judicious waiting, SRP attains the same delay performance but manages to significantly lower the transmission cost, thereby saving energy.

8 Conclusion

In this paper we studied a routing problem in wireless sensor networks where sensors are randomly duty-cycled. We developed an optimal stochastic routing framework in the presence of duty-cycling as well as unreliable wireless channels. Using this framework, we presented and analyzed an optimal centralized stochastic routing algorithm, and then simplified the algorithm when only local sleep/wake states of neighbors are available. We further developed a distributed algorithm utilizing local sleep/wake states of neighbors which performs better than some existing distributed algorithms such as ExOR.

References

[1] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *ACM Mobile Networks and Applications Journal*, Oct. 1996.

[2] Charles E. Perkins and Pravin Bhagwat, "Highly dynamic destination sequenced distance vector routing (dsdv) for mobile cmputers," in *Conference of the Special Interest Group on Data Communication (SIGCOMM)*, Oct. 1994.

[3] David B. Johnson and David A. Maltz, *Dynamic source routing in ad hoc wireless networks*, Mobile computing, Kluwer Academic Publishers, 1996.

[4] 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.

[5] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva, "Directed diffusion for wireless sensor networking," *IEEE Transactions on Networking*, vol. 11, no. 1, 2003.

[6] Tommaso Melodia, Dario Pompili, and Ian 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.

[7] Karim Seada, Marco Zuniga, Ahmed Helmy, and Bhaskar Krishnamachari, "Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2004.

[8] Dario Ferrara, Laura Galluccio, Alessandro Leonardi, Giacomo Morabito, and Sergio 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.

[9] Christopher Lott and Demosthenis Teneketzis, "Stochastic routing in ad-hoc networks," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, 2006.

[10] Sanjit Biswas and Robert Morris, "ExOR: Opportunistic multi-hop routing for wireless networks," in *Conference of the Special Interest Group on Data Communication (SIGCOMM)*, Aug. 2005.

[11] Zifei Zhong and Srihari Nelakuditi, "On the efficacy of opportunistic routing," in *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2007.

[12] 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.

[13] Sanjit Biswas and Robert Morris, "Opportunistic routing in multi-hop wireless networks," in *Workshop on Hot Topics in Networks (HotNets-II)*, Nov. 2003.

A Detailed Calculation for the Optimal Policy in Example 1

In Example 1, we show the optimal policy $\pi^*(W, A)$ for the network illustrated in Figure 1, given $W = \{1, 2, 4\}$ and $A \in F(W)$. Note that $F(W) = \{\{3a, 5a\}, \{3a, 5s\}, \{3s, 5a\}, \{3s, 5s\}\}$. When all nodes are awake, i.e., $A = \{3a, 5a\}$,

$$\begin{aligned} &V^{\pi^*}(\{1,2,4\},\{3a,5a\}) \\ &\quad = \max_{i\in\{1,4,I\}}\{-c_i + \sum_{W'\supseteq\{1,2,4\}}\sum_{A'\in F(W')} P^i(W',A'|\{1,2,4\},\{3a,5a\})V^{\pi^*}(W',A')\} \\ &\quad = -1 + \max_{i\in\{1,4,I\}}\{\sum_{W'\supseteq\{1,2,4\}}\sum_{A'\in F(W')} P^i(W',A'|\{1,2,4\},\{3a,5a\})V^{\pi^*}(W',A')\}, \end{aligned}$$

where the second equality is based on the assumption of unit cost.

When node 1 is transmitting, possible W' is $\{1, 2, 4\}$ with probability 0.2 or $\{1, 2, 3, 4\}$ with probability 0.8. Then, the term in max function with $i = 1$ is calculated as follows.

$$\begin{aligned} &\sum_{A'\in F(\{1,2,4\})} 0.2P(A')V^{\pi^*}(\{1,2,4\},A') + \sum_{A'\in F(\{1,2,3,4\})} 0.8P(A')V^{\pi^*}(\{1,2,3,4\},A') \\ &\quad = \sum_{A'\in F(\{1,2,4\})} 0.2P(A')V^{\pi^*}(\{1,2,4\},A') + 0.08V^{\pi^*}(\{1,2,3,4\},\{5a\}) \\ &\qquad + 0.72V^{\pi^*}(\{1,2,3,4\},\{5s\}), \end{aligned} \tag{24}$$

where $V^{\pi^*}(\{1, 2, 3, 4\}, \{5a\})$ and $V^{\pi^*}(\{1, 2, 3, 4\}, \{5s\})$ are calculated similarly.

Since $\pi^*(\{1, 2, 3, 4\}, \{5a\}) = 3$ and $\pi^*(\{1, 2, 3, 4\}, \{5s\}) = I$,

$$\begin{aligned} V^{\pi^*}(\{1,2,3,4\},\{5a\}) &= -1 + 0.028V^{\pi^*}(\{1,2,3,4\},\{5a\}) \\ &\quad + 0.252V^{\pi^*}(\{1,2,3,4\},\{5s\}) + 0.72R_5 \\ V^{\pi^*}(\{1,2,3,4\},\{5s\}) &= -1 + 0.1V^{\pi^*}(\{1,2,3,4\},\{5a\}) + 0.9V^{\pi^*}(\{1,2,3,4\},\{5s\}). \end{aligned}$$

From the above simultaneous equations, we obtain $V^{\pi^*}(\{1, 2, 3, 4\}, \{5a\}) = -4.8889 + R_5$ and $V^{\pi^*}(\{1, 2, 3, 4\}, \{5s\}) = -14.8889 + R_5$. Thus, Eqn. (24) becomes

$$\sum_{A'\in F(\{1,2,4\})} 0.2P(A')V^{\pi^*}(\{1,2,4\},A') - 11.1111 + 0.8R_5.$$

If node 4 is transmitting, i.e., $i = 4$, possible W' is $\{1, 2, 4\}$ with probability 0.4 or $\{1, 2, 4, 5\}$ with probability 0.6 and the term in max function is

$$\begin{aligned}
&\sum_{A' \in F(\{1,2,4\})} 0.4P(A')V^{\pi^*}(\{1,2,4\}, A') + \sum_{A' \in F(\{1,2,4,5\})} 0.6P(A')V^{\pi^*}(\{1,2,4,5\}, A') \\
&= \sum_{A' \in F(\{1,2,4\})} 0.4P(A')V^{\pi^*}(\{1,2,4\}, A') + 0.06V^{\pi^*}(\{1,2,4,5\}, \{3a\}) \\
&\qquad + 0.54V^{\pi^*}(\{1,2,4,5\}, \{3s\}) \\
&= \sum_{A' \in F(\{1,2,4\})} 0.4P(A')V^{\pi^*}(\{1,2,4\}, A') + 0.6R_5.
\end{aligned}$$

If I is chosen, it is just $\sum_{A' \in F(\{1,2,4\})} P(A')V^{\pi^*}(\{1, 2, 4\}, A')$.
Let $S \triangleq \sum_{A' \in F(\{1,2,4\})} P(A')V^{\pi^*}(\{1, 2, 4\}, A')$. Then, combining these together, we have

$$V^{\pi^*}(\{1, 2, 4\}, \{3a, 5a\}) = \max\{0.2S + 0.8R_5 - 12.1111, 0.4S + 0.6R_5 - 1, S - 1\} \tag{25}$$

Similarly, $V^{\pi^*}(\{1, 2, 4\}, A)$ is calculated for the remaining $A \in F(\{1, 2, 4\})$. Then, we have $V^{\pi^*}(\{1, 2, 4\}, \{3a, 5s\}) = \max\{0.2S+0.8R_5-12.1111, S-1\}$, $V^{\pi^*}(\{1, 2, 4\}, \{3s, 5a\}) = \max\{0.4S+0.6R_5 - 1, S - 1\}$ and $V^{\pi^*}(\{1, 2, 4\}, \{3s, 5s\}) = S - 1$. Intuitively, the optimal choices are straight-forward for some A so that $\pi^*(\{1, 2, 4\}, \{3a, 5s\}) = 1$, $\pi^*(\{1, 2, 4\}, \{3s, 5a\}) = 4$ and $\pi^*(\{1, 2, 4\}, \{3s, 5s\}) = I$. Thus,

$$\begin{aligned}
S = &0.01V^{\pi^*}(\{1, 2, 4\}, \{3a, 5a\}) + 0.09(0.2S + 0.8R_5 - 12.1111) \\
&+ 0.09(0.4S + 0.6R_5 - 1) + 0.81(S - 1).
\end{aligned}$$

For $A = \{3a, 5a\}$, $S = R_5 - 15.7546$ when node 1 is chosen whereas $S = R_5 - 98.6447$ when node 4 is chosen. Thus, the maximum of $V^{\pi^*}(\{1, 2, 4\}, \{3a, 5a\})$ is achieved when node 4 is transmitting. Hence, $\pi^*(\{1, 2, 4\}, \{3a, 5a\}) = 4$.

B The proof of Lemma 1

If $\pi(W_1, A_1) = \pi^*(W_1, A_1) = r_i$ for some $i \in W_1$ and some A_1, Eqn. (10) holds.

Next, we consider the case where both policies π and π^* do not retire but transmit or wait. Suppose $\pi(W_1, A_1) = \pi^*(W_1, A_1) = I$ for some A_1. Let (W_2, A_2) and (W_2^*, A_2^*) be the state after the idle action when in (W_1, A_1) for π and π^*. Obviously, W_2 and W_2^* are the same as W_1 while A_2 is the same as A_2^* for any given sample path, but not necessarily the same as A_1. Both π and π^* pay the idle costs until they reach the state for transmission.

Suppose $\pi(W_1, A_1) = \pi^*(W_1, A_1) = i \in W_1$ for some A_1. Let (W_2, A_2) and (W_2^*, A_2^*) be the state after i's transmission when in (W_1, A_1) for π and π^*. Since node i is transmitting for both policies, $W_2 = W_2^* \supseteq W_1$. Again A_2 is the same as A_2^* for any given sample path. By Eqn. (8), we have $V^{\pi}(W_2, A_2) = V^{\pi^*}(W_2^*, A_2^*)$ for $W_2 = W_2^* \supset W_1$ if at least one node receives the packet successfully. Otherwise, we have $W_2 = W_2^* = W_1$ and $A_2 = A_2^*$, which may or may not be different from A_1. Similar to the case of choosing the idle action, by Eqn. (9), π chooses the same action (the idle action or transmission but fail) as π^* until it reaches the state where $W_2 \supset W_1$ and any A_2. Hence, Eqn. (10) holds.

C The proof of Lemma 2

We define a new policy π on state (W_1, A_1) as follows. Suppose (W_4, A_4) is the state after transmission or idle action by π^* when in (W_2, A_2), where $W_4 \supseteq W_2$. Let π make the same decision as π^* did, which is possible because the node in $W_2 \cup \{I\}$ chosen by π^* is also available in the set $W_1 \cup \{I\} \subseteq W_2 \cup \{I\}$. Let (W_3, A_3) be the state after transmission or idle action by π when in (W_1, A_1). The nodes which are not in W_1 and receive the packet are included in W_3 as well as W_4. However, the nodes which are not in W_2 but in W_1 and receive the packet are included in W_4 whereas W_3 contains all nodes in W_1. Hence, $W_3 \supseteq W_4$. Accordingly, the sleep/wake states of the nodes in $\Omega - W_3$ are the same as A_3 while the nodes in $W_3 - W_4$ may be in different sleep/wake states. Therefore, $A_4 \in F(W_4|W_3, A_3)$.

At the next step, π acts on (W_3, A_3) by choosing the same node as π^* acts on (W_4, A_4). The process repeats in the same way until π retires when π^* does. Let (W_{f1}, A_{f1}) and (W_{f2}, A_{f2}) be the states at retirement for π and π^*, respectively. We have $W_{f2} \subseteq W_{f1}$ and $A_{f2} \in F(W_{f2}|W_{f1}, A_{f1})$. Total cost incurred by π is the same as π^* because both policies chose the same nodes at every step before retirement. At retirement, π and π^* receive rewards $R(W_{f1})$ and $R(W_{f2})$, respectively. $R(\cdot)$ is a G-index function because it satisfies Eqn. (5). Thus, $W_{f1} \supseteq W_{f2}$ results in $R(W_{f1}) \geq R(W_{f2})$ which proves $V^{\pi}(W_1, A_1) \geq V^{\pi^*}(W_2, A_2)$. Finally, because π^* is optimal, $V^{\pi^*}(W_1, A_1) \geq V^{\pi}(W_1, A_1)$ holds. This completes the proof.

D The proof of Lemma 3

The proof is constructive. Let us define π recursively using the following rules:

$$\pi(\Omega, A) = \pi^*(\Omega, A), \quad A = \{1, 1, \cdots, 1\}, \tag{26}$$

$$\pi(W - \{j\}, A) = \pi(W, A_i), \quad \forall W \subseteq \Omega, \forall A_i \in F(W), \forall A \in F(W - \{j\}|W, A_i), \\ \forall j \in W : \pi(W, A_i) \neq j, r_j \text{ for } \forall A_i, \tag{27}$$

$$\pi(W - \{j\}, A) = \pi^*(W - \{j\}, A), \quad \forall W \subseteq \Omega, \forall A \in F(W - \{j\}|W, A_i), \\ \forall j \in W : \pi(W, A_i) = j, r_j \text{ for some } A_i. \tag{28}$$

If $N = 1$, the lemma is true directly by Eqn. (26). Hence, we assume that $N \geq 2$.

Eqn. (27) shows that π satisfies Eqn. (11) and Eqn. (12) in the first property of this lemma. We now focus on its second property. We prove Eqn. (13) by backward induction on the cardinality of W. As the induction basis, we show Eqn. (13) is true for $W = \Omega$ and $A = \{1, 1, \cdots, 1\}$. We know that $\pi^*(\Omega, A) = r_i$ for some i such that $i = \arg\max_{k \in \Omega} R_k$ because π^* is optimal. Thus, $V^{\pi^*}(\Omega, A) = R_i$. According to Eqn. (26), $\pi(\Omega, A) = r_i$ and $V^{\pi}(\Omega, A) = R_i$ which proves the second equality in Eqn. (13). In order to show the first and third equalities, let A_1 be all ones but zero for node $j \in \Omega - \{i\}$. By Eqn. (27), we have $\pi(\Omega - \{j\}, A) = \pi(\Omega - \{j\}, A_1) = \pi(\Omega, A) = r_i$ which means that π retires and receives R_i. Thus, $V^{\pi}(\Omega - \{j\}, A) = V^{\pi}(\Omega - \{j\}, A_1) = R_i$. This proves its first equality of Eqn. (13). For $\forall j \in \Omega - \{i\}$, we have $\pi^*(\Omega - \{j\}, A) = r_i$ and $V^{\pi^*}(\Omega - \{j\}, A) = R_i$ because the optimal policy π^* chose node i in Ω which is still in the set $\Omega - \{j\}$ and has the highest reward among nodes in $\Omega - \{j\}$. Similarly, $\pi^*(\Omega - \{j\}, A_1) = r_i$ and $V^{\pi^*}(\Omega - \{j\}, A_1) = R_i$. This proves the last equality of Eqn. (13) for $W = \Omega$.

As the induction hypothesis, assume that Eqn. (13) holds for any state (W, A) where $|W| = L + 1$ and any possible $A \in F(W)$. If $N = 2$, the basis completes the proof of Eqn. (13). Thus, we assume $N > 2$ and $2 \leq L < N$. Consider a state (W_1, A_i) where $|W_1| = L$ and $A_i \in F(W_1)$. If there is $j \in \Omega - W_1$ such that $\pi(W_1 \cup \{j\}, F(W_1 \cup \{j\}|W_1, A_i)) \neq j, r_j$, then we have $\pi(W_1, A_i) = \pi(W_1 \cup \{j\}, F(W_1 \cup \{j\}|W_1, A_i))$ by Eqn. (27). By the induction hypothesis,

Eqn. (13) is true for $W = W_1 \cup \{j\}$. Thus, we have $V^{\pi}(W_1 \cup \{j\} - \{j\}, A_i) = V^{\pi^*}(W_1 \cup \{j\} - \{j\}, A_i)$ which proves the second equality of Eqn. (13). That is,

$$V^{\pi}(W_1, A_i) = V^{\pi^*}(W_1, A_i). \tag{29}$$

On the other hand, if there is $j \in \Omega - W_1$ such that $\pi(W_1 \cup \{j\}, F(W_1 \cup \{j\}|W_1, A_i)) = j$ or r_j, then by Eqn. (28) $\pi(W_1, A_i) = \pi^*(W_1, A_i)$. By the induction hypothesis, we have $V^{\pi}(W, A_i) = V^{\pi^*}(W, A_i)$ for $\forall W \supset W_1$. By Lemma 1 we proved that Eqn. (29) holds for this case. We have shown that the second equality of Eqn. (13) holds for any W_1 where $|W_1| = L$ and any $A_i \in F(W_1)$.

In order to show the first and third equalities of Eqn. (13) below, we note that there are two cases: either $\pi(W_1, A_i) = n_i \in W_1 \cup \{I\}$ or $\pi(W_1, A_i) = r_{n_i}$ for all $A_i \in F(W_1)$. Let $j \in W_1$, $j \notin N_{W_1}$ where $N_{W_1} = \bigcup_{i=1}^{m_1} n_i - \{I\}$ and $m_1 = 2^{N-|W_1|}$. Consider the case where $\pi(W_1, A_i) = r_{n_i}$. By Eqn. (27) $\pi(W_1 - \{j\}, A') = r_{n_i}$, $\forall A' \in F(W_1 - \{j\}|W_1, A_i)$. This implies that $V^{\pi}(W_1, A_i) = V^{\pi}(W_1 - \{j\}, A') = R_{n_i}$, $\forall A' \in F(W_1 - \{j\}|W_1, A_i)$. For an optimal policy π^*, since $W_1 - \{j\} \subset W_1$ and $j \notin N_{W_1}$, by Lemma 2 we get $V^{\pi^*}(W_1 - \{j\}, A') \leq V^{\pi^*}(W_1, A_i)$, $\forall A' \in F(W_1 - \{j\}|W_1, A_i)$. By Eqn. (29) we have $V^{\pi^*}(W_1 - \{j\}, A') \leq R_{n_i}$, $\forall A' \in F(W_1 - \{j\}|W_1, A_i)$. On the other hand, because $i \in W_1 - \{j\}$ and π^* is an optimal policy, $V^{\pi^*}(W_1 - \{j\}, A') \geq R_{n_i}$, $\forall A' \in F(W_1 - \{j\}|W_1, A_i)$. Hence,

$$V^{\pi^*}(W_1 - \{j\}, A') = R_{n_i}, \forall A' \in F(W_1 - \{j\}|W_1, A_i).$$

This completes the proof of the Eqn. (13) for $\pi(W_1, A_i) = r_{n_i}$.

We now prove the first and the third equalities of Eqn. (13) in the case of $\pi(W_1, A_i) = n_i \in W_1 \cup \{I\}$. Let us prove the first equality as follow. Let $W \supseteq W_1 - \{j\}$, $j \notin N_W$. We first show the following.

$$\pi(W, A) \neq j, r_j, \quad \forall A. \tag{30}$$

We prove this in two cases: $j \in W$ and $j \notin W$. If $j \notin W$, $\pi(W, A) \neq j, r_j$ for any A. If $j \in W$, $W_1 \subseteq W$ and $|W| \geq L$. If $|W| = L$, $W = W_1$ and $\pi(W, A) \neq j, r_j$ for any A because of $j \notin N_W$ as given. Assume $|W| > L$. If $\pi(W, A) = j$ for some A, removing all nodes from $W - W_1$ one by one results in $\pi(W_1, A_i) = j$ by Eqn. (27), for some A_i which has the same values for nodes in $\Omega - W$ and arbitrary values for nodes in $W - W_1$. This contradicts the hypothesis which is $\pi(W_1, A_i) = n_i \neq j$. Similarly if $\pi(W, A) = r_j$, we have $\pi(W_1, A_i) = r_j$ for some A_i. We have shown that Eqn. (30) is true in all cases when $\pi(W_1, A_i) = n_i$. Then, the following is true for any $W' \supseteq W_1$, any $A' \in F(W')$, and any $\tilde{A} \in F(W_1 - \{j\}|W_1, A_i)$:

$$P^{n_i}(W', A'|W_1, A_i) = P^{n_i}(W', A'|W_1 - \{j\}, \tilde{A}) + \sum_{A'' \in F(W' - \{j\}|W', A')} P^{n_i}(W' - \{j\}, A''|W_1 - \{j\}, \tilde{A}). \tag{31}$$

By Eqn. (30) and Eqn.(31), we have $V^{\pi}(W_1 - \{j\}, \tilde{A}) = V^{\pi}(W_1, A_i)$ for $\forall \tilde{A} \in F(W_1 - \{j\}|W_1, A_i)$. Next, we prove the third equality of Eqn. (13) in case of $\pi(W_1, A_i) = n_i$. By Lemma 2 we have $V^{\pi^*}(W_1, A_i) \geq V^{\pi^*}(W_1 - \{j\}, \tilde{A})$ for $\forall \tilde{A} \in F(W_1 - \{j\}|W_1, A_i)$. In addition, π^* is optimal so that $V^{\pi^*}(W_1 - \{j\}, \tilde{A}) \geq V^{\pi}(W_1 - \{j\}, \tilde{A})$. Since $V^{\pi}(W_1, A_i) = V^{\pi^*}(W_1, A_i)$ by Eqn. (29),

$$V^{\pi}(W_1 - \{j\}, \tilde{A}) = V^{\pi}(W_1, A_i) = V^{\pi^*}(W_1, A_i) \geq V^{\pi^*}(W_1 - \{j\}, \tilde{A}) \geq V^{\pi}(W_1 - \{j\}, \tilde{A}). \tag{32}$$

This proves Eqn. (13) for $\pi(W_1, A_i) = n_i$. We have shown that Eqn. (13) is true for all $W \subseteq \Omega$ where $|W| \geq 2$ and all possible $A_i \in F(W)$.

We prove now that π is an optimal Markov policy. As we showed in the second property, $V^{\pi}(W, A) = V^{\pi^*}(W, A)$ for any W where $|W| \geq 2$ and any A. From this relationship Eqn. (13), we also have $V^{\pi}(\{i\}, A) = V^{\pi^*}(\{i\}, A)$ for $\forall i \in \Omega$ such that $\pi(\{i\} \cup \{j\}, A) = i$ or I for all A where $j \in \Omega$. If there is no such i left, we still have $V^{\pi}(\{j\}, A) = V^{\pi^*}(\{j\}, A)$ for $\forall j \in \Omega$ by Eqn. (28) when $\pi(\{i\} \cup \{j\}, A) = i \in \Omega$ for all A.

E The proof of Lemma 4

By Eqn. (32), we know $V^{\pi^*}(\cdot)$ satisfies

$$V^{\pi^*}(W \cup \{i\}, F(W \cup \{i\}|W, A)) \geq V^{\pi^*}(W, A), \forall W \subseteq \Omega, \forall A \in F(W), i \in W. \tag{E-1}$$

Consider a Markov policy π which satisfies Eqn. (11) and Eqn. (12). Suppose $V^{\pi}(W, A) = V^{\pi}(W_1, A_1)$ for some $W_1 \subset W$ and $A_1 \in F(W_1)$ s.t. $i \notin W_1$. This implies $\pi(W, A) \neq i$ for all $A \in F(W)$. Then, by Eqn. (13) we have $V^{\pi^*}(W, A) = V^{\pi^*}(W - \{i\}, \tilde{A})$, $\forall \tilde{A} \in F(W - \{i\}|W, A)$. From the above properties of $V^{\pi^*}(\cdot)$, we conclude that $V^{\pi^*}(W, A) \geq V^{\pi^*}(W_1, A_1)$, $\forall W_1 \subseteq W$ and $\forall A_1 \in F(W_1)$. There exists $W_1 \subseteq W$ such that $V^{\pi^*}(W, A) = V^{\pi^*}(W_1, A_1)$ for each $A \in F(W)$. This satisfies Eqn. (5).